IEKP-KA/2003-1

Eine Hard- und Software-Umgebung für Physik-Analysen bei CDF II

Patrick Schemitz

Zur Erlangung des akademischen Grades eines
DOKTORS DER NATURWISSENSCHAFTEN
von der Fakultät für Physik der Universität (TH)
Karlsruhe

genehmigte

DISSERTATION

von

Dipl. phys. Patrick Schemitz
aus Karlsruhe

Tag der mündlichen Prüfung: 20.12.2002
Referent: Prof. Dr. Michael Feindt, Institut für Experimentelle Kernphysik
Korreferent: Prof. Dr. Günter Quast, Institut für Experimentelle Kernphysik

Die Blätter fall'n, wer heute schreit ist morgen schon gewesen.
Die Zeile die mein Leben schreibt wird niemand lesen.

– Subway to Sally: *Zu spät*

Inhaltsverzeichnis

Übersicht

Moderne Teilchenphysik-Detektoren erzeugen enorme Datenmengen, die nur mit Hilfe einer äußerst leistungsfähigen Computer-Infrastruktur aufgezeichnet, verarbeitet und ausgewertet werden können. Die vorliegende Arbeit beschreibt die Rechner-Infrastruktur des *Collider Detector at Fermilab* (CDF), eines Detektors am mit 1.96 TeV derzeit stärksten Teilchenbeschleuniger der Welt, dem Tevatron. Das CDF-Experiment wurde in den vergangenen Jahren sehr verbessert. So wurde unter anderem ein neues Silizium-System mit 8 Lagen eingebaut; drei dieser Lagen messen φ und z, vier messen φ und 1.2° und eine Lage mißt nur φ.

Der erweiterte Detektor machte die Überarbeitung der Rekonstruktionssoftware unausweichlich. Dabei wurden im großen Stil objekt-orientierte Methoden eingesetzt, was einen radikalen Bruch mit der alten Software darstellt. Die vorliegende Arbeit gibt einen Überblick über die CDF II-Analysekette. Es wurde ein Zugang zu den Monitoringprogrammen über das WWW eingerichtet. Damit ist es möglich, mit einem Web-Browser Run-Informationen wie Qualitätsmerkmale oder Überwachungshistogramme abzurufen. Diese werden von den *Consumern* definiert, die Ereignisse in Echtzeit analysieren und nach verschiedenen Kriterien wie Triggerrate, Strahlposition und Aktivität im Silizium bewerten.

Der Entwurf der *Kaltrak*-Spurrekonstruktionssoftware stellt einen weiteren Schwerpunkt dieser Arbeit dar. Es wird gezeigt, wie mit den Methoden des *Multi Paradigm Design* ein flexibles und schnelles Silizium-Tracking-Paket entworfen und implementiert wurde. Dabei wurde großen Wert auf einen schnellen Entwicklungszyklus (Editieren-Compilieren-Testen) und auf gute Wartbarkeit gelegt. Das resultierende Design ist einfach und leicht zu handhaben. Auch die verwendeten Rekonstruktionsalgorithmen werden kurz vorgestellt.

Die enormen Datenmengen und zum Teil sehr rechenintensiven Algorithmen verlangen nach einer leistungsfähigen Rechner-Infrastruktur. Diese wurde in Form des EKPplus-Linux-Clusters aufgebaut, der ebenfalls in dieser Arbeit beschrieben wird. Dabei wird zunächst die Software vorgestellt, die auf dem Cluster laufen soll. Dies ist in erster Linie die CDF II-Umgebung sowie benötigte Fremdprodukte, vor allem der KAI C++ Compiler und der Oracle-Datenbank-Client. Auch auf die Software zur Zuteilung der Rechenleistung, das Batchsystem OpenPBS, und auf das speziell für die EKPplus entwickelte Überwachungsprogramm Fleximon wird eingegangen. Nach einer allgemeinen Einführung in die Thematik wird die Struktur des EKPplus-Netzwerks inclusive Firewall erläutert. Die Konfiguration der einzelnen Maschinen bildet den Abschluß des Kapitels.

Anhand einer Beispielanalyse im Kanal $J/\Psi \to \mu^+\mu^-$ wird gezeigt, daß die vorgestellte Hard- und Software-Umgebung tatsächlich funktioniert und physikalische Ergebnisse liefert. Dazu wurden Monte Carlos und knapp 240 nb^{-1} neu vom CDF II-Detektor genommene Daten mit dem *Kaltrak*-Paket prozessiert und anschließend analysiert. Anhand der Zerfallslänge

der rekonstruierten J/Ψ wird gezeigt, daß die Daten sowohl prompte J/Ψ als auch solche aus b-Zerfällen enthalten.

Der Ausblick am Ende der Arbeit schließlich weist auf zukünftige Entwicklungen im Bereich Teilchenphysik und Computing, etwa neue Netze und Speichertechnologien, hin und erläutert das *Grid*, ein Konzept zum weltweit verteilten Rechnen, mit dem die Herausforderungen der enormen Datenmengen, die der *Large Hadron Collider* (LHC) liefert, bewältigt werden sollen.

Kapitel 1

Einführung und Motivation

1.1 Teilchenphysik und Rechnernutzung

Da die Teilchenphysik sehr stark auf statistische Methoden als Analysetechnik zurückgreift, besteht seit jeher der Wunsch nach immer mehr Ereignissen. Auch sind viele Teilchen und Zerfallsmodi so selten, daß nur immer größere Ereignisraten aussagekräftige Messungen zulassen. Dabei wurde es sehr bald zwingend notwendig, weite Teile der Analysekette dem Computer zu überlassen.

Auch die Detektoren haben sich in diese Richtung weiterentwickelt. Wurden in der Anfangszeit der Teilchenphysik Ereignisse noch photographiert, so werden die heutigen Detektoren vollständig elektronisch ausgelesen, und die Zahl der Auslesekanäle wächst mit jedem neuen Detektor.

Heutige Hochenergiephysik-Detektoren haben $O(10^6)$ Auslesekanäle, die mit einem Takt von mehreren 10 MHz ausgelesen werden; die Meßapparatur nimmt also größenordnungsmäßig mehrere Terabyte Daten pro Sekunde[1]. Dies erklärt die überragende Bedeutung von Computern in der Teilchenphysik.

1.2 Entwicklung

Die frühen Detektoren, Blasen-, Nebel- und Funkenkammern, wurden "ausgelesen", indem die Ereignisse, die der Detektor sichtbar machte, photographiert wurden. Diese Photographien wurden dann von Hand ausgewertet. Die Nachteile dieses Verfahrens liegen auf der Hand: viel Statistik ist nicht machbar. Die Detektoren selbst haben sehr große Latenzzeiten, auch das Photographieren ist ein zeitaufwendiger Prozess. Das Entwickeln schließlich und die manuelle Auswertung der Aufnahme sind sowohl zeit- als auch arbeitsintensive Schritte.

Drei Voraussetzungen zum Erreichen großer Statistik sind:

- *Luminosität des Beschleunigers* bzw. die Anzahl der vom Beschleiniger erzeugten Ereignisse. Auf die Funktionsweise der verschiedenen Beschleunigertypen wird in dieser Arbeit nicht weiter eingegangen.

[1] Nur ein geringer Anteil davon wird tatsächlich genauer analysiert. Die Filterung durch die sogenannten "Trigger" obliegt ebenfalls Computern.

- *Schnelligkeit des Detektors.* Sowohl die Meßzeit, als auch die Totzeit zwischen den Messungen muß gering sein, um die hohe Luminosität des Beschleunigers nutzen zu können, da es sonst zu Überlagerungen von Ereignissen bzw. zu verlorenen Ereignissen kommt.

- *Schnelligkeit der Auswertung.* Die Daten müssen zeitnah an der Messung ausgewertet werden, um Probleme im Detektor oder Beschleuniger schnell erkennen zu können.

1.2.1 Detektor-Entwicklungen

Die wichtigste Entwicklung im Detektorbereich zum Erzielen höherer Datenraten war sicherlich die Entwicklung von vollständig elektronisch ausgelesenen Detektoren. Doch das allein ist nicht ausreichend; so wäre es durchaus denkbar, beispielsweise eine Blasenkammer statt mit herkömmlicher, chemischer Photographie mit Digitalkameras auszulesen. Damit wäre aber nur ein Teil des Problems gelöst, denn der Detektor selbst hätte immer noch enorm lange Meß- und Totzeiten, bis die sich Blasen ausgebildet haben bzw. bis sie wieder verschwunden sind. Die derzeit eingesetzten Detektoren umgehen diese Problematik durch verschiedene Maßnahmen. Bei Driftkammern zum Beispiel ist die Driftgeschwindigkeit der Elektronen und vor allem die der Ionen der beschränkende Faktor. Wichtig beim Entwurf ist also, die Driftwege möglichst kurz zu halten[2]. Siliziumdetektoren, die als Streifen- oder als Pixel-Ausführung bei allen modernen Hochenergiephysik-Experimenten zum Einsatz kommen, sind hier nicht so kritisch, da die Driftwege innerhalb der Sperrschicht lediglich wenige $100\mu m$ betragen und die Signale danach elektronisch weiterlaufen. Andererseits haben Siliziumdetektoren den Nachteil, daß sie aus dichterem Material bestehen als das Gas einer Driftkammer und dadurch stärker mit den zu messenden Teilchen wechselwirken: der Energieverlust der durchgehenden Teilchen ist größer, ebenso wie die Ablenkung von ihrer ursprünglichen Bahn durch Vielfachstreuung. Dazu kommen je nach Bauweise auch noch die Ausleseelektronik, die sich oft ebenfalls im Zentralbereich des Detektors befindet (so auch bei CDF).

1.2.2 Rechner-Entwicklungen

In der Entwicklung von Computern gilt nach wie vor Moore's Gesetz, welches besagt, daß sich die Rechenleistung von Computern etwa alle 18 Monate verdoppelt[3]. Die Speicherkapazität von Festplatten hat sich im letzten Jahrzehnt vertausendfacht, also jährlich verdoppelt[4]. Dagegen nimmt sich der Faktor 60 bei der Hauptspeicherausstattung schon beinahe bescheiden aus[5]. Während ein Arbeitsplatz-PC nach drei Jahren noch ausreichend sein mag, ist ein Rechenknoten nach dieser Zeit völlig veraltet. Bei vergleichbarer Stellfläche, Abwärme

[2] Aus diesem Grund sind Zeitprojektionskammern (TPC) auch nicht für höchste Luminositäten geeignet – hier wird schließlich gerade die Driftzeit zur Positionsbestimmung in z ausgenutzt. Daher werden in Hadronmaschinen, in denen die Teilchendichten ja besonders groß sind, keine TPCs eingesetzt.

[3] Moore's Gesetz gilt rein empirisch; es gibt keine Theorie, die gerade diese (oder eine andere) Wachstumsrate begründen würde.

[4] Festplatte eines typischen Arbeitsplatz-PCs 1992: 50 MB → 2002: 40 GB.

[5] Hauptspeicher eines typischen Arbeitsplatz-PCs 1992: 4 MB → 2002: 256 MB.

und Anschaffungskosten ist ein aktuelles System dann viermal so leistungsstark. Anders ausgedrückt, ist ein Rechner-Cluster mit 5 modernen Knoten genauso leistungsstark wie ein drei Jahre altes System mit 20 Knoten. Es ist daher unerläßlich, bei der Anschaffung eines Clusters den Austausch aller Rechenknoten spätestens alle drei, besser alle zwei Jahre, mit einzuplanen, um Veralten vorzubeugen.

1.2.3 Eine Meta-Entwicklung: Entwicklung der Software-Entwicklung

Die Entwicklungsgeschichte der Programmierung läßt sich anhand der vorherrschenden Programmier-Paradigmen einteilen, die im folgenden kurz vorgestellt werden. Ein Programmier-Paradigma beschreibt die Art, ein Programm zu entwerfen und zu strukturieren. Dabei ist es ein großer Unterschied, ob eine Programmiersprache ein Paradigma wie zum Beispiel strukturierte oder objekt-orientierte Programmierung *erlaubt* oder *unterstützt*. Man kann auch in Fortran mehr oder weniger strukturiert programmieren oder in C objekt-orientiert, jedoch nur eingeschränkt, und die Sprache behindert dabei mehr als daß sie hilft. Eine Sprache *unterstützt* ein Paradigma genau dann, wenn sie dessen Verwendung leicht macht, zum Beispiel durch spezielle Sprachelemente wie `virtual` in C++. Abbildung 1.1 bietet einen Überlick über die Entwicklung einiger Sprachen.

- Imperative Programmierung.
  Ein Programm wird als eine Folge von Anweisungen aufgefaßt. Das Paradigma der imperativen Programmierung liegt den meisten heute verbreiteten Programmiersprachen direkt oder indirekt zugrunde, jedoch sind rein imperative Sprachen, wie beispielsweise Assembler und einige primitive BASIC-Dialekte (z.B. Commodore-BASIC und GW-BASIC), sehr unübersichtlich. Der Ansatz der imperativen Programmierung geht letztlich direkt auf die von Neumann-Architektur von Universalrechnern von 1946 zurück.

- Prozedurale Programmierung.
  Eine naheliegende Erweiterung des imperativen Paradigmas. Ein Programm wird wieder als eine Folge von Anweisungen aufgefaßt, jedoch können eine oder mehrere Anweisungen zu einer Prozedur, quasi einer neuen, aber komplexeren Anweisung kombiniert werden; das gleiche gilt für Funktionen. Dazu gehört auch das Konzept von lokalen Variablen. Die Gliederung von Programmen in Prozeduren stammt aus den frühen 50er Jahren. Eine typische rein prozedurale Programmiersprache ist Fortran 77.

- Funktionale Programmierung.
  Ein Programm wird als eine Funktion angesehen, die eine Menge von Eingabewerten in eine Menge von Ausgabewerten abbildet. Diese Funktion ist selbst definiert durch die Zusammensetzung anderer, einfacherer Funktionen, analog zur prozeduralen Programmierung. Dieser Ansatz wurde in den 50er Jahren entwickelt und ist zum Beispiel in LISP umgesetzt. Außerhalb von Informatiker-Kreisen ist diese Methodik nicht besonders weit verbreitet.

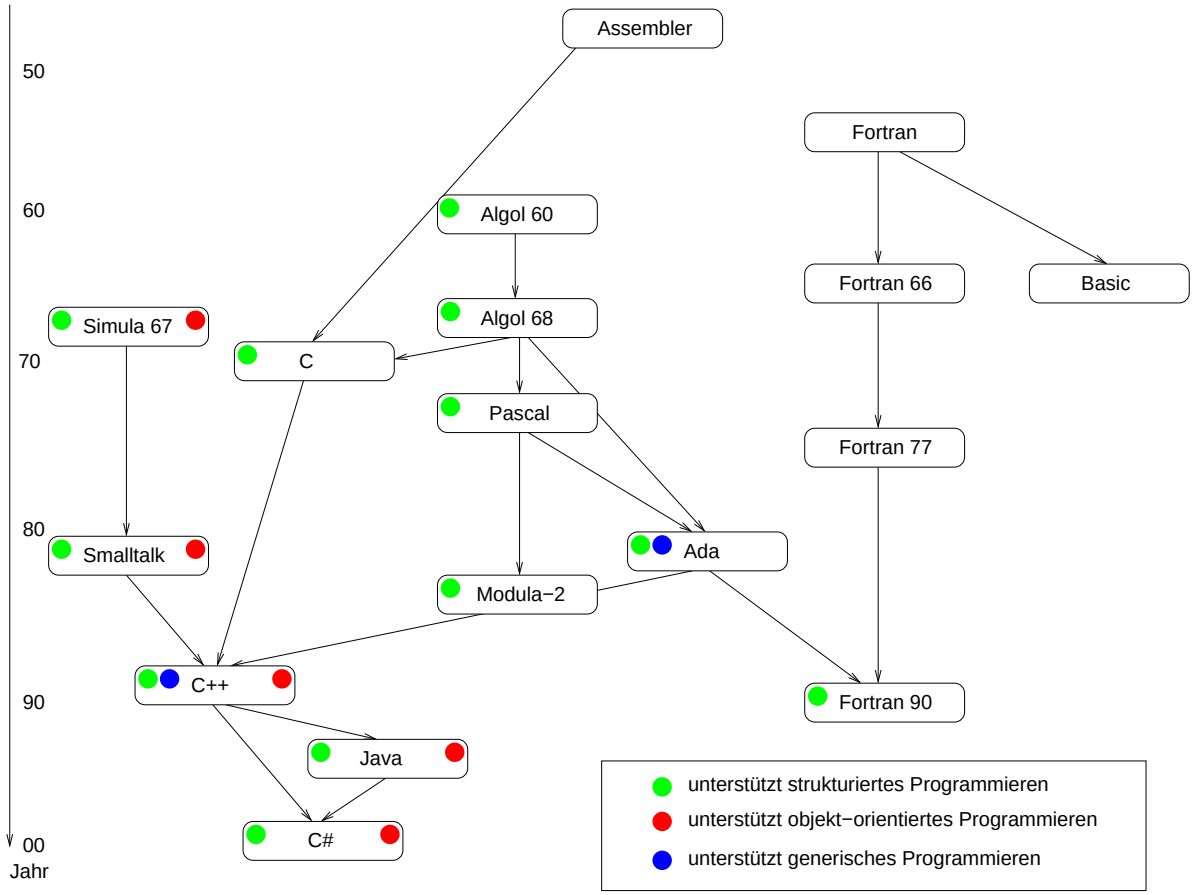

Abbildung 1.1: Zeitliche Entwicklung einiger Programmiersprachen (Überblick). Es ist deutlich zu sehen, wie sehr sich Sprachen gegenseitig beeinflussen.

- Strukturierte Programmierung.
  Der Grundgedanke hier ist, daß die Struktur des Programmes die Struktur des Problems widerspiegeln soll. Zugrunde liegt dabei die prozedurale Programmierung (in der Anweisungen strukturiert werden), jedoch wird sie um Möglichkeiten zur Strukturierung der *Daten* erweitert. 1960 entstand mit Algol die erste strukturierte Programmiersprache; weitere typische Vertreter sind Pascal und C. In Fortran hat die Möglichkeit, Daten zu strukturieren, mit dreißigjähriger Verspätung erst 1990 Einzug erhalten.

- Prädikative Programmierung
  sei hier nur am Rande erwähnt. Hier besteht das Programm aus einer Sammlung von Fakten, also gültigen Prädikaten, sowie Fragen, die das Programm mit Hilfe der Prädikaten beantwortet. Dabei kommt dem Laufzeitsystem der Sprache und besonders dessen effizienter Implementierung eine entscheidende Bedeutung zu.

- Objekt-orientierte Programmierung.
  Hier besteht ein Programm aus Objekten, die miteinander kommunizieren. Ein Objekt ist ein Verbund aus Daten, die den Zustand des Objekts definieren, und Methoden,

die auf diesen Zustand wirken. Einem Objekt eine Nachricht zu schicken bedeutet, eine seiner Methoden aufzurufen, woraufhin das Objekt mit einer Zustandsänderung reagiert. Ende der 60er kam mit Simula die Idee des objekt-orientierten Paradigmas auf; heute unterstützen die meisten Sprachen das OOP, wobei C++ und Java die verbreitetsten sind.

- Generische Programmierung
  ist kein alleinstehendes Merkmal einer Programmiersprache. Es handelt sich hierbei um die Möglichkeit, Algorithmen oder Objekt-Klassen typunabhängig zu formulieren – beispielsweise einen Sortieralgorithmus, der für alle Datentypen funktioniert, die mittels $<$ und $=$ vergleichbar sind[6], oder Listen. Dieses Konzept kam Ende der 80er auf und wird von C++ und insbesondere von Ada unterstützt.

- Aspekt-orientierte Programmierung.
  Hier wird ein Programm nicht entlang von Prozeduren oder Objekten aufgeteilt, sondern anhand von Aspekten, wie zum Beispiel Persistenz (Abspeicherbarkeit) oder Benutzerinteraktion. Dies ist eine aktuelle Entwicklung der Informatik und wird bisher noch von keiner verbreiteten Programmiersprache aktiv unterstützt; man behilft sich mit sogenannten "code weavers", die die verschiedenen, getrennt programmierten Aspekte zu Objekt-Klassen zusammenfügen.

All diese Entwicklungen fanden und finden vor dem Hintergrund immer leistungsfähigerer Computer und immer größerer Speicher statt. Die Erkenntnis, daß man zwar ein 500 Zeilen-Programm zur Not auch ohne jegliche Strukturierung funktionsfähig bekommt, nicht jedoch ein 10000 Zeilen-Programm, zwang die Programmierer, ihre Methoden und Werkzeuge weiterzuentwickeln. Mit der Verbreitung von Bildschirmarbeitsplätzen (und dem Verschwinden der Lochkarten) entstanden visuelle Editoren; Betriebssysteme wie Unix und VMS ermöglichten interaktive Programmentwicklung und damit wesentlich effizientere Fehlersuche. Ende der 80er entstanden integrierte Entwicklungsumgebungen, später CASE-Werkzeuge[7]. Durch die zunehmende Größe der Softwareprojekte verschob sich der Fokus bei der Softwareentwicklung allmählich vom Programmieren zum Programm*design*. Aktuelle Entwicklungen wie die Unified Modelling Language (UML), die zur graphischen Beschreibung des Designs dient, und entsprechenden graphischen UML-Editoren spiegeln dies wieder.

Heutige Großprojekte mit mehreren 10 Millionen Zeilen Code verwenden in der Regel mehrere verschiedene Paradigmen: strukturierte Programmierung, Modularität, objektorientierte Analyse und Entwurf, sowie oft auch generische Programmierung. Sprachen wie C++ unterstützen den Entwickler dadurch, daß sie mehr als nur ein Paradigma anbieten [1] [2].

[6] Die Einschränkung auf Datentypen mit Eigenschaften wie Vergleichbarkeit wird als Generizität bezeichnet. In Ada können diese Eigenschaften explizit angegeben werden, in C++ nicht (unbeschränkte Generizität).

[7] Computer-Aided Software Engineering, computer-unterstützte Softwareentwicklung.

Kapitel 2

Das CDF II Experiment

Der Collider Detector at Fermilab (CDF) ist ein Mehrzweck-Detektor, der am Strahl des Tevatron betrieben wird. Der Tevatron-Beschleuniger ist mit 1.96 TeV derzeit der leistungsstärkste Beschleuniger der Welt. Das CDF-Experiment ist Teil des Fermi National Accelerator Laboratory (FNAL) und liegt in der Nähe von Batavia, Illinois (USA), rund 60 km westlich von Chicago. Die CDF-Kollaboration hat über 500 Mitglieder aus 50 Instituten in 10 Ländern; das Institut für Experimentelle Kernphysik (EKP) der Universität Karlsruhe ist das einzige deutsche Institut.

2.1 Das Tevatron

Das Tevatron ist ein Proton-Antiproton-Kollider mit einem Umfang von ca. 6 km. Protonen und Antiprotonen werden aus dem Main Ring zugeführt, wo sie auf 120 GeV vorbeschleunigt werden. Protonen werden aus dem Booster, einem 8 GeV Synchrotron, in den Main Ring geschossen. Der Booster wiederum erhält seine Protonen aus einem 150 m langen Linearbeschleuniger mit einer Endenergie von 400 MeV, der selbst von einem 750 keV Cockroft-Walton-Beschleuniger gefüllt wird. Antiprotonen werden erzeugt, indem Protonen aus dem Main Ring abgezweigt und auf ein Wolfram-Ziel geschossen werden. Die Antiprotonen aus den dabei entstehenden $p\bar{p}$-Paaren werden im Debuncher gesammelt, wo der Strahl stochastisch gekühlt wird. Beim Einschießen in den Main Ring ist der Antiprotonenstrahl monochromatisch.

Im Beschleunigerlauf I wurde der Beschleuniger mit einer Schwerpunktsenergie von $\sqrt{s} =$ 1.8 TeV betrieben. Das Tevatron ist derzeit der einzige Beschleuniger, mit dem Top-Quark-Paare produziert werden können, und wird dies bis zum Start des Large Hadron Colliders (LHC) voraussichtlich im Jahre 2007 auch bleiben. Für den derzeit laufenden Run II wurde die Schwerpunktsenergie auf 1.96 TeV erhöht, wodurch der theoretische Wirkungsquerschnitt für die Produktion von $t\bar{t}$-Paaren, $\sigma_{t\bar{t}}$, von 4,8 pb auf 6,8 pb erhöht wurde [3]. Außerdem wurde die Luminosität erhöht. Die angestrebte integrierte Luminosität von 0.5 fb^{-1} pro Jahr konnte bislang jedoch aufgrund technischer Schwierigkeiten im Betrieb des Beschleunigers noch nicht erreicht werden. Bei der Zielluminosität ist mit im Mittel drei Wechselwirkungen pro Bunchcrossing zu rechnen. Abbildung 2.1 zeigt den schematischen Aufbau des Tevatrons.

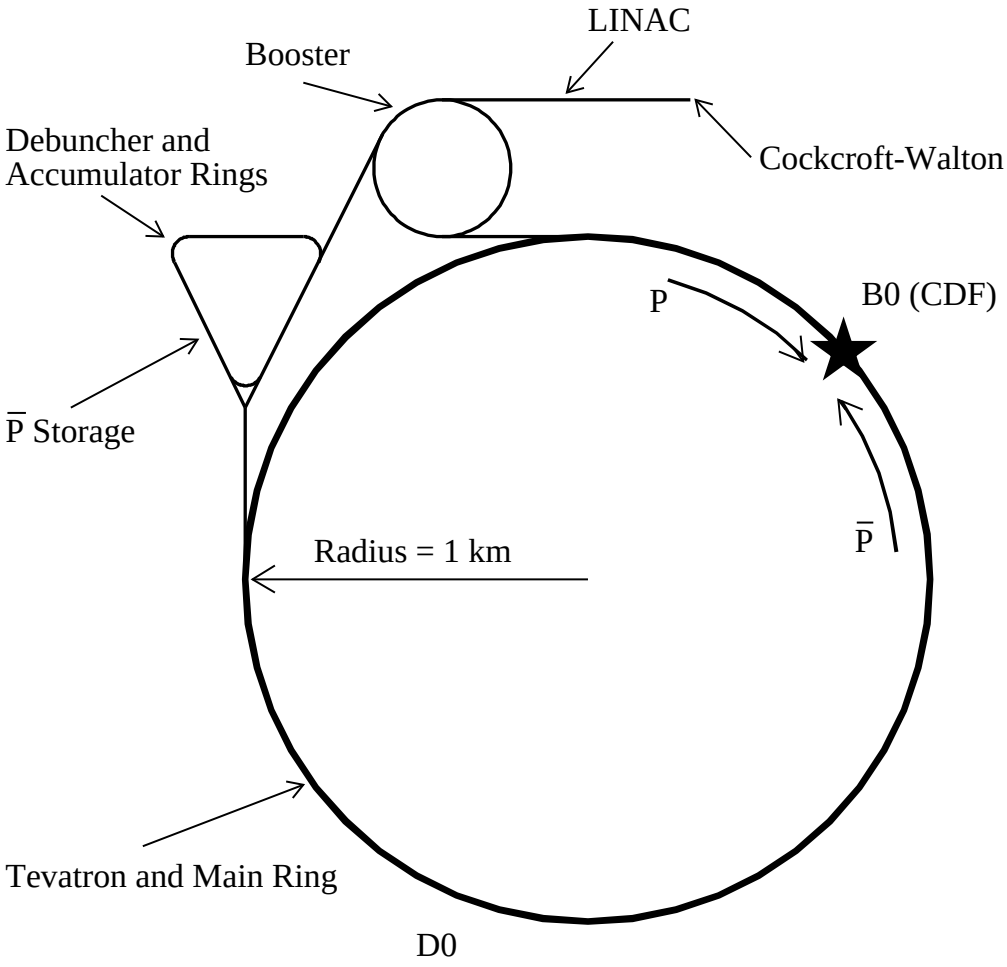

Abbildung 2.1: Schematischer Aufbau des Tevatron-Colliders. Der CDF-Detektor befindet sich am Wechselwirkungspunkt B0.

2.2 Der Collider Detector at Fermilab

Der CDF-Detektor hat in Run 0 und I eine integrierte Luminosität von 130 pb^{-1} aufgezeichnet. Unter den Entdeckungen des CDF-Experiments sind die des Top-Quarks 1994/95 (gemeinsam mit der D0-Kollaboration) [4] und des B_c im Jahre 1998 [5]. Desweiteren war die CDF-Publikation [6] die damals genaueste Messung des CP-Verletzungsparameters $\sin 2\beta$ aus dem Zerfall von $B \to J/\Psi\ K_s^0$.

2.3 Das CDF II Detektor-Upgrade

Der CDF-Detektor wurde in den letzten Jahren grundlegend überarbeitet und erweitert, um die erhöhte Luminosität von Run II besser nutzen zu können [3]. Alle Spurdetektoren wurden durch neuere Systeme ersetzt; Lücken im Myon-System wurden geschlossen; die gesamte Elektronik zur Datennahme wurde ersetzt. Außerdem wurde die Raumwinkel-Abdeckung

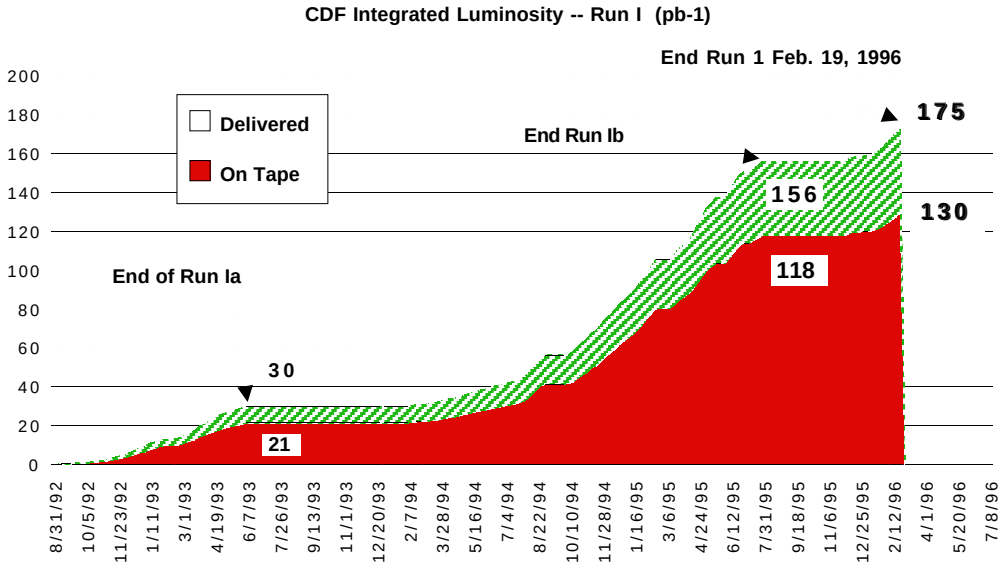

Abbildung 2.2: CDF integrierte Luminosität während Run I. Für Run II wird eine integrierte Luminosität von $2fb^{-1}$ erwartet.

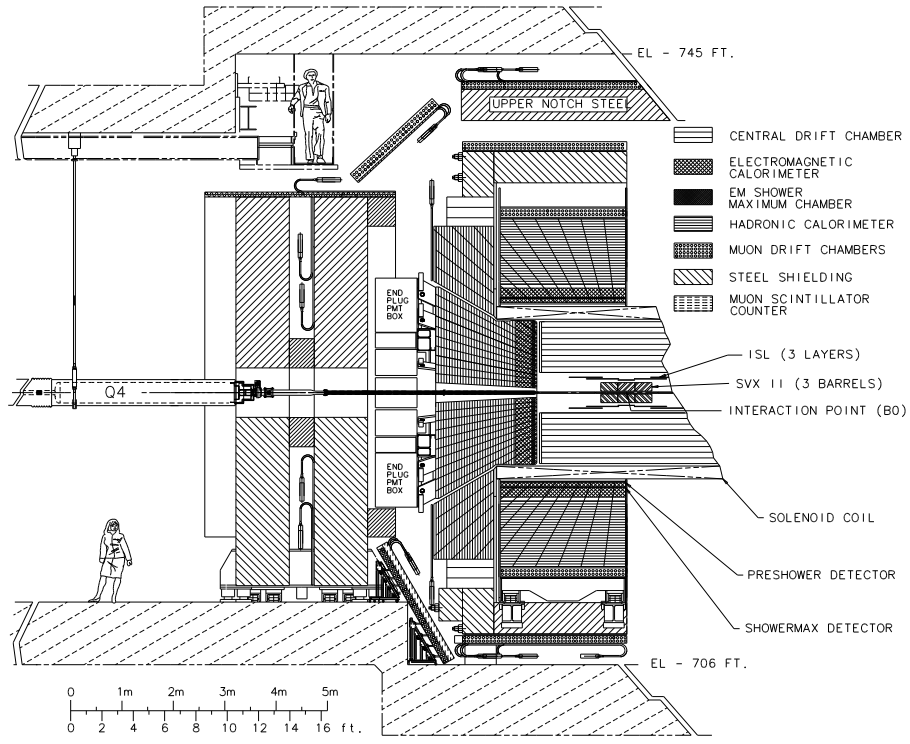

Abbildung 2.3: Der modernisierte CDF II-Detektor. Die Silizium-Detektoren im Zentrum, der ISL und der SVX II, sind 195 cm lang und haben einen Durchmesser von 60 cm.

des vorwärtsgerichteten Myon-Systems erweitert. Abbildung 2.3 zeigt den überarbeiteten Detektor. Das Silizium-Subsystem wurde stark verbessert: der frühere SVX' (ein 80 cm langer, vierlagiger Detektor mit einem Durchmesser von 20 cm) wurde durch drei neue Detektoren ersetzt: der Intermediate Silicon Layer (ISL), der Silicon Vertex Detector II (SVX II) und die Lage 00 (Layer 00).

ISL

Der *Intermediate Silicon Layer* – die Silizium-Lagen 5 und 6 – besteht aus doppelseitig ausgelesenen Streifendetektoren-Chips, die die $r\varphi$- und die rz-Koordinaten von Teilchenspuren messen. Die Streifen auf der einen Seite des Wafers sind parallel zur z-Achse, während die der anderen Seite um den Winkel von 1.2° gedreht sind. Dies ermöglicht die Messung der z-Koordinate. Die Länge des ISL beträgt 195 cm, die Radien der beiden Lagen sind 20 cm und 30 cm.

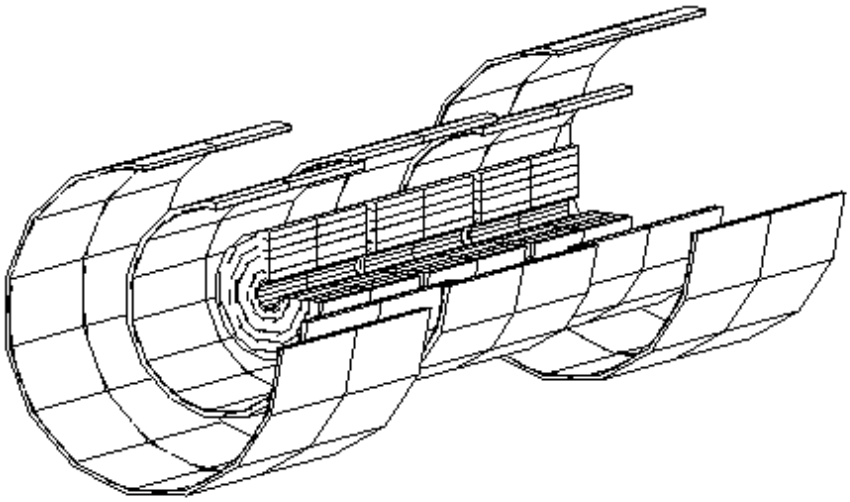

Abbildung 2.4: Der ISL und der SVX II. Der Radius der innersten Lage beträgt lediglich 2.54 cm, der Radius der äußersten Lage 30 cm.

SVX II

Der SVX II besteht aus fünf doppelseitig ausgelesenen Lagen Silizium, die in jeweils zwölf Winkelsegmente unterteilt sind. Die Gesamtlänge von 87 cm setzt sich aus drei Teilstücken von je 29 cm zusammen. Auf den Lagen 0, 1 und 3 werden die φ- und die z-Koordinate gemessen; auf der einen Seite der Wafer sind die Streifen parallel zur z-Achse, auf der anderen Seite senkrecht zu z. Die Streifen der Lagen 2 und 4 sind auf der einen Seite ebenfalls parallel

zur z-Achse und messen die φ-Koordinate, auf der anderen Seite wie beim ISL um 1.2° gedreht. Abbildung 2.4 zeigt SVX II und ISL.

Lage 00

Nachträglich wurde noch eine weitere Lage Silizium eingebaut, die Lage 00, eine einseitig ausgelesene Lage, die in zwölf Segmenten direkt auf das Beryllium-Strahlrohr geklebt wurde, wie Abbildung 2.5 zeigt [7]. Der mittlere Radius der Lage 00 beträgt 1,6 cm, die sensitive Länge ca. 80 cm. Durch die Lage 00 wird die Auflösung des *Impact Parameters* beträchtlich verbessert, was an eben diesem äußerst geringen mittleren Radius liegt. Auch verschlechtert die Lage 00 die Messungen nicht durch Mehrfachstreuung, da sie im zentralen Bereich aus einer dünnen Silizium-Schicht besteht, während die Ausleseelektronik weiter außen bei $|z| > 45$ cm liegt. Abbildung 2.6 verdeutlicht die Auswirkung von Lage 00 auf die Auflösung.

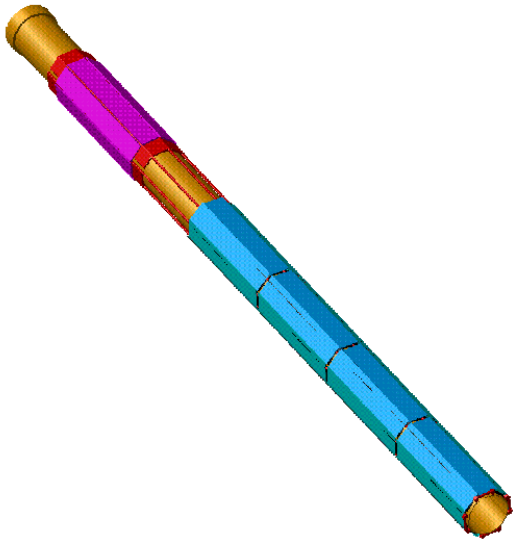

Abbildung 2.5: Eine Hälfte der Lage 00. Die vier Teilstücke sind zusammen 40 cm lang. Die Ausleseelektronik befindet sich am äußeren Rand.

2.4 Physikalische Ziele

Das Physik-Programm von CDF für Run II ist sehr umfangreich und vielfältig. Mit einer integrierten Luminosität von $2fb^{-1}$ wird CDF II genügend Statistik haben, um zum Beispiel Präzisionsmessungen von α_s und m_W durchzuführen. Die fünf physikalischen Hauptziele des CDF-Experiments für Run II sind:

- **B-Physik**

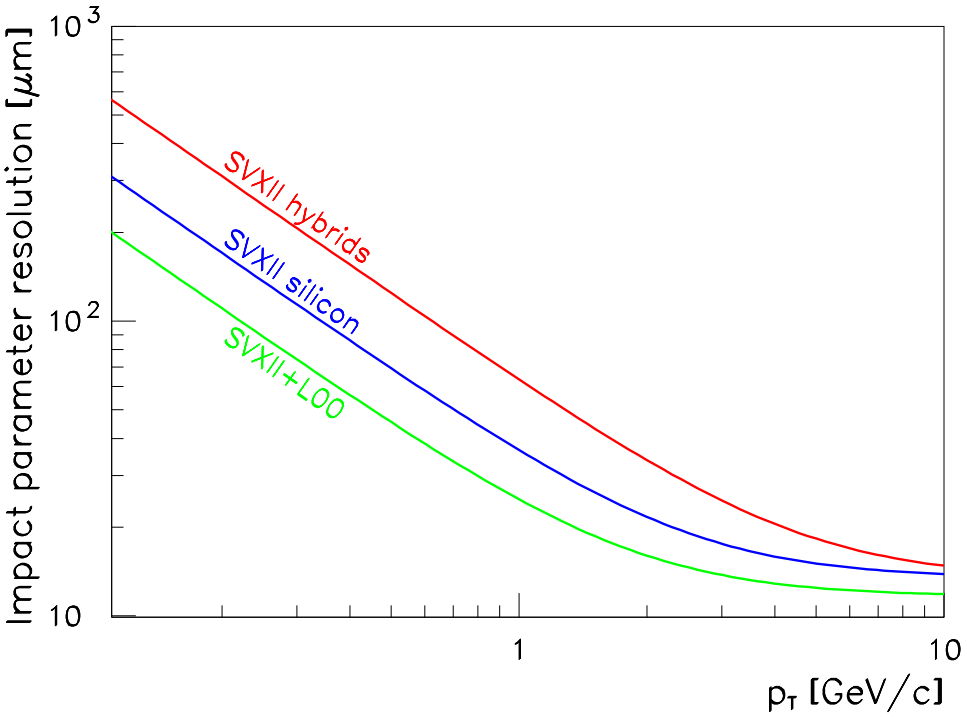

Abbildung 2.6: Impact Parameter-Auflösung mit und ohne Lage 00. Die SVX II-Hybride – die Ausleseelektronik – verringern die Auflösung durch von ihnen verursachte Mehrfachstreuung.

wird in Run II mit hoher Statistik möglich sein. Bei einer integrierten Luminosität von $2fb^{-1}$ wird mit folgenden Ereigniszahlen gerechnet:

- $B^0 \to J/\Psi K_s$ – 15.000 Ereignisse
- $B^0 \to \pi^+\pi^-$ – 10.000 Ereignisse
- $B_s \to J/\Psi\Phi$ – 9.000 Ereignisse

B-Fabriken erzielen zwar in den ersten beiden Kanälen höhere Ereigniszahlen, sind jedoch nicht in der Lage, B_s-Mesonen zu erzeugen. Die Untersuchung dieser und anderer Zerfälle ermöglicht die Messung von CP-Verletzung (in Zerfällen $B^0 \to J/\Psi K_s$ und $B^0 \to \pi^+\pi^-$, und auch in $B_s \to J/\Psi\Phi$) und auch Konsistenzprüfungen der CKM-Matrix. Durch Vermessen der Mischung von $B_s^0 - \bar{B}_s^0$ kann das Verhältnis der Matrixelemente $|V_{td}/V_{ts}|$ bestimmt werden. Auch Lebensdauermessungen, beispielsweise von B^0- oder B^+-Mesonen, werden von der erhöhten Statistik gegenüber Run I profitieren.

- **Top-Physik**
  profitiert genau wie die B-Physik von der höheren Statistik in Run II. Geschätzte Ereignisraten für $2fb^{-1}$ sind:

  - rein leptonisch – 150 Ereignisse
  - W + 3 Jets (davon einer mit b-Tag) – 900 Ereignisse

- W + 4 Jets (davon einer mit b-Tag) – 725 Ereignisse
- W + 4 Jets (davon zwei mit b-Tag) – 180 Ereignisse

Bei so vielen Ereignissen mit mindestens einem b-Tag wird es möglich sein, eine wesentlich genauere Untersuchung der Top-Eigenschaften vorzunehmen. So wird der Fehler in der Top-Masse auf unter 4 GeV sinken, der des totalen Wirkungsquerschnittes auf 9 %. Elektroschwache Produktion einzelner Top-Quarks wird ebenfalls meßbar.

- **QCD bei großem Impulsübertrag**
  Da das Tevatron eine Hadron-Maschine ist, wird der Impulsübertrag Q^2 den gesamten Bereich von $(10\ \text{GeV})^2$ bis $(500\ \text{GeV})^2$ abdecken, sodaß beispielsweise das Laufen der starken Kopplungskonstante α_s über die gesamte Skala untersucht werden kann; Abweichungen von der Standardmodell-Vorhersage deuten auf Loop-Beiträge neuer Teilchen hin. Auch QCD-Vorhersagen zu Jet-Produktion und -Fragmentation können in NLO geprüft werden, ebenso wie die Produktion von W und Z.

- **Elektroschwache Präzisionsmessungen**
  Der verbesserte Detektor (insbesondere der größere Pseudorapiditätsbereich) führt zu einer erheblichen Verbesserung der Statistik im Bereich der für CDF so wichtigen leptonischen Zerfälle der Vektorbosonen:

  - $W \to l\nu(e, \mu)$ – 4.300.000 Ereignisse
  - $Z \to l^+l^-(e, \mu)$ – 600.000 Ereignisse
  - $W\gamma, W \to e\nu$ – 4.000 Ereignisse
  - $Z\gamma, Z \to e^+e^-$ – 1.800 Ereignisse

  Auch seltenere Prozesse wie Produktion von zwei Vektorbosonen sind nachweisbar: $W^+W^- \to l\nu l\nu$ (200 Ereignisse) und $W^+Z^0 \to l^+\nu l^+l^-$ (50 Ereignisse). Ein Ziel ist die Messung der W-Masse mit weniger als 40 MeV Fehler, was zusammen mit der Top-Masse Rückschlüsse auf die Higgs-Masse im Standardmodell ermöglicht.

- **Suche nach neuen Phänomenen**
  An vorderster Stelle bei der Suche nach neuen Phänomenen steht natürlich die Suche nach dem Higgs-Boson, ob nun innerhalb des Standardmodells oder in einer supersymmetrischen Erweiterung. Wie Abbildung 2.7 zeigt, besteht eine (wenngleich geringe) Chance, ein leichtes Higgs (unter 120 GeV) in einer kombinierten Analyse von CDF und D0 zumindest mit 95% Vertrauensniveau zu messen. Für eine Entdeckung (mind. 5σ CL) wären in jedem Fall mehr als $2fb^{-1}$ nötig. Bis zu einer Skala von 5 TeV wäre auch eine Quark-Compositeness nachweisbar: es wird an Composite-Modellen wie Technicolor und, davon abgeleitet, Topcolor geforscht.

Eine ausführliche Darstellung der physikalischen Ziele in Run II findet sich in [3].

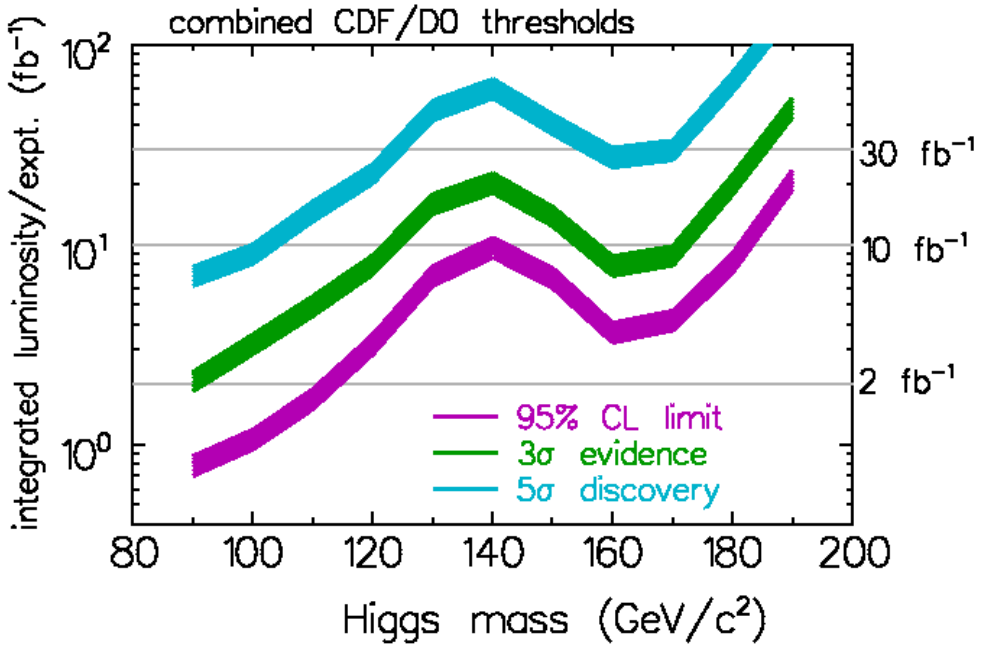

Abbildung 2.7: Nachweiswahrscheinlichkeit geladener Higgs-Bosonen in Run II (kombiniertes Limit von CDF und D0).

2.5 Das CDF II Software-Upgrade

Die vielfältigen Veränderungen am Detektor machten es notwendig, die Software, die die Ereignisse rekonstruiert, ebenfalls zu überarbeiten. Die Kollaboration entschied sich dafür, die Rekonstruktionssoftware objekt-orientiert in C++ neu zu schreiben. Dies stellt einen radikalen Bruch dar, da bisher stets das prozedurale Fortran verwendet wurde.

Die Entwicklung der neuen Software erwies sich als überraschend schwierig, und auch heute noch, Monate nach dem Start von Run II, ist die Entwicklung noch nicht überall abgeschlossen. Einige Gründe, warum die Entwicklung so schwierig war bzw. ist:

- **Erfahrung mit der objekt-orientierten Methodologie.**
  Anders als prozedurales Programmieren ist es bei objekt-orientierter Programmierung zwingend notwendig, sich *vorher* über die Struktur des Programms klarzuwerden. Dies war den Entwicklern sehr deutlich bewußt, und sie haben das Projekt *extrem* allgemein und *sehr* flexibel entworfen. Die Warnung "Abstraction is *bad*. Abstraction is selective ignorance." [8] wird dabei nicht immer beachtet.

- **Vereinte Tracking- *und* C++-Erfahrung.**
  Die Kollaborationsmitglieder mit Erfahrung bei der Entwicklung von Rekonstruktionssoftware wurden durch den rigorosen Einsatz von objekt-orientierten Methoden abgeschreckt, ihre Erfahrungen weiterzugeben. Umgekehrt haben die wenigen Entwickler mit umfassender Erfahrung im Bereich der objekt-orientierten Entwicklung oft keine oder nur sehr begrenzte Erfahrung mit Rekonstruktionsalgorithmen.

- **Wahl der Werkzeuge.**
  Aufgrund des oben erwähnten "Über-Designs" war in den ersten Jahren der Entwicklung nur der C++-Compiler von Kuck & Associated, Inc. (KAI) in der Lage, die Programme zu übersetzen. Der KAI-Compiler ist jedoch teuer und langsam, die gesamte Entwicklungsumgebung sehr komplex und pflegeintensiv. Dadurch gibt es nur wenige Entwicklungsmaschinen, auf denen sich die Programmierer drängen. Der Entwicklungszyklus ist sehr langsam, eine winzige Änderung einzubauen dauert mehrere Minuten; die Fehlersuche mit Debugger ist ebenfalls sehr aufwendig und nur eingeschränkt möglich.

- **"Glaube" an objekt-orientierte Techniken.**
  Der Enthusiasmus von Entwicklern, die an ihrem ersten größeren objekt-orientierten Projekt arbeiten, ist erfahrungsgemäß überaus groß. Sie setzen oft einen Glauben in objekt-orientierte Techniken, die selbst (oder gerade) langgediente Experten diesen Techniken nicht zubilligen. Informatikern ist längst klar, daß objekt-orientierte Methoden kein Allheilmittel sind, sondern daß nur eine Vielzahl von Techniken ein optimales Ergebnis bringen kann [1] [2]. So ist es meist nicht sinnvoll, Algorithmen objekt-orientiert umzusetzen.

Bei fortschrittlichen Techniken der Softwareentwicklung besteht offenbar die Gefahr, daß sich der Fokus der Entwickler von den eigentlichen Algorithmen weg und hin zu technischen Details verschiebt. Die Thematik der Softwareentwicklung bei CDF wird ausführlich in Kapitel 3 besprochen.

Kapitel 3

Die CDF II Analysekette

Der Weg, den Daten vom Detektor bis hin zur fertigen Analyse nehmen, läßt sich in zwei Bereiche untergliedern:

1. *Online*
2. *Offline*

Unter dem Begriff *Online* faßt man alle Vorgänge zusammen, die unmittelbar während der Datennahme geschehen; ein Stocken dieses Teils der Analysekette führt zu Datenverlust. *Online* bedeutet hier also, auf noch nicht permanent abgespeicherten Daten zu operieren. *Online*-Vorgänge umfassen unter anderem:

- Auslesen der Detektoren
- Level 3-Trigger
- Erstellen der Ereignis-Datensätze
- Speichern der Ereignisse in verschiedenen Datenströmen
- Überwachung des Detektors

Der Begriff *Offline* hingegen zielt darauf ab, daß die entsprechenden Teile der Analysekette nicht (nötigenfalls) unmittelbar bei Datennahme durchlaufen werden müssen, sondern unabhängig vom Vorgang des Messens ablaufen. *Offline*-Software operiert also auf Daten von Festplatte oder Bandlaufwerk. Zu den *Offline*-Aktivitäten zählen:

- Hit-Clustering[1]
- Spurrekonstruktion
- Vertexfits

[1]Eine Besonderheit des CDF-Level 3-Triggers ist, daß auch schon dort Hits und Spuren rekonstruiert werden; jedoch nicht mit allen verfügbaren Algorithmen.

- Verbinden von Spuren und Kalorimeter-Signalen
- Teilchenidentifikation
- Jet-Clustering
- die eigentliche Analyse

Im folgenden werden die beiden großen Teilgebiete separat vorgestellt und auf einige Details eingegangen, mit denen sich der Autor besonders befaßt hat.

3.1 Online

Abbildung 3.1 gibt einen groben Überblick über den Datenfluß im Online-System von CDF II. Der Datenfluß beginnt mit der Auslese der *Front End Crates*, also der Detektor-Hardware, und endet damit, daß die noch nicht rekonstruierten Daten in Form von nach Triggern vorsortierten Datensätzen auf Band (und teilweise auf Festplatte) vorliegen, bereit für die Rekonstruktionssoftware.

Detektorauslese

Der CDF II-Detektor wird über Standard-VME-Bus ausgelesen. Dabei übertragen die *Front End Crates* die Daten zu den VME-Auslesepuffern (*VME Readout Buffers*, VRBs), die wiederum von Scanner-CPUs (SCPUs) ausgelesen werden. Die SCPUs wiederum schicken die Daten über 155 Mbit/s ATM-Leitungen über den Event Builder ATM-Switch an die Level-3 PC-Farm (siehe Abbildung 3.1). Dort stehen die Ereignisse erstmals zusammenhängend zur Verfügung.

Trigger-Stufe 3

Die Trigger-Stufe 3 ist ein reiner Software-Trigger. Die Ereignisse mit einer mittleren Größe von 150-250 kB kommen mit einer Rate von 300-1000 Hz vom *Event Builder Switch* und werden auf 200 PCs verteilt (Level-3 PC farm). Dort wird entschieden, ob ein Ereignis weiterverarbeitet werden soll. Ist dies der Fall, so wird das Ereignis in das ROOT-Datenformat umformatiert [9], mit Informationen über den oder die entscheidenden Trigger ausgestattet und an den Consumer-Server/Logger weitergereicht. Dabei werden Raten von bis zu 75 Hz erreicht, was bedeutet, daß für jedes Ereignis maximal 2.6 s CPU-Zeit zur Verfügung stehen. Dabei werden folgende Algorithmen eingesetzt:

- Jet- und fehlende Transversalenergie-Trigger.
  Ausgehend von den Kalorimetern, werden die Energien von Einfach-, Doppel- und Mehrfachjets bestimmt; der Schnitt erfolgt auf diese Energien, sodaß Ereignisse mit besonders hochenergetischen Teilchenjets ausgewählt werden können. Ergänzend dazu wird eine Bilanz des Transversalenergie[2] aufgestellt; geht die Bilanz nicht auf, löst der Missing E_T-Trigger aus.

[2] eigentlich: des Transversalimpulses, aber da $v = c = 1$ ist das dasselbe.

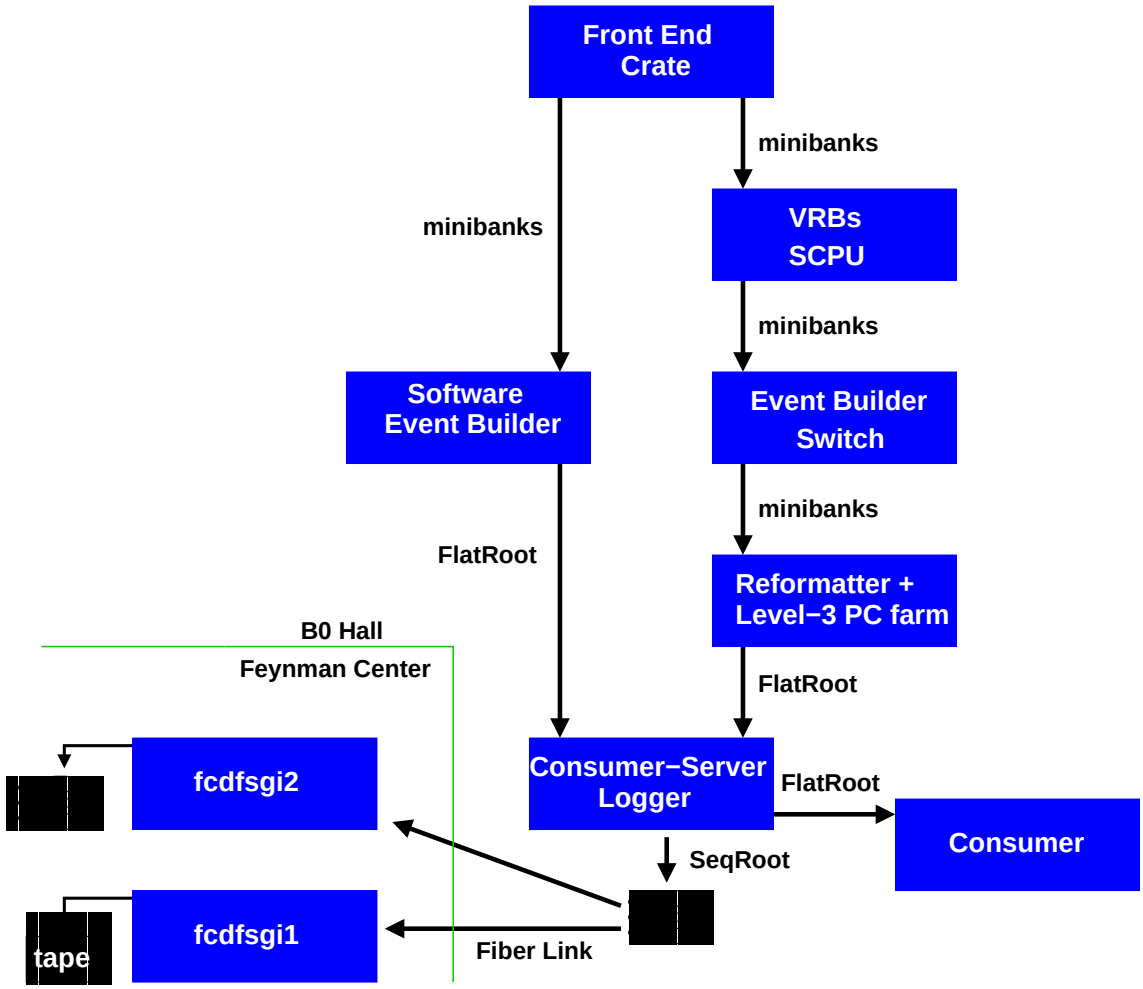

Abbildung 3.1: Online-Kette bei CDF II. Der linke Pfad von den *Front End Crates* zum Consumer-Server Logger dient zu Testzwecken.

- Trigger auf inklusive COT-Spuren.
  Hier wird auf die Pseudorapidität $|\eta|$ und auf den Transversalimpuls von Driftkammerspuren geschnitten. Dieser Trigger kann als Vertretung für den zentralen Lepton-Trigger fungieren. Dann muß eine entsprechende isolierte COT-Spur vorliegen.

- Trigger auf inklusive zentrale Elektronen.
  Ausgehend vom elektromagnetischem Kalorimeter werden passende COT-Spuren gesucht. Diese Elektron-Kandidaten werden dann nach ihrer Pseudorapidität sowie Transversalenergie und -impuls ausgewählt.

- Trigger auf inklusive zentrale Myonen.
  Analog zum zentralen Elektron-Trigger werden hier statt des EM-Kalorimeters die Myonkammern genutzt.

Die Trigger-Stufen 1 und 2 sind reine Hardware-Trigger und damit nicht Thema dieser Arbeit. Sie werden beispielsweise in [10] beschrieben.

3.1.1 Consumer-Server/Logger

Immer, wenn ein Knoten der Level-3 Farm ein Ereignis für weiterverarbeitungswert befunden hat, kontaktiert er den Consumer-Server/Logger. Dies ist eine Vierprozessor-Maschine von SGI, die mit einer besonders starken Netzwerkanbindung ausgestattet ist [11]. Dies ist erforderlich, da die Maschine als eine Art Daten-Switch eingesetzt wird: die eingehenden Daten kommen mit einer mittleren Datenrate von 20 MB/s; ebenso schnell müssen die Daten in das Rechenzentrum des Fermilabs, das Feynman Computing Center (FCC), übertragen werden. Hinzu kommen noch die zu bedienenden Consumer mit bis zu 10 MB/s (siehe Abbildung 3.2). Insgesamt benötigt der Consumer-Server/Logger also 50 MB/s Außenanbindung.

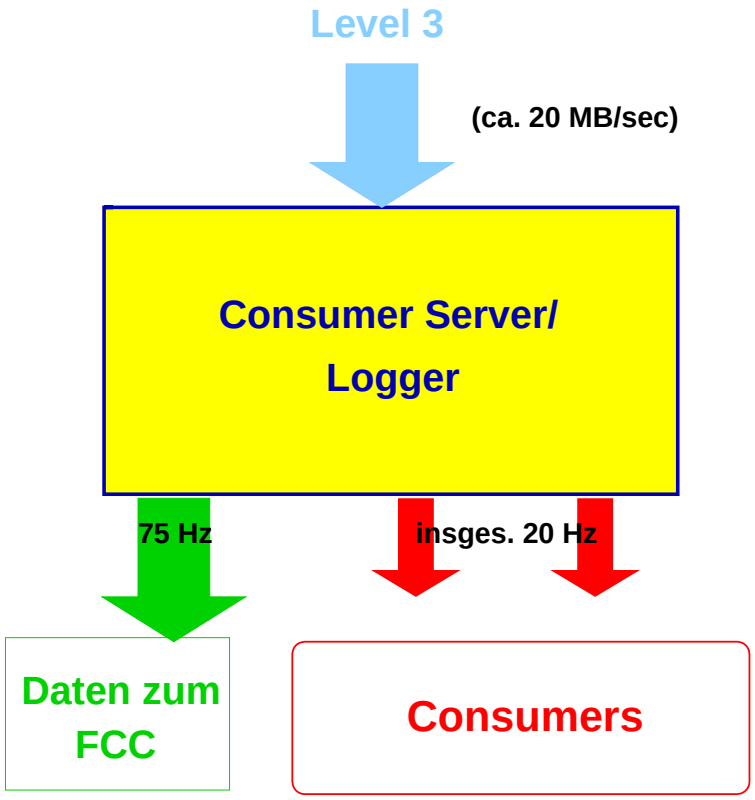

Abbildung 3.2: Grober Überblick über den Consumer Server/Logger.

Um diese Bandbreite zu realisieren, werden verschiedene Hardware-Typen eingesetzt. Die Verbindung zur Level-3 Farm geht über vier Fast Ethernet-Leitungen, die eine Datenrate von 40 MB/s ermöglichen. Die Anbindung an das eineinhalb Kilometer entfernte Rechenzentrum erfolgt über FibreChannel[3] mit 50-70 MB/s. Für die Consumer stehen zwei weitere

[3]Diese auf Glasfasern basierende Übertragungstechnologie (theoretisch ist auch Kupfer möglich) ermöglicht unter anderem, Festplatten an mehrere Rechner gleichzeitig anzuschließen. FibreChannel wird

Fast Ethernet-Anschlüsse mit bis zu 20 MB/s zur Verfügung. Insgesamt sind also noch Leistungsreserven für kurzzeitige Höchstlast vorhanden; vor allem aber hat die hardwaremäßige Unterteilung der Bandbreite den Vorteil, daß sich die drei Verbindungspartner – Level-3 Farm, Rechenzentrum und Consumer – nicht gegenseitig behindern können.

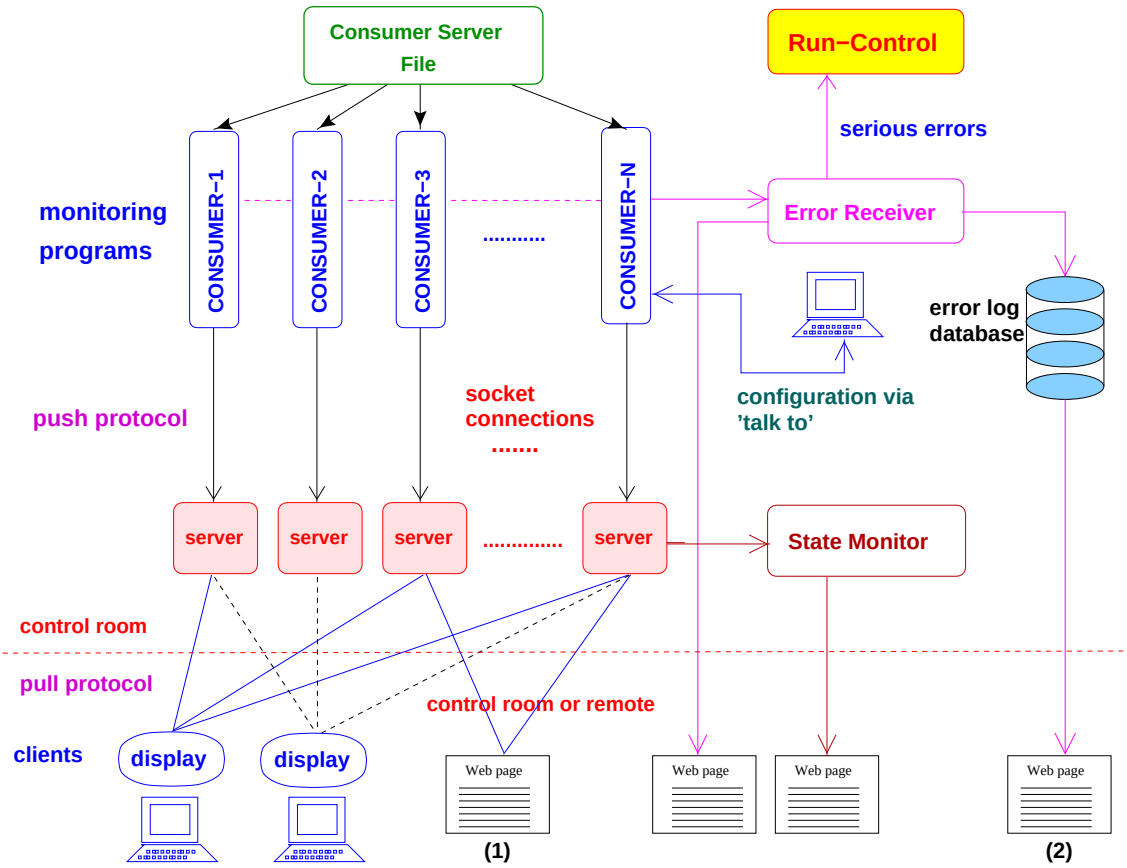

Abbildung 3.3: Struktur des Consumer Server/Loggers. (1) bezeichnet den Consumer Histogram Browser, (2) den Consumer Database Browser. Beide werden in diesem Kapitel vorgestellt.

Logger

Der Logger ist dafür verantwortlich, daß die Daten ordnungsgemäß auf Platte geschrieben werden. Dazu werden die Ereignisse den mit den jeweiligen Triggern korrespondierenden Datenströmen zugeordnet (die Information hierfür liefert die Level-3 Farm), und in jeweils 1 GB großen Dateien abgelegt. Hierbei werden FibreChannel-Platten verwendet, die dann sowohl für den Logger als auch im Rechenzentrum direkt sichtbar sind, wo die Daten dann auch auf Bänder geschrieben werden.

in Kapitel 6.2.2 beschrieben.

Consumer und Consumer-Server

Im CDF-Jargon ist ein Consumer ein Programm, das *Online*, also während der Datennahme, die genommenen Daten analysiert und überwacht, ob ein bestimmter Aspekt des Detektors korrekt funktioniert. Einige der Consumer sind:

- **XMon** – überwacht die Triggerrate
- **LumMon** – beobachtet die Luminosität
- **Stage0** – für die Kalibration der Driftkammer
- **BeamMon** – mißt die Strahlposition
- **ObjectMon** – inspiziert in Level-3 rekonstruierte Objekte
- **SiliMon** – plottet Verteilungen der Silizium-Hits

Der Consumer-Server wiederum ist das Programm, von dem die Consumer ständig mit den neuesten Ereignissen versorgt werden (mit 1-2 Hz pro Consumer). Dabei funktioniert er ähnlich wie ein Webserver: der ursprünglich gestartete Prozess wartet auf Verbindungen von Consumern. Dabei bearbeitet er die Anfrage jedoch nicht selbst, sondern verzweigt in einen neuen Prozess, der die Anfrage bearbeitet und der sich beendet, wenn der Consumer die Verbindung unterbricht. Der ursprüngliche Vater-Prozess ist jedoch dafür zuständig, daß nahe der maximalen Netzwerklast kein Consumer einen anderen völlig blockieren kann.

Consumer haben zwei Wege, ihre Ergebnisse nach außen zu leiten. Für Textmitteilungen wie Fehlermeldungen oder Zusammenfassungen gibt es den im folgenden beschriebenen Fehler-Logger. Detaillierter sind jedoch von den Consumern bereitgestellte Histogramme. Jeder Consumer erhält einen ihm zugeordneten Server-Prozess, der diese Histogramme (oder auch andere ROOT-Objekte) für das Display-Programm und das in Kapitel 3.1.3 beschriebene Web-Interface zur Verfügung stellen. Welche Server momentan verfügbar sind, ist über eine automatisch aktualisierte Webseite ersichtlich.

Es mag auf den ersten Blick unnötig kompliziert erscheinen, den Consumer-Server, Consumer und zu den Consumern gehörige Server in so viele Prozesse aufzuspalten, jedoch hat dieses Verfahren den Vorteil, daß, falls dies nötig sein sollte, jeder Consumer auf einem dedizierten Rechner laufen kann. Dadurch ist es möglich, auch sehr rechenzeitintensive Consumer einzusetzen.

Das Consumer-Framework wird ausführlich in [12] und [13] beschrieben.

Fehler-Logs

Ebenfalls Teil des Consumer-Frameworks ist eine zentrale Sammelstelle für Fehler- und andere Meldungen, der *Error Receiver*. Über ihn melden die Consumer Probleme und Run-Zusammenfassungen, die der Error Receiver direkt in den Kontrollraum weiterleitet, so sie denn wichtig genug sind. Unabhängig davon werden die Meldungen in einer Oracle-Datenbank gespeichert. Diese Datenbank kann über ein Web-Interface, den in Kapitel 3.1.2 beschriebenen Consumer Database Browser, abgefragt werden.

Monitoring im Kontrollraum

Im Kontrollraum laufen die in Abbildung 3.3 unten links eingezeichneten Display Clients, das ConsumerDisplay. Dieses verfügt über eine in ROOT implementierte graphische Benutzeroberfläche, mit der schnell und bequem durch die Histogramme der verschiedenen Consumer navigiert werden kann. Auch werden dringende Fehlermeldungen der Consumer direkt in den Kontrollraum geleitet. Abbildung 3.4 zeigt ein Photo des CDF II Kontrollraums mit laufenden Monitor-Programmen. Weitere Informationen zum ConsumerDisplay finden sich in [14].

Abbildung 3.4: Der CDF II Kontrollraum.

3.1.2 Der Consumer Database Browser

Da der direkte Zugriff auf die Online-Datenbanken von außerhalb des Fermilab-Netzwerks[4] gesperrt ist, kann die Kontrollraum-Software nur dort genutzt werden, nicht jedoch an anderen Instituten. Es wäre auch nicht unbedingt wünschenswert, die Kontrollraum-Software außerhalb des Kontrollraumes zu installieren, denn zum einen bietet diese Software auch Steuerfunktionen, und es ist natürlich wünschenswert, daß nur vom Kontrollraum aus in den Zustand des Detektors eingegriffen werden kann. Zum anderen muß gerade diese Software immer aktuell gehalten werden, da sich Programmfehler hier besonders schmerzlich bemerkbar machen würden. Hieraus leitet sich der Bedarf für Überwachungssoftware ohne Steuerfunktionen ab, die über das WWW von beliebigen Rechnern abgefragt werden kann.

Mit dem Consumer Database Browser wurde nun eine Möglichkeit geschaffen, mit einem beliebigen Web-Browser in der Online-Datenbank Run-Informationen zu recherchieren. Dazu genügt es, mit dem Web-Browser folgende URL zu öffnen [15]:

[4] genauer: von außerhalb des *Online*-Subnetzes.

`http://www-cdfonline.fnal.gov/clservlet/servlet/CdfConsumerServlet`

Diese Seite enthält Eingabemöglichkeiten für Run-Nummern und ermöglicht die Auswahl der gewünschten Consumer. Abbildung 3.5 zeigt die Eingabemaske.

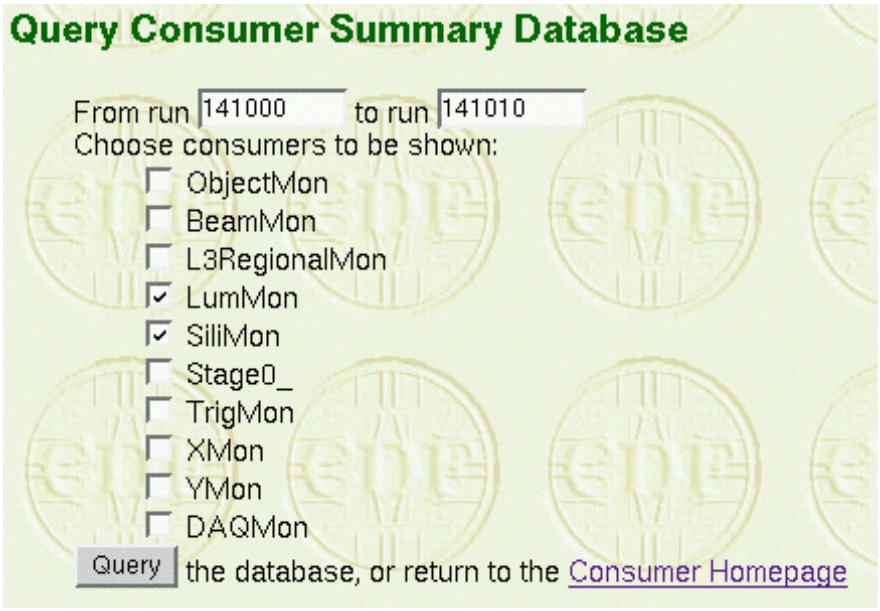

Abbildung 3.5: Der Consumer Database Browser-Startbildschirm.

Nachdem der Benutzer den gewünschten Run-Bereich (im Beispiel Run 141000 bis 141010) angegeben und die gewünschten Consumer (hier: LumMon und SiliMon) ausgewählt hat, werden die entsprechenden Daten aus der Datenbank tabellarisch dargestellt, wie Abbildung 3.6 zeigt. Folgt man dem "Errors"-Link eines Runs, so erhält man weitere Details zu den während des Runs aufgetretenene Fehlern (Abbildung 3.7).

Java – Servlet- und JDBC-Technologie

Zur Implementation des Consumer Database Browsers wurde die Programmiersprache Java [16] verwendet. Java ist eine Entwicklung von Sun Microsystems, die dank ihrer umfangreichen und meist gutdurchdachten Bibliotheken und aufgrund der Tatsache, daß sie zwar ähnlich, aber wesentlich einfacher als C++ ist, weite Verbreitung gefunden hat[5]. Ein weiterer Vorteil von Java, die Plattformunabhängigkeit, kommt bei diesem Projekt nicht zum tragen.

Die obengenannten umfangreichen Bibliotheken waren auch der Hauptgrund dafür, daß sich der Autor für Java als Implementationssprache entschied, denn jene Bibliotheken enthalten auch zahlreiche leistungsfähige Klassen für die beiden Hauptprobleme, WWW- und Datenbankzugriff:

[5]Viele Konzepte in C++, die unerfahrene Programmierer oft verwirren, sind in Java nicht enthalten, so etwa Zeiger, das Überladen von Operatoren, Templates, und echte Mehrfachvererbung – eben die Konzepte, die C++ so mächtig machen. Außerdem nimmt Java dem Entwickler die Speicherverwaltung ab, indem es nicht mehr genutzten Speicher automatisch wieder freigibt ("*Garbage Collection*").

Consumer Summary

Run#	Consumer	# Evt	Evaluation	Comment
141000	LumMon	3475	UNDEFINED (Errors)	NoneNoneNone
141000	SiliMon	106	PROBLEM (Errors)	No_clusters_detected_in_layer_No_ADC_counts_
141002	SiliMon	931	PROBLEM (Errors)	No_clusters_detected_in_layer_No_ADC_counts_
141004	SiliMon	149	PROBLEM (Errors)	No_clusters_detected_in_layer_No_ADC_counts_
141007	SiliMon	172	PROBLEM (Errors)	No_clusters_detected_in_layer_No_ADC_counts_
141010	SiliMon	227	PROBLEM (Errors)	No_clusters_detected_in_layer___0____1____2_

Abbildung 3.6: Der Consumer Database Browser zeigt die Ergebnisse einer Anfrage.

Consumer Error CID 47264

Message:	Severity:	Issuer:	# issued:
CLC single channel r	warning	CLAD	57
CdfJetCollPuffer:	warning	makeJetAlgPara	4
CdfMetPuffer:	warning	makeJetAlgPara	41
LumMon Summary	unspecified	LumMonModule	6
OCI unique constrain	warning2	-	1
SiAlignmentManager:	severe	makeJetAlgPara	1

6 entries found.

Abbildung 3.7: Übersicht über die Fehler während eines Runs.

- **Java Servlets**
  Servlets sind eine Möglichkeit, Java zur Erzeugung dynamischer Webseiten einzusetzen [17]. Hierbei wird ein in Java geschriebener Mini-Webserver (die Servlet Engine) gestartet, der durch die sogenannten Servlets – das sind Klassen, die von der Basisklasse `HttpServlet` erben – erweitert wird. Wird von einem Browser eine URL, die mit `/servlet/` beginnt, angefordert, so ruft die Servlet Engine die `doGet()`-Methode des gleichnamigen Servlets auf. Diese Methode ist dann für die Erzeugung der der HTML-Seite verantwortlich; entweder (bei einfachen Seiten) erzeugt sie sie selbst, oder sie delegiert diese Aufgabe an eine sogenannte Business Class.

  Mittlerweile gibt es die Java Server Pages-Technologie (JSP), die Servlets beim Erstellen umfangreicherer dynamischer Seiten mehr und mehr ablöst und auf Enterprise Java Beans basiert. Da der Consumer Database Browser jedoch lediglich zwei Seiten beinhaltet, bringt der Einsatz von JSP hier keine nennenswerte Vorteile.

- **Java Database Connectivity** (JDBC)
  Mit JDBC bietet Java ein einheitliches Interface für eine Vielzahl von SQL-Datenbank-

Servern[6] an, so auch für den bei CDF verwendeten Oracle-Server. Dabei wird zunächst einmalig der Datenbanktreiber geladen; dann können mittels des Treibers Verbindungen zur Datenbank aufgebaut werden (Klasse `Connection`). Über diese Verbindung werden dann Befehle an die Datenbank gegeben (Klasse `Statement`). Handelt es sich bei den Befehlen um Abfragen (`Statement::executeQuery()`), so werden die Ergebnisse der Abrage in einer Instanz der Klasse `ResultSet` zurückgeliefert, über deren Einträge dann iteriert wird. Danach wird die Verbindung wieder geschlossen, da aufgrund lizenzrechtlicher Aspekte nur eine begrenzte Zahl gleichzeitiger Verbindungen mit der Oracle-Datenbank möglich ist. Eine Einführung in JDBC gibt unter anderem [18].

Aufbau des Consumer Database Browser-Servlets

Das Consumer Database Browser-Servlet ist in zwei Teile untergliedert, die die beiden Aspekte, Servlet und JDBC, widerspiegeln. Abbildung 3.8 zeigt das Klassendiagramm in OMT-Notation[7] [20].

Die erste Klasse, `CdfConsumerServlet`, ist dabei für das Protokoll zuständig. Sie wertet den HTTP-Parameter `function` aus, der festlegt, welche Business-Klasse aufgerufen wird. Dementsprechend müssen alle Business-Klassen in `CdfConsumerServlet` aufgeführt sein.

`CdfConsumerBusinessClass` ist die abstrakte Basisklasse für alle Business-Klassen. Sie stellt die Schnittstelle zum Servlet bereit. Außerdem ist hier das Erscheinungsbild der HTML-Seite definiert: Hintergrundbild, Farbe und Position des Titels, sowie Links auf andere Seiten. Schließlich stellt die Klasse auch noch eine Log-Funktion zur Verfügung, mit der abgeleitete Klassen Fehler ausgeben und protokollieren können.

Die beiden von `CdfConsumerBusinessClass` abgeleiteten Klassen schließlich, `CdfConsumerSummary` und `CdfConsumerError`, sind für die Bereitstellung des eigentlichen Inhalts zuständig. `CdfConsumerSummary` greift auf die `consumer_summary`-Tabelle der `cdfonprd`-Datenbank[8] zu und extrahiert für den gewünschten Run-Bereich Run-Nummern, Consumer (soweit gewünscht), Anzahl der Ereignisse im Run, Bewertung des Runs durch die Consumer, sowie weitere Kommentare und für jeden Run einen Link auf die entsprechende Fehlerseite. Die `CdfConsumerSummary`-Klasse stellt auch das Formular bereit, in dem Run-Nummern und Consumer ausgewählt werden. Sie wird auch aufgerufen, wenn kein `function`-Parameter angegeben ist.

Die `CdfConsumerError`-Klasse schließlich stellt die während eines Runs aufgetretenen Fehler im Detail dar. Dazu werden die zum Run gehörigen Meldungen, jeweils mit Informationen zum Grad der Schwere des Fehlers, wie oft der Fehler auftrat und wer ihn aufgenommen hat, aus der `consumer_errors`-Tabelle gelesen und tabellarisch dargestellt (siehe Abbildung 3.7).

[6] SQL, *Structured Query Language*, ist der Industriestandard für Datenbankabfragen. Alle mittleren und großen Datenbanksysteme (Oracle, IBM DB/2, Informix, Sybase, mySQL, PostgreSQL und der Microsoft SQL Server) unterstützen diesen Standard.

[7] OMT, *Object Modeling Technique*, ist eine graphische Notation für objekt-orientiertes Design, die in [19] vorgestellt wird.

[8] CDF Online Production-Datenbank.

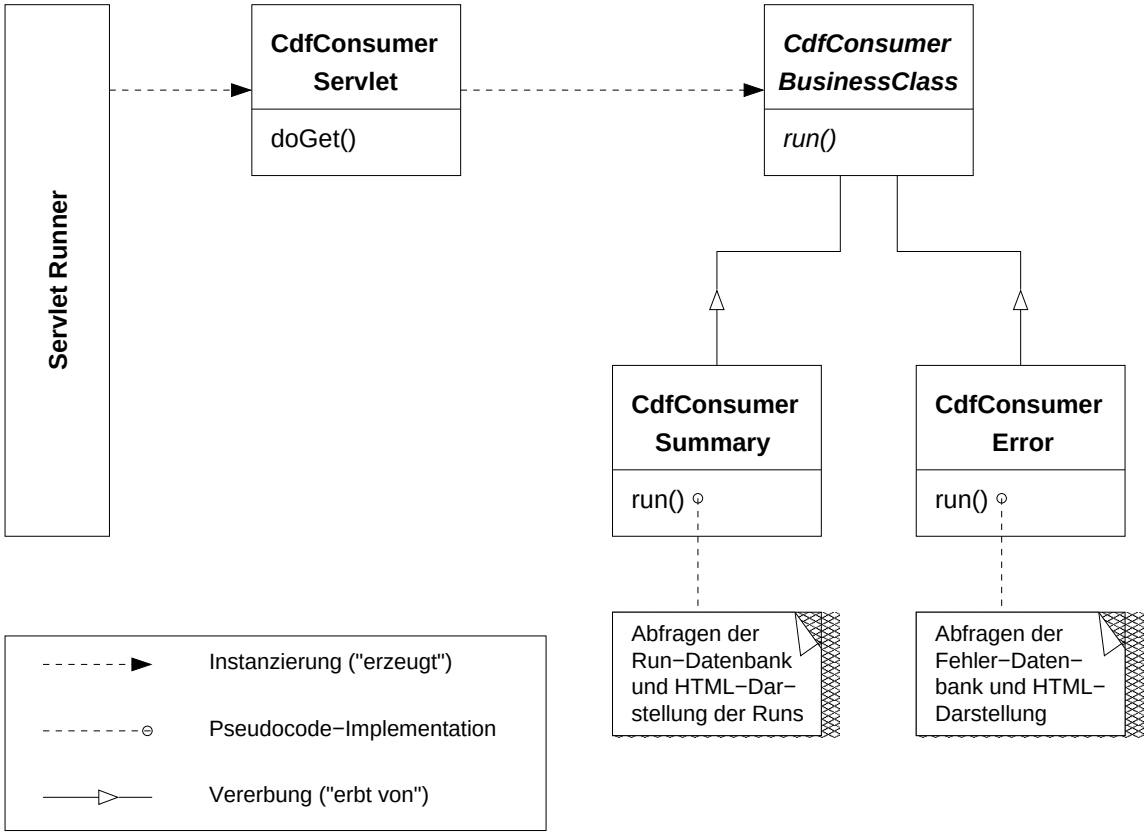

Abbildung 3.8: Architektur des Consumer Database Browser-Servlets.

Der Quellcode des Consumer Database Browsers ist im CDF-Paket `ConsumerFramework`, im Verzeichnis `ErrorLogger` zu finden [21].

3.1.3 Der Consumer Histogram Browser

Die Problemstellung beim Consumer Histogram Browser ist analog zu der des Consumer Database Browsers: der Benutzer will von außerhalb des *Online*-Subnetzes die ROOT-Histogramme[9], welche von den Consumern gefüllt werden, ohne spezielle Softwareinstallation von einem beliebigen Rechner aus überwachen. Wieder wird dies verhindert, zum einen von der B0-Firewall (die das *Online*-Netz schützt und Zugriffe von außen auf den ROOT-Server unterbindet), zum anderen dadurch, daß ROOT und die entsprechenden Erweiterungsklassen für die Consumer nicht überall installiert sind. Entsprechend wurde als Lösung wieder eine interaktive WWW-Seite implementiert [22]:

`http://www-cdfonline.fnal.gov/~schemitz/cgi-bin/chb.cgi`

[9]Neben Histogrammen kann der Consumer Histogram Browser auch andere Graphiken darstellen. Ursprünglich waren jedoch nur Histogramme möglich; im folgenden wird stets von Histogrammen gesprochen.

Lädt man diese URL mit einem graphischen WWW-Browser[10], wird man aufgefordert, den gewünschten Consumer (Rechnername sowie Port) anzugeben, wie Abbildung 3.9 zeigt. Welche Consumer momentan verfügbar sind, erfährt der Benutzer unter

`http://www-cdfonline.fnal.gov/consumer/consumer_status.html`

Es ist nötig, beim ersten Aufruf die Checkbox **Hide List** (die die zum Teil sehr umfangreichen Listen angebotener Histogramme unterdrückt, was für wiederholtes Anzeigen einzelner Histogramme hilfreich ist) *nicht* zu markieren. Ein Click auf den **Go!**-Knopf listet die vom gewählten Consumer bereitgestellten Histogramme auf (Abbildung 3.10).

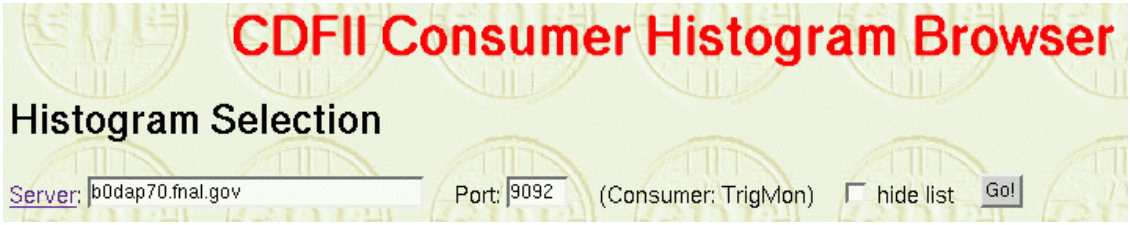

Abbildung 3.9: Auswahl des Consumers im Consumer Histogram Browser.

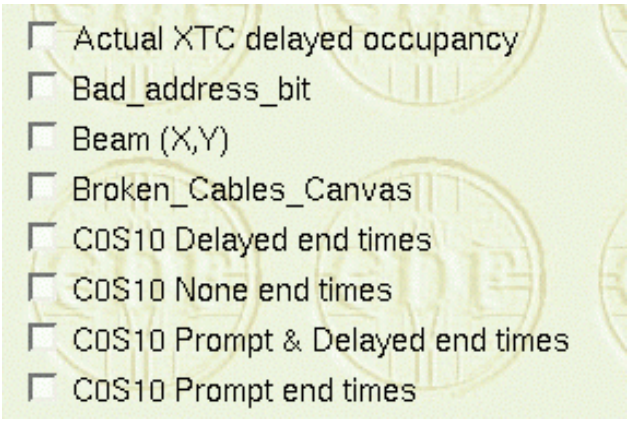

Abbildung 3.10: Auswahl der Histogramme im Consumer Histogram Browser.

Manche Consumer bieten sehr viele Histogramme an, die auch noch sehr lange Namen tragen, sodaß die Liste der Histogramme mehrere Seiten umfassen kann. Will man die zeitliche Entwicklung eines (oder mehrerer) Histogrammes untersuchen, also das Histogramm alle paar Minuten aktualisieren, sind solche langen Listen störend und können durch Markieren von **Hide List** abgeschaltet werden. Ein weiterer Click auf **Go!** zeigt die gewünschten Histogramme an (Abbildung 3.11 zeigt ein Beispiel).

[10] Anders als beim Consumer Database Browser funktionieren Text-basierte Browser wie Lynx oder Links hier nicht, da diese die Histogramme – Bilder – nicht darstellen können.

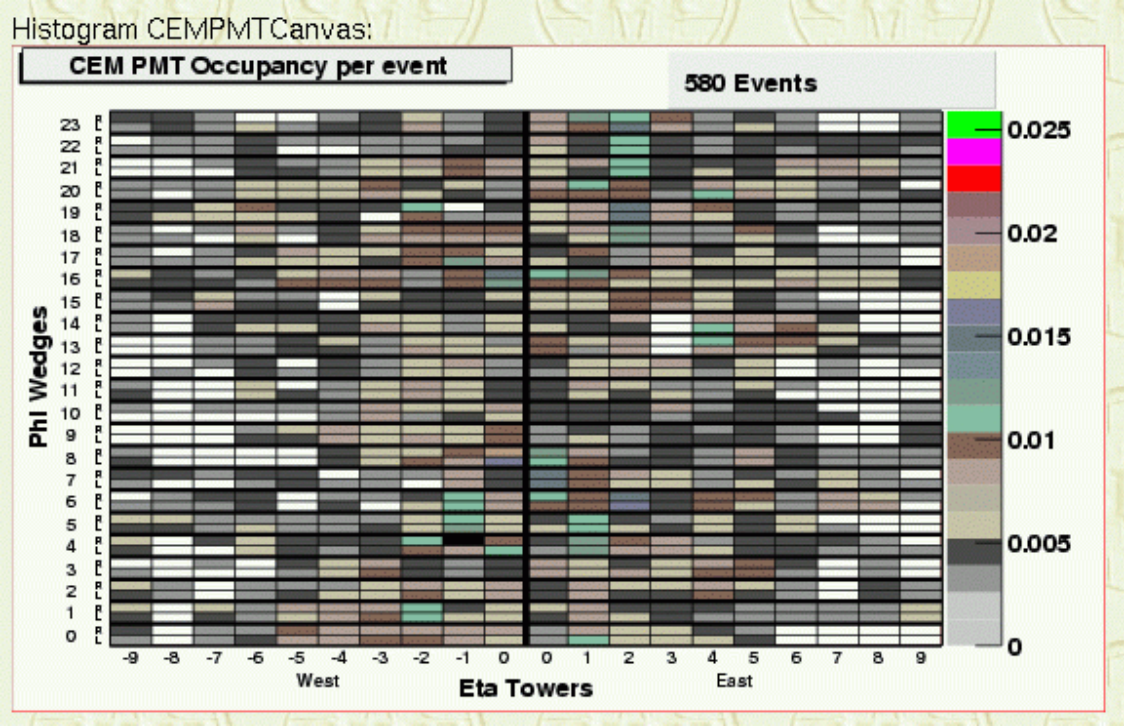

Abbildung 3.11: Ein Plot im Consumer Histogram Browser.

Implementation

Die naheliegendste Art, den Consumer Histogram Browser zu implementieren, wäre wieder ein Servlet analog zum Consumer Database Browser gewesen. Dies wird dadurch unmöglich gemacht, daß es keine Möglichkeit gibt, mit Java auf ROOT-Server zuzugreifen: erstens gibt es keine Unterstützung für das verwendete Netzwerkprotokoll, zweitens stehen die benötigten ROOT-Klassen (in diesem Fall vor allem die Histogrammklassen) nur unter C++ zur Verfügung. Damit bleibt für die Implementation nur die Möglichkeit, in C++ geschriebene und auf die ROOT-Bibliotheken zurückgreifende CGI-Programme zu verwenden. Dabei wird die Funktionalität auf zwei Teilprogramme aufgeteilt, die im folgenden beschrieben werden. Abbildung 3.12 zeigt schematisch die Funktionsweise der Consumer Histogram Browsers.

1. **chb**
   Das erste CGI-Programm fragt den angegebenen Consumer-Server ab und erstellt ein Menü mit den vom Consumer gefüllten Histogrammen, aus dem der Benutzer auswählen kann. Für jedes ausgewählte Histogramm erzeugt das Programm ein HTML-Bild-Tag, das auf das unten beschriebene `histo2png` verweist und diesem Consumer Server-Name und -Port sowie den Namen des Histogramms übergibt.

   Dabei nutzt der `chb` die Tatsache, daß jeder Consumer Server Metainformationen über sich selbst in Form einer Instanz der Klasse `TConsumerInfo` bereitstellt, in der die Namen der Histogramme aufgeführt sind.

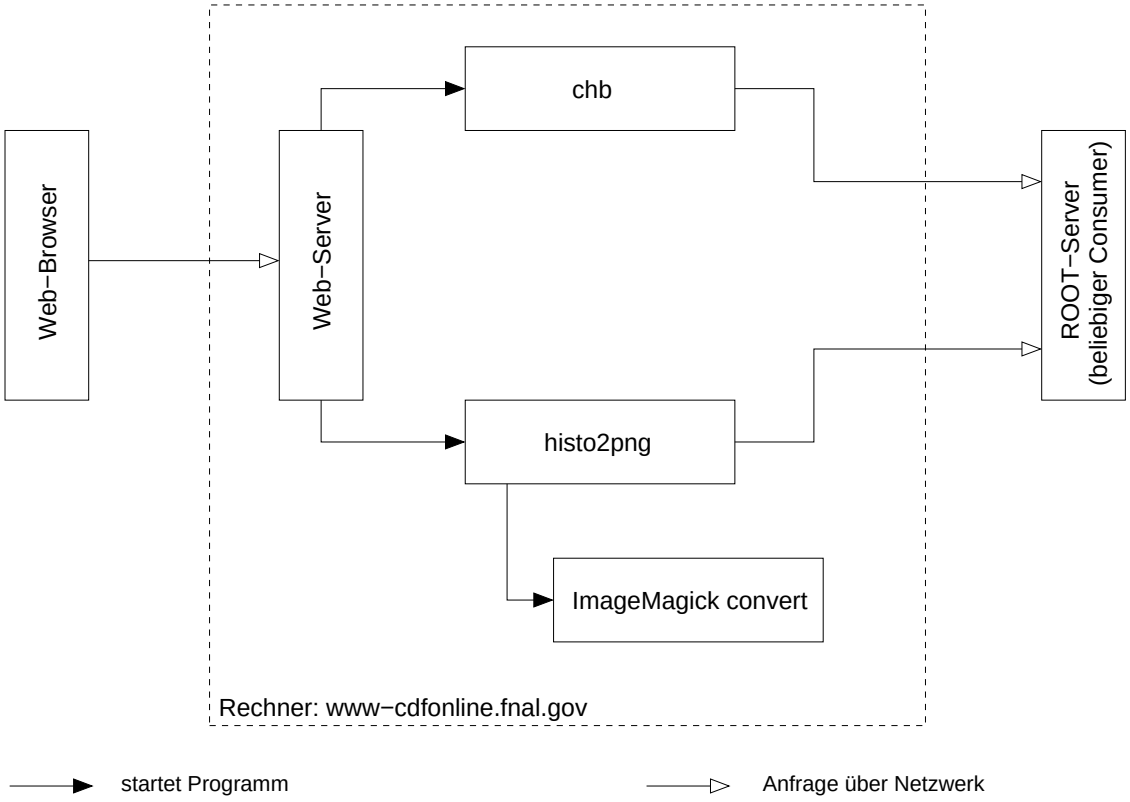

Abbildung 3.12: Funktionsschema des Consumer Histogram Browser.

2. **histo2png**
   Das zweite CGI-Programm ist dafür zuständig, ein bestimmtes, mit dem `chb` ausgewähltes Histogramm vom angegebenen Consumer-Server zu holen und in ein von Web-Browsern darstellbares Graphikformat umzuwandeln.

   Der `histo2png`-Konvertierer stellt dabei eine Verbindung zum angegebenen Consumer Server her und holt sich eine Kopie des gegebenen Histogramms, die schließlich in eine Bilddatei ausgegeben wird. Hierzu sind gleich zwei Umwege nötig: es ist nämlich nicht möglich, die ROOT-Funktion, ein Histogramm in ein GIF-Bild auszugeben, im Batchbetrieb zu nutzen[11], was jedoch für ein CGI-Programm zwingend notwendig ist[12]. Also wird zunächst ein Postscript-Bild des Histogramms erzeugt. Dieses wird dann mittels des freien Konvertierungswerkzeuges ImageMagick [23] in ein PNG-Bild gerendert. Das weiter verbreitete GIF-Dateiformat wird aus patentrechtlichen Gründen nicht verwendet[13].

[11] Der Grund hierfür ist, daß diese Funktionalität von ROOT eigentlich ein verkappter Screenshot ist. ROOT verfügt über keine Möglichkeit, ohne den Umweg über X-Windows in eine GIF-Datei zu zeichnen.

[12] CGI-Programme werden vom Webserver gestartet, der wiederum als Daemon läuft, ohne Zugang zu X11, einer Konsole oder anderen Ein-/Ausgabe-Geräten. Daher müssen CGI-Programme, die ROOT benutzen, dieses in den sogenannten Batch-Modus schalten, in dem ROOT auf jegliche Form von graphischer Ausgabe verzichtet – sogar auf den sonst unvermeidlichen *Splash Screen*.

[13] Unisys Corp. hält ein Patent auf den im GIF-Dateiformat verwendeten Kompressionsalgorithmus.

Der Quellcode des Consumer Histogram Browsers ist im CDF-Paket `ConsumerFramework`, im Verzeichnis `HistogramBrowser` zu finden [21].

3.2 Offline I – Spurrekonstruktion

Für viele Analysen sind gut rekonstruierte Teilchenspuren von herausragender Bedeutung. Hier besonders hervorzuheben ist die Bestimmung von Primär- und Sekundärvertex in einem Ereignis, deren genaue Bestimmung beispielsweise Lebensdauermessungen von B-Mesonen überhaupt erst ermöglicht. Überhaupt sind Primär- und Sekundärvertices für die gesamte Bottom-Physik essenziell, wie auch für die Top-Physik, da Top-Quark-Zerfälle immer auch b-Quarks enthalten. Die Existenz eines Sekundärvertex ist auch ein wichtiges Trigger-Kriterium.

Der CDF-Detektor verfügt über zwei Tracking-Systeme, den Silizium-Detektor (bestehend aus Lage 00, SVX II und ISL) und die zentrale Driftkammer (COT). Von der Auflösung her ist das Silizium-System der Driftkammer weit überlegen, da der Abstand zum Wechselwirkungpunkt geringer ist (und damit Extrapolationsfehler weniger schwerwiegend), und da mit Siliziumstreifen ein sehr viel geringerer Abstand realisierbar ist als mit dem Drähten der Driftkammer. Trotzdem ist die Driftkammer von großem Nutzen. So ist durch die größeren Spurlängen innerhalb der Driftkammer eine genauere Messung der Spurkrümmung und damit des Teilchenimpulses möglich. Auch sind im Silizium-System oft so viele Kanäle aktiv, daß es aufgrund der Kombinatorik zuviel Rechenzeit kosten würde, sie "direkt" mit *Silicon Standalone*-Strategien zu rekonstruieren. Hier verwendet man *Outside-In*-Strategien, bei denen COT-Spuren in den Silizium-Detektor hinein fortgeführt werden. Das ist deshalb vorteilhaft, weil in der Driftkammer die Spurdichte geringer ist, sodaß die Spuren dort in recht kurzer Zeit rekonstruiert werden können. Ein weiterer Vorzug der Driftkammer ist, daß sie ein großes Volumen umschließt und dabei nur wenig wechselwirkende Materie darstellt (und damit wenig Vielfachstreuung und Energieverlust der passierenden Teilchen verursacht), und so gut für Impulsmessungen geeignet ist. Diese Arbeit beschränkt sich im folgenden jedoch auf Spurrekonstruktion mit dem Silizium-System.

3.2.1 Kaltrak – Kalmanfitter-basiertes Tracking

Ende 1999 begann am EKP die Entwicklung eines neuen Tracking-Pakets für das Silizium-System [24]. Dies war nötig geworden, nachdem sich herausgestellt hatte, daß es mit dem ursprünglichen Entwurf nicht möglich war, den schnellen Kalman-Fitter, der im Delphi-Experiment zur Spurrekonstruktion verwendet wurde, effektiv einzusetzen. Der Kalman-Fitter, dessen grundsätzliche Funktionsweise in [25] ausführlich erläutert wird, liefert quasi als Nebenprodukt zum eigentlichen Fit auch die Verschiebungen der einzelnen Meßpunkte gegen den Fit, die für die Bestimmung des Misalignments des Detektors benötigt werden. Allerdings ist der Fitter unter anderem daher so viel schneller als der Kalman-Fitter des ersten Entwurfs, weil er sich die Tatsache zunutze macht, daß die Teilchenspuren zum einen stets Helices sind, zum anderen daß die Achsen dieser Helices immer entlang des Magnetfeldes im Detektor, also entlang der z-Achse des Detektors verlaufen. Es gibt keine naheliegende Möglichkeit, den Fitter in den ersten Entwurfs zu integrieren. Wohl könnte er verwendet

werden, doch liefe das sowohl dem Design des Tracking-Pakets als auch dem des Fitters entgegen, letzteres insbesonders aufgrund der Tatsache, daß der Fitter Helices anders parametrisiert als der erste Entwurf des Tracking-Pakets. So müßten also die Parameter vor und nach jedem Fit-Vorgang konvertiert werden, was weder der Geschwindigkeit noch der Genauigkeit zuträglich wäre.

Erklärte Ziele bei der Umsetzung des Neuentwurfs waren:

- Einhalten der CDF II-Standard-Schnittstellen mit bestehendem Code.
- Ein schneller Entwicklungszyklus.
- Verbesserte Wartbarkeit gegenüber dem ersten Entwurf.

Einhalten der CDF II-Standard-Schnittstellen

Der neue Entwurf muß im Standard-CDF II-Framework funktionieren, also in AC++-Moduln[14] mit entsprechenden Konfigurationsmöglichkeiten nutzbar sein. Details zum CDF II-Software-Framework finden sich in [26] und [27]. Außerdem muß das Paket CDF II-Daten im EDM2-Format lesen und schreiben können [28].

Schneller Entwicklungszyklus

Die Arbeit an Kaltrak hat fast drei Jahre nach dem ersten Entwurf begonnen, und um diese Zeit aufzuholen und noch rechtzeitig zum Beginn von Run II ein lauffähiges Programm vorzeigen zu können, mußte die Entwicklung sehr schnell vonstatten gehen. Weder für eine lange Design-Phase noch für lange Lern-Phasen für neue Mitarbeiter war Zeit, sodaß das Design sehr übersichtlich blieb (Abbildung 3.14 im Vergleich zu 3.13). Außerdem wurde Kaltrak so angelegt, daß der Zyklus Editieren-Übersetzen-Linken-Testen möglichst kurz ist; denn wenn dieser Zyklus einige Minuten dauert, ist dies abträglich für die Produktivität der Entwickler. Daher müssen Abhängigkeiten zwischen Paketen minimiert werden.

Wartbarkeit

Der überarbeitete CDF-Detektor ist, was nicht ungewöhnlich ist, noch nicht voll verstanden. Daher unterliegt die Rekonstruktionssoftware auch jetzt noch kontinuierlichen, möglicherweise drastischen Änderungen. Es hat also höchste Priorität, den Code wartbar zu halten. Auch daher ist ein einfaches, intuitives Design angeraten, denn der Code wird einmal von neuen Mitarbeitern gewartet werden, und die ursprünglichen Autoren werden an anderen Projekten arbeiten oder die Physik ganz verlassen.

3.2.2 Entwurfs-Ansatz

Statt sich auf das objekt-orientierte Paradigma zu beschränken, wurde auf den wesentlich flexibleren Ansatz des *Multi Paradigm Designs* (MPD) zurückgegriffen. Dieser Ansatz wird in [1] vorgestellt und propagiert, wie der Name schon sagt, die Benutzung mehrerer Paradigmen

[14] *Analysis Control++* (AC++) ist das Analyseframework von CDF II.

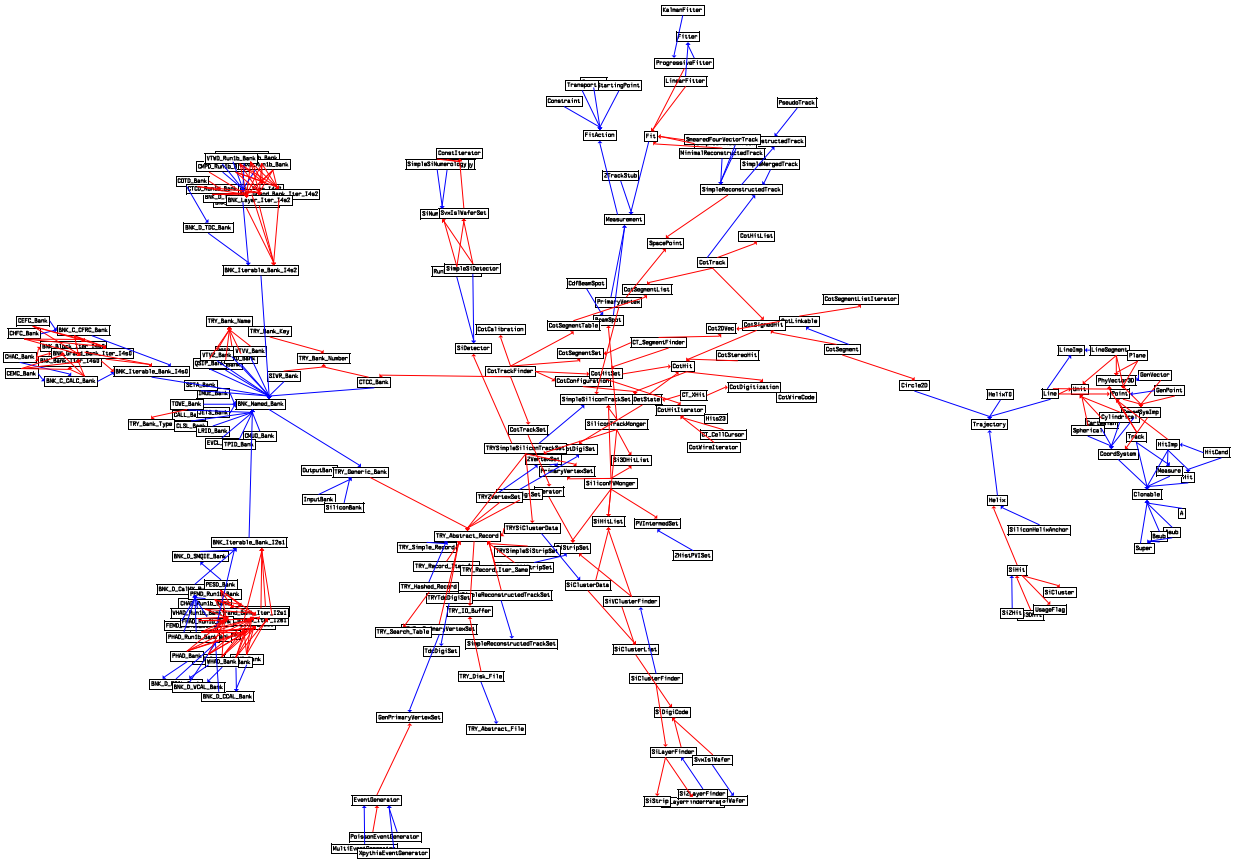

Abbildung 3.13: Klassenstruktur des ersten Tracking-Pakets. Die äußerst komplexen Klassenhierarchien erfordern eine lange Einarbeitungszeit.

innerhalb eines Projekts, wobei das Projekt dabei in Teilbereiche ("*Domains*") aufgegliedert wird, für die dann jeweils das geeignetste Paradigma zu wählen ist. In dieser Hinsicht ist MPD gewissermaßen ein Meta-Paradigma. C++ unterstützt explizit mehrere Paradigmen [2].

Ein Beispiel, bei dem das objekt-orientierte Paradigma ungeeignet ist, sind Container, zum Beispiel Listen oder Vektoren. Natürlich ist es möglich, eine Containerklasse objekt-orientiert zu implementieren, wie beispielsweise die Java-Klasse `java.util.Vector`. Hierbei wird in der Containerklasse für Elemente stets der Typ *Referenz auf* `Object` verwendet, sodaß alle Elemente im Container von `Object` abgeleitet sein müssen[15]. Der Schwachpunkt des rein objekt-orientierten Containers besteht jedoch darin, daß *alle* von `Object` abgeleiteten Instanzen in denselben Container passen – es gibt keine Typsicherheit. Es wäre also möglich, Tracks in die Hitlisten zu füllen oder umgekehrt. Nun kann man zwar Container

[15] In C++ wäre bereits das ein gravierendes Problem, da die elementaren Datentypen (`int`, `float` etc.) von keiner Basisklasse wie `Object` erben, sodaß die Listen nicht für elementare Datentypen zur Verfügung stünden.

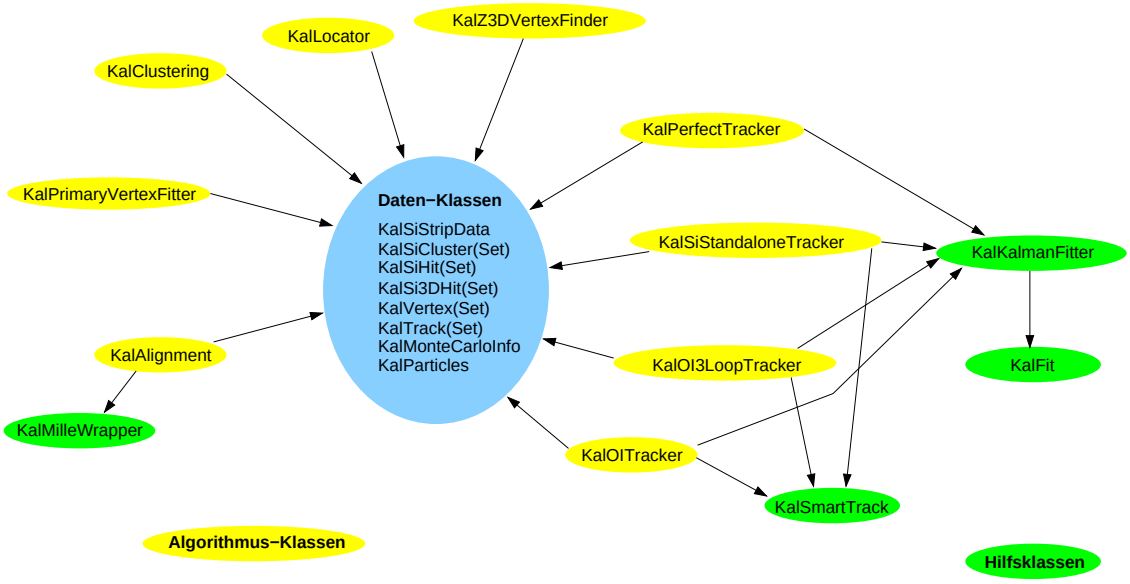

Abbildung 3.14: Klassenstruktur des Kaltrak-Pakets. Das Design ist sehr einfach, die Klassen bilden wenn überhaupt nur sehr flache Hierarchien.

dadurch spezialisieren, daß man die Hit-Liste von der ursprünglichen Listenklasse (etwa `java.util.Vector`) ableitet und die Einfüge-Methode mit einem Test versieht, der sicherstellt, daß das einzufügende Objekt tatsächlich ein Hit ist. Dies ist jedoch nur ein Test zur *Laufzeit*. Ein sauberer Test durch den Compiler, bereits beim Übersetzen, *ist nicht möglich*. In dieser Hinsicht ähnelt die Verwendung von `Object` dem C-Konzept des `void*`.

Hier ist die von C++ genutzte generische Programmierung wesentlich besser geeignet. Dabei wird die Containerklasse für einen zunächst unspezifizierten Typ `T` implementiert; so entsteht ein *Template* (eine Schablone) eines Containers, aus dem der Compiler bei Bedarf die tatsächlichen spezialisierten Container erzeugt. Dabei ist es immer noch möglich, den rein objekt-orientierten Ansatz nachzuahmen, indem man beispielsweise einen Container für `Object*` (oder `void*`) erzeugen läßt.

Algorithmen

Zur Implementation von Algorithmen eignen sich strukturiert prozedurale Ansätze am besten, sind Algorithmen doch Verarbeitungs*anweisungen*. Da viele Algorithmen über Parameter konfigurierbar sind, und da es bisweilen wünschenswert ist, einen Algorithmus mehrfach, aber mit verschiedenen Parametern anzuwenden, werden Algorithmen in Kaltrak als einfache Klassen implementiert, die über eine `process()`-Methode verfügen. Diese Methode bekommt die vom Algorithmus benötigten Datensätze sowie Listen, in denen die Ergebnisse zu speichern sind, übergeben. Abbildung 3.15 verdeutlicht dies.

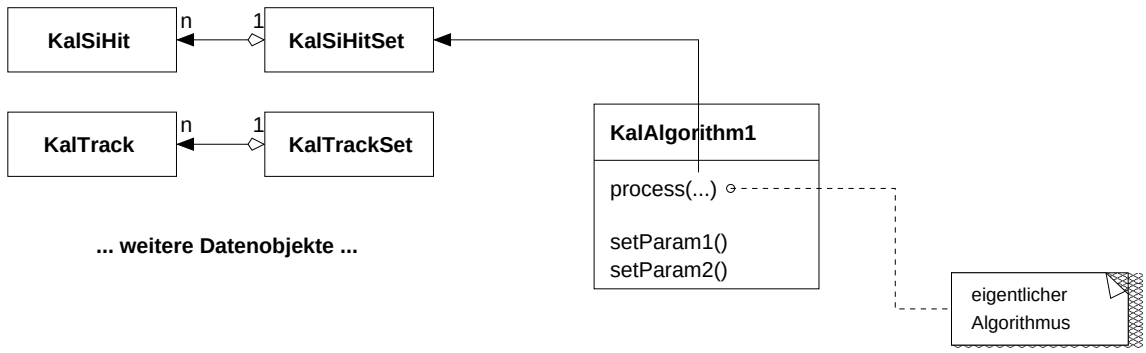

Abbildung 3.15: Implementation eines Algorithmus im Kaltrak-Paket. Nur die tatsächlich benötigten Daten werden der `process()`-Methode übergeben.

Vererbung

Vererbung dient dazu, Gemeinsamkeiten zwischen Klassen – sowohl gemeinsame Interfaces als auch gemeinsame Implementationen – explizit zu machen. Dies kommt im Kaltrak-Paket lediglich an zwei Stellen bei der Darstellung interner Datenstrukturen vor[16], ansonsten wird auf dieses Feature der Sprache verzichtet.

Die einzige andere Stelle, an der es sich noch angeboten hätte, Vererbung einzusetzen, wäre ein gemeinsames Interface der Algorithmenklassen, die `process()`-Methode. Nun benötigen die verschiedenen Algorithmen jedoch nicht alle dieselben Daten (siehe Abbildung 3.18). Um ein einheitliches Interface zu erzwingen wäre es nötig gewesen, jedem Algorithmus *alle* Daten zugänglich zu machen, und zwar nicht nur lesend, sondern auch schreibend. So wird das im CDF II-Software-Framework gehandhabt, und es hat eben den gravierenden Nachteil, daß man nicht mehr sieht, welche Daten ein Algorithmus nun tatsächlich benutzt und wie. Die Gemeinsamkeit zwischen den Algorithmus-Klassen, die Existenz der `process()`-Methode, ist nicht groß genug ist, um eine Klassenhierarchie mit diesem Nachteil zu rechtfertigen. In Anhang A.1 findet sich ein Code-Fragment, mit dem eine gezielte Parameterübergabe über ein einheitliches Interface *halbwegs* elegant möglich ist. Der Nachteil dieser Lösung ist dem der oben erwähnten objekt-orientierten Liste ähnlich: erst zur Laufzeit kann bestimmt werden, ob die Parameter korrekt waren oder zuwenig oder zuviel oder im falschen Modus übergeben wurden. Das Kaltrak-Design bevorzugt die statische Typprüfung durch den Compiler.

"Leuchtspur-Programmierung"

Um die Entwurfsphase zu verkürzen und schneller einen lauffähigen Prototyp zu haben, wurde die Methode der "Leuchtspur-Programmierung" (*tracer bullet method*) angewandt, wie sie in [29] beschrieben wird:

> "Es gibt zwei Möglichkeiten, im Dunkeln mit einem Maschinengewehr zu schie-

[16] Innerhalb der Fitter-Struktur wird Vererbung zur Beschreibung der verschiedenen Fit-Lagen benutzt, in der Darstellung der Detektorgeometrie zur Anbindung an die Alignment-Datenstruktur.

ßen. Sie können die genaue Position des Ziels herausfinden. Sie können die Umweltbedingungen (Wind etc.) bestimmen. Sie können die genauen Spezifikationen von Treibladung und Kugel ermitteln und ihre Wechselwirkung mit dem verwendeten MG. Dann können Sie mittels Tabellen oder eines Feuerleitcomputers die richtige Richtung und Neigung des Laufs berechnen. Wenn sich alles gemäß den Spezifikationen verhält, die Tabellen korrekt sind und sich die Umweltbedingungen nicht ändern, sollten die Kugeln ziemlich nahe am Ziel einschlagen.

Oder Sie können Leuchtspurmunition verwenden."

Analog zu der Maßnahme, in bestimmten Intervallen Leuchtspur-Geschosse im Patronengurt zu plazieren, um sehen zu können, wie nahe man beim Ziel liegt, wurde beim Entwurf von Kaltrak verfahren: nachdem einige wenige Klassen entworfen waren, wurden sie rudimentär implementiert und der Entwurf wurde mit Dummy-Daten getestet. Dadurch wurde schnell sichtbar, welche Teile des Entwurfs als gelungen bezeichnet werden konnten und welche einer Überarbeitung bedurften. So war es möglich, Fehlentscheidungen im Design frühzeitig zu korrigieren und keine Zeit damit zu verschwenden, aussichtslose Konzepte zu verfolgen. Außerdem konnte so die Ausführungsgeschwindigkeit schon während der Design-Phase unter (relativ) realistischen Bedingungen abgeschätzt und damit designbedingte Geschwindigkeits-Engpässe von vorneherein vermieden werden, anstatt später aufwändig optimieren zu müssen.

3.2.3 AC++ versus Standalone

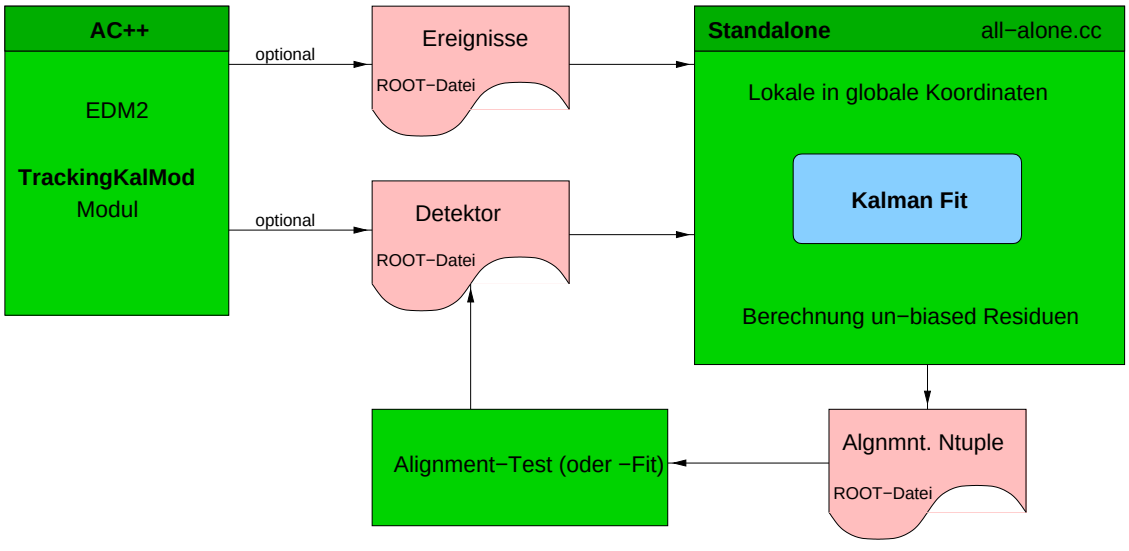

Abbildung 3.16: Kaltrak als AC++-Modul und alleine, ohne AC++.

Die Kaltrak-Rekonstruktionssoftware kennt zwei Modi: als AC++-Modul (`TrackingKalMod`)

und ohne zusätzliche CDF-Software[17] (siehe Abbildung 3.16). Als offizieller Teil der CDF II-Rekonstruktionssoftware kommt das AC++-Modul zum Einsatz. Hier stehen alle AC++-Features zur Verfügung, wie Auswahl der Ereignisse, Kombination mit anderen Moduln und automatischer Abgleich mit der Detektor-Alignment-Datenbank. Da Kaltrak intern keine EDM2-Objekte, sondern andere Datenformate verwendet, wird ein Konverter (`KalConverter`) zwischengeschaltet. Er importiert EDM2-Daten (Silizium-Hits, COT-Spuren), sodaß diese in Kaltrak zur Verfügung stehen, und exportiert Resultate (Spuren, Vertices).

Nachteilig an der AC++-Umgebung ist die Tatsache, daß jegliche Softwareentwicklung extrem träge und langsam ist. Allein das Linken (ohne Compilieren!) des fertigen Programmes aus den Modul-Bibliotheken dauert ca. zwei Minuten[18]. Um zügiger entwickeln zu können, wurde die *Standalone*-Option geschaffen, mittels derer Kaltrak auch ohne weitere CDF II-Software auskommt und den GNU C++-Compiler [30] nutzt, was den Compile-Link-Zyklus von einigen Minuten auf wenige Sekunden reduziert.

Da die Standalone-Version nicht in der Lage ist, EDM2-Daten zu lesen, mußte eine andere Möglichkeit geschaffen werden, in Standalone-Modus an Daten zu kommen. Dazu wurde der *persistente Cache* eingeführt.

Persistenter Cache

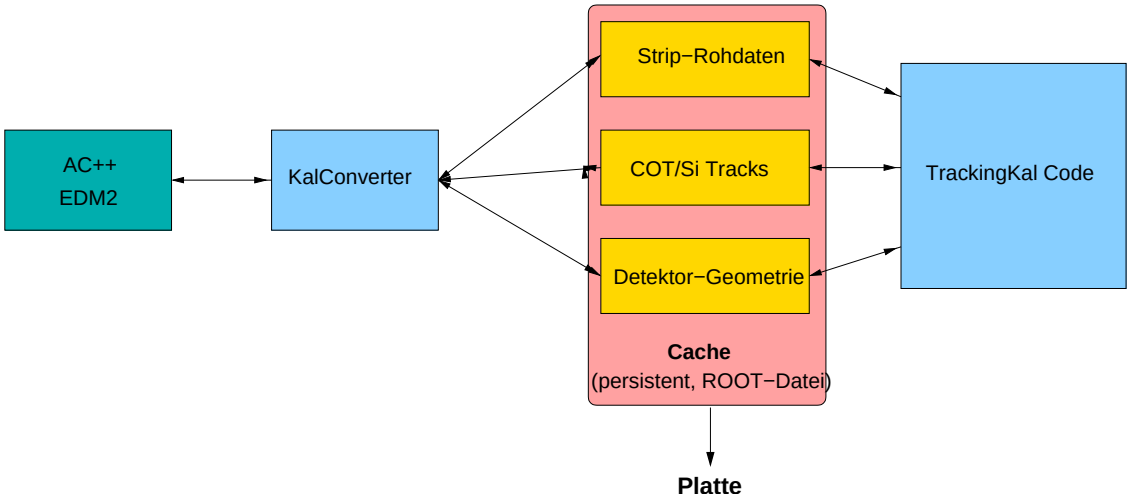

Abbildung 3.17: Persistenter Cache..

Um außerhalb des AC++-Frameworks (und damit ohne EDM2-Daten) arbeiten zu können, muß das Kaltrak-Paket die Daten zwischenspeichern. Dazu können die internen Datenstrukturen, die eigentlich als Cache für die bisweilen schwer zugänglichen EDM2-Datenstrukturen dienen, auf Platte geschrieben werden, wie dies in Abbildung 3.17 verdeutlicht wird. Kaltrak bleibt funktionsfähig, wenn die beiden Blöcke links des Cache entfallen. Man muß lediglich

[17] Anforderungen von Kaltrak im Standalone-Modus: GNU Make; GNU Scientific Library (GSL); GNU oder KAI C++-Compiler; ROOT; CLHEP-Bibliotheken. Betriebssysteme: Linux oder IRIX.

[18] Referenzmaschine EKPCDF1: 2x 1400 MHz Athlon, 2 GB RAM, 100 MB/s Festplattendurchsatz.

einen zuvor abgespeicherten Cache-Inhalt laden (die Ereignis- und die Detektor-Datei in Abbildung 3.16).

Es muß betont werden, daß die Standalone-Option hauptsächlich für Entwickler gedacht ist, da jedes Ereignis, das *standalone* bearbeitet werden soll, zunächst im AC++-Framework mittels des `KalConverter` in das Kaltrak-Format umgewandelt und schließlich gespeichert werden muß. Dieses Speichern kostet zum einen natürlich Zeit, zum anderen – schlimmer – Plattenplatz. Der persistente Cache eignet sich also nur für kleine Datensamples.

3.2.4 Datenfluß

Abbildung 3.18 zeigt den Datenfluß innerhalb des Kaltrak-Pakets. Dabei steht jede horizontale Linie für einen elementaren Datentyp, wie beispielsweise Hits oder Spuren, die durch die verschiedenen Algorithmen – Clustering, Outside-In Tracking etc. – fließen. Pfeile auf den Algorithmus weisen Eingabe-Daten aus, Pfeile aus dem Algorithmus heraus stehen für Ergebnisse. Dabei ist zu beachten, daß Spuren sowohl zu den Eingabe- als auch zu den Ausgabedaten gehören: die Outside-In-Strategien benutzen importierte Driftkammer-Spuren als Startpunkte; und natürlich erzeugen alle Tracker Spuren. Desweiteren muß Kaltrak die 2D-Hits nicht importieren (was es für gewöhnlich tut), sondern kann das Clustering (die Konstruktion von Hits aus Rohdaten) auch selbst übernehmen.

Die Ergebnisse des Kaltrak-Pakets sind 3D-Hits, Spuren und Vertices.

Die in 3.18 extra aufgeführten Datenströme dienen zur Untersuchung der Algorithmen selbst. Es sind dies:

(1) von KalEffPurity:
Histogramme mit Effizienz und Reinheit der gefundenen Spuren, Spektrum des Transversalimpulses, N-Tupel mit den Spurparametern von rekonstruierten Spuren, perfekten Spuren und Monte Carlo-Wahrheit (ROOT-Datei).

(2) von KalCheck:
Spuren mit Messungen in den einzelnen Lagen, zur Untersuchung von Residuen (ROOT-Datei).

(3) von KalWriteFile:
Spuren mit Messungen in den einzelnen Lagen, Monte Carlo-Informationen falls vorhanden, für Alignment-Studien (ASCII-Datei).

(4) von KalVertexEfficiency:
Histogramme mit Effizienz des Vertexsuchers: Abweichung des rekonstruierten vom wahren Vertex in x, y, z sowie Korrelation von z gegen z-Abweichung bei z-Vertices (ROOT-Datei).

3.2.5 Rekonstruktionsalgorithmen

Die Tracking-Algorithmen von CDF II werden in [31] ausführlich beschrieben. Hier werden nur kurz die wichtigsten Ansätze vorgestellt. Es gibt zwei Möglichkeiten, das Silizium-System

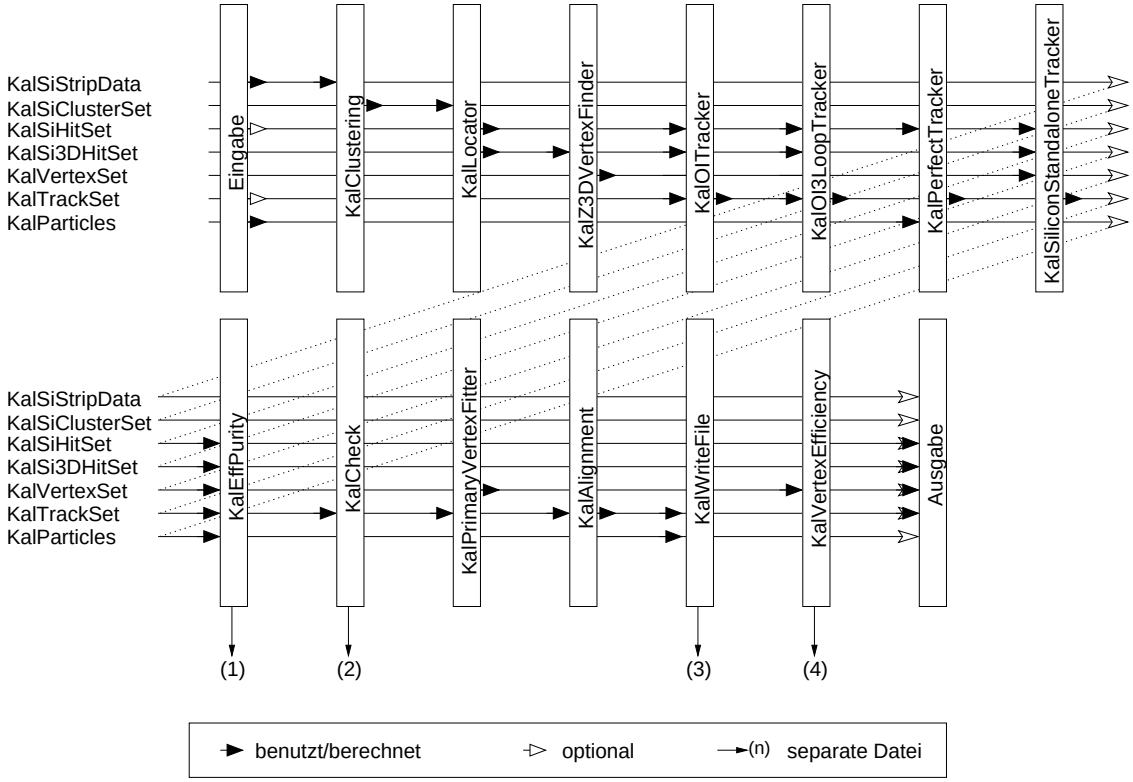

Abbildung 3.18: Ablaufschema des Kaltrak-Pakets. Die Ergebnisdaten (1) bis (4) werden im Text beschrieben.

in die Spurfindung einzubeziehen, eigenständig ("*Silicon Standalone Tracking*") oder in Verbindung mit bereits gefundenen Driftkammerspuren ("*Outside-In Tracking*"). Beide Ansätze haben ihre Vor- und Nachteile, die im folgenden kurz dargestellt werden, und werden deshalb auch beide eingesetzt, sofern genügend Rechenleistung verfügbar ist.

Eigenständiges Silizium-Tracking

Im Magnetfeld des CDF-Spurkammersystems beschreiben Teilchen eine Helixbahn, wobei die Helixachse entlang des Magnetfelds, also parallel zur Strahlache verläuft. Um eine Helix mit ihren fünf Parametern vollständig zu bestimmen, sind drei 3D-Messungen notwendig. Beim eigenständigen Silizium-Tracking werden nun je ein Meßpunkt aus zwei der vier 1.2°-Lagen mit dem Primärvertex zu einer Helix kombiniert. So erhält man einen "Kandidaten". Dieser Kandidat wird nun von außen nach innen durch die Lagen verfolgt und kann dort weitere Hits aufsammeln. Dadurch wird die Helix überbestimmt, und anhand des χ^2 der neuen Hits kann die Güte der Spur berechnet werden. Wird das χ^2 pro Freiheitsgrad zu groß, so wird der Kandidat verworfen.

Die Wahl der Startlagen ist dabei wichtig, da ansonsten Spuren verfehlt werden können, wie Abbildung 3.19 zeigt. Daher muß die eigenständige Si-Tracking-Strategie mehrmals aufgerufen werden, mit jeweils einer anderen Startlagen-Kombination (2,4; 2,5; 2,6; 4,5; 4,6;

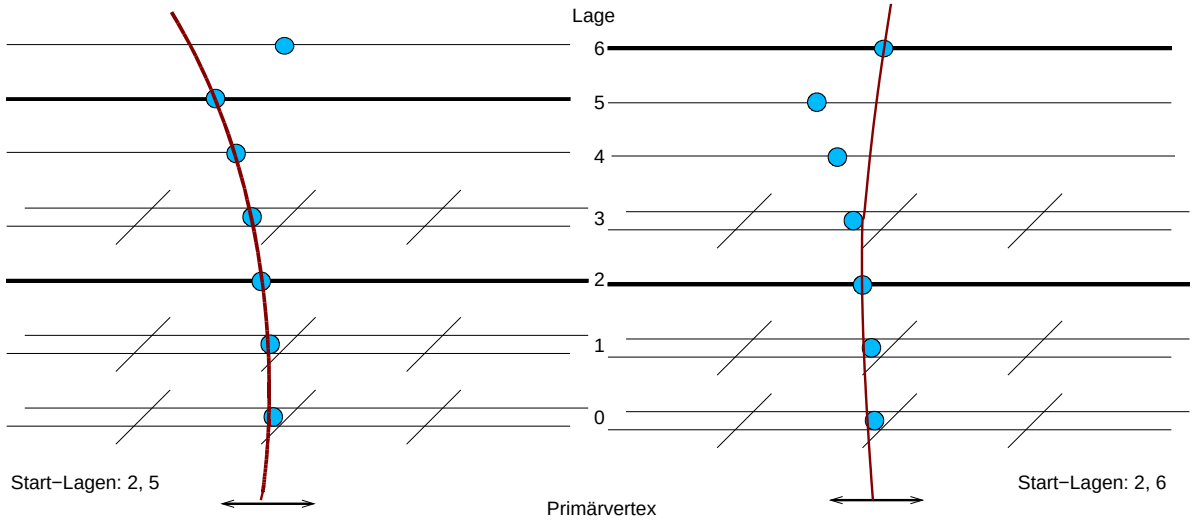

Abbildung 3.19: Eigenständiges Silizium-Tracking. Die Wahl der Startlagen (breite Linien) entscheidet über den Erfolg der Strategie.

5,6). Das führt auch zum größten Nachteil dieser Strategie: die Kombinatorik ist sehr groß, es müssen sehr viele Kandidaten verfolgt und überprüft (und zumeist verworfen) werden. Dazu kommt, daß die Hit-Dichte innen im Detektor ohnehin größer ist als weiter außen in der Driftkammer. Eigenständiges Silizium-Tracking ist also recht CPU-aufwändig. Ein großer Vorteil ist jedoch, daß damit auch Spuren bis zu einer Pseudorapidität von $|\eta| = 2$ rekonstruiert werden können, was mit Outside-In nicht möglich ist, da COT-Spuren nur bis $|\eta| < 1$ rekonstruiert werden können.

Outside-In Tracking

Beim Outside-In Tracking wird, wie Abbildung 3.20 illustriert, eine Driftkammer-Spur in den Silizium-Detektor hinein extrapoliert. Von außen nach innen gehend, sammelt die Spur Hits auf. Der entscheidende Unterschied zum eigenständigen Silizium-Tracking ist dabei, daß der Spurkandidat von der COT kommt und nicht aus den 1.2°-Lagen gebildet wird. Das weitere Vorgehen hingegen ist analog.

Der Vorteil dieses Verfahrens ist, daß es nur relativ wenige ($O(100)$) COT-Spuren gibt, verglichen mit den Zigtausenden von Kombinationsmöglichkeiten von Hits auf 1.2°-Lagen. Dadurch ist das Outside-in Tracking wesentlich schneller. Der Nachteil hingegen ist, daß nur Teilchen, die eine rekonstruierbare COT-Spur hinterlassen, so rekonstruierbar sind. Das stellt eine starke Einschränkung im Vorwärtsbereich ($|\eta| > 1$) dar.

3.3 Offline II – Vertexrekonstruktion

Der richtige Zeitpunkt zur Rekonstruktion von Vertices ist etwas kritisch. Zum einen möchte man die Vertexposition zur Unterdrückung von Kombinatorik bei der Spursuche verwenden;

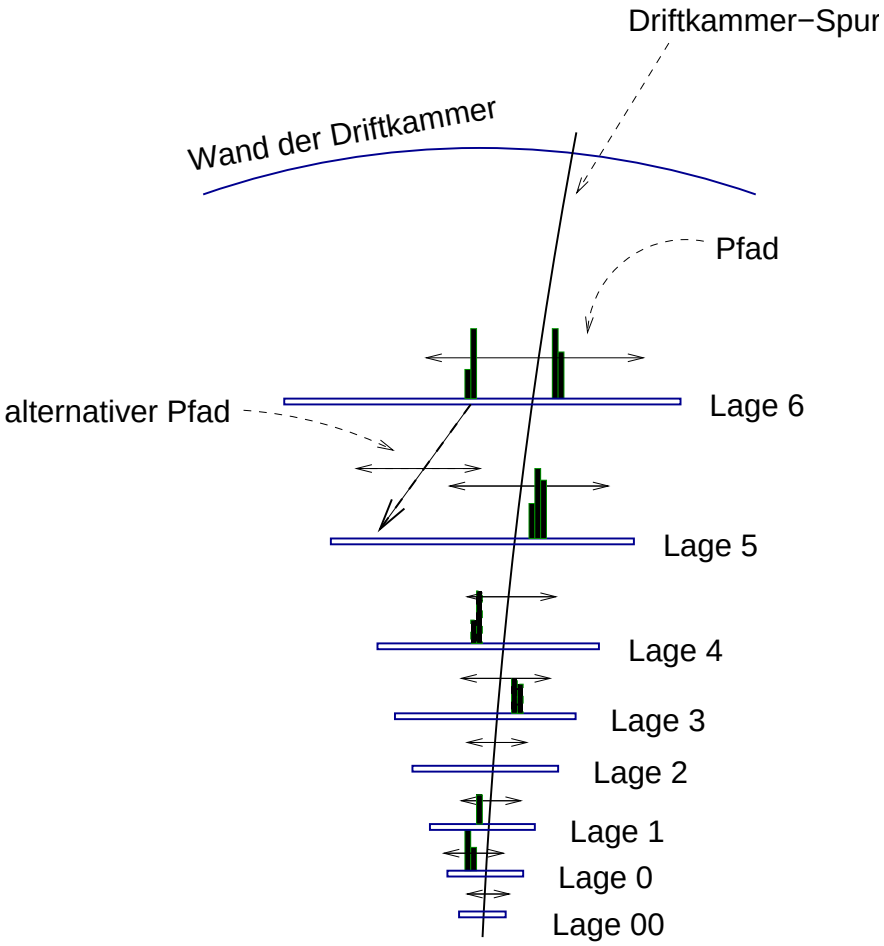

Abbildung 3.20: Outside-In Tracking-Schema.

zum anderen braucht man eben jene Spuren, um überhaupt einen Vertex fitten zu können. Die Lösung liegt in der Verwendung zweier unterschiedlicher Algorithmen: der erste läuft *vor* der Spurrekonstruktion, nutzt Hits statt Spuren und liefert eine Abschätzung der z-Position des Primärvertex'. Der zweite läuft *nach* der Spurrekonstruktion und fittet den Primär- (und gegebenenfalls auch Sekundär-) Vertex aus den rekonstruierten Spuren.

3.3.1 Schneller Z-Vertex-Sucher

Der schnelle Z-Vertex-Sucher `KalZ3DVertexFinder` nutzt Histogramme, um aus 3D-Hits (bestehend aus einer kombinierten φ- und 1.2°-Messung) die z-Position des Primärvertex zu bestimmen. Dazu wird nach folgendem Verfahren vorgegangen (siehe Abbildung 3.21):

1. Je ein 3D-Hit auf zwei verschiedenen Lagen und in einem φ-Fenster von $\pi/20$ (9°) werden kombiniert.

2. Auf den dazwischenliegenden 1.2°-Lagen wird in einem z-Fenster von 0.5 cm (Voreinstellung) ein dritter Hit gesucht.

3. Ohne passenden dritten Hit wird stattdessen der Strahlkreuzungspunkt verwendet.

4. Von der sich so ergebenden Helix werden Transversalimpuls, Impact Parameter sowie z_0, der Schnittpunkt mit der Strahlachse in r/z, berechnet.

5. Helices mit $|z_0| > 48.0$ cm (also außerhalb des Akzeptanzbereichs des Detektors) werden verworfen.

6. Die z_0 von Helices aus drei Hits mit einem Impact Parameter von unter 2 cm und einem Transversalimpuls von über 250 MeV kommen in das *High Quality Histogram*.

7. Die z_0 der übrigen Helices kommen in das *Low Quality Histogram*, sofern sie einen Transversalimpuls von über 500 MeV haben.

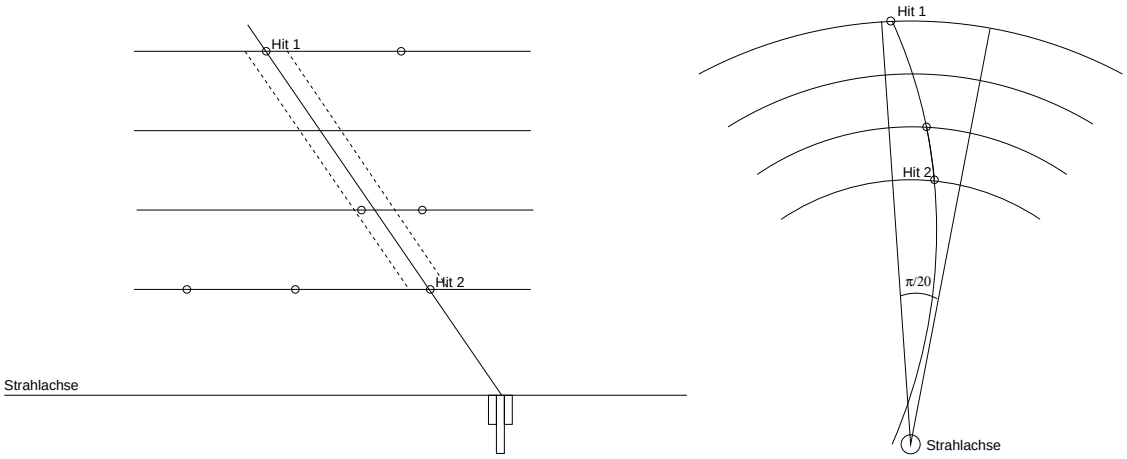

Abbildung 3.21: Histogrammfüllung des schnellen Z-Vertex-Suchers.

Die so gefüllten Histogramme (*High Quality* und *Low Quality*) werden mit drei verschiedenen Algorithmen auf Maxima durchsucht, wobei jedes Maximum mit einem Primärvertex identifiziert wird. Dabei wird zunächst von beiden Histogrammen eine im Fourierraum Tiefpaß-gefilterte Version errechnet. Dann wird wie folgt vorgegangen:

1. Alle Maxima über einem Schwellwert im gefilterten *High Quality Histogram* werden als Primärvertex der Güteklasse 3 (höchste Güteklasse) klassifiziert und aus allen Histogrammen entfernt.

2. Das Maximum des ungefilterten *Low Quality Histogram* wird, falls es über einem bestimmten Schwellwert liegt, als Primärvertex der Güteklasse 2 eingestuft und ebenfalls entfernt.

3. Im gefilterten *Low Quality Histogram* werden Peaks gesucht, die um einen Faktor f größer sind als die mittlere Bin-Belegung des Histogramms und dabei einen absoluten Schwellwert überschreiten. Diese werden als Primärvertices der Güteklasse 1 (niedrigste Güteklasse) eingestuft.

Die mit diesem Verfahren erziele Auflösung beträgt 1.0 mm. Der Algorithmus und die verschiedenen Optimierungen sowie Leistungsmerkmale werden ausführlich in [32] besprochen.

3.3.2 Primärvertex-Fitter

Als Primärvertexfitter kommt der bereits in Run I verwendete und in [33] beschriebene VxPrim zum Einsatz. Er wurde von Fortran nach C++ übersetzt und in das Run II-Framework eingepaßt [34]. Der Algorithmus beginnt, wie Abbildung 3.22 zeigt, mit der Auswahl geeigneter Spuren durch Mindestanforderungen an Transversalimpuls und Güte der Spur. Der eigentliche Fit wird iterativ verbessert, indem in jedem Durchgang die Spur, die den größten Beitrag zum χ^2 des Fits leistet, vor der nächsten Iteration entfernt wird, solange dieser Beitrag über einem bestimmten Schwellwert liegt. Der Fit wird fortgesetzt, bis entweder dieser Schwellwert unterschritten ist, oder nur noch die Mindestanzahl von Spuren übrig ist.

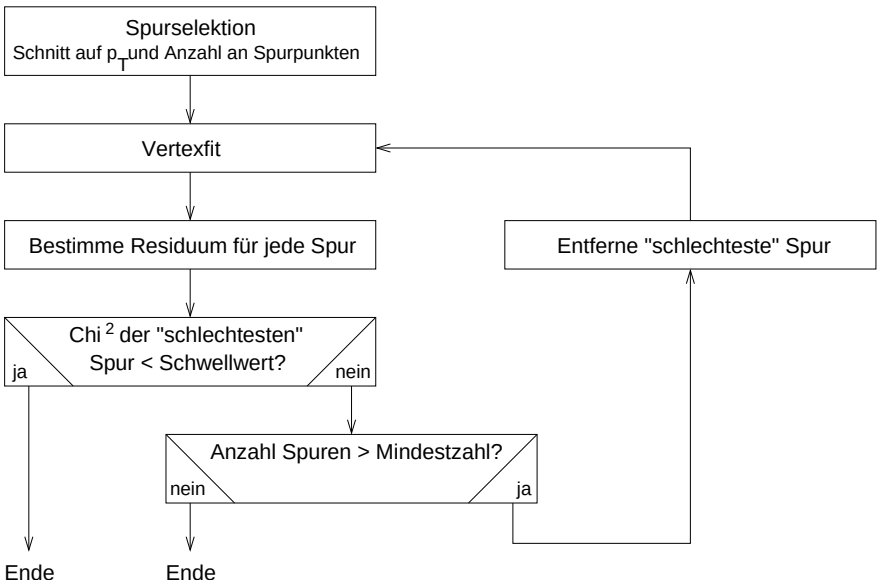

Abbildung 3.22: Ablaufschema des Primärvertex-Fitters VxPrim.

Kapitel 4

Der EKPplus Analysecluster

Zur Bewältigung der enormen Datenmengen des CDF II-Experiments wurde der EKPplus-Cluster konzipiert und aufgebaut. Es handelt sich dabei um einen Verbund von Linux-Rechnern, dessen Aufbau im folgenden beschrieben werden soll: zunächst wird ein allgemeiner Überblick über die Anforderungen gegeben, gefolgt von einer Aufstellung der Software, die auf dem Cluster laufen soll. Nach einer Betrachtung der Netzwerktopologie wird die Hardware und ihre Konfiguration vorgestellt.

4.1 Motivation und Anforderungen

Bei der Anschaffung von Rechenleistung bieten Linux-Cluster mit Abstand das beste Preis-/Leistungsverhältnis, vorausgesetzt, die zu lösenden Probleme lassen sich effizient parallelisieren. Bei den Analysen der Teilchenphysik ist dies der Fall, da entlang der Grenzen der einzelnen Ereignisse getrennt werden kann.

Ziele der EKPplus sind:

- eine leistungsstarke Umgebung für CDF-Analysen bereitzustellen
- die Entwicklung von Software für das CMS-Experiment[1] zu ermöglichen
- offen für Grid-Projekte zu sein[2]
- ein gutes Preis-/Leistungs-Verhältnis zu bieten
- auch für die anderen Experimente, an denen das EKP beteiligt ist (DELPHI, AMS), einen Einstieg zu ermöglichen

Dabei ist die Entscheidung zugunsten einen Linux-Clusters (und gegen eine proprietäre Lösung, etwa eine Mehrprozessor-Sun) dadurch gegeben, daß Linux auf PC-Hardware die einzige Schnittmenge von CDF-, CMS- und Grid-Software darstellt; die CDF-Software läuft ansonsten nur auf SGI-Hardware.

[1] Der *Compact Muon Solenoid* [35] ist einer der Detektoren am Large Hadron Collider; das Institut für Experimentelle Kernphysik ist an dessen Entwicklung von CMS beteiligt.

[2] Das Grid-Projekt wird in Kapitel 6.3 vorgestellt.

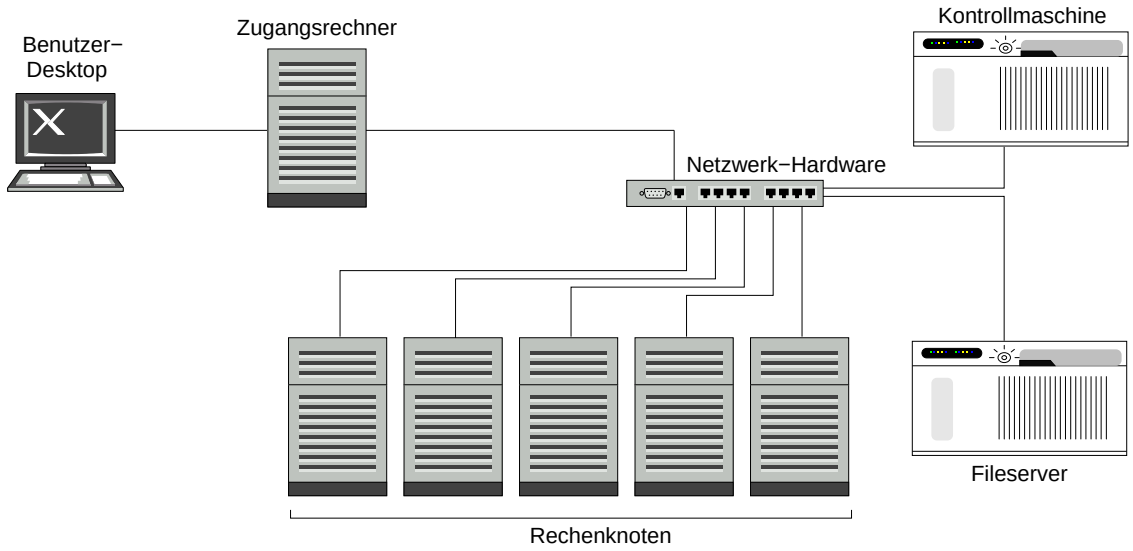

Abbildung 4.1: Grundsätzlicher Aufbau eines Linux-Clusters.

Allgemein besteht ein Linux-Cluster aus den in Abbildung 4.1 gezeigten Komponenten. Dabei bestimmen die oben aufgeführten Anforderungen zusammen mit dem Budget, wie die jeweilige Komponente ausgeführt wird. Im einzelnen:

- **Rechenknoten**
  stellen die Rechenleistung bereit; hier laufen die Analysejobs. Die Knoten können als Ein- oder Mehrprozessorsysteme ausgelegt sein. Oft verfügen sie über eine eigene Festplatte. Wenn genug Platz zur Verfügung steht, können auch einfache PCs als Knoten verwendet werden. Rechenknoten sind nicht für normalen, interaktiven Benutzerbetrieb gedacht.

- **Fileserver**
  stellen gemeinsam genutzte Daten und Platz für Ergebnisse bereit. Da sie alle Knoten zu bedienen haben, ist eine starke Netzwerkanbindung wichtig. Je nach Aufgabe des Clusters reicht ein mit zusätzlichen Platten ausgestatteter PC aus; in der Regel sind jedoch spezielle Gehäuse und Plattenkontroller nötig.

- **Zugangsrechner** ("Portale")
  ermöglichen den Zugang zu den Rechenknoten. Hier werden Jobs entwickelt und schließlich auf die Knoten abgeschickt, wobei meist ein *Batch Queueing*-System zum Einsatz kommt, das die Jobs möglichst effizient auf die Knoten verteilt. Da mehrere Benutzer gleichzeitig auf den Portalen interaktiv arbeiten, sind hier leistungsstarke (Mehrprozessor-) Systeme zu bevorzugen.

- **Kontrollmaschine**
  Sie steuert die Knoten – stellt das Betriebssystem bereit, verteilt Jobs – und übernimmt die Benutzerverwaltung inklusive der Loginverzeichnisse.

- **Netzwerk-Hardware**
  – zumeist Switches – verbindet die Komponenten des Clusters miteinander. Ihre Auslegung (und auch ihr Preis) hängt von den zu verarbeitenden Datenmengen ab und davon, ob die Knoten untereinander kommunizieren müssen oder nicht. Oft sind gemischte Gigabit/100 Mbit-Netze eine gute Lösung.

Bei kleinen, einfachen Clustern können die Funktionen von Zugangsrechner, Kontrollmaschine und Fileserver auch ganz oder teilweise kombiniert werden.

4.1.1 "Typische" Jobs

Um die Hardware-Anforderungen festzulegen, werden üblicherweise Tests mit "typischen" Jobs vorgenommen, anhand derer der Ressourcenverbrauch bestimmt wird. Unglücklicherweise gibt es in der Hochenergiephysik keinen "typischen" Job. Die Anforderungen unterscheiden sich ganz erheblich, je nachdem, welchen Teil der CDF-Analysekette man betrachtet:

Ressource	bei **MC Produktion**	bei **Analysen**
CPU 1 GHz Athlon	großer Bedarf, besonders die Detektorsimulation ist sehr CPU-intensiv: O(5 s/Ereignis)	kleiner bis mittlerer Bedarf, auch komplizierte Schnitte brauchen nur wenig CPU-Zeit: $\ll$ 0.01 s/Ereignis
I/O	vollständiges Ereignis wird abgespeichert: <500 kB/Ereignis (schreibend)	kleiner Teil des Ereignisses wird betrachtet: 100 kB/Ereignis (lesend)
Bandbreite	100 kB/s	10 MB/s

Diese Angaben machen klar, daß es soetwas wie eine typische benötigte Bandbreite nicht gibt. Auch die angegebenen Zahlen sind nur beispielhaft, und es ist leicht, sich eine Analyse vorzustellen, die mit weit weniger Bandbreite auskommt. Im schlimmsten Fall jedoch liest ein Filter-Job ein komplettes Ereignis ein, prüft ein einziges Bit, und verwirft das Ereignis (geschätzte benötigte Bandbreite: 10 GB/s; dies ist natürlich nicht realisierbar, nicht einmal im Arbeitsspeicher werden derartige Datenraten erzielt). Zu beachten ist, daß die genannten Werte nur eine Abschätzung der Größenordnung darstellen, da die Laufzeitangaben natürlich vom verwendeten Prozessor abhängen.

Für CMS ist der Kontrast noch stärker: für ein Monte Carlo-Ereignis (Größe: 1 MB) sind hier ca. 120 s CPU-Zeit nötig, was eine Bandbreite von lediglich 8 kB/s erfordert. Bei Analysen ist mit vergleichbaren Anforderungen zu rechnen wie bei CDF.

4.1.2 Komponenten und Gesamtsystem

Bei der Wahl der Komponenten ist darauf zu achten, daß deren Leistungsfähigkeit zusammenpaßt und keine Engpässe entstehen. Kenngrößen sind hierbei die Auslastung der CPU und die der Netzwerkkarte der Knoten. Dabei ist darauf zu achten, *beide* möglichst gut auszunutzen. Es ist nicht sinnvoll, mehr CPU-Leistung zu kaufen, als man über das Netzwerk

mit Daten versorgen kann; genausowenig, das Netzwerk überzudimensionieren, und dann nicht über eine CPU zu verfügen, die diese Datenmengen verarbeiten kann.

Aufgrund der unterschiedlichen Lastprofile ist es zwar unmöglich, einen optimalen Cluster zu entwerfen, denn die von der CPU benötigte Bandbreite schwankt ja zwischen 100 kB/s (mit BNC-verkabeltem Ethernet machbar) und 50 MB/s (Gigabit Ethernet unter Idealbedingungen). Dennoch sollte der suboptimale Cluster möglichst konsistent sein und keine weiteren Flaschenhälse aufweisen. So ist es nicht sinnvoll, einen Fileserver mit besonders schnellen Platten auszustatten, wenn er lediglich mit 100 Mbit-Ethernet angebunden werden soll. Das Gesamtbild muß stimmig sein.

Bei der EKPplus ist zu beachten, daß sich die Kommunikation nicht unter den Knoten, sondern zwischen den Knoten *und dem Fileserver* abspielt. Dadurch bündelt sich der Datenverkehr vor dem Fileserver: dessen Anbindung muß also stark genug sein, um mehrere Knoten gleichzeitig zu bedienen. Auch bei der Wahl der Festplatten und Kontroller ist dies berücksichtigt worden.

4.1.3 Zukunftssicherheit

Die CPU-Leistung verdoppelt sich nach Moore's Gesetz alle 18 Monate, das heißt, nach eineinhalb Jahren könnten halbsoviele aktuelle Knoten dasselbe leisten wie der ursprüngliche Cluster. Beim Aufbau des Clusters ist darauf zu achten, daß Neuerungen gut integriert werden können. Dabei wachsen jedoch nicht alle Komponenten gleich schnell: Plattenplatz wächst schneller als CPU-Leistung, die wiederum schneller wächst als Netzwerkbandbreite. Das bedeutet jedoch, daß sich mit jedem Update der in 4.1.2 gewählte Arbeitspunkt verschiebt.

Bei Erweiterungen müssen neben pekuniären Gesichtspunkten auch räumliche Beschränkungen in Betracht gezogen werden. So stellen Stellfläche und vor allem die Kälteleistung der Klimaanlage ein Limit dar, das jedoch absolut gesehen immer weniger wichtig wird: durch immer weiter verkleinerte Chip-Strukturen wächst der Energieverbrauch neuer CPUs langsamer als ihre Rechenleistung.

4.2 Software

Im folgenden wird diskutiert, welche Software auf dem EKPplus-Cluster zum Einsatz kommt, sowohl an Systemprogrammen als auch an spezieller Software für die Teilchenphysik.

4.2.1 Linux

Zum Einsatz auf den Portalen und den Knoten kommt eine vom Rechenzentrum des Fermilabs leicht modifizierte Version von Redhat Linux 7.1 mit dem Namen Fermi Linux 7.1.1. Die Modifikationen beschränken sich auf einige zusätzliche bzw. aktualisierte Softwarepakete und auf einen Satz vorgefertigter Konfigurationsprofile für verschiedene Einsatzgebiete im Rahmen des Fermilabs, zum Beispiel als CDF-Rechner, der nicht vor Ort steht (die *CDF-Offsite*-Konfiguration, die das EKP verwendet), oder als D0-Entwicklungsmaschine und dergleichen mehr. Diese Profile bestimmen den Umfang der installierten Software, das heißt

welche vorgefertigten Programmpakete eingespielt werden. Dabei ist zu beachten, daß eine als *CDF-Offsite* konfigurierte Maschine nach der Installation noch keinerlei CDF-spezifische Software enthält, der Bezeichnung verführt also zu voreiligem Optimismus.
Die wichtigsten Eckdaten der Fermi 7.1.1/Redhat 7.1 Linux-Distribution sind:

- Linux-Kernel 2.4.3
- GNU C-Bibliothek (glibc) 2.2.4
- GNU C/C++/F77 Compiler 2.96

Während die ersten beiden Komponenten eine gute Wahl darstellen (und der Kernel überdies leicht ausgetauscht werden kann), ist die Wahl der Compiler-Version umstritten. Es handelt sich dabei nämlich nicht um eine offizielle, von den Compilerentwicklern freigegebene Version, sondern um ein inoffizielles Zwischenrelease, das in erster Linie einen Meilenstein zur nächsten regulären Version darstellt[3]. Redhat verfolgt die sehr lobenswerte Politik, den Compiler (und einige andere zentrale Komponenten; der Kernel zählt nicht dazu) innerhalb einer Serie (7.x) nicht zu ändern, damit die Serie binärkompatibel bleibt. Dies ist sehr hilfreich für den Systemverwalter, da so das Aktualisieren vieler Rechner mit verschiedenen Versionen (7.1, 7.2) einer Serie wesentlich vereinfacht wird, und sei anderen Distributoren sehr zur Nachahmung empfohlen. In diesem Fall jedoch hat sich Redhat damit für über eineinhalb Jahre auf einen kaum getesteten Compiler festgelegt.

Dieser Compiler machte dann auch die Umstellung des CMS-Portals von Fermi Linux 7.1.1 auf SuSE Linux 7.2 nötig. Der Hintergrund dazu wird in Kapitel 4.2.3 erläutert. SuSE 7.2 setzt standardmäßig Kernel 2.4.4, die GNU C-Bibliothek (glibc) 2.2.2 und den GNU Compiler 2.95.3 ein.

Aufgrund der verwendeten zum Teil recht neuen Hardware (Platten-Kontroller und Gigabit Ethernet-Karten) wurde der zum damaligen Zeitpunkt aktuelle Kernel 2.4.16 und, wo notwendig, Patches für die Gigabit-Netzwerkkarten installiert. Die Portale liefen über zwei Monate lang störungsfrei, bis sie ein Stromausfall ausschaltete.

4.2.2 CDF Software

Die CDF-Software unterteilt sich in folgende Gruppen: die allgemeine Fermi-Softwareumgebung, Fremdprodukte, sowie die Software der in Kapitel 3 beschriebenen CDF-Analysekette. Die Fermi-Softwareumgebung (*Fermi Unix Environment*, FUE) besteht im wesentlichen aus einem eigenen Paketmanagement, dem unten beschrieben UPS. An Fremdsoftware kommt neben dem der ROOT-Software [9] und einigen anderen Programmen vor allem der C++-Compiler von Kuck & Associated, Inc. (KAI) zum Einsatz.

Die CDF-spezifische Software wird bei der Installation von Fermi Linux nicht mitinstalliert, sondern muß mit dem Fermi-eigenen Paketmanagementwerkzeug UPS/UPD nachinstalliert werden.

[3]Die reguläre Version zum Zeitpunkt der Entscheidung war 2.95.3.

Das Paketsystem UPS/UPD

Das am Fermilab entwickelte und verwendete Paketverwaltungssystem UPS/UPD ("Unix Product Support/Unix Product Distribution") hat gegenüber dem von Redhat (und vielen anderen Linux-Distributionen) verwendeten RPM-Paketformat einen entscheidenden Vorteil: es lassen sich mehrere Versionen eines Programmpakets gleichzeitig installieren, und der Benutzer kann mit einem einfachen Befehl (`setup emacs 19.3`) die gewünschte Version auswählen. Das Teilprogramm UPD dient dazu, vorcompilierte Produkte vom zentralen Softwareserver am Fermilab herunterzuladen und zu installieren.

Intern handelt es sich bei UPS und UPD um Konglomerate aus Shell-Skripten, die auf einer Textdatei – der Produktdatenbank – operieren und verschiedene Environment-Variablen passend zu den gewünschten Produkten aufsetzen, wie beispielsweise den `PATH` oder den `LD_LIBRARY_PATH`. Das oben erwähnte `setup`-Skript etwa entfernt eventuell vorher aufgesetzte andere Versionen des gewünschten Produkts aus dem Environment der Shell und setzt dieses für das Produkt auf. Der Nachteil daran, daß es sich bei UPS/UPD um Shell-Skripte handelt, ist, daß von allen Skripten unterschiedliche Versionen für die bash-Shell und die (t)csh-Shell gewartet werden müssen[4].

Die beiden folgenden Fremdprodukte sind für die CDF-Software von besonderer Bedeutung:

Der KAI C++ Compiler

In der Anfangsphase der Entwicklung der CDF-Software wurde die Entscheidung getroffen, den C++-Compiler von Kuck & Associated, Inc. (KAI) zu verwenden. Dieser Compiler steht auf allen bei CDF verwendeten Rechnerplattformen[5] zur Verfügung und unterstützt den vollen ANSI-C++-Standard. Die damals zur Verfügung stehenden freien Compiler, GNU C++ 2.7.2 und EGCS 1.0.2, hatten einige Schwachstellen vor allem bei der Unterstützung von C++-Templates, waren jedoch durchaus alltagstauglich[6]. Die Entscheidung zugunsten von KAI C++ wurde vor allem dadurch begründet, daß die freien Compiler einige der komplexeren Features von C++[7], die der CDF-Code ausgiebig nutzt, nicht unterstützte. Die damaligen Autoren der CDF-Software waren nicht bereit, den Code zu vereinfachen; so wurde der KAI-Compiler gekauft.

Heute sind starke Bemühungen im Gange, den Code wieder mit dem aktuellen GNU C++-Compiler zum laufen zu bringen, da KAI von Intel aufgekauft wurde und der KAI C++-Compiler nicht weiter gepflegt wird bzw. als Teil des Intel-C++-Compilers nur noch auf PC-Hardware entwickelt wird. Ein weiterer Nachteil des KAI-Compilers ist, daß die Fehlersuche nur mit dem teuren TotalView-Debugger effizient möglich ist.

[4] Seit Anfang 2002 sind kollaborationsinterne Bemühungen im Gange, von der (t)csh loszukommen, da diese zahlreiche Nachteile gegenüber der Bourne-Shell-Familie (sh, bash) hat [36], unter anderem ein Limit von 1024 Zeichen für die Länge des `PATH`; dieses Limit wird in der CDF-Umgebung gelegentlich erreicht.

[5] heute: Linux auf Intel- oder AMD-PCs, und die IRIX-Rechner von Silicon Graphics, Inc.

[6] Die erste Version der weitverbreitete Unix-Benutzeroberfläche KDE wurde mit diesen Compilern entwickelt.

[7] Bei CDF vor allem geschachtelte Templates.

Oracle

Zur Speicherung einiger Kalibrationskonstanten und für das Fehlerlog der Monitoring-Programme verwendet das CDF-Experiment eine Oracle-Datenbank, wie bereits in Kapitel 3.1.1 erwähnt. Demzufolge müssen fast alle selbstentwickelten CDF-Programme (Monte Carlo-Generatoren, Rekonstruktionssoftware) in der Lage sein, Datenbankzugriffe zu tätigen, was die Notwendigkeit der in Kapitel 4.3.3 beschriebenen Netzstruktur zur Folge hat.

4.2.3 CMS Software

Die CMS-spezifische Software ist mindestens so komplex wie die des CDF-Experiments. Auch sie beinhaltet einen speziellen C++-Compiler und eine Datenbank, allerdings kein Paketmanagementsystem. Auch wird hier die Übersetzung der eigenen Software nicht wie bei CDF über Makefiles, sondern über Shellskripte erledigt.

Objectivity

Die LHC-Experimente CMS und ATLAS verwenden die objekt-orientierte Datenbank Objectivity des gleichnamigen Herstellers zur Speicherung der Ereignisse[8]. Die verwendete Objectivity-Version ist eigentlich nur für Redhat 6.2 zertifiziert und läuft nicht auf Redhat 7.1 oder Fermi Linux 7.1.1. Daher war es nötig, das CMS-Portal von Fermi Linux 7.1.1 auf SuSE 7.2 umzustellen, was den Cluster uneinheitlich und schwerer zu warten macht. Das zwei Jahre alte Redhat 6.2 kann aufgrund der verwendeten modernen Hardware nicht eingesetzt werden; die Tatsache, daß die CMS-Software auf SuSE 7.2 läuft, ist als glücklicher Zufall zu werten.

Objectivity ist außerdem die Ursache dafür, daß die CMS-Software nur mit einer speziellen, angepaßten Version von GNU C++ 2.95.2 übersetzt werden kann, da die Objectivity-Bibliotheken sonst nicht korrekt eingebunden werden können[9].

4.2.4 Jobverwaltung mit OpenPBS

Aus der Sicht der Benutzer stellt der Cluster zunächst einmal einen Pool an Rechenzeit dar. Ein Programm hat einen gewissen Rechenzeitbedarf, üblicherweise in CPU-Minuten oder CPU-Stunden gemessen[10]. Häufig wird jedoch der Bedarf an Rechenzeit größer als

[8] Noch. Ende 2002 soll die Software von Objectivity auf eine CERN-Lösung – ROOT-basiert – umgestellt werden.

[9] Die sogenannte ABI (*Application Binary Interface*) von C++ legt fest, wie aus einer Funktionssignatur ein Linkersymbol generiert wird. Bei C ist das trivial, es wird einfach der Funktionsname selbst verwendet; bei Fortran wird ein Unterstrich angehängt. C++ unterstützt jedoch das Überladen von Funktionen. Dadurch reicht der Funktionsname allein nicht mehr aus, um eine eindeutige Zuordnung zu erreichen; die Typen der Parameter müssen in das Linkersymbol mit eingehen, ebenso wie der Klassenname, falls es sich bei der Funktion um eine Methode handelt. Wie dies zu geschehen hat, legt das ABI fest. Allerdings befindet sich das ABI derzeit noch in Entwicklung und ändert sich mit beinahe jeder *Major Release* des GNU C++-Compilers.

[10] Die Einheit CPU-Minuten bzw. -Stunden ist rechnerabhängig: ein Programm, daß auf einem alten Pentium eine Stunde benötigte, wird auf einem modernen Athlon nur noch wenige Minuten brauchen. Dennoch sind CPU-Minute und -Stunde die gebräuchlisten Einheiten für Rechen-"Arbeit", da sie die tatsächliche Ablaufzeit angeben, die für Benutzer und Umgebung entscheidend sind.

das Angebot sein, und die Ressource muß "gerecht" verteilt werden. Dies ist Aufgabe des Batch-Queueing-Systems.

Will ein Benutzer ein Programm laufenlassen, übermittelt er diesen Wunsch in Form eines Submittierungskommandos an die Batch-Queueing-Software. Diese reiht den Job in eine Queue, eine Liste aller Anfragen, ein. Der Scheduler des Batch-Queueing-Systems prüft in gewissen Zeitabständen, ob Rechenknoten frei sind, und füllt diese mit den Jobs aus den Queues. Dabei greift der Scheduler nach folgenden Kriterien in die Sortierung der Queues ein:

- Reihenfolge der Submittierung. "Wer zuerst kommt, mahlt zuerst."
- Angemeldeter CPU-Bedarf des Jobs. Kurze Jobs werden vorrangig behandelt.
- eventuell CPU-Bedarf der Benutzer in den letzten Wochen.

Ein kurzer Job K brauche 30 CPU-Minuten, ein gleichzeitig submittierter langer Job L brauche 5 CPU-Stunden. Nach 5:30 Stunden liegen also beide Ergebnisse vor; der zuerst bearbeite Job hat eine Laufzeitverlängerung von 0%. Wird K zuerst bearbeitet, so liegt das Ergebnis von L nach 5:30 Stunden mit einer effektive Laufzeitverlängerung von 10% vor. Wird hingegen L zuerst abgearbeitet, so liegt das Ergebnis von K ebenfalls nach 5:30 Stunden vor, jedoch beträgt die vom "benachteiligten" Benutzer beobachtete Laufzeitverlängerung 1000%! Ziel des Schedulers muß also sein, das Verhältnis von Wartezeit zu CPU-Zeit global zu minimieren.

In der Regel ist es nicht möglich, die Laufzeit eines Programmes vorab genau abzuschätzen. Der EKPplus-Cluster arbeitet mit einer Laufzeit-Staffelung in drei Queues: die *short*-Queue, in der Programme bis maximal eine Stunde CPU-Zeit laufen, die *medium*-Queue mit acht Stunden Laufzeit, sowie die *long*-Queue mit maximal 120 Stunden. Dazu kommt die *io_only*-Queue für Ein-/Ausgabe-intensive Jobs mit ebenfalls 120 Stunden. Ist ein Job sehr viel früher fertig als sein Zeitlimit, so ist sein Verhältnis von Warte- zu CPU-Zeit unnötig schlecht. Überschreitet ein Job sein Zeitlimit, so wird er abgebrochen. Es ist also im Interesse der Benutzer, Jobs in die "richtige" Queue zu submittieren.

Außerdem gibt es noch eine Queue für Ein-/Ausgabe-limitierte Jobs, also Programme, die den größten Teil ihrer Laufzeit Daten lesen oder schreiben und nur verhältnismäßig wenig rechnen. Die Einführung einer speziellen Queue für I/O-intensive Jobs begründet sich darin, daß die I/O-Leistung und die CPU-Leistung quasi orthogonale Ressourcen sind. Das Einlesen oder Schreiben von Daten, sei es nun von bzw. auf Festplatte oder über Netzwerk, ist so langsam, daß der Prozessor dabei kaum belastet wird. Daher ist es sinnvoll, einen I/O- und (mindestens) einen CPU-intensiven Job gleichzeitig ablaufen zu lassen, um beide Ressourcen möglichst gut auszunutzen. Dies wird durch das Einrichten einer speziellen Queue ermöglicht. Der Scheduler sorgt nun dafür, daß auf jedem Knoten zusätzlich zu CPU-intensiven Jobs auch ein I/O-intensiver Job läuft.

OpenPBS

Der EKPplus-Cluster verwendet als Batch-Queueing-System OpenPBS[11] von Veridian Systems [37]. OpenPBS bietet folgende Vorzüge:

- Plattformunabhängigkeit. OpenPBS läuft auf einer Vielzahl von Unix-Systemen, darunter Linux.
- Verfügbarkeit des Quellcodes. Zwar ist OpenPBS keine Open Source Software oder Freie Software[12], doch können lokale Anpassungen (etwa am Scheduler) vorgenommen werden.
- Kostenlos für Universitäten. Für die Industrie gibt es die Version PBSPro, für welche auch Support vom Hersteller erhältlich ist.
- Grid-Anbindung. Dies ist wichtig, da der EKPplus-Cluster später in das weltweite Rechner-Grid eingebunden werden soll.

Eingabe- und Ergebnisdateien

Die Jobs, die auf den Knoten der EKPplus laufen, lassen sich nach ihren Bedürfnissen hinsichtlich Daten-Ein-/Ausgabe unterscheiden:

- Schwerpunkt **Schreiben (einmalig).** Hierunter fallen zum Beispiel die Monto Carlo-Produktion von Delphi, bei der lineare Dateien geschrieben werden.
- Schwerpunkt **Schreiben (mehrfach).** Auf den ersten Blick scheint es widersinnig, eine Ergebnisdatei mehrfach zu schreiben. Bei objektorientierten Dateiformaten (wie das von CDF verwendete ROOT-Dateiformat) ist dies jedoch nötig, da Kontrollstrukturen und das "Inhaltsverzeichnis" der Datei nach jedem gespeicherten Ereignis aktualisiert werden müssen.
- Schwerpunkt **Lesen (einmalig).** In diese Rubrik fallen beispielsweise Filterprogramme, die Schnitte auf Datensätze anwenden und DSTs erzeugen, und generell Analysejobs.
- Schwerpunkt **Lesen (mehrfach).** Programme, die mehrfach über einen Datensatz iterieren.

Bei einmaligem Lesen oder Schreiben ist es das einfachste, die Daten direkt vom jeweiligen Netzwerklaufwerk zu lesen bzw. darauf zu schreiben. Bei wiederholtem Zugriff kann es unter Umständen sinnvoll sein, die Daten nur einmal vom bzw. zum Fileserver zu übertragen und ansonsten mit einer lokalen, auf der Festplatte des Rechenknoten gespeicherten Kopie zu

[11] *P*ortable *B*atch *S*ystem.

[12] Die gängige Definition von Freier Software verlangt, daß modifizierte Versionen weitervertrieben werden dürfen (ja sogar müssen, außer bei Inhouse-Produkten), was bei OpenPBS nicht der Fall ist. Die genaue Definition von Freier Software ist bei der Free Software Foundation [38] einsehbar.

arbeiten. OpenPBS unterstützt dieses "Staging", also das Verwalten der lokalen Kopie. Der Benutzer kann angeben, welche Dateien der Job liest und schreibt. Bevor der Job dann auf dem Knoten gestartet wird, kopiert OpenPBS die benötigten Eingabedateien auf die schnelle lokale Platte des Knotens. Nachdem der Job beendet ist, werden die erzeugten Ergebnisdateien an ihren angegebenen Zielort übertragen. Anschließend werden die nun nicht mehr benötigten lokalen Kopien gelöscht.

4.2.5 Überwachungssoftware Fleximon

Für ein großes System wie die EKPplus ist es angeraten, Kenndaten wie Temperatur der Prozessoren, Raumtemperatur und Funktionstüchtigkeit der verschiedenen Lüfter der Rechner ständig zu überwachen. Zu diesem Zweck wurde vom Autor die Fleximon-Software entwickelt, ein Programmpaket zur netzwerktransparenten Überwachung beliebiger Betriebsparameter [39]. Dabei verwendet Fleximon eine Client-Server-Architektur zur Fernabfrage von Meßwerten, und einen Plugin-Mechanismus zur Realisierung der eigentlichen Meßvorgänge (siehe Abbildung 4.2).

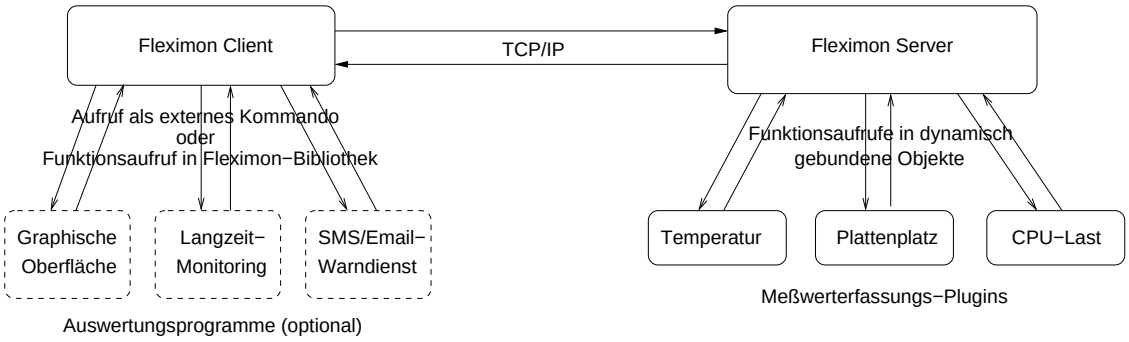

Abbildung 4.2: Architektur des Fleximon-Frameworks.

Jedes Plugin stellt genau ein Kommando zur Verfügung, welches vom Client aus ausgeführt werden kann. Der Client nimmt die Ausgabe des Plugins entgegen und stellt sie dem Benutzer dar, hier etwa die CPU-Last der Knoten 1 und 2:

```
% fleximon -i 192.168.101.1 "load"
2.92
% fleximon -i 192.168.101.2 "load"
1.01
%
```

Manche Kommandos benötigen weitere Parameter, etwa der `PROCRUN`-Befehl, der den Namen des zu überprüfenden Prozesses erwartet:

```
% fleximon -i 192.168.101.1 "procrun /usr/sbin/pbs_mom"
root   522  0.0  0.1  1476  660 ?     S   07:57  0:00  /usr/sbin/pbs_mom
%
```

Der Kommando-Name dient zur Erkennung des richtigen Plugins; weitere Parameter werden an das Plugin übergeben.

Aufbau des Fleximon-Pakets

Das Fleximon-Programmpaket besteht im wesentlichen aus vier Teilen:

1. **libfleximon** – diese Bibliothek stellt die grundlegenden Funktionen zur Verfügung: Netzwerkverbindungen für Client, Server und den unten erklärten Proxy, sowie den Mechanismus zum Laden der Plugins.

2. **fleximon** – der Kommandozeilen-Client. Hiermit ist die Abfrage des Servers möglich, wie oben im Text bereits demonstriert. Der Client kann auch aus Shellskripten oder anderen Programmen heraus aufgerufen werden; alternativ kann der Entwickler beispielsweise einer graphischen Oberfläche auch direkt auf die Client-C-Funktionen der libfleximon-Bibliothek zurückgreifen.

3. **fleximond** – der Server und Proxy. Dieses Programm muß auf dem zu überwachenden Rechner laufen, um Anfragen nach Meßwerten entgegenzunehmen, sie eventuell –als Proxy– an einen anderen Rechner weiterzuleiten, oder den Plugins zu übermitteln.

4. **Plugins** – sie dienen zum Erfassen der verschiedenen Kenngrößen. Standardmäßig stehen folgende Plugins zur Verfügung:

   - `load`: die derzeitige CPU-Last.
   - `procrun`: überprüft, ob der angegebene Prozess läuft.
   - `diskfree`: freier Festplattenplatz der lokalen Festplatten.
   - `sensors`: Temperaturen (CPU, Gehäuse) und Lüfterdrehzahlen. Details zu Konfiguration und Auslese der Motherboard-Sensoren finden sich in [40].

Ein Beispiel-Plugin

Die Entwicklung eigener Plugins ist sehr einfach. Ein Plugin besteht aus einer C-Funktion, die eine Zeichenkette entgegennimmt und gegebenenfalls auswertet, und das Ergebnis seiner Messung wiederum als Zeichenkette zurückgibt. Dazu muß lediglich noch ein C-Makro ausgefüllt werden, in dem das Kommando (als Zeichenkette, zum Beispiel `LOAD`), eine Kurzbeschreibung, sowie die eigentliche C-Funktion aufgeführt sind. Der C-Compiler erzeugt dann die Kontrollstrukturen, die zum Einbinden in das Fleximon-Framework notwendig sind.

Im folgenden wird ein Plugin vorgestellt, das die Größe von Dateien (etwa Log-Dateien) überwacht. Das Kommando heiße `SIZE`, als Parameter wird der vollständige Name der Datei erwartet. Das Plugin liefert die Größe der Datei in Bytes zurück. Dazu benutzt es die C-Funktionen `fseek()` und `ftell()`. Mittels `fseek()` wird zunächst das Ende der Datei angesprungen, dann wird mit `ftell()` die momentane Position innerhalb der Datei (in Bytes) abgefragt; dies ist die Größe der Datei.

```
#include <stdio.h>
#include <string.h>

#include "libfleximon.h"  /* Einbinden der Fleximon-Bibliothek */

static char *respond_size (const char *query)
{
   static char size_buf [128];  /* Platz fuer das Ergebnis */
   FILE *file;
   long int size;
   file = fopen(query,"rt");  /* Versuch, die Datei zu oeffnen */
   if (!file) return "-1";  /* -1 falls Datei nicht da/lesbar */
   fseek(file,0,SEEK_END);
   size = ftell(file);
   sprintf(size_buf,"%li",size);  /* long integer -> Zeichenkette */
   fclose(file);
   return size_buf;  /* Zeichenkette zurueckgeben */
}

/* Das FM_PLUGIN Makro erzeugt die noetige Struktur fuer den Lader */
FM_PLUGIN("SIZE","File size in bytes",respond_size)
```

Der Plugin-Lader

Der programmiertechnisch interessanteste Teil ist sicherlich das `FM_PLUGIN`-Makro. Dieses Makro erzeugt eine Funktion mit dem für jedes Plugin gleichen Namen `plugin_info`, in der alle nötigen Angaben zum neuen Plugin in eine dafür vorgesehene Datenstruktur gefüllt werden: Kommando (`SIZE`), Kurzbeschreibung ("File size in bytes") sowie die Funktion selbst (`respond_size`).

Da nun diese Funktion bei allen Plugins gleich heißt, geht der Plugin-Lader folgendermaßen vor: er lädt das Plugin als dynamisches Objekt mittels des Systemaufrufs `dlopen()` und fordert danach mittels `dlsym()` einen Zeiger auf die Funktion `plugin_info` an. Sollte dies scheitern, so ist das vermeintliche Plugin unzulässig und wird wieder entfernt. Ist der Zeiger gültig, so wird die Funktion aufgerufen, und der Plugin-Lader kommt auf diesem Weg an die relevanten Daten des Plugins, die in die globale Liste aller Kommandos eingetragen werden. Damit ist das neue Kommando, welches das Plugin implementiert, verfügbar. Das Elegante an dieser Vorgehensweise ist, daß es für den Plugin-Programmierer einen minimalen Aufwand darstellt, da die meiste Arbeit vom Präprozessor beim Expandieren des Makros erledigt wird. Es gibt bereits Plugins von anderen Autoren [41].

Proxy-Fähigkeit

Die in Abbildung 4.2 aufgezeichnete Architektur ermöglicht lediglich den Einsatz innerhalb des Clusters. Ein Zugriff von außerhalb, etwa zur graphischen Darstellung der Temperatur-

und Lastverteilung im WWW, ist so nicht möglich, da die Fleximon-Server auf den Rechenknoten aufgrund der Netzwerkarchitektur nicht von außerhalb der EKPplus zugänglich sind. Der Fleximon-Server kann jedoch auch im sogenannten Proxy-Modus (auch Forwarding-Modus genannt) arbeiten; hier wirkt er als Vermittler zwischen dem Client, der beispielsweise auf dem Web-Server läuft, und den eigentlichen Servern auf den Rechenknoten. Abbildung 4.3 verdeutlicht das Geschehen.

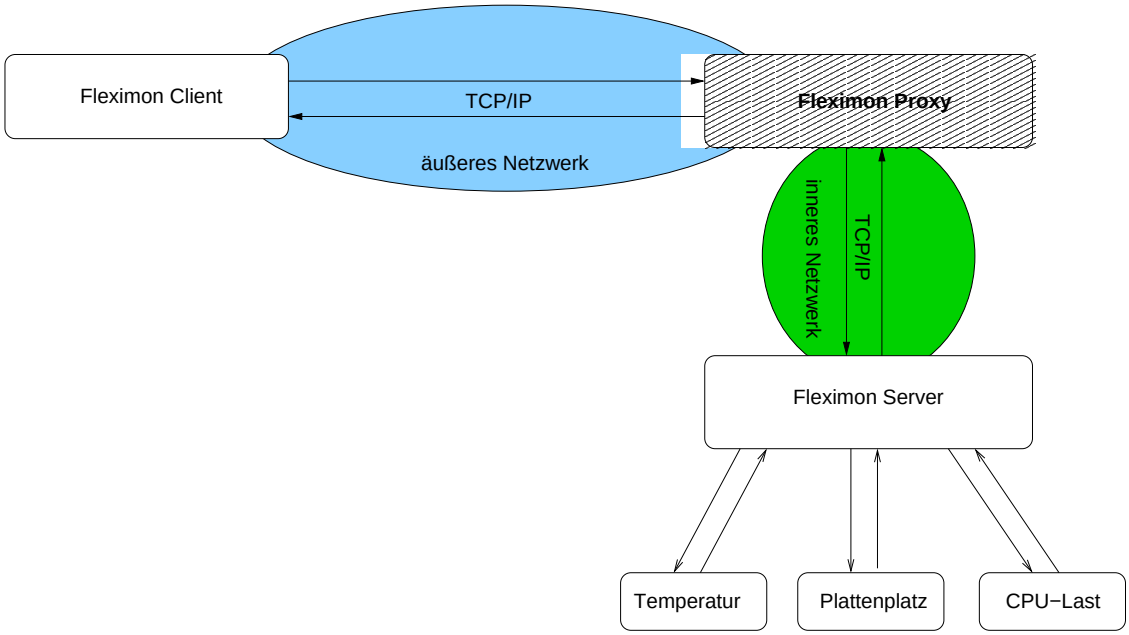

Abbildung 4.3: Fleximon-Architektur mit Proxy-Funktion.

Wird beim Start des Fleximon-Daemons die Forwarding-Fähigkeit aktiviert und somit der Daemon als Proxy gestartet (dies geschieht durch die Option `-f`), dann darf die an den Proxy geschickte Anfrage eine Ziel-IP-Adresse sowie einen Ziel-Port beinhalten, an den dann die eigentliche Anfrage weitergeleitet wird. Um beispielsweise von Institutswebserver aus die Temperatur und den Lüfterstatus des Knotens 1 abzufragen, ist folgende Anfrage (vom Webserver aus) nötig:

```
% fleximon -i 129.13.102.42 "192.168.101.1 sensors"
Mobo fan: 0
CPU fan: 5000
Mobo temperature: 28
CPU temperature: 45
%
```

Der Client schickt nun die Anfrage (der Text in Anführungszeichen) an den Proxy, der auf dem CDF-Portal (129.13.102.42) läuft. Dieser erkennt, daß der erste Teil der Anfrage selbst wieder eine IP-Adresse ist (nämlich 192.168.101.1), und leitet den Rest der Anfrage an eben diese IP-Adresse weiter. Sollte eine Anfrage keine IP-Adresse beinhalten, so bezieht

der Proxy die Anfrage auf sich und reagiert wie im normalen Server-Modus, indem er eben *seine* Temperatur und Lüfterdrehzahlen zurückliefert.

4.3 Netzwerkarchitektur

Bevor die physikalische und logische Netzwerkarchitektur der EKPplus vorhgestellt wird, werden hier zunächst einige Grundlagen der Rechnervernetzung mittels des IP-Protokolls erleutert. Der in diesen Dingen kundige Leser mag die folgenden Kapitel überspringen und gleich bei 4.3.4 weiterlesen.

4.3.1 Netzwerk-Grundlagen

Das wichtigste Netzwerkprotokoll heutzutage ist sicherlich IP, das "Internet Protocol". Jeder Rechner, der an ein IP-Netzwerk angeschlossen ist, hat eine IP-Adresse, die, soll der Rechner von "außen" erreichbar sein, weltweit eindeutig sein muß[13]. Der Adressraum ist 32 bit groß, umfaßt also rund 4 Milliarden Adressen. Die Adressen werden üblicherweise in der sogenannten *dotted decimal notation* dargestellt, also als vier Bytes (Werte von 0 bis 255), getrennt durch Punkte. Der Webserver der Universität Karlsruhe beispielsweise ist 129.13.64.33. In der Regel werden jedoch statt der IP-Adressen Namen verwendet, an die man sich leichter erinnern kann und die in der Regel eindeutig auf eine Adresse abgebildet werden[14].

Der IP-Adressraum ist zunächst einmal linear, was nicht der realen Struktur des Internet entspricht. Dieses ist hierarchisch in voneinander unabhängige Netze untergliedert. Anders ausgedrückt, ist das Internet ein Meta-Netzwerk, ein Netz aus Netzen. Um die Untergliederung zu ermöglichen, wird der lineare Adressraum in sogenannte Subnetze strukturiert. Ein Subnetz wird definiert durch seine Netzwerkadresse und Netzmaske. Die Netzwerkadresse ist der Teil, der allen Rechnern des Subnetzes gemein ist, um Nullen ergänzt. So beginnen etwa alle Adressen der Univ. Karlsruhe mit 129.13; die Netzwerkadresse des Uninetzes ist demnach 129.13.0.0. Um festzustellen, ob ein Rechner zu einem Subnetz gehört, wird seine IP-Adresse mit der Netzmaske des Subnetzes binär *und*-verknüpft. Wenn der Rechner im Subnetz ist, dann stimmt das Ergebnis mit der Netzwerkadresse des Subnetzes überein. Die Netzmaske gibt also an, welche Teile der Netzwerkadresse signifikant sind und welche Nullen einfach nur aufgefüllt sind. Das Uninetz hat demnach als Netzmaske 255.255.0.0 [15]. Oft wird auch

[13]Muß ein (oder mehrere) Rechner nicht von außen erreichbar sein, so stehen hier verschiedene sogenannte private IP-Adressbereiche zur Verfügung. Rechner mit privaten IP-Adressen haben keine direkte Verbindung mit dem Internet; dadurch können diese Adressen problemlos mehrfach vergeben werden. Verbindungen zur Außenwelt sind erst über den in Kapitel 4.3.3 erläuterten Trick möglich. Die EKPplus verwendet solche privaten Adressen (192.168.x.y) in ihrem inneren Netz.

[14]Diese Abbildung wird übernommen vom Domain Name Service (DNS), einer hierarchischen, verteilten Namensdatenbank.

[15]Ein etwas komplizierteres Beispiel wäre die Netzmaske des EKP-Netzes. Das EKP-Netz beinhaltet die Adressen 129.13.102.1 bis 129.13.102.63. Der gemeinsame Teil dieser Adressen und damit die Netzwerkadresse ist 129.13.102.0. Doch das ist nur ein Teil der Wahrheit, denn die naheliegende Netzmaske 255.255.255.0 ist falsch. Sie würde nämlich auch beispielsweise den Rechner 129.13.102.240 (eine Maschine des Rechenzentrums!) ins EKP-Netz einordnen. Die Netzmaske muß also auch berücksichtigen, daß die letzte Zahl der IP-Adresse kleiner oder gleich 63 sein muss. In Binärdarstellung ist $63_{dez} = 00111111_{bin}$. Die ersten beiden

die kombinierte Netzwerkadresse angegeben in der Form Netzwerkadresse/Netzmaske, für die Universität Karlsruhe also 129.13.0.0/255.255.0.0 oder 129.13.0.0/16. Dabei sagt die /16 aus, daß die oberen 16 bit der Netzwerkadresse, also 129.13, relevant sind[16]. /8-Netzwerke nennt man auch *Class A*-Netze (rund 16 Mio. Adressen), /16-Netzwerke werden als *Class B* (gut 64000 Adressen) bezeichnet; bei /24-Netzwerken spricht man von *Class C*-Netzen (254 Adressen).

Auf dem IP-Adressierungsmodell aufsetzend gibt es verbindungsorientierte (*Transmission Control Protocol*, TCP) und verbindungslose (*User Datagram Protocol*, UDP) Möglichkeiten der Datenübertragung[17]. Bei TCP-Übertragungen sorgt der Netzwerktreiber (*TCP/IP-Stack*) des Betriebssystems dafür, daß die Datenpakete in der richtigen Reihenfolge ankommen; auch wird anhand von Seriennummern im Paket-Header geprüft, ob Pakete verloren gegangen sind. Die Protokolle vom Web (HTTP), von Mail (SMTP) und FTP arbeiten mit TCP-Datenströmen. UDP hingegen prüft nicht, ob Pakete verlorengegangen sind; dafür sind hier die Latenzzeiten geringer, denn wenn ein Paket verloren gegangen ist, muß nicht lange gewartet werden, ob es nun doch noch eintrifft oder sein Nachfolgepaket. Daher wird UDP gerne für IP-Telephonie und -Videokonferenzen eingesetzt, wo es verschmerzt werden kann, wenn ein Teilbild kurz nicht aktualisiert wird oder 0.1 s Ton fehlen, hingegen lange Latenzzeiten sehr störend wären.

Ein Rechner kann sowohl mehrere Services gleichzeitig anbieten (etwa ein kombinierter Mail- und Webserver) als auch in Anspruch nehmen (zum Beispiel Daten vom CERN herkopieren und eine Datenbank am Fermilab abfragen). Damit dies mit nur einer Netzwerkadresse möglich ist, hat jeder Rechner 65535 sogenannte Ports (genauer: je 65535 TCP und UDP-Ports), an denen Services auf Anfragen warten können bzw. von denen aus Anfragen gestellt werden können. Dabei ist wichtig, daß bestimmte Services immer an bestimmte Ports gebunden sind: der Webserver beispielsweise immer an Port 80[18]. Einige häufig genutzte Ports sind (vollständige Liste in `/etc/services`):

- TCP/22 – die Secure Shell.
- TCP/25 – Mail-Server.
- TCP/80 – WWW-Server.

Das Internet besteht, wie eingangs schon erwähnt, aus zahlreichen Teilnetzen, die sehr komplexe Hierarchien ausbilden können. In Deutschland beispielsweise ist ein Teilnetz das

Binärziffern, die Nullen, sind also noch allen Adressen des EKP-Netzes gemein, und müssen daher in der Netzmaske berücksichtigt werden. Die letzte Zahl der Netzmaske lautet also $11000000_{bin} = 192_{dez}$, die gesamte Netzmaske 255.255.255.192. Die kombinierte Netzwerkadresse ist demnach 129.13.102.0/255.255.255.192.

[16]Die Kurzform der kombinierte Netzwerkadresse des in der vorherigen Fußnote erklärten EKP-Netzes wäre demnach 129.13.102.0/26.

[17]Es gibt noch weitere IP-basierte Protokolle wie zum Beispiel das *Internet Control Message Protocol* (ICMP), die in `/etc/protocols` aufgelistet sind, auf die hier jedoch nicht eingegangen wird.

[18]Man kann einen Webserver auch so konfigurieren, daß er nicht an Port 80, sondern an einen anderen Port gebunden ist. Dann muß man dem Client, also dem Browser, jedoch explizit angeben, an welchen Port des Servers er sich wenden soll.

Deutsche Forschungs-Netz (DFN), in dem unter anderem die Universitäten zusammengefaßt sind. Die baden-württembergischen Universitäten, Fachhochschulen, Schulen und andere Landesdienststellen sind außerdem im Belwue-Netz organisiert. Die Universität Karlsruhe ist also Teil des Belwue-Netzes und selbst wiederum in Teilnetze gegliedet: Physik, das Rechenzentrum, Mathematik usw. Oft sind auch diese Netze auf Fakultätsebene noch weiter in Institutsnetze (etwa das EKP-Netz) untergliedert. Das nächste Kapitel beschäftigt sich damit, wie sich nun Rechner aus verschiedenen Subnetzen finden.

4.3.2 IP-Routing und Subnetze

Das Weiterleiten von Daten von einem Netz zum anderen nennt man Routing, die Rechner, die zwei (oder mehr) Netze miteinander verbinden, heißen Router oder Gateways. An das Internet angeschlossene Rechner müssen lediglich wissen, wie sie aus ihrem jeweiligen Netz herauskommen, welches also ihr Gateway zur Außenwelt ist. Diese Information bildet zusammen mit Auskünften über lokale, also ohne Gateway erreichbare Netze die sogenannte Routing-Tabelle des Rechners.

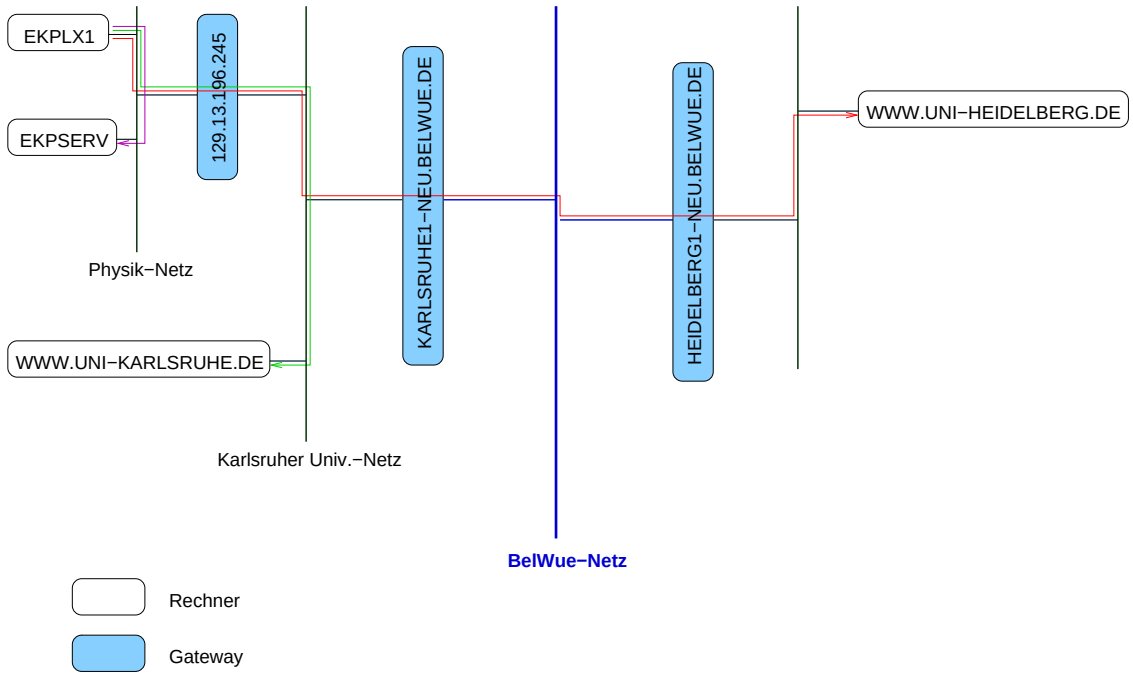

Abbildung 4.4: Verbindungswege von der EKPLX1 zu verschiedenen Rechnern im Internet. Gateways stellen Verbindungen zwischen Netzen her.

In Abbildung 4.4 ist zu sehen, wie ein Desktop-PC des Instituts (EKPLX1) Verbindung mit drei anderen Rechnern in verschiedenen Netzen aufnimmt:

- Verbindung zur EKPSERV, einem Server am Institut (oben rechts). Diese Verbindung läuft direkt über den Switch im Institut.

- Verbindung zum Webserver der Univ. Karlsruhe (unten rechts). Hier stellt ein Gateway die Verbindung zum Karlsruher Campus-Netz her, von dem aus der Webserver direkt zu erreichen ist.
- Verbindung zum Webserver der Univ. Heidelberg (ganz links). Wieder stellt ein Gateway die Verbindung zum Karlsruher Campus-Netz her; von dort aus führt ein weiterer Gateway ins baden-württembergische BelWue-Netz, an das auch das Netz der Univ. Heidelberg über einen Gateway angeschlossen ist.

Die Routing-Tabelle dient dazu, Datenpakete auf den richtigen Weg zu ihrem Ziel zu bringen. Im Falle des in Abbildung 4.5 gezeigten Desktop-PCs ist das recht einfach, da dieser PC nur über eine Netzwerkkarte (eth0) verfügt[19]. Die Routingtabelle hat hier zwei Einträge:

- der erste Eintrag gibt an, welche Rechner direkt (also ohne Gateway) an der Netzwerkkarte eth0 erreichbar sind. Die Netzwerk-/Netzmaskenkombination 129.13.0.0/255.255.0.0 besagt, daß dieser Eintrag für alle Rechner gilt, deren IP-Adresse mit 129.13 beginnt (entspricht der gesamten Universität Karlsruhe).
- der zweite Eintrag ist die sogenannte *default route*, die immer dann greift, wenn kein anderer Eintrag zutrifft. Sie verweist Datenpakete an den für den Rechner zuständigen Gateway. Wichtig ist dabei, daß der Gateway-Rechner selbst in einem direkt erreichbaren Netz ist, also eine Gateway-lose Route zu ihm führt.

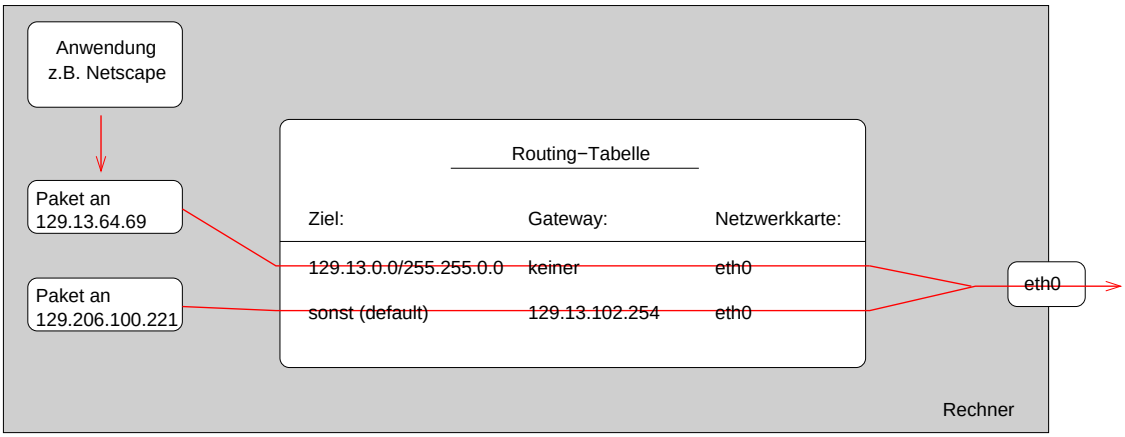

Abbildung 4.5: Routing-Tabelle auf einem Desktop-PC.

Abbildung 4.6 zeigt die Routing-Tabelle eines Portals mit zwei Netzwerkkarten (entsprechend dem Zugangsrechner in Abbildung 4.1). Die erste Netzwerkkarte (eth0) arbeitet wie die eines Desktop-PCs (vergleiche Abbildung 4.5), was nicht überrascht, denn die Außenanbindung soll ja gleich funktionieren. Neu ist die zweite Netzwerkkarte, die für das innere

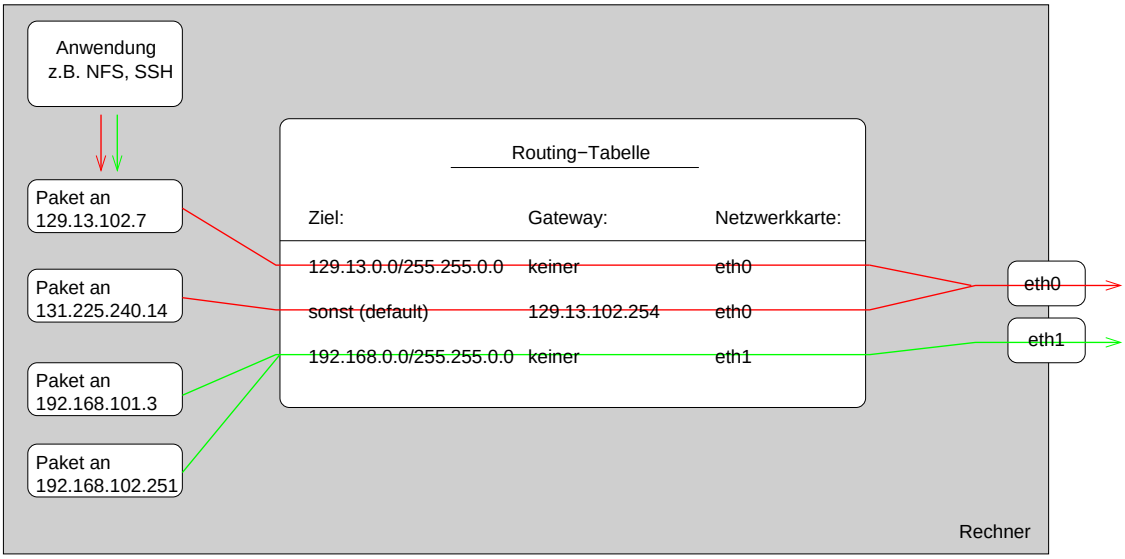

Abbildung 4.6: Routing-Tabelle auf einem Portal mit zwei Netzwerkkarten.

Netz des Clusters zuständig ist, also für Rechner, deren IP-Adresse mit 192.168 beginnt[20].

Als drittes Beispiel zeigt Abbildung 4.7 die Routing-Tabelle eines mit vier Netzwerkkarten (eth0-eth3) ausgestatteten Portals. Im Unterschied zu Abbildung 4.6 ist hier das innere Netz auf drei Netzwerkkarten aufgesplittet. Dazu werden drei logische Subnetze[21] gebildet: 192.168.101.0/24, 192.168.102.0/24 und 192.168.103.0/24. Man beachte, wie die beiden unteren Beispielpakete nun auf verschiedenen Netzwerkkarten den Rechner verlassen, während sie in Abbildung 4.6 noch dieselbe Karte benutzt hatten. Routing kann also zur Lastverteilung im Falle von mehreren Netzwerkkarten genutzt werden, und tatsächlich ist dies auch der Grund, warum das innere Netz der EKPplus in die drei obengenannten Subnetze gegliedert wurde. Abbildung 4.8 zeigt, daß ohne die Aufspaltung in Subnetze keine optimale Lastverteilung möglich wäre. Zwar würden Daten auf verschiedenen Karten eingehen, wenn sie entsprechend addressiert sind, jedoch gehen ausgehenden 192.168.0.0/16-Pakete alle über dieselbe Karte eth1, und eth2 und eth3 bleiben ungenutzt. Bei Rechnern, die vor allem Daten importieren (lesen), fällt dies nicht so sehr ins Gewicht wie bei Servern, also Rechnern, die in erster Linie Daten bereitstellen.

4.3.3 Firewall und Network Address Translation

Eine Firewall stellt eine Trennwand zwischen zwei Netzen oder Netzbereichen dar: mit einer Netzwerkkarte steht sie im äußeren, potentiell feindlichen, mit der anderen im inneren, ver-

[19] Die beiden Beispiel-IP-Adressen gehören zum Webserver der Univ. Karlsruhe und dem der Univ. Heidelberg; daher wird in Abbildung 4.5 auch Netscape als Anwendungsbeispiel gewählt.

[20] Die Zieladressen in diesem Beispiel und in Abbildung 4.7 gehören der Reihe nach zu folgenden Rechnern: EKPLX7, ein Desktop-PC; FCDFORA1.FNAL.GOV, der Oracle-Datenbank-Server des CDF-Experiments; EKPPLUS003, ein Rechenknoten des EKPplus-Clusters; EKPFS1, der Fileserver des EKPplus-Clusters.

[21] "Logisch" im Gegensatz zu "physikalisch", denn es werden physikalisch dieselben Switches genutzt.

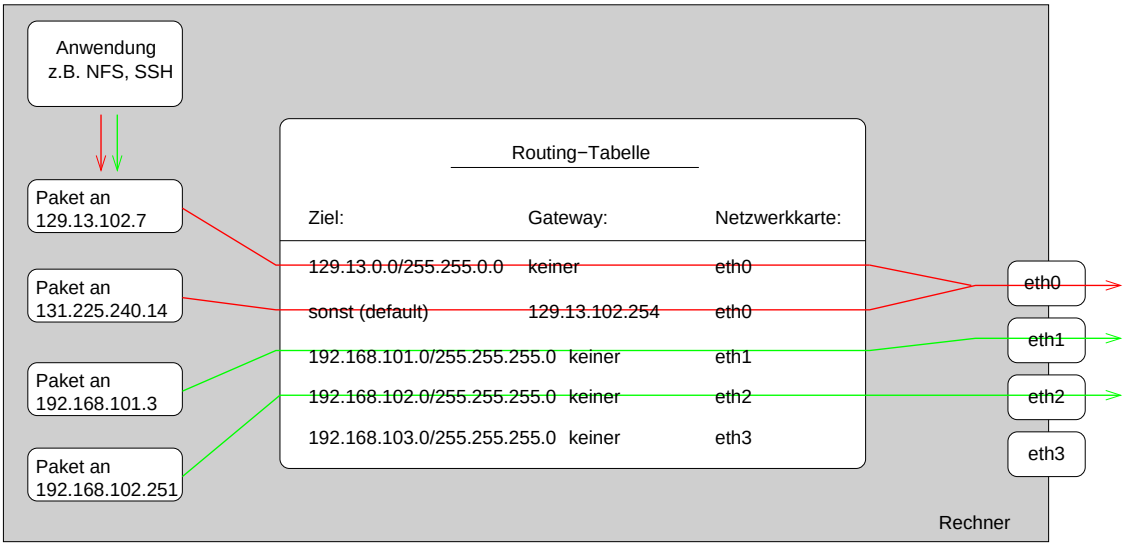

Abbildung 4.7: Routing-Tabelle auf einem Portal mit vier Netzwerkkarten.

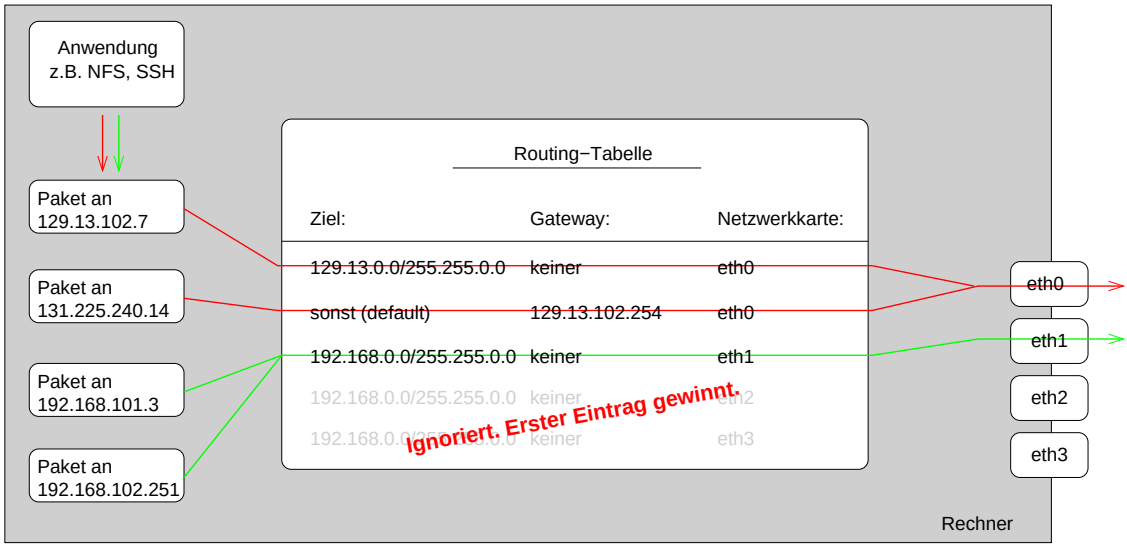

Abbildung 4.8: Ungünstige Routing-Tabelle auf einem Portal mit vier Netzwerkkarten. Beide Pakete erreichen das innere Netz über dieselbe Karte, während eth2 und eth3 ungenutzt bleiben.

trauenswürdigen Netz. Verbindungen, speziell von außen nach innen, werden selektiv durchgeschaltet. Man unterscheidet zwei Arten von Firewalls: der Paketfilter und der Proxy[22].

Ein Proxy ist ein zwischengeschalteter Vermittler zwischen Server (beispielsweise einem

[22]Der Begriff “Firewall” wird unterschiedlich verwendet. Oft finden sich auch die Begriffe “Packet-Filter Firewall” und “Application-Level Firewall” (letzteres ein Synonym zu Proxy), und “Firewall” ohne Zusatz steht für eine Kombination von beidem. In der vorliegenden Arbeit ist mit Firewall ein Paketfilter gemeint.

Webserver) und Client (dem Browser). Der Proxy analysiert die Anfragen des Clients auf Anwendungsprotokoll-Ebene (in diesem Fall das HTTP-Protokoll) und filtert sicherheitsgefährdende oder unerlaubte Anfragen aus. Dazu nutzt der Proxy sein Wissen um das verwendete Protokoll[23]. Dabei hat er allerdings keine Möglichkeit, die Echtheit der Absender-IP-Adressen zu prüfen.

Dies obliegt dem Paketfilter. Dieser untersucht eine Ebene tiefer die Header der IP-Datenpakete (also Ziel- und Absenderadresse sowie IP-Flags) und entscheidet anhand von vorgegebenen Regeln, welche Pakete passieren dürfen. Beim Aufstellen dieser Regeln nutzt man das Wissen darüber aus, welche Rechner auf welcher Seite der Firewall stehen. Kommt von außen ein Paket, das als Absender einen Rechner hat, der auf der inneren Seite der Firewall steht, so weiß man, daß der Absender des Pakets gefälscht ist ("*spoofing*"), und verwirft es. Die wichtigste Fähigkeit des Paketfilters ist jedoch, gezielt Ports (und damit Services) im inneren Netz für die Außenwelt zu sperren. So sollen beispielsweise die Netzwerklaufwerke (NFS) des EKP nur innerhalb des Institutsnetzes zur Verfügung stehen. Die Lösung ist, die Ports für NFS an der Firewall zu sperren. Man beachte, daß der Paketfilter nur die Header, nicht jedoch den Inhalt, also die eigentlichen Nutzdaten des IP-Pakets untersucht. So können auch potentiell gefährliche Pakete (etwa Löschbefehle an einen Datenbankserver) durch den Paketfilter, solange nur Ziel- und Absenderadresse den Regeln entsprechen: der Paketfilter kann mit dem Inhalt der Pakete nichts anfangen. Dies obliegt dem Proxy.

Oft wird eine Verbindung von Paketfilter und Proxy eingesetzt: der Paketfilter überprüft die Header, und der Proxy prüft auf destruktive Inhalte. Zum Abschirmen der EKPplus wird ein reiner Paketfilter eingesetzt. Dies ist für diesen Zweck ausreichend, da nur ein Dienst, nämlich die Secure Shell (ssh), von außen durchgelassen wird, und diese benötigt keinen Proxy, im Gegenteil, ein Proxy würde die Authifizierungsmechanismen der Secure Shell stören.

Auch der als *Network Address Translation* (NAT)[24] bekannte Mechanismus stellt ebenfalls eine Verbindung zwischen Netzwerken her, jedoch mit einem anderen Schwerpunkt. Bei NAT werden keine Pakete gefiltert, sondern es werden die Adressen im Paket-Header umgeschrieben – daher auch die Bezeichnung. Abbildung 4.9 verdeutlich dies. Nach außen sieht es also so aus, als kämen alle Netzwerkverbindungen aus dem inneren Netz von der NAT-Maschine selbst. Diese erhält auch die Antwort-Pakete, schreibt wiederum die Adressen um und leitet sie weiter.

Der Nutzen dieses Verfahrens ist, daß damit Rechner, die selbst keine öffentliche IP-Adresse haben und somit keinen Internetzugang, mit der NAT-Maschine als Mittler nun doch Verbindungen nach außen aufnehmen können. In gewisser Weise teilen sich also die Rechner hinter einer NAT-Maschine eine öffentliche IP-Adresse – eben die Adresse der NAT-Maschine. Gleichzeitig sind sie für die Außenwelt unsichtbar, denn der Versuch, etwa eine SSH-Verbindung zu ihnen aufzubauen, wird lediglich zum SSH-Port des NAT-Rechners führen. Über eine NAT-Maschine hinweg kommen nur Verbindungen zustande, die von den

[23]Daraus folgt, daß jede Applikation (HTTP, Datenbankserver, Mailserver usw.) ihr eigenes Proxy-Programm benötigt; diese können jedoch auf demselben Rechner laufen.

[24]Unter Linux wird NAT als *Masquerading* bezeichnet, was ebenfalls zutreffend; der Ausdruck NAT entstammt der BSD-Welt. Der NAT-Rechner läuft unter OpenBSD, daher wird hier diese Bezeichnungsweise gewählt.

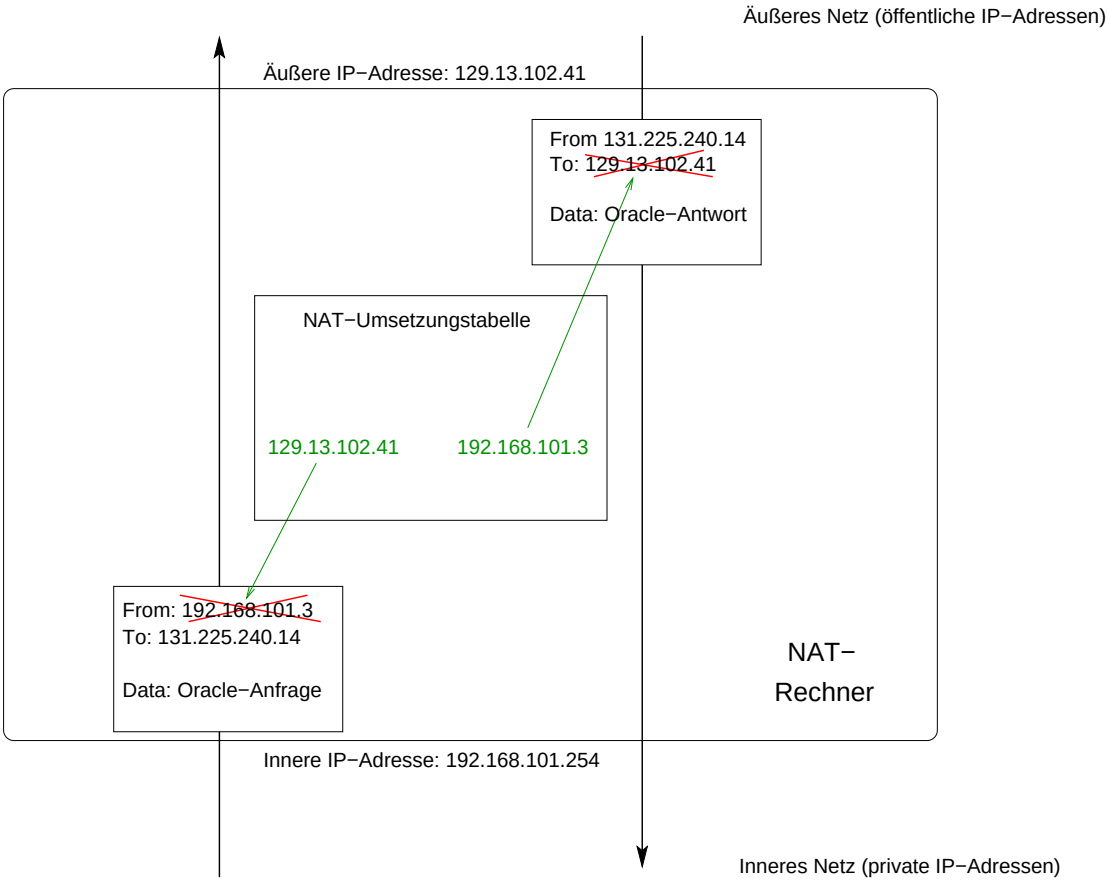

Abbildung 4.9: Network Address Translation am Beispiel einer Datenbankverbindung. Pakete, die die NAT-Maschine passieren, werden umgeschrieben, sodaß für das äußere Netz der Eindruck entsteht, alle Verbindungen kämen von der NAT-Maschine; in den Zugriffslogs des Oracle-Servers 131.225.240.14 wird nur die IP-Adresse des NAT-Rechners stehen. Die Antwortpakete werden zurückübersetzt und an die Maschinen hinter dem NAT-Rechner weitergeleitet.

Rechnern *hinter* ihr initiiert worden sind. Für die Zwecke der EKPplus – den Rechenknoten Zugriff auf den Oracle-Server am Fermilab zu ermöglichen und dergleichen – ist NAT die ideale Lösung.

4.3.4 Netzwerkstruktur (physikalisch)

Um eine sinnvolle und effiziente Netzwerkarchitektur für ein System wie den EKPplus-Cluster zu entwerfen, ist es notwendig, sich bereits im Vorfeld Gedanken über die spätere Lastverteilung zu machen und mögliche Engpässe rechtzeitig zu erkennen. In Falle der EKPplus hilft die Problemstellung selbst: die Rechenknoten müssen nicht untereinander kommunizieren, sondern in erster Linie mit den zentralen Datenspeichern. Betrachten wir die zwei Hauptaufgaben der EKPplus unter dem Gesichtspunkt der Datenübertragung (siehe auch Kapitel

4.1.1):

- **Monte-Carlo-Produktion**
  Die Knoten erzeugen große Datenmengen, die dann auf Fileservern abgelegt werden. Die zur Monte-Carlo-Produktion benötigten Kontrolldateien sind vernachlässigbar klein; bei CMS kommt noch eine Datenbank mit vorgefertigten *underlying events* hinzu. Kommunikation unter den Knoten findet nicht statt. Die von den Knoten benötigte Bandbreite ist relativ gering, da viel CPU-Zeit pro Ereignis nötig ist.

- **Datenprozessierung**
  Die Knoten prozessieren große Datenmengen, die sie von Fileservern lesen. Die erzeugten Ntupel-Dateien sind wesentlich kleiner als die Eingabedateien. Auch hier ist keine Kommunikation der Knoten untereinander nötig. Die von den Knoten benötigte Bandbreite ist jedoch recht hoch. Die in 4.1.1 abgeschätzten 50 MB/s sind nicht zu erreichen; man wird eine konservativere Schätzung von 5 MB/s (also eine etwas aufwändigere Analyse) ansetzen.

Daraus ergibt sich als möglicher Flaschenhals vor allem der Zugriff auf die zentralen Datenspeicher, wo sich der Netzwerkverkehr der Knoten bündelt. Es ist also darauf zu achten, daß der Fileserver mit genügend Übertragungskapazität ausgestattet ist. Die Anbindung der Knoten selbst ist unkritisch, hier reicht Fast Ethernet aus. Eine Anbindung der Knoten mit Gigabit Ethernet würde nur den Engpaß vor dem Fileserver verschärfen.

Ferner ist darauf zu achten, daß das öffentliche und das private Subnetz physikalisch voneinander getrennt sind, da es sonst zu empfindlichen Netzwerkstörungen käme, wie sie in Anhang A.2 beschrieben sind.

Diese Überlegungen führten schließlich zu einer Vernetzung wie sie Abbildung 4.10 zeigt. Die dort und in folgenden erwähnten Rechner (1) bis (8) werden in Kapitel 4.4 noch ausführlich beschrieben. Die Anbindung ans Internet erfolgt über die kombinierte Firewall/NAT-Maschine (1), die mögliche Angriffe aus den eingehenden Verbindungen herausfiltert, und an die sich das äußere Netz der EKPplus (als Teil des EKP-Desktop-Netzes) anschließt. An das äußeren Netz sind alle Netzwerkkarten mit öffentlichen IP-Adressen angeschlossen; dies sind die Portale EKPGRID (2), EKPCDF1 (3), EKPCDF2 (4), EKPCMS1 (5) und EKPDELPHI (6). Alle diese Rechner haben über zusätzliche Netzwerkkarten auch Zugriff auf das innere Netz mit den privaten IP-Adressen. Auf diesen inneren Netzbereich ist von außen kein direkter Zugriff möglich, da keiner der infragekommenden Rechner 2, 3, 4, 5 und 6 dafür konfiguriert ist, die entsprechenden Netzwerkpakete vom äußeren ins innere Netz weiterzuleiten. Indirekt ist der Zugriff auf das innere Netz aber möglich, indem sich der Benutzer auf einem der Portale einloggt und von dort aus auf die Ressourcen des inneren Netzes – die Knoten (ohne Nummer), den Fileserver (7) und die Kontrollmaschine (8) – zugreift.

Der NAT-Maschine (1) kommt in diesem Zusammenhang eine Sonderrolle zu. Sie vermittelt Verbindungen von den Knoten zur Außenwelt. Ein Beispiel für solche Verbindungen ist die CDF-Software, die verschiedene Oracle-Datenbanken mit Kalibrationskenngrößen am Fermilab abfragt. Eine denkbare Anwendung wäre auch das automatische Übertragen von Ergebnissen auf die Arbeitsplatzrechner der Benutzer (von den Knoten initiiertes *Secure Copy* `scp`).

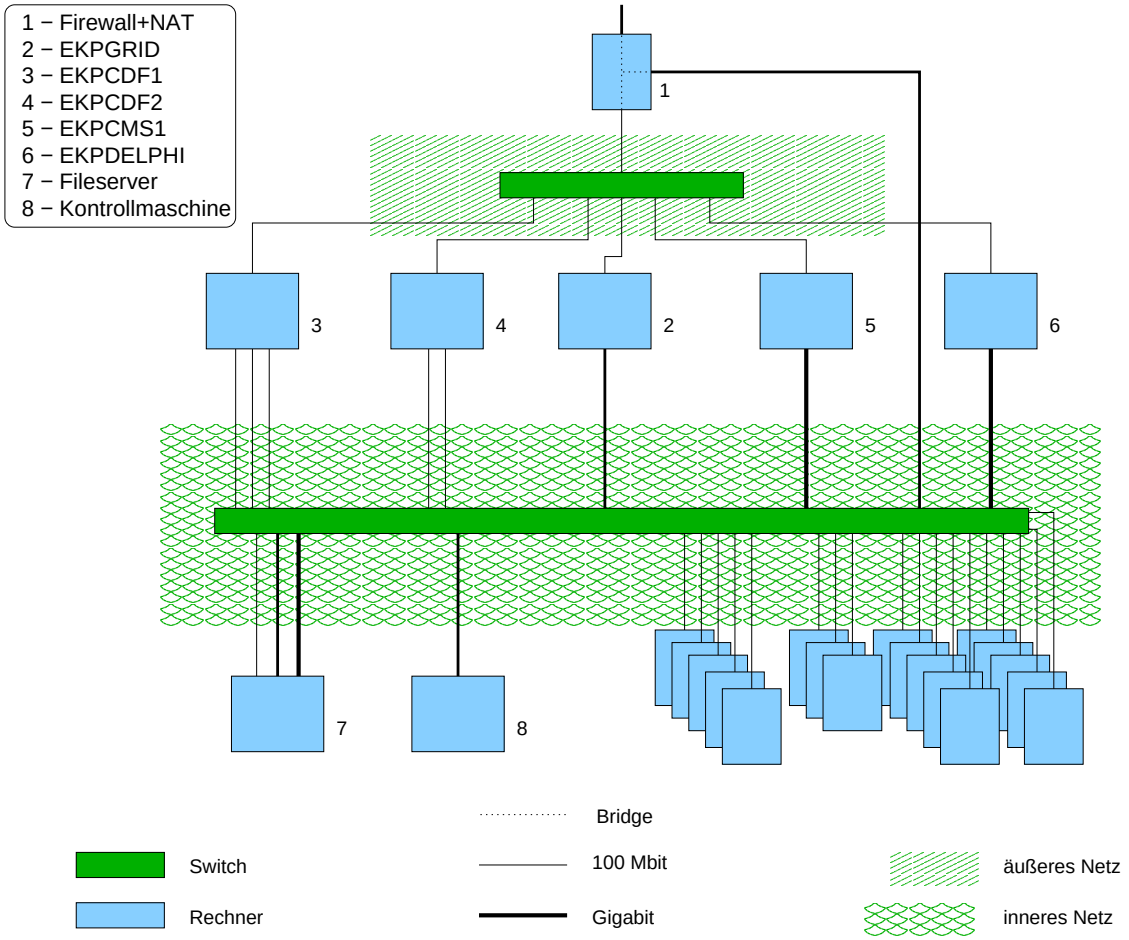

Abbildung 4.10: Physikalischer Netzaufbau. Die nummerierten Rechner sind im Text ausführlich beschrieben.

Das äußere Netz ist ein reines Fast Ethernet-Netz und verwendet als Teil des EKP-Institutsnetzes die beim Rechenzentrum üblichen Allied Telesyn 24 Port-Switches. Die Firewall/NAT-Maschine hingegen ist auf Gigabit ausgelegt: die NAT-Verbindung zum inneren Netz der EKPplus nutzt eine Intel-Gigabit-Karte; auch eine Glasfaserkarte zur Anbindung ans Universitätsnetz ist bereits eingebaut, jedoch noch nicht in Betrieb: eine entsprechende Gegenstelle im Rechenzentrum ist noch nicht verfügbar, es wird jedoch daran gearbeitet. Das Ziel ist eine schnelle Anbindung an das LHC Tier 1-Zentrum, das derzeit am Forschungszentrum Karlsruhe aufgebaut wird. Die Anforderungen an eine Gigabit-Firewall sind verhältnismäßig hoch: bei einer Datenrate von 30 MB/s und einer mittleren Paketgröße von 1 kB sind pro Sekunde die Header von bis zu 30000 Paketen zu analysieren, der PCI-Bus wird mit ca. 60 MB/s immerhin zur Hälfte ausgelastet.

Das innere Netz der EKPplus ist ein gemischtes 100 Mbit- und Gigabit Ethernet-Netz. Stark belastete Verbindungen sind in Gigabit Ethernet ausgeführt; dies sind die Anbindung des Fileservers (zwei Gigabit-Karten), des CMS-, Grid- und DELPHI-Portals und der Kontrollmaschine (je eine Gigabit-Karte) sowie die Verbindung zur NAT-Maschine. Für die

beiden CDF-Portale waren ursprünglich auch Gigabit-Karten vorgesehen, jedoch erwiesen sich diese als inkompatibel zum verwendeten Motherboard. Stattdessen wurden dort mehrere 100 Mbit-Karten eingebaut.

Abbildung 4.11 zeigt, wie die Umsetzung der physikalischen Netzwerkstruktur in natura aussieht.

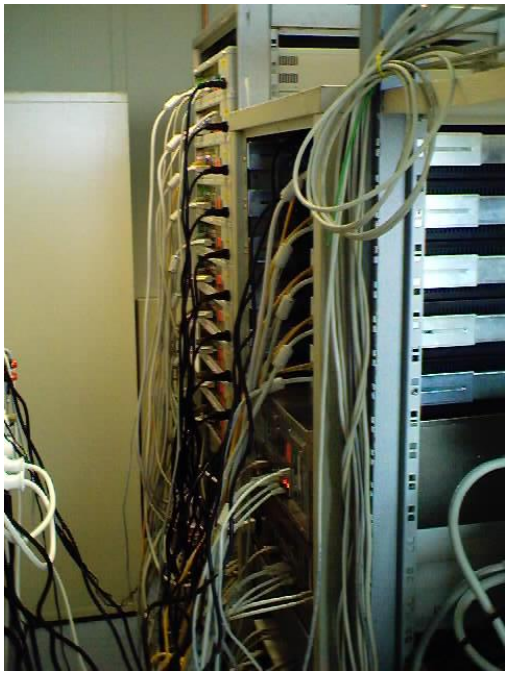

Abbildung 4.11: Verkabelung der EKPplus.

4.3.5 Netzwerkstruktur (logisch)

Wie Abbildung 4.12 zeigt, ist der logische Netzwerkaufbau dem physikalischen sehr ähnlich. Aus dem physikalischen Aufbau nicht ersichtlich ist jedoch die Untergliederung des privaten Netzes in drei Subnetze, 192.168.101.0/24, 192.168.102.0/24 und 192.168.103.0/24. Diese Unterteilung wird dadurch nötig, daß einige Rechner mehr als eine Netzwerkkarte im privaten Netz haben. Die Erklärung der Unterteilung wird aus den Erläuterungen zum Routing in Kapitel 4.3.2 deutlich: die Bandbreite der zusätzlichen Netzwerkkarten kann nur bei einer Unterteilung in Subnetze genutzt werden. Daher ist die Zahl der Subnetze genau gleich der maximalen Anzahl von Netzwerkkarten im privaten Netz, die ein Rechner hat. Bei der EKPplus sind dies die EKPCDF1 (3) und die EKPFS1 (7) mit ebenfalls drei Karten im privaten Netz.

Die logische Struktur des äußeren Netzes entspricht genau der physikalischen Verkabelung. Die Rolle des Firewall/NAT-Rechners (1) als Brücke zwischen innerem und äußerem Netz ist in Kapitel 4.3.3 ausführlich beschrieben.

Um eine möglichst gleichmäßige Lastverteilung zwischen den Netzwerkkarten zu erzielen, wurde folgende Aufteilung für NFS-Laufwerke festgelegt:

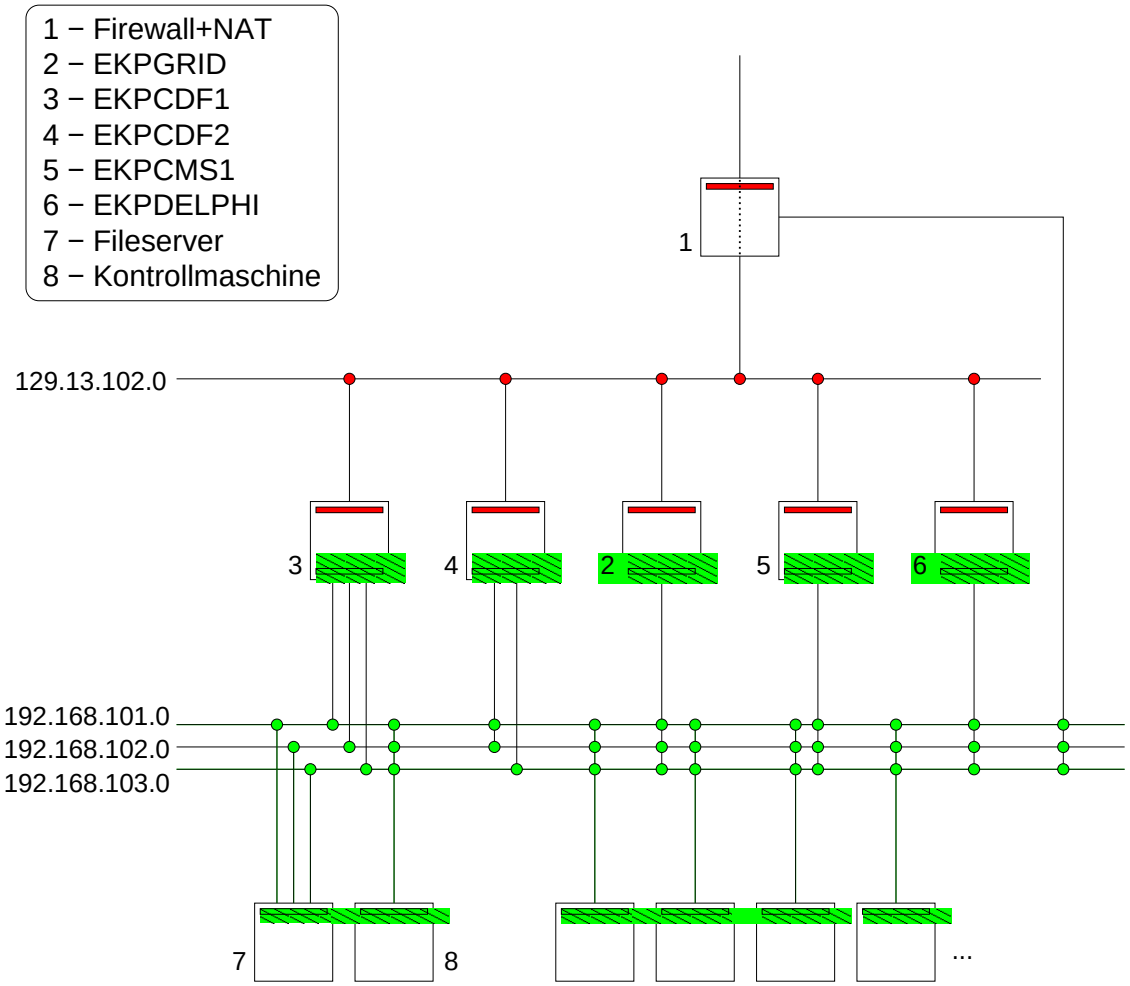

Abbildung 4.12: Logischer Netzaufbau. Nummerierung wie in Abbildung 4.10. Dunkle Markierungen in der Abbildung stehen für öffentliche IP-Adressen, schraffierte für private.

Subnetz:	**Verwendung:**
192.168.101.0/24	Benutzerverzeichnisse und Administration (Jobkontrolle, Booten der Rechenknoten sowie Wartung).
192.168.102.0/24	Home-Verzeichnisse der Portale; erstes RAID5-Laufwerk des Fileservers.
192.168.103.0/24	Scratch-Verzeichnisse der Portale; zweites RAID5-Laufwerk des Fileservers.

4.4 Die Komponenten der EKPplus

Im folgenden wird auf die einzelnen Rechner eingegangen, die zusammen die EKPplus bilden, und auf die Kriterien, nach denen die Hardware (und Software) ausgewählt wurde.

Dabei taucht ein Merkmal immer wieder auf: alle Rechner haben 19"-Gehäuse mit 1 bis 5 Einheiten Höhe, die gesamte Hardware wurde in zwei 19"-Schränke eingebaut. Der Grund für die Verwendung der teureren 19"-Technik liegt im geringeren Platzbedarf verglichen mit herkömmlichen PC-Gehäusen.

Ein Kriterium, das bei der Anschaffung unterschätzt wurde, ist die von den Geräten produzierte Wärme. Im Betrieb stellte sich heraus, daß die angeblichen 15 kW Kälteleistung, die die Klimaanlage des Raumes laut Hersteller erreicht, in der Praxis lediglich zur Kühlung einer elektrische Leistung von 4000-4500 W ausreicht. Ein Grund dafür ist die Temperatur des Kühlwassers: beträgt diese im Winter noch unter 10° C, so liegt sie im Sommer bei 15° C. Unter dem Gesichtspunkt der Kühlung sind Intel-CPUs gegenüber AMD-Prozessoren im Vorteil, da sie 10-20% weniger Abwärme erzeugen.

Das Preis-/Leistungs-Verhältnis war ein weiteres Schlüsselkriterium. Bei Prozessoren etwa zahlt man beim Spitzenmodell weit mehr für die CPU-Leistung als für die 10% langsamer getaktete Version. Auch bei der Wahl des Arbeitsspeichers (und damit des Motherboards) spielt dies eine Rolle: so war zum Anschaffungszeitpunkt DDR-RAM (*Double Data Rate RAM*) fast doppelt so teuer wie der übliche SD-RAM (*Syncronous Dynamic RAM*), jedoch haben Tests ergeben, daß der Geschwindigkeitsvorteil für die Testjobs im einstelligen Prozentbereich liegt. Auch hier wurde im Sinne des Preis-/Leistungs-Verhältnisses und nicht nach der absoluten Leistung entschieden.

4.4.1 Übersicht

Abbildung 4.13 zeigt eine Übersicht über die Komponenten der EKPplus und ihre Verbindungen untereinander. Dabei sind Netzwerkkomponenten grau, Rechner selbst weiß dargestellt. Zu beachten ist, daß der Funktion nach unterschieden und damit die Firewall/NAT-Maschine den Netzwerkkomponenten zugeschlagen wird, obwohl es sich eigentlich um einen Rechner handelt. In den folgenden Abschnitten werden die Rechner im Einzelnen vorgestellt, sowohl von der verwendeten Hardware, als auch von der Funktion. Die Details der Hardwareausstattung finden sich im Anhang A.3.

Zugangsrechner (Portale)

erfüllen zwei Aufgaben: zum einen führt über sie der Zugang zu den Ressourcen des Clusters (daher ihr Name), zum anderen dienen sie als Entwicklungsplattformen und Software-Server für das jeweilige Experiment. In ihrer Funktion als Entwicklungsplattform sind sie mit viel Arbeitsspeicher und schnellen Platten ausgestattet, die einen zügigen Entwicklungszyklus Editieren-Compilieren-Linken-Testen ermöglichen. Außerdem sind eventuell erforderliche Werkzeuge wie etwa spezielle Compiler installiert. Die Maschinen sind so dimensioniert, daß auch mehrere Entwickler gleichzeitig effizient arbeiten können, wobei sich die Dimensionierung nach der tatsächlichen Größe der Arbeitsgruppe und der Komplexität der Software richtet.

Die Portale sind die einzigen Rechner des Clusters, die von außerhalb zugänglich sind. Nur sie (und die nicht für Benutzer zugängliche Firewall/NAT-Maschine) haben öffentliche IP-Adressen. Von den Portalen aus sind die Platten des Fileservers sichtbar; von hier aus können Jobs in die Queues submittiert werden. Außerdem exportieren die Portale ihre

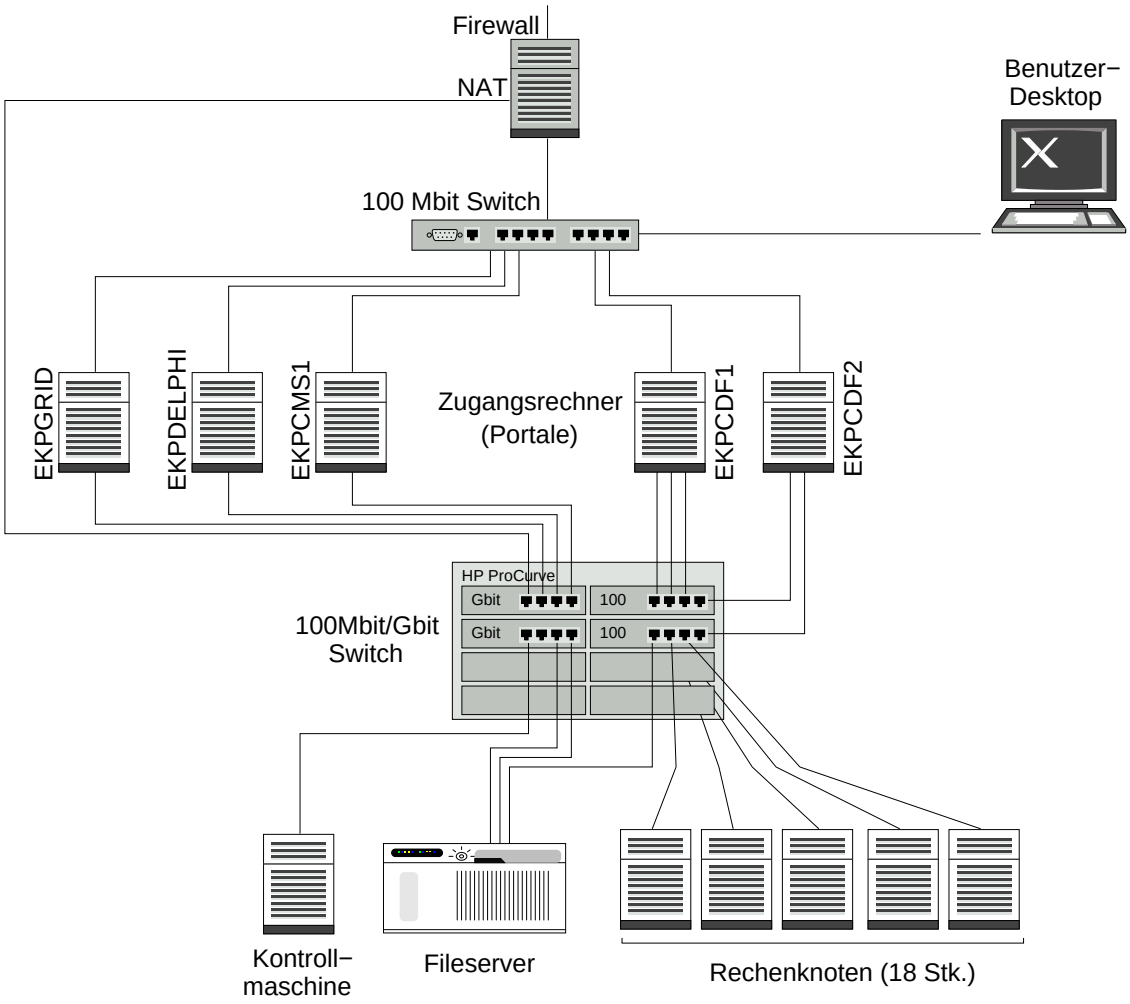

Abbildung 4.13: Übersicht über die Komponenten der EKPplus.

gesicherten (`/portal/<portalname>/home`) und ungesicherten (`/portal/<portalname>-/scratch`) Benutzerbereiche an die Rechenknoten.

Rechenknoten

stellen Rechenleistung bereit. Die Knoten sind darauf ausgelegt, daß ein CPU-intensiver und ein I/O-intensiver Job gleichzeitig dort laufen kann.

Fileserver

Der Fileserver verfügt über möglichst viel Plattenplatz und stellt diesen den Maschinen im Cluster zur Verfügung.

Kontrollmaschine

Auf ihr werden die Benutzer verwaltet, hier liegen auch die Loginverzeichnisse. Außerdem steuert die Kontrollmaschine den Ablauf der Jobs: hier läuft der Scheduler des Batch-Systems. Die Rechenknoten beziehen ihre Netzwerkkonfiguration und ihr gesamtes Betriebssystem von der Kontrollmaschine.

Firewall/NAT-Maschine

Ihr obliegt das Filtern der Verbindung vom Internet zur EKPplus. NAT ermöglicht es den Knoten, auf Daten im Internet zuzugreifen, etwa auf die Kalibrationsdatenbank von CDF.

Switches

bilden die Netzwerk-Infrastruktur.

Hardware zur Überwachung und Wartung

Unter diese Rubrik fällt zum Beispiel die Temperaturüberwachung des Raums und die Wartungskonsole.

4.4.2 CDF-Portal `EKPCDF1`

Die Software des CDF-Experiments befindet sich derzeit noch stark in der Entwicklung. Daher sind die Anforderungen an die Portale, die auch als Entwicklungsmaschinen dienen, besonders groß; es gibt derzeit zwei CDF-Portale.

Hardware

Die `EKPCDF1` ist eine Athlon-Doppelprozessormaschine mit 1400 MHz Prozessortakt. Die Maschine ist mit 2 GB Hauptspeicher ausgestattet. Als Systemplatte kommt eine schnelldrehende 40 GB IDE-Platte von IBM zum Einsatz. Zusätzlich verfügt die `EKPCDF1` über vier IDE-Festplatten von Maxtor mit einer Kapazität von je 80 GB. Diese vier Platten sind an einen RAID-Kontroller Escalade 7410 von 3ware [42] angeschlossen und als RAID0 konfiguriert. In diesem RAID-Modus werden die Daten gleichmäßig auf die Platten verteilt (*Striping*), sodaß sich bei n Festplatten eine theoretische Gesamtübertragungsrate von n mal der Einzelübertragungsrate ergibt. Es wurden Übertragungsraten von über 100 MB/s gemessen[25].

Das verwendete Motherboard Tyan Thunder K7 [43] enthält bereits einen einfachen Graphikadapter. Auch zwei 100 Mbit Ethernet-Adapter sind auf das Board integriert. Dieses verfügt außerdem über fünf 64 bit PCI-Steckplätze, was besonders für den Einsatz des Escalade 7410 vorteilhaft ist. Ursprünglich war geplant, die `EKPCDF1` mit einer Gigabit Ethernet-Karte an das innere Netz anzuschließen, jedoch wurde dies durch Inkompatibilitäten zwischen

[25] Zur Plattengeschwindigkeit siehe Anhang A.4.1; RAID wird in Anhang A.5 erläutert.

Motherboard und Netzwerkkarte vereitelt. Stattdessen wurden zusätzlich zu den beiden integrierten Netzwerkadaptern noch zwei weitere 100 Mbit-Netzwerkkarten eingebaut, sodaß ca. 30 MB/s (3 × 10 MB/s) ins innere Netz übertragen werden können.

Als Hauptspeicher finden Double Data Rate-Module mit Error Correction Code (ECC-DDR-RAM) Verwendung. Beim Double Data Rate-Verfahren wird nicht nur ein Speicherzugriff pro normalen Speichertakt (üblich sind 100 oder 133 MHz) durchgeführt, sondern zwei: einer bei ansteigender Flanke des Taktsignals, der andere bei abfallender Flanke. Damit ergibt sich theoretisch eine Verdopplung des Speicherdatenrate gegenüber herkömmlichem SD-RAM. In der Praxis fällt der Zuwachs jedoch wesentlich geringer aus, da die Athlon-Prozessoren mit ihren Level 1- und Level 2-Caches über einen recht wirkungsvollen Kompensationsmechanismus verfügen und darauf ausgelegt sind, mit der bescheideneren Datenrate von SD-RAM auszukommen. So wurde bei typischen CDF II-Programmen ein Geschwindigkeitszuwachs von 5% gegenüber ansonsten gleichausgestatteten Rechnern mit SD-RAM gemessen. Der Mehrpreis von ECC-DDR-RAM war zum Anschaffungszeitpunkt nicht gerechtfertigt, jedoch unterstützt das verwendete Motherboard lediglich diese Speicherart.

Aufgaben der `EKPCDF1`

Als CDF-Entwicklungsmaschine ist hier die komplette, in Kapitel 3 beschriebene CDF-Software installiert, inklusive KAI C++-Compiler. Neben ihrer Aufgabe als Entwicklungsmaschine und Zugangsrechner hat die `EKPCDF1` weitere Sonderrollen inne: für das CDF-Experiment wurde ein MySQL-Datenbank-Server aufgesetzt, der eine Kopie der Kalibrationsdatenbank von CDF bereithält für den Fall, daß die Zentraldatenbank am Fermilab nicht erreichbar ist (Netzwerkstörung, zuviele gleichzeitige Zugriffe o.ä.). Die `EKPCDF1` stellt außerdem die Master-Installation der CDF-Software für die Knoten bereit (`/ekpplus/cdf`).

4.4.3 CDF-Portal `EKPCDF2`

Die `EKPCDF2` ist im wesentlichen baugleich mit der `EKPCDF1`, verwendet jedoch die neueren Athlon XP-Prozessoren, ebenfalls mit 1400 MHz Prozessortakt (*Quantispeed*-Bezeichnung: XP 1600+). Auch wurde nur eine zusätzliche Netzwerkkarte eingebaut statt zwei wie in der `EKPCDF1`. Die `EKPCDF2` ist ein reines Entwicklungsportal (verfügt also ebenfalls über die komplette CDF-Softwareumgebung inklusive Compiler). Hier ist außerdem die CDF-Datenverteilungssoftware SAM installiert.

4.4.4 CMS-Portal `EKPCMS1`

Die CMS-Software ist sehr komplex, was eine spezielle Konfiguration und leistungsstarke Hardware erforderlich macht.

Hardware

Die `EKPCMS1` ist eine Athlon-Doppelprozessormaschine mit zwei Athlon MP 2100+ CPUs mit einem Takt von 1733 MHz. Der Speicherausbau beträgt 2 GB. Als Systemplatte kommt

eine 40 GB IDE-Platte von Maxtor zum Einsatz.

Die Maschine verfügt außerdem über zwei 3ware 7450 RAID-Kontroller: die Homeverzeichnisse und die zentrale CMS-Installation liegen auf einem RAID5-Laufwerk, bestehend aus 4 120 GB-Platten mit einer Nettokapazität von 360 GB. Der zweite Kontroller ist als RAID0 konfiguriert und dient als schneller Scratch-Speicher. Hier beträgt die nutzbare Kapazität 480 GB (wiederum bestehend aus 4 120 GB-Platten).

Die Netzwerkanbindung nach außen übernimmt die auf dem Board integrierte 3COM Fast Ethernet-Karte, die Anbindung ins innere Netz eine Intel Pro/1000 Gigabit Ethernet-Karte.

Aufgaben der `EKPCMS1`

Aufgrund der in Kapitel 4.2.3 geschilderten Problematik läuft die `EKPCMS1` nicht unter Fermi Redhat 7.1.1 wie der Rest des Clusters, sondern unter SuSE 7.2, um ihrer Rolle als CMS-Entwicklungsmaschine gerecht zu werden. Die CMS-Entwicklungsumgebung ist installiert, inklusive gepatchtem GNU Compiler.

Eine Besonderheit ist der Objectivity-Server, der auf der `EKPCMS1` läuft, denn dieser schreibt seine Daten nicht auf eine lokale Platte, sondern über NFS auf den Fileserver. Dazu ist eine spezielle Konfiguration des Fileservers erforderlich.

4.4.5 Grid-Portal `EKPGRID`

Da die Karlsruher Grid-Software-Gruppe noch im Aufbau befindlich ist, sind die Hardwareanforderungen an das Grid-Portal im Moment noch eher moderat; doch wird sich dies in den nächsten Monaten ändern. Das Grid-Portal soll später einmal die Einbindung der EKPplus in das in Kapitel 6.3 vorgestelle *Computational Grid* ermöglichen.

Hardware

Die `EKPGRID` verfügt über einen Athlon-Prozessor mit 1400 MHz und ist mit 1 GB Speicher ausgestattet. Als Massenspeicher dient eine 120 GB-Festplatte von Maxtor. Das Motherboard verfügt bereits über eine einfache Graphikkarte und eine 100 Mbit-Netzwerkkarte. Um eine hinreichend schnelle Anbindung an den Fileserver zu ermöglichen, wurde eine Gigabit Ethernet-Karte von Typ D-Link DGE-550T (die ursprünglich für die `EKPCDF1` vorgesehen war) eingebaut.

4.4.6 DELPHI-Portal `EKPDELPHI`

Da die DELPHI-Software noch in Fortran geschrieben und damit deutlich effizienter ist, und da die Datenmengen an Lepton-Collidern allgemein geringer sind als bei Hadron-Collidern, genügt für das DELPHI-Portal eine relativ kleine Maschine.

Hardware

Die `EKPDELPHI` ist eine Athlon-Maschine mit 1400 MHz und demselben Motherboard wie die `EKPGRID`. Bei der Speicherausstattung genügen hier 512 MB, dazu kommen 120 GB

Festplattenplatz. Als Gigabit Ethernet-Karte zum Anschluß an das innere Netz kommt eine 3com 3C996T zum Einsatz, die ursprünglich für die `EKPCDF2` vorgesehen war.

Aufgaben der `EKPDELPHI`

Auf der `EKPDELPHI`, die unter Fermi Linux 7.1.1 läuft, sind die CERN-Bibliotheken und der GNU Fortran 77-Compiler installiert. Außerdem wurde auf Anfrage der LEP 2-Gruppe der Fortran 90-Compiler von Nagware eingerichtet, für den die Universität eine Campus-Lizenz besitzt.

4.4.7 Rechenknoten `EKPPLUS001` - `EKPPLUS018`

Für die Knoten kamen nach einigen Vorabtests zwei Bauformen in die engere Wahl: zum einen Athlon-Doppelprozessorknoten basierend auf dem auch in den CDF-Portalen verwendeten Tyan ThunderK7 Board, zum anderen Athlon-Einprozessorknoten basierend auf dem im DELPHI- und Grid-Portal verwendeten Asus A7S-VM Board. Die Intel-Prozessoren Pentium III und 4 waren aufgrund ihrer schwächeren Fließkommaleistung und ihres höheren Preises für den Einsatz in Rechenknoten im Nachteil; die überraschend schwachen Leistung der Klimaanlage, die die Intel-Prozessoren in einem besseren Licht erscheinen läßt, war zum damaligen Zeitpunkt noch nicht bekannt. Zur endgültigen Entscheidung stellte die Firma FMS Computer freundlicherweise je ein Testsystem für eine Woche zur Verfügung. Die Rahmenbedingungen, die die Auswahl letztlich bestimmten, waren:

- die Größe des klimatisierten Raumes. Zur Unterbringung des gesamten Clusters inklusive Arbeitstisch, Ersatzteillager und Überwachungskonsole stehen nur ca. $15.6m^2$ zur Verfügung. Die in speziellen 1 HE-Gehäusen untergebrachten Doppelprozessorknoten benötigen pro Prozessor nur ein Viertel des Platzes, den ein 2 HE großer Einprozessorknoten belegt.

- die Kälteleistung der Klimaanlage. Die gesamte elektrische Leistungsaufnahme des Clusters, und der im selben Raum befindlichen sonstigen Server, darf die 15 kW, die die Klimaanlage angeblich an Wärme abführen kann, nicht überschreiten. Die mittlere Leistungsaufnahme eines Einprozessorknotens beträgt ca. 100 W, die eines Doppelprozessorknotens ca. 200 W, sodaß hier keine Aussage zugunsten eines der beiden Systeme getroffen werden kann. Eine Abschätzung ergab eine Kapazität von bis zu 100 Prozessoren (100×100 W $= 10$ kW, mit 5 kW für sonstige Rechner). Im Betrieb diesen Sommer zeigte sich allerdings, daß diese Abschätzung allzu optimistisch war und die Klimaanlage bereits mit zwanzig Knoten nicht mehr allzu viel Reserven hat. Messungen zufolge bewirken 5 Knoten einen Temperaturanstieg um 2° C. Derzeit beträgt die Raumtemperatur $24 - 25°$ C.

- die Wiederverwendbarkeit von Gehäusen und Netzteilen. 19"-Gehäuse sind mit ca. 200 Euro wesentlich teurer als normale PC-Gehäuse (Midi-Tower ca. 80 Euro) und machen damit etwa 20% des Gesamtpreises für einen Knoten aus. Aus diesem Grund wäre es wünschenswert, die Gehäuse bei einer späteren Aufrüstung der Knoten mit neuen

Prozessoren und Boards wiederverwenden zu können, was bei den 2 HE-Gehäusen der Einprozessorknoten problemlos möglich ist. Leider ist bei den 1 HE-Gehäusen der Doppelprozessorknoten eine Wiederverwendung ausgeschlossen: spezielle Luftkanäle sowie ein proprietäres Netzteil, wie es für das Tyan ThunderK7 Board benötigt wird, verhindern dies.

- die Kosten pro Rechenleistung. Hier waren die Doppelprozessorknoten mit 1600 Euro pro 1.4 GHz Athlon deutlich schlechter gestellt als die Einprozessorknoten mit 1100 Euro. Die höheren Kosten für die Doppelprozessorsysteme sind unter anderem begründet durch das teure Board (500 Euro gegenüber 2x150 Euro), den teureren Speicher (ECC-DDR-RAM gegenüber SD-RAM), sowie das besondere Gehäuse mit Spezialnetzteil. Dies kann die eine eingesparte Festplatte nicht ausgleichen.

Keine Rolle bei der Entscheidung spielt hingegen die Anzahl der Knoten in Hinblick auf Software-Wartung. Würde die benötigte Software auf jedem Knoten separat installiert, so stiege der Wartungsaufwand linear mit der Zahl der Knoten, und folglich wäre eine möglichst kleine Anzahl möglichst leistungsfähiger Knoten wünschenswert. Da jedoch von vorneherein klar war, daß auf den Knoten selbst keine Software installiert wird, ist dieses Argument für die EKPplus hinfällig: die Software wird zentral an einer Stelle gewartet, unabhängig von der Anzahl der Knoten.

Hardware

Die obengenannten Kriterien führten zu einer Entscheidung für Einprozessorknoten; der erhöhte Platzbedarf wird in Kauf genommen, insbesondere vor dem Hintergrund des Budgets, das ohnehin den Kauf von zunächst nur ca. 20 Prozessoren zuließ. Die entgültige Entscheidung fiel zugunsten von fünf Knoten der Firma FMS Computer und 13 Knoten der Firma Kircher EDV; diese unterscheiden sich lediglich im Festplattenmodell (FMS verwendet IBM-Platten, Kircher Maxtor) sowie im verwendeten Gehäuse. Hierzu ist anzumerken, daß es einige Mühe bereitete, die Knoten der Firma Kircher hinreichend gut zu kühlen, da die verbauten Gehäuse keinen geeigneten Luftstrom zulassen. Durch den Einbau weiterer, leistungsstarker Lüfter gelang es jedoch nach einigem Experimentieren, eine angemessene Gehäusetemperatur von 40° C zu erzielen. Alle Knoten verfügen über einen 1.4 GHz Athlon, 512 MB RAM, eine 40 GB IDE-Platte und eine auf dem Board integrierte, bootfähige 100 Mbit-Netzwerkkarte.

Konfiguration der Knoten

Die Knoten beziehen alle Software über das innere Netz. Die 40 GB-Platten sind ausschließlich als temporärer Speicher (`/tmp`) gedacht; sie werden beim Hochfahren des Knotens gelöscht (die Platte enthält auch eine 512 MB große Swap-Partition). Auch der Kernel wird über Netzwerk geladen. Dies ist möglich, da die auf dem A7S-VM-Board integrierte Netzwerkkarte bootfähig ist: die Firmware der Karte beinhaltet einen DHCP- und einen TFTP-Client. Beim Booten eines Knotens vom Netzwerk sendet die Karte zunächst eine DHCP-Anfrage ins Netz. Der DHCP-Server (auf der Kontrollmaschine) beantwortet diese

mit den Netzwerkdaten des Knotens (IP-Adresse, Netzmaske, Gateway) und mit der IP-Adresse des TFTP-Servers, der den Kernel für den Knoten bereithält (die `EKPPLUSCTL`). Die Netzwerkkarte lädt dann den Boot-Lader und den Kernel vom angegebenen TFTP-Server.

Der Kernel für die Knoten ermöglicht es, das Root-Verzeichnis `/` des Knotens via NFS zu importieren; dies wird in Kapitel 4.4.2 beschrieben.

4.4.8 Fileserver `EKPFS1`

In Anbetracht der zu erwartenden sehr großen Datenmengen von bis zu 10 TB, die die verschiedenen Analysegruppen am EKP in Karlsruhe benötigen, ist vor allem der Preis pro TB ausschlaggebend für die Wahl der Massenspeicher. Dieses Kriterium schließt SCSI-Platten von vorneherein aus, denn diese sind zwischen drei und acht mal so teuer wie gleichgroße IDE-Platten. Gleichzeitig verhindern die großen Datenmengen ein vollständiges Backup, sodaß der Datenspeicher selbst Redundanz beim Ausfall von Festplatten bereitstellen muß, wie dies bei RAID-Kontrollern der Fall ist[26]. Diese Überlegungen legen die Verwendung von IDE-Platten an RAID 5-Kontrollern nahe. Die Plattengeschwindigkeit ist relativ unkritisch, da die 30 MB/s, die ein Gigabit Ethernet-Adapter maximal über NFS exportieren kann, schon von einer einzigen modernen IDE-Platte erreicht wird.

Hardware

Angeschafft wurde der "Tera-Server" der Firma FMS Computer; dieser verfügt über zwei 3ware Escalade 7850 RAID-Kontroller [42], an die je 8 100 GB-Platten mit einer Gesamtkapazität von $2 \times 8 \times 100$ GB = 1.6 TB angeschlossen sind. Da die Platten als RAID 5 geschaltet sind, stehen effektiv 1.4 TB zur Verfügung. Zum Export der Daten stehen eine 100 Mbit- und zwei Gigabit Ethernet-Netzwerkkarten zur Verfügung und ermöglichen eine Gesamtübertragungsrate von annähernd 70 MB/s. Auf dem Serverworks-Board, das neben drei 32 bit PCI-Steckplätzen auch zwei 64 bit PCI-Slots bietet (die die Netzwerkkarten beherbergen), sitzen zwei Pentium III-Prozessoren mit 1.2 GHz sowie 2 GB Speicher, der im wesentlichen als Plattencache genutzt wird.

Der Tera-Server ist in einem 4 HE 19"-Gehäuse untergebracht, das über 16 *hot-swap*-Rahmen für die Platten verfügt.

Konfiguration des Fileservers

Beide RAID-Kontroller stellen sich für den Linux-Kernel als SCSI-Kontroller mit je einer angeschlossenen 700 GB-Festplatte dar. Auf der "Systemplatte", also der Pseudo-Platte des ersten Kontrollers, wurde vom Hersteller auf einer 10 GB-Partition SuSE Linux 7.3 mit Kernel 2.4.10 installiert. Der Rest steht (abzüglich 1 GB für Swap) als eine 660 GB große Partition unter `/storage/1` zur Verfügung. Die andere Pseudo-Platte ist (ebenfalls bis auf 1 GB für Swap) als 670 GB große Partition unter `/storage/2` verfügbar. Als Dateisystem kommt in beiden Fällen das *Journalling*-Dateisystem EXT3 zum Einsatz.

[26] RAID-Kontroller erlauben es, je nach RAID-Level Platten zu spiegeln oder über Prüfsummenverfahren genügend Redundanz zu erzeugen, um den Ausfall einer Platte im RAID-Verbund ohne Datenverlust zu überstehen. Mehr Details dazu im Anhang A.5.

Beide großen Partitionen werden jeweils einer Gigabit Ethernet-Karte zugeordnet. Dabei wird die Segmentierung des inneren Netzes in drei Subnetze ausgenutzt: das 100 Mbit-Adapter des Fileservers wird an die IP-Adresse 192.168.101.251 (in ersten Subnetz) gebunden, die erste Gigabit-Karte an 192.168.102.251 (zweites Subnetz) und die zweite Gigabit-Karte an 192.168.103.251 (drittes Subnetz). Client-Rechner greifen nun auf `/storage/1` über 192.168.102.251 (erste Gigabit-Karte) zu und auf `/storage/2` über 192.168.103.251 (zweite Gigabit-Karte). Es ist zu beachten, daß dies technisch nicht erzwungen ist; es handelt sich lediglich um eine Konvention[27].

Um dem Objectivity-Server auf der `EKPCMS1` Zugang zum Fileserver zu ermöglichen, müssen die Laufwerke mit der Option `insecure` in `/etc/exports` exportiert werden; diese Option gestattet den Zugriff auf den NFS-Server von nicht-privilegierten Ports, wie sie Objectivity verwendet.

Der Fileserver läuft unter dem von SuSE mitgelieferten und nicht mehr aktuellen Kernel 2.4.10, da die RAID-Überwachungstools von 3ware bisher nicht unter neueren Kerneln laufen. Bei den CDF-Portalen ist dies nicht relevant, da diese als RAID 0 (keinerlei Redundanz, maximale Geschwindigkeit) konfiguriert sind: hier könnten die Überwachungswerkzeuge ohnehin nur Totalverlust diagnostizieren – und dazu braucht es kein Spezialwerkzeug. Also wurde auf den Portalen einem neueren Kernel der Vorzug gegeben. Bei RAID 5 jedoch ist beim Ausfall einer einzelnen Platte noch nichts verloren, wenn sie rechtzeitig (also bevor eine weitere Platte ausfällt) ersetzt wird. Der 3ware *Disk Monitoring Daemon* (3DMD) überwacht die Platten und meldet Ausfälle per Email.

4.4.9 Kontrollrechner `EKPPLUSCTL`

An den Kontrollrechner, der Dienste wie Benutzerverwaltung und dergleichen übernimmt, werden keine besonderen Anforderungen gestellt, sodaß hier ein preiswerter Rechner genügt. Die Kontrollmaschine ist ein AMD Duron mit 800 Mhz Taktfrequenz und 512 MB Speicher (vor allem als Plattencache genutzt). Zwei 40 GB IDE-Platten, die als Software-RAID1 geschaltet sind (sich also gegenseitig spiegeln, um die Ausfallsicherheit zu erhöhen) beheimaten das System für die Kontrollmaschine selbst und das für die Knoten, sowie die Loginverzeichnisse der Clusterbenutzer.

Aufgaben des Kontrollrechners

Die Aufgaben des Kontrollrechners sind vielfältig. Zum einen läuft hier der DHCP-Server, der für die Vergabe der IP-Adressen an die Knoten zuständig ist. Die MAC-Adresse jedes Knotens ist in der Konfigurationsdatei des DHCP-Servers gespeichert, zusammen mit der dem Knoten zugewiesenen IP-Adresse. Diese und weitere Netzwerkparameter werden beim Booten eines Knotens an diesen übermittelt, zusammen mit der IP-Adresse des TFTP-Servers (die `EKPPLUSCTL` selbst).

[27] Um die Zuordnung zu erzwingen, könnte man `/storage/1` nur für das 192.168.102.0/26-Subnetz exportieren und `/storage/2` nur für 192.168.103.0/26. Dann müßten jedoch die Knoten neben ihrer regulären IP-Adresse 192.168.101.1-18 auch noch je eine 192.168.102.1-18 und 192.168.103.1-18 zugewiesen bekommen, was aufwändiger ist als die kooperative Lösung.

Für die Rechenknoten stellt die `EKPPLUSCTL` das Betriebssystem bereit: diese beziehen sowohl ihren Kernel als auch ihre jeweiligen Root- (`/`) und weiteren System-Dateisysteme (`/bin, /lib, /usr, /sbin`) via NFS aus `/cluster/os` von der `EKPPLUSCTL`. Dazu muß der für die Knoten bereitliegende Kernel mit der Option `CONFIG_ROOT_NFS` übersetzt sein[28]. Auch muß der Kernel den NFS-Client-Code fest eincompiliert enthalten und nicht etwa als Kernel-Modul. Zur Übertragung des Kernels (und des Laders für den Kernel) wird TFTP (*Trivial File Transfer Protocol*, ein stark vereinfachtes FTP) verwendet. Dabei kommen die boot-fähigen Netzwerkkarten der Knoten zum Zug: diese enthalten nämlich in ihrer Firmware auch einen TFTP-Client. Die Root-Verzeichnisse für die Knoten werden mittels eines Skipts aus einer Vorlage (`/cluster/nodes/MASTER-192.168.101.X`) generiert. Die Vorlage enthält das Root-Verzeichnis eines (gedachten) Knotens mit dem Namen `EKPPLUS0_XXX_` und der IP-Adresse 192.168.101._XXX_. Das selbstentwickelte Skript `/cluster/nodes/setup-node` kopiert die Vorlagen in das neue Root-Verzeichnis des Knotens und ersetzt `_XXX_` durch die entsprechende Knotennummer (1-18).

Außerdem ist die Kontrollmaschine für die Benutzerverwaltung zuständig. Hier werden neue Benutzer eingetragen; der YP-Server, der die clusterweite Authifizierung übernimmt, läuft ebenfalls hier. Schließlich exportiert der Kontrollrechner noch die clusterweiten Loginverzeichnisse (`/users/<experiment>/<benutzer>`) der Benutzer.

Als dritte wichtige Aufgabe ist der Kontrollrechner für die Verteilung der Jobs auf die Knoten zuständig: hier läuft der Server und der Scheduler des in Kapitel 4.2.4 beschriebenen OpenPBS Batchsystems.

4.4.10 Firewall/NAT-Maschine `EKPPLUSNAT`

Da die Firewall/NAT-Maschine für Gigabit-Verbindungen ausgelegt ist und somit 30000 Pakete pro Sekunde analysieren können muß, wurde entsprechend leistungsfähige Hardware angeschafft. Ein Pentium 4 mit 1800 MHz und 512 MB Arbeitsspeicher filtern den Netzwerkverkehr und sorgen für die Internetanbindung der Knoten. Für die Anbindung an das Universitätsnetz wird derzeit noch eine Fast Ethernet-Verbindung genutzt, eine Schneider & Koch Glasfaserkarte vom Typ SK-9D21 ist jedoch bereits eingebaut und betriebsbereit, sobald die Glasfaseranbindung vom Rechenzentrum freigeschaltet wird. Der Desktop-Bereich und das äußere EKPplus-Netz werden über die Onboard-Intel Pro/100 Fast Ethernet-Karte angebunden, der NAT-Link ins innere Netz der EKPplus geht über eine Intel Pro/1000 XT Gigabit Ethernet-Karte.

Da auf der Firewall/NAT-Maschine kein Benutzerbetrieb erlaubt ist, andererseits aber höchste Sicherheitsanforderungen gestellt werden, wird diese Maschine unter OpenBSD 3.2 (beta) betrieben. OpenBSD hat mit einem einzigen *Remote Root Exploit*[29] in den letzten sechs Jahren eine unübertroffene Sicherheitsgeschichte [44]. Außerdem ist die Formulierung der Filterregeln mit dem in OpenBSD integrierten überaus leistungsfähigen Paketfilter PF

[28] Im Linux-Kernel-Konfigurationsmenü unter *Filesystems* → *Network file systems* → *Root filesystem on NFS*.

[29] Unter *Exploit* versteht man einen Angriff auf einen Rechner unter Ausnutzung eines Sicherheitsloches in der installierten Software. *Remote* bezeichnet Angriffe, die keine normale (unprivilegierte) Zugangskennung für den Rechner voraussetzen.

sehr elegant möglich.

Zur Erhöhung der Ausfallsicherheit wurden zwei 40 GB IDE-Platten als Software-RAID1 (Spiegeln) konfiguriert, wobei die in OpenBSD integrierte RAIDframe-Software [45] zum Einsatz kommt.

Konfiguration der Firewall

Um den Paketfilter zu aktivieren, muß zunächst die Fähigkeit des Kernels, IP-Pakete weiterzuleiten, angeschaltet werden. Dies geschieht mit einem Eintrag in `/etc/sysctl.conf`:

```
net.inet.ip.forwarding=1
```

Des weiteren ist in `/etc/rc.conf` die Variable `pf=YES` zu setzen, damit der Filter beim Systenstart automatisch aktiviert wird. Die Filterregeln selbst stehen in `/etc/pf.conf`. Einige der Regeln sollen hier erläutert werden; die vollständige Liste befindet sich im Anhang A.6. Beim Lesen der Regeln ist zu beachten, daß OpenBSD andere Bezeichnungen für Netzwerkkarten verwendet als Linux, wo Netzwerkkarten mit `eth0`, `eth1` usw. durchnummeriert werden: bei OpenBSD reflektiert der Name den Hersteller der Karte bzw. des verwendeten Chips. So heißt die Onboard-Intel 100 Mbit-Karte `fxp0`, die beiden Intel Pro/1000 XT-Karten heißen `gx0` (inneres EKPplus-Netz) und `gx1` (Anbindung ans Universitätsnetz); die noch unbenutzte Glasfaserkarte heißt `bge0` und soll später anstelle von `gx1` benutzt werden.

Nachdem das Forwarding aktiviert ist, müssen die Netzwerkkarten noch in einer sogenannten *Bridge*, einer logischen Verbindungsbrücke zwischen Netzwerken, zusammengefaßt und diese aktiviert werden werden. Dazu wird folgende Zeile in die Datei `/etc/bridgename.bridge0` eingetragen:

```
add fxp0 add gx0 add gx1 up
```

Die Filterregeln wurden so formuliert, daß jede Karte als Eingangsfilter funktioniert, d.h. auf der Karte eingehende Pakete werden gefiltert, während Pakete, die auf der Karte ausgegeben werden, nicht gefiltert werden. Dabei wird natürlich an der mit dem Universitätsnetz verbundenene Karte besonders gründlich gefiltert, während zwischen dem Desktop- und dem inneren Netz lediglich gezielt einige Dienste gesperrt werden, die sonst zu Störungen führen könnten.

Am äußeren Interface `gx1` werden zunächst Pakete ausgesondert, die überhaupt nicht an einen Computer des EKP adressiert sind. Dann werden simple Täuschungsversuche abgeblockt, bei denen der Angreifer vorgibt, ein Rechner des EKP zu sein. Dabei nutzt die Firewall das Wissen, daß alle Rechner des EKP auf der *inneren* Seite der Firewall stehen. Behauptet also ein Rechner auf der äußeren Seite, zum EKP zu gehören, so ist dies offenkundig gelogen. Solche *spoofing*-Versuche werden abgewiesen und protokolliert. Die entsprechenden Filterregeln sehen (verkürzt) so aus:

```
block in log quick on gx1 from 129.13.102.0/26 to any
block in log quick on gx1 from 172.22.9.0/26 to any
```

Alsdann wird der Zugriff auf Systemdienste eingeschränkt: nur die Secure Shell sowie die Dienste zur Namensauflösung und zum Zeitabgleich werden freigegeben, sowie Zugang zum

Web- und Mailserver des Instituts (129.13.102.59); der Rest wird unverzüglich abgeblockt und protokolliert:

```
pass  in     quick on gx1 proto tcp          from any to any port ssh \
   flags S/SA keep state
pass  in     quick on gx1 proto { tcp, udp } from any to any port domain \
   flags S/SA keep state
pass  in     quick on gx1 proto { tcp, udp } from any to any port ntp \
   flags S/SA keep state
pass  in     quick on gx1 proto tcp from any to 129.13.102.59 port smtp \
   flags S/SA keep state
pass  in     quick on gx1 proto tcp from any to 129.13.102.59 port www \
   flags S/SA keep state
block in log quick on gx1 all
```

An den Karten `fxp0` (Desktop-Maschinen) und `gx0` (inneres EKPplus-Netz) wird lediglich verhindert, daß Netzwerklaufwerke und vor allem DHCP-Anfragen ihr jeweiliges Netz verlassen.

Die Funktionsweise von Firewall und NAT ist in Kapitel 4.3.3 ausführlich beschrieben.

Konfiguration der NAT

Der NAT-Service wird konfiguriert, indem man die Regeln zur Adressumschreibung in der Datei `/etc/pf.conf`, die auch zur Konfiguration der Firewall genutzt wird, einträgt. Im Falle der EKPplus sind zwei Regeln notwendig, eine zum Erreichen des Desktop-Netzes und eine für Verbindungen nach außen:

```
nat on fxp0 from 192.168.0.0/16 to 129.13.102.0/26 -> 129.13.102.40
nat on gx1  from 192.168.0.0/16 to any -> 129.13.102.40
```

In beiden Fällen werden Adressen aus dem 192.168.0.0/16-Subnetz (das innere Netz der EKPplus) umgeschrieben auf die Adresse der NAT-Maschine, 129.13.102.40 (rechts des Pfeiles in den Regeln).

Die `EKPPLUSNAT` ist leistungsstark genug, um einen direkten Datenimport von außen (Fermilab, CERN) auf den Fileserver zu ermöglichen, ohne Umweg über ein Portal.

4.4.11 Switches

Bei 37 benötigten Netzwerkanschlüssen (“*Ports*”) bereits in Grundausbau, und in Hinblick auf Erweiterbarkeit in Bezug auf Portale, Datenspeicher und Knoten ist eine sorgfältige Auswahl der aktiven Netzkomponenten zur Verschaltung der Rechner unerläßlich.

Bei aktiven Netzwerkkomponenten unterscheidet man zunächst einmal Switches und Hubs. Ein Hub gibt alle Datenpakete, die er an einem Port empfängt, einfach auf allen anderen Ports aus. Der Nachteil dieses Verfahrens liegt auf der Hand: statt nur die Leitung zum eigentlichen Zielrechner des Datenpakets zu belasten, werden alle Leitungen verstopft, also auch Leitungen zu Rechnern, die mit dem ursprünglichen Datenpaket gar nichts zu tun

haben und es nach Empfang sofort verwerfen. Im Gegensatz dazu verfügt der Switch über einen Mechanismus, dies zu vermeiden und Datenpakete nur auf den zum Zielrechner führenden Port auszugeben: der sogenannte ARP-Cache speichert die MAC-Adressen[30] der an die Ports angeschlossenen Rechner. Auf diese Weise vermeidet der Switch unnötiges Versenden von Datenpaketen. Bei unbekannten Zieladressen genügt eine ARP-*who-has*-Anfrage auf allen Ports, um den korrekten Zielport zu ermitteln. Die Daten selbst werden nur an einen Port übertragen. Bis vor ein bis zwei Jahren waren Switches wesentlich teurer als Hubs; heute ist der Preisunterschied relativ gering, und Switches haben Hubs weitgehend verdrängt.

Im EKPplus-Cluster treten zwei Netzwerkgattungen in Erscheinung: zum einen das weitverbreitete 100 BaseT-Netzwerk, auch Fast Ethernet genannt, mit einer Übertragungsrate von 100 Mbit/s; zum anderen das noch relativ seltene 1000 BaseT-Netzwerk, auch bekannt als Gigabit Ethernet, mit 1 Gbit/s = 1000 Mbit/s. Beide Formen sind wechselseitig kompatibel, das heißt man kann eine Gigabit-Karte an einen 100 Mbit-Switch anschließen bzw. umgekehrt eine 100 Mbit-Karte an einen Gigabit-Switch[31]. Natürlich limitiert in beiden Fällen die 100 Mbit-Komponente die Übertragungsrate.

Zu Gigabit Ethernet sei noch angemerkt, daß dies wahrscheinlich die letzte Kupferbasierte Netzwerktechnologie ist. Bereits bei Gigabit Ethernet sind die Qualitätsanforderungen an die Kabel sehr hoch: schon geringe Unterschiede in den Kabellängen der einzelnen Adern führen durch den extrem hohen Takt zu Übertragungsfehlern aufgrund von Laufzeitunterschieden. Netzwerktechnologien mit noch höheren Übertragungsraten werden, soweit sie sich nicht auf wenige Meter beschränken, auf Glasfasern basieren, wie etwa die in Storage Area Networks (SAN) eingesetzte FibreChannel-Technologie.

Nach sorgfältigen Vergleichen und Beratung durch das Rechenzentrum der Universität Karlsruhe wurden die folgenden Netzwerkkomponenten angeschafft:

- **Allied Telesyn AT-8224XL** [46].
  Ein Fast Ethernet-Switch mit 24 Ports und zwei (derzeit ungenutzten) Erweiterungsslots. Bauform 19", Höhe 1,5 HE. Dieser Switch, der auch vom Rechenzentrum verwendet wird, bedient das äußere Netz und ist Teil des Desktop-Netzes.

- **Hewlett-Packard ProCurve 4108** [47].
  Ein modularer Switch mit acht Einschüben, von denen zwei mit 6 Port Gigabit-Moduln und zwei mit 24 Port 100 Mbit-Moduln belegt sind (12 Gigabit-Ports und 48 100 Mbit-Ports). Die Backplane des Switches leistet 36 Gbit. Bauform 19", Höhe 4 HE. Dieser Switch bildet das innere Netz der EKPplus.

Die folgende Tabelle zeigt die derzeitige Port-Belegung der Switches. Dabei gibt die erste Spalte den Rechner an, die weiteren Spalten, wieviele Ports der Rechner im jeweiligen Netz

[30] Auf jeder Netzwerkkarte ist eine weltweit eindeutige "Seriennummer" gespeichert, die sogenannte MAC-Adresse, oft auch als "Hardwareadresse" der Netzwerkkarte bezeichnet. Diese umfaßt 6 Bytes, was einen Adressraum von über 10^{12} MAC-Adressen ergibt. Hersteller von Netzwerkkarten erhalten verschiedene Adressblöcke zugewiesen.

[31] Eventuell kann es nötig sein, die Gigabit-Komponenten manuell in den 100 Mbit-Modus zu schalten; meist ist dies jedoch nicht nötig, und die maximale Datenrate wird automatisch ausgehandelt – "*auto negotiated*".

belegt. Das äußere Netz wird, wie bereits erwähnt, vom AT-8224XL bedient und das innere vom HP ProCurve.

Rechner	**außen**	**innen 100 Mbit**	**innen Gigabit**	**gesamt**
EKPCDF1	1	3	0	4
EKPCDF2	1	2	0	3
EKPCMS1	1	0	1	2
EKPDELPHI	1	0	1	2
EKPGRID	1	0	1	2
EKPPLUSNAT	1	0	1	2
EKPPLUSCTL	–	0	1	1
EKPFS1	–	1	2	3
Knoten (18 Stk.)	–	18	0	18
Summe	6	24	7	37
Vorhanden	24	48	12	84
Frei	18	24	5	47

Die Netzwerkkomponenten sind auf Zuwachs gekauft: die insgesamt 72 Fast Ethernet-Ports sind nur zu rund 40 % belegt, die Gigabit Ethernet-Ports zu 60 %. Somit bleiben bei der Anschaffung weiterer Fileserver, Portale oder Knoten noch genug Reserven. Außerdem ist es möglich, den HP ProCurve mit bis zu vier weiteren Moduln aufzurüsten. Damit sind weitere 24 Gigabit-Ports oder 96 100 Mbit-Ports oder eine Kombination daraus möglich.

4.4.12 Überwachung und Wartung

Konsolenumschalter

Der EKPplus-Cluster besteht derzeit aus 26 Rechnern. Um diese Rechner warten zu können – speziell bei Netzwerkproblemen –, wurden vier Konsolenumschalter vom Typ KVT-K1080 angeschafft. Diese sind kaskadierbar, können also in Serie geschaltet werden, wodurch bis zu 32 Rechner mit einem Bildschirm und einer Tastatur gewartet werden können. Die Konsolenumschalter verfügen über ein *On-Screen*-Menü, in dem die Namen der angeschlossenen Rechner eingetragen werden können, was die Bedienung des Umschalters sehr vereinfacht. Bei der Installation der an die Konsoleumschalter angeschlossenen Rechner ist darauf zu achten, daß alle an derselben Tastatur angeschlossen sind, also dasselbe Tastaturlayout (US) haben.

Unterbrechungsfreie Stromversorgung

Der Fileserver, die Kontrollmaschine und die Firewall sind an eine unterbrechungsfreie Stromversorgung (USV) von American Power Conversion (APC) angeschlossen, die Stromausfälle von bis zu fünf Minuten überbrücken kann und die angeschlossenen Rechner dann kontrolliert herunterfährt. Gerade für den Fileserver ist das unerläßlich, da hier große Mengen Daten gelagert werden, von denen keine regelmäßigen Sicherungskopien gemacht werden können.

Überwachung der Raumtemperatur

Ohne funktionierende Klimatisierung kann der Betrieb der EKPplus nicht aufrecht erhalten werden, da binnen weniger Stunden eine Raumtemperatur von über 35° C erreicht würde, wodurch die Rechner überhitzen würden. Zur Überwachung der Raumtemperatur an strategischen Positionen im Raum (Abbildung 4.14) wird ein Conrad Digital-Thermometer 304 mit vier Meßfühlern verwendet. Dieses ist an die serielle Schnittstelle der `EKPDELPHI` angeschlossen und wird über Fleximon (Kapitel 4.2.5) ausgelesen. Das zugehörige Plugin ist unter [41] verfügbar. Steigt eine der Temperaturen über einen Schwellwert (je nach Position zwischen 15° C und 25° C), so wird dies über Email an die Administratoren gemeldet. In jedem Fall werden die Temperaturen protokolliert und sind über WWW einsehbar [48].

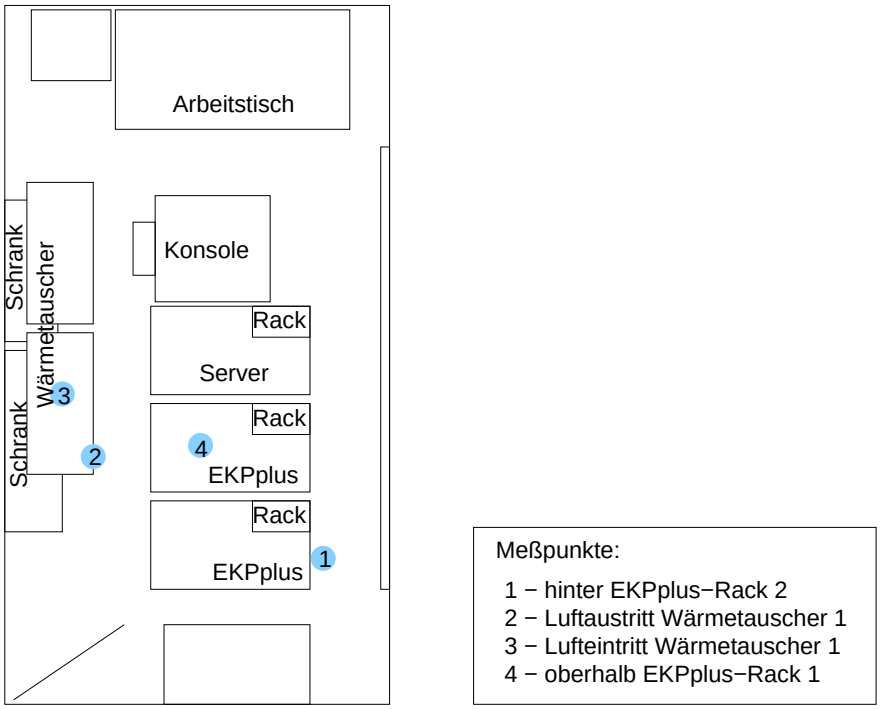

Abbildung 4.14: Temperaturmeßpunkte im klimatisierten Rechnerraum.

4.5 Skalierbarkeit des Clusters

Erweiterungen der EKPplus lassen sich in drei Kategorien unterteilen: die Anschaffung weiterer Knoten, weitere Portale und weitere Fileserver. Die Möglichkeiten und Beschränkungen des Wachstums in diesen drei Bereichen werden im folgenden diskutiert.

Die Netzwerkanbindung stellt sich in keinem Ausbauszenario als problematisch dar. Der HP ProCurve ist derzeit nur halb ausgebaut, und von den Ports dieser Ausbaustufe ist kaum mehr als die Hälfte belegt. Eine Verdreifachung des derzeitigen Clusters stellt aus dieser Sicht keine Herausforderung dar, es müssen lediglich einige Switch-Moduln nachgekauft werden.

Auch der Switch selbst ist ausreichend dimensioniert, die Backplane mit 36 Gbit ist in der Lage, realistische Lastszenarien bei Vollausbau zu bewältigen.

Die beiden 19"-Schränke, die derzeit die EKPplus beherbergen, sind vollständig belegt. An Stellfläche ist noch Platz für ein weiteres Rack mit üblicherweise 40 HE. Dies ermöglicht die Unterbringung von:

- ca. 20 Rechenknoten a 2 HE **–oder–**
- ca. 10 Fileservern a 4 HE **–oder–**
- ca. 10 Portalen a 4 HE **–oder–**
- einer Kombination, z.B. 2 Portale, 2 Fileserver und 12 Knoten

Wird die Zahl der Rechenknoten als nicht ausreichend angesehen, so muß die Anschaffung wesentlich teurer 1 HE-Rechenknoten in Betracht gezogen werden. Dies stellt allerdings auch eine große Herausforderung hinsichtlich Wiederverwendbarkeit der Gehäuse dar.

4.5.1 Grenzen des Wachstums

Die limitierte Kälteleistung der Klimaanlage stellt die größte Herausforderung für den Ausbau der EKPplus dar. Bereits jetzt beträgt die mittlere Raumtemperatur bei maximaler Kühlung 25° C. Jeder weitere Knoten erzeugt ca. 100 W Abwärme, Fileserver und Portale bis zu 400 W. Es erscheint dringend angeraten, bei Anschaffung eines weiteren Racks unbedingt ein selbstgekühltes Modell zu wählen.

Bis jetzt ist noch nicht abzusehen, in welcher Kategorie als nächstes Bedarf besteht, wahrscheinlich jedoch wird der Aufbau der CMS-Software-Gruppe die baldige Anschaffung eines weiteren, leistungsstarken CMS-Portals nach sich ziehen. Wie bald ein neuer Fileserver benötig wird, hängt stark von der tatsächlich erzielten Luminosität des Tevatron ab.

Die jetztige logische Netzwerkstruktur beschränkt die EKPplus auf 199 Rechenknoten (192.168.101.1 bis 192.168.101.199; .200 bis .254 sind für Portale und andere Maschinen reserviert). Dies wird aufgrund der baulichen Voraussetzungen (Raumgröße, Klima-Leistung) als ausreichend angesehen.

Kapitel 5

Beispielanalyse $J/\Psi \to \mu^+\mu^-$

Die in diesem Kapitel vorgestellte Analyse des Prozesses $J/\Psi \to \mu^+\mu^-$ soll zeigen, daß die entwickelten Werkzeuge (Software wie Hardware) funktionieren und grundsätzlich dazu in der Lage sind, physikalische Analysen durchzuführen. Neue Physik hingegen ist in diesem Kapitel nicht zu erwarten: die Eigenschaften des J/Ψ-Teilchen sind so genau bekannt, daß sie zur Kalibration des Magnetfeldes und des Teilchen-Energieverlusts in der Detektor-Materie benutzt werden [49]. Vielmehr zeigt dieses Kapitel, daß sowohl die in Kapitel 3 vorgestellte Software als auch der in Kapitel 4 beschriebenen EKPplus-Linux-Cluster voll funktionstüchtig und einsatztauglich sind.

Die gesamte hier vorgestellte Analyse (Monte Carlo-Generator, Rekonstruktions- und Analyseprogramme) basiert auf dem CDF II-Software-Release 4.6.2.

5.1 Daten-Sample

Ausgewählt zur Analyse wurde ein Run Set mit 239.25 nb^{-1} Daten aus dem "A"-Stream. Dieser *Express Stream* beinhaltet unter anderem die folgenden Prozesse [50]:

- $J/\Psi \to \mu\mu$ (Wirkungsquerschnitt nach Level 3-Trigger: 5 nb)
- $W \to e\nu$ (3 nb)
- $W \to \mu\nu$ (5 nb)
- $Z \to ee$ (<1 nb)
- $Z \to \mu\mu$ (<1 nb)

Dabei wird mindestens ein isoliertes Lepton gefordert. Einzelheiten über die Trigger in *Stream A* können in [51] nachgelesen werden. Der für diese Analyse besonders wichtige $J/\Psi \to \mu\mu$ Trigger verlangt mindestens ein Myon mit mehr als 1.35 GeV Energie (die Nachweisschwelle der Myonkammern).

Die 239.25 nb^{-1} enthalten 22551 Ereignissen in 6.1 GB an prozessierten Dateien. Dies entspricht einer mittleren Ereignisgröße von 270 kB. Die Ereignisse wurden als Rohdaten importiert und wie auch die im folgenden beschriebenen Monte Carlos auf dem EKPplus-Cluster rekonstruiert und analysiert.

5.2 Monte Carlo-Produktion

Als Referenz wurden mit dem PYTHIA-Generator (Version 6.203) [52] J/Ψ-Monte Carlo-Ereignisse generiert und anschließend durch die auf GEANT [53] basierende Detektorsimulation verfolgt. Beide Teilprogramme sind in das CDF II-Standard-Monte Carlo-Programm `cdfSim` integriert.

Zur Verbesserung der Effizienz, vor allem beim Erzeugen von J/Ψ aus b-Zerfällen, wurde zwischen Generator und Simulation ein Filter integriert, der sicherstellt, daß auch tatsächlich ein J/Ψ vorhanden ist und daß die Myonen im Akzeptanzbereich des Detektors liegen. Ersteres ist beim Generieren des $B \to J/\Psi$ Samples wichtig; die Einhaltung des Akzeptanzbereichs ist in jedem Fall zu berücksichtigen, denn die Myonkammern können Myonen mit einem Transversalimpuls $p_T < 1.5$ GeV nicht nachweisen. Auch ist die Pseudorapidität auf $|\eta| < 1$ beschränkt, da nur in diesem Zentralbereich die geforderte Driftkammerspur rekonstruierbar ist. Entsprechend werden unmittelbar nach dem Generator Ereignisse, die diesen Kriterien ($\exists\ J/\Psi$; $p_{T,\mu} > 1.5$ GeV; $|\eta_\mu| < 1$) nicht entsprechen, verworfen. Dabei ist zu beachten, daß *beide* Myonen die Kriterien zu erfüllen haben.

Die Detektorsimulation von CDF II benutzt das GEANT 3-Paket (Version 3.21/13), um die vom Physik-Generator erzeugten Teilchen durch den Detektor zu verfolgen und ihre Wechselwirkung mit den Materialien und dem Magnetfeld zu simulieren. Das Ergebnis der Simulation sind Dateien, die sich im Idealfall nur durch die zusätzlich vorhandene "Wahrheit" über die Ereignisse von gemessenen Daten unterscheiden.

Die Detektorsimulation ist ein extrem CPU-intensiver Prozess (mehrere CPU-Sekunden pro Ereignis). Daher ist es wichtig, den oben beschriebene Filter *vor* der Simulation einzubinden, um die CPU-Zeit für eigentlich nutzlose Ereignisse zu sparen.

5.2.1 Generierte Prozesse

Als Generator kam das vom CDF II verwendete PYTHIA Version 6.203 zum Einsatz. Als Monte Carlo wurden zwei Prozesse generiert: zum einen prompte J/Ψ, zum anderen solche aus b-Zerfällen. In beiden Fällen wurde der Zerfall des J/Ψ in zwei Myonen auf Generatorebene erzwungen, indem die anderen Zerfallskanäle des J/Ψ abgeschaltet wurden. Die genauen Einstellungen des Generators finden sich in Anhang A.7.

Prompte J/Ψ

PYTHIA kennt vier Kanäle, in denen prompte J/Ψ entstehen [54]:

- $gg \to J/\Psi g$ (Prozess 86)
- $gg \to J/\Psi\gamma$ (Prozess 106)
- $g\gamma \to J/\Psi g$ (Prozess 107)
- $\gamma\gamma \to J/\Psi\gamma$ (Prozess 108)

Die genaue PYTHIA-Konfiguration findet sich im Anhang A.7.1. Die Abbildungen 5.1 und 5.2 zeigen die Verteilungen von η, ${p_T}^2$ und η über ${p_T}^2$ der J/Ψ bzw. der Myonen, wie sie der Generator produziert, also noch vor dem Filter.

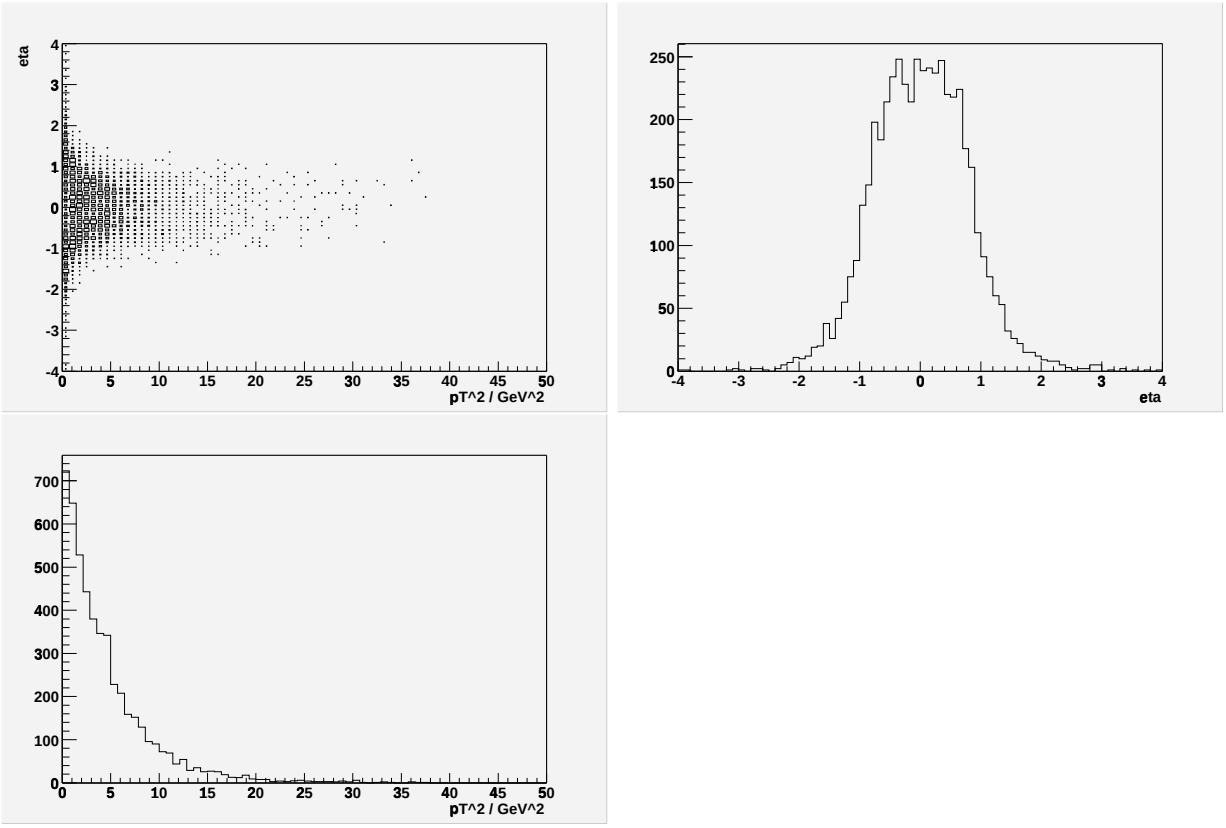

Abbildung 5.1: η und ${p_T}^2$ prompter J/Ψ aus dem Generator (ungefiltert).

Es ist bereits aus den Abbildungen ersichtlich, daß der Filter lediglich einen relativ kleinen Anteil der produzierten prompten J/Ψ akzeptieren wird. Tatsächlich beläuft sich der Anteil auf 10.5 %. Da der Generator sehr viel schneller ist als die Detektorsimulation, bedeutet dies eine Rechenzeitersparnis von fast 90 %.

J/Ψ aus b-Zerfällen

J/Ψ aus b-Zerfällen werden erzeugt, indem zunächst einmal b-Quarks erzeugt werden, und dann nur diejenigen Ereignisse verwendet werden, in denen ein J/Ψ vorkommt. Es wird folgender PYTHIA-Prozess verwendet:

- $gg \to Q\bar{Q}$ (Prozess 82)

Dabei wurde b als zu produzierendes Quark angegeben. Der Zerfall $t\bar{t} \to Wb$ mit J/Ψ-Produktion in der b-Zerfallskette spielt aufgrund der geringen Top-Produktionrate keine Rolle. (Tatsächlich würden so produzierte J/Ψ aufgrund der anderen Topologie von Top-Ereignissen das Ergebnis verfälschen.) Dabei wurde der Zerfall von $J/\Psi \to \mu^+\mu^-$ erzwungen,

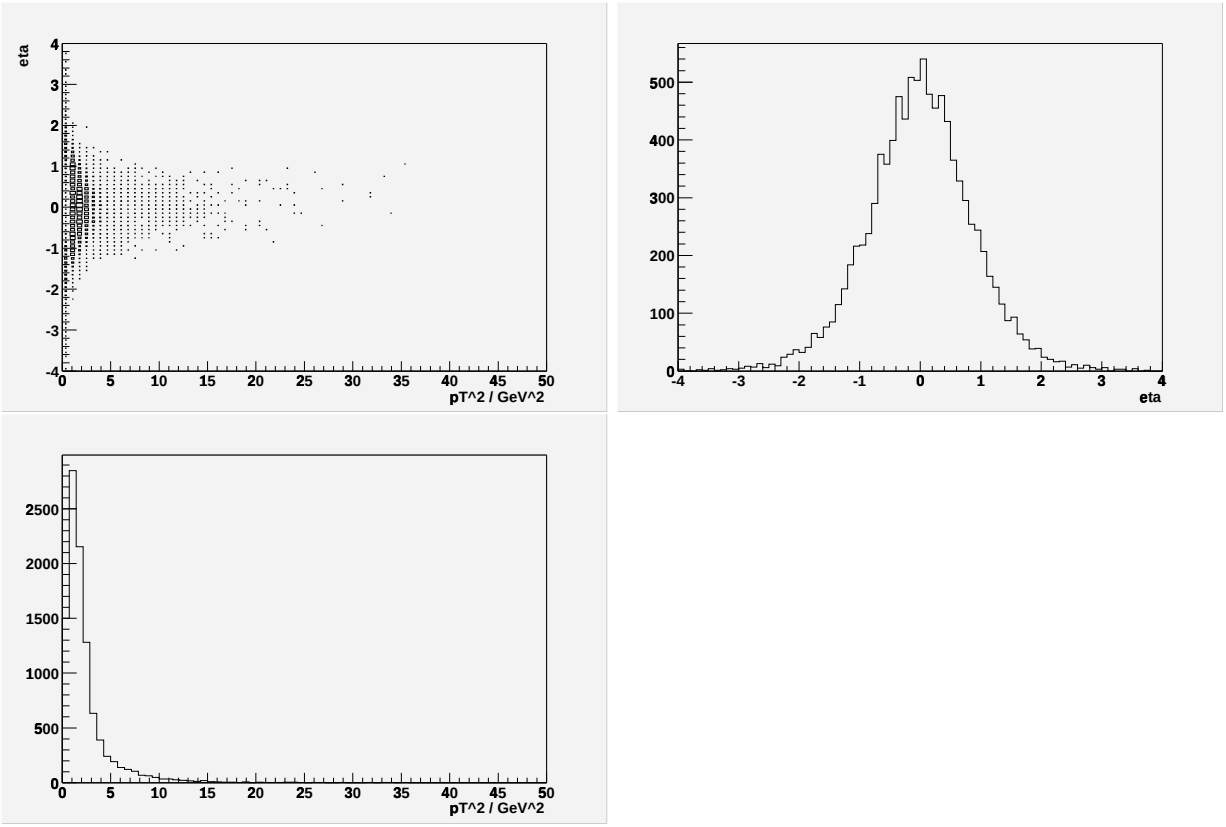

Abbildung 5.2: η und ${p_T}^2$ von Myonen aus prompten J/Ψ (nach dem Filter).

alle anderen Zerfallskanäle, insbesondere der hadronische, wurden abgeschaltet. Zu beachten ist, daß keine Eingriffe in den Zerfall der b-Hadronen vorgenommen wurden, da ein Erzwingen von $B \to J/\Psi$ unweigerlich zu *zwei* J/Ψ pro Ereignis führen würde, da der Generator die b-Quarks paarweise erzeugt.

Die genaue PYTHIA-Konfiguration findet sich im Anhang A.7.2. Die Abbildungen 5.3 und 5.4 zeigen wieder die Verteilungen von η, ${p_T}^2$ und η über ${p_T}^2$ der J/Ψ bzw. der Myonen, wie sie der Generator produziert, also ebenfalls vor dem Filter.

Der Anteil der vom Filter akzeptierten Ereignisse ist mit 0.4 % noch wesentlich kleiner als bei den prompten J/Ψ. Dadurch ist das Erzeugen dieser Monte Carlos ein sehr CPU-intensives Unterfangen. Auch mit PYTHIA-Steuerparametern (`ckin`) ist keine Verbesserung zu erzielen, denn hier kann lediglich die Energie und die Winkelverteilung des *primären* Prozesses (also der $Q\bar{Q}$-Produktion) eingestellt werden, nicht jedoch der von späteren Zerfallsprodukten.

5.3 Rekonstruktion

Sowohl Daten als auch Monte Carlos wurden mit dem CDF-Standard-Programm zur Rekonstruktion, `ProductionExe` (Version 4.6.2) bearbeitet. Dabei wurde gegenüber der Stan-

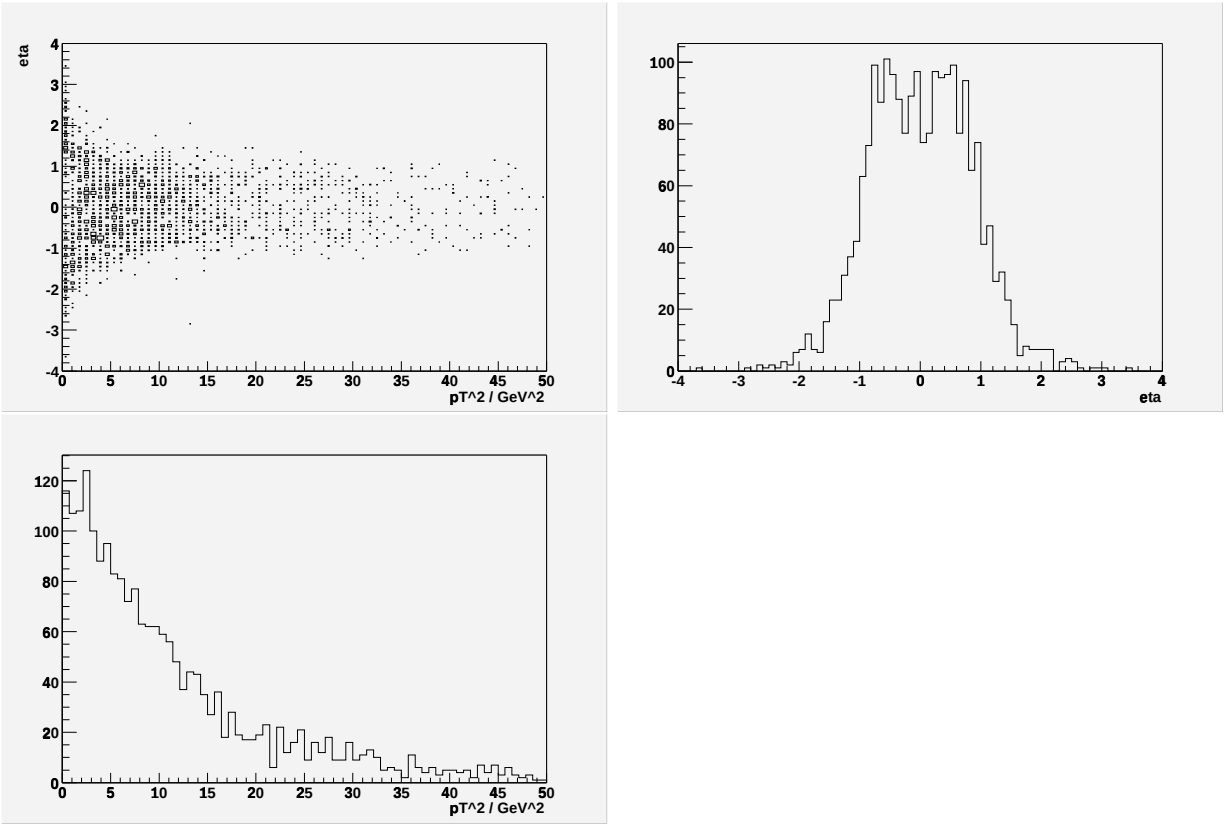

Abbildung 5.3: η und ${p_T}^2$ von J/Ψ aus b-Zerfällen aus dem Generator. (Ungefiltert.)

dardkonfiguration lediglich das Outside-In-Tracking auf die Karlsruher Version umgestellt. Die genaue Konfiguration der Trackingsoftware findet sich in Anhang A.8.

5.4 Die J/Ψ Analyse

Die eigentliche Analyse wurde mit einer um einige Histogramme erweiterten Version des `JPsiModules`-Pakets, welches Teil der CDF II-Softwareinstallation ist, durchgeführt. Dieses Paket ermöglicht die Suche nach J/Ψ und erstellt zahlreiche Histogramme.

5.4.1 Verwendete Schnitte

Das `Jpsi`-Modul des `JPsiModules`-Pakets führt, um J/Ψ zu identifizieren, gemäß den Einstellungen eine Anzahl von Schnitten aus, die im folgenden diskutiert werden. Dabei wird jedes Ereignis getrennt behandelt. Zunächst werden folgende kinematische Schnitte durchgeführt:

- $p_{T_\mu} > 1.5$ GeV
  Für beide Myonen wird ein Mindesttransversalimpuls von $p_T > 1.5$ GeV gefordert. Dies

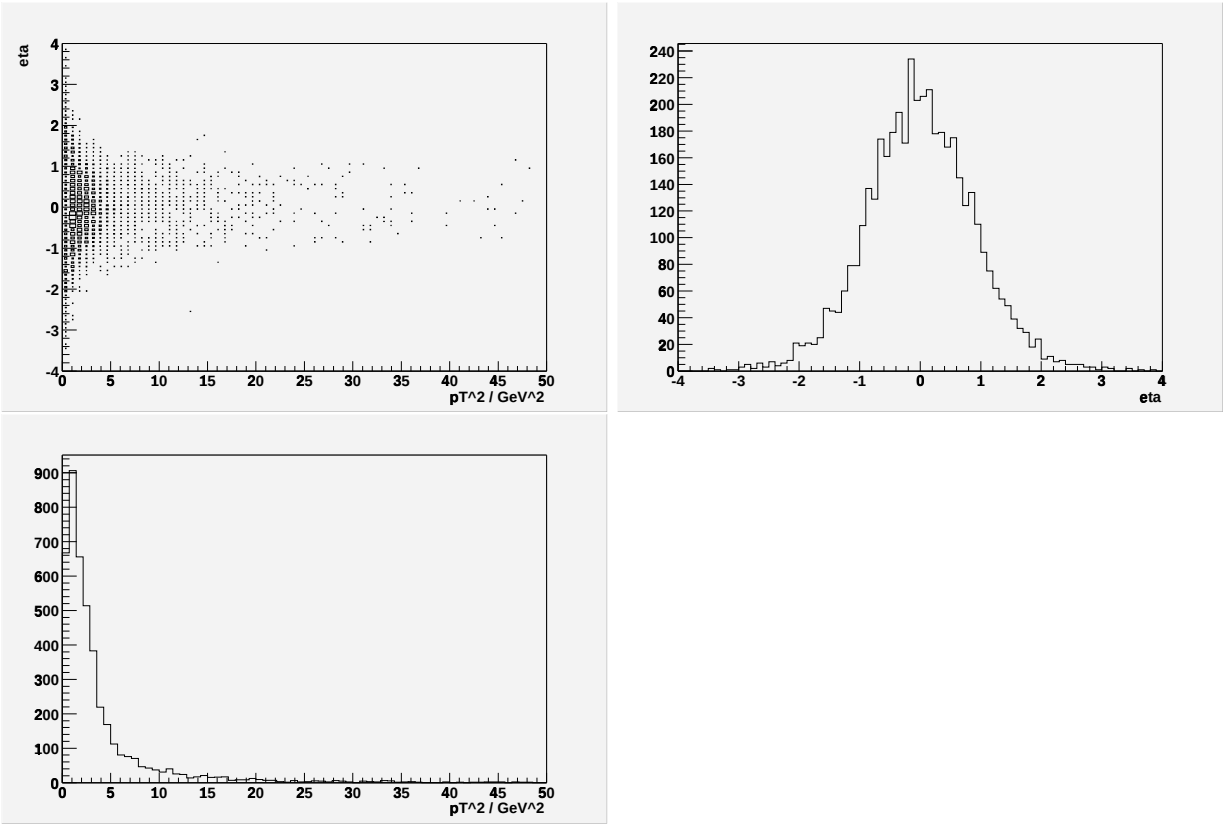

Abbildung 5.4: η und ${p_T}^2$ von Myonen aus J/Ψ aus b-Zerfällen. (Ungefiltert.)

liegt deutlich über dem im Trigger geforderten Wert von 1.35 GeV, der Nachweisschwelle der Myonkammern. Auch ist die Spurrekonstruktion bei größeren Teilchenimpulsen besser.

- $|\eta_\mu| < 1.0$
  Die Spuren müssen im Zentralbereich des Detektors verlaufen, die Pseudorapidität $|\eta|$ für beide Myonen unter 1 liegen, damit die Spuren vollständig in der Driftkammer liegen. (Das Siliziumsystem ließe $|\eta| < 2.0$ zu.)

- $n_\mu \geq 2$
  Bleiben weniger als zwei Myonen übrig, so wird das Ereignis ganz verworfen.

Von jedem der verbleibenden Myonen wird ein Eintrag in einer Myonkammer und eine rekonstruierte Driftkammerspur verlangt. Es werden alle Zweierkombinationen dieser Myonen gebildet und folgenden weiteren Schnitten unterzogen:

- $n_{\mu,\ SiHits} \geq 3$
  Mindestens 3 Hits im Silizium pro Myon werden wegen der Genauigkeit des Vertex gefordert. Vertexfits aus reinen Driftkammer-Spuren sind zu ungenau: für die Bestimmung der Zerfallslänge ist eine präzise Vertexmessung unerläßlich.

- $q_{\mu_1} = -q_{\mu_2}$
  Die beiden Myonen müssen gegensätzliche Ladung haben. Dadurch werden die Fehlkombinationen $\mu^+\mu^+$ und $\mu^-\mu^-$ vermieden.

- 2 GeV $< m_{J/\Psi} <$ 4 GeV
  Die invariante Masse des J/Ψ, bestimmt aus der Summe der Vierervektoren der beiden Myonen, muß zwischen 2 und 4 GeV liegen. (Dieser Schnitt wurde recht weit gewählt, um eine bessere Abschätzung für den Untergrund zu finden.)

Die Myon-Kombinationen, die diesen Kriterien entsprechen, liefern Einträge in zahlreichen Histogrammen.

5.4.2 J/Ψ-Massenpeak

Aus den Vierimpulsen der beiden Myonen wird die invariante Masse des $\mu^+\mu^-$-Paares berechnet. Die Anhäufung bei 3.1 GeV in Abbildung 5.5 weist auf die Existenz von J/Ψ-Mesonen hin. Die Abbildung zeigt die Massenverteilung für Monte Carlo und Daten.

Der Untergrund kommt durch zufällige Kombinationen von echten und misidentifizierten Myonen zustande. Zur Bestimmung des Untergrunds wurde zunächst die Massenverteilung im Bereich von 2.8 bis 3.0 GeV und von 3.2 bis 3.4 GeV an eine Gerade gefittet. Danach wurde die Massenverteilung im Bereich von 2.8 bis 3.4 GeV an diese Gerade mit einer überlagerten Gauß-Verteilung gefittet (siehe Abbildung 5.6). Die Ergebnisse dieses Fits:

$$m_{J/\Psi} = 3.0867 \pm 0.0005\ (stat)\ \text{GeV}$$

Der Literaturwert beträgt 3.09688 ± 0.00004 GeV. Der gemessene Wert ist etwas niedriger als der Literaturwert. Dies weist darauf hin, daß die Detektorbeschreibung noch nicht vollständig ist; die zusätzliche Materie sorgt für einen nicht kompensierten Energieverlust. Außerdem sind kleine Korrekturen zum als homogen angenommenen Magnetfeld nötig. Zu diesem Ergebnis kommt auch [49].

5.4.3 Zerfallslänge

Abbildung 5.7 zeigt die Zerfallslänge L_{xy} des J/Ψ, also die Entfernung des J/Ψ-Zerfallsvertex (gefitteter Vertex der beiden Myon-Spuren) vom Primärvertex, projeziert auf die Richtung der Resultierenden der beiden Myonspuren, in der x-y-Ebene. Die Zerfallslänge korrespondiert somit mit der Lebensdauer des Mutterteilchens des J/Ψ.

Sei $\vec{PV}$ der Primärvertex in r/φ, $\vec{SV}$ der Sekundärvertex (also der gefittete Vertex der beiden Myonen) in r/φ, und seien $\vec{\mu_1}$ und $\vec{\mu_2}$ die 3-Impulse der beiden Myonen, so ergibt sich:

$$L_{xy} = (\vec{SV} - \vec{PV}) \cdot \frac{\vec{\mu_1} + \vec{\mu_2}}{|\vec{\mu_1} + \vec{\mu_2}|}$$

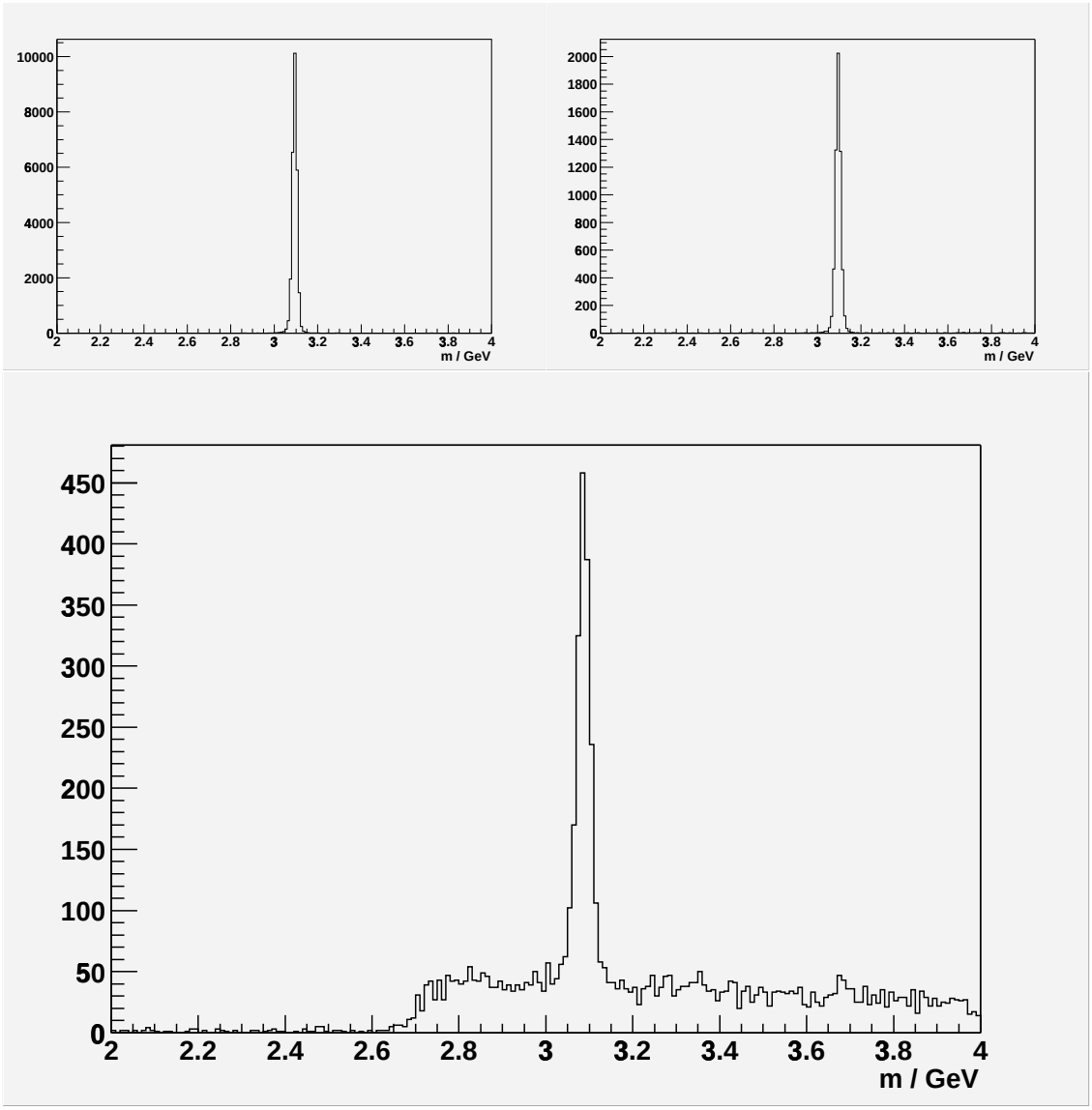

Abbildung 5.5: Der J/Ψ-Massenpeak in der $\mu^+\mu^-$-Paarmassen-Verteilung. Links oben das prompte J/Ψ Monte Carlo, rechts oben das aus b-Zerfällen, unten Daten.

Die beiden oberen Plots in Abbildung 5.7 zeigen das unterschiedliche Verhalten der prompten Komponente und der aus B-Zerfällen. Prompte J/Ψ entstehen direkt am Primärvertex, haben also eine sehr kleine Zerfallslänge. Die Breite des Peaks von 60 μm entspricht der Auflösung des Primärvertex'. Oben rechts hingegen entsteht das J/Ψ aus dem Zerfall eines B-Mesons. B-Mesonen haben eine Lebensdauer von ca. 1.6 ps, mithin haben die entstehenden J/Ψ eine exponentiell verteilte Zerfallslänge von bis zu einigen mm. Die Asymmetrie in der Verteilung erklärt sich aus der Definition von L_{xy}: negative Zerfallslängen treten nur auf, wenn die Projektion von J/Ψ-Vertex auf die Richtung der Myonen negativ ist, wenn also die Myonen in entgegengesetzter Richtung des ursprünglichen B-Mesons fliegen. Dies

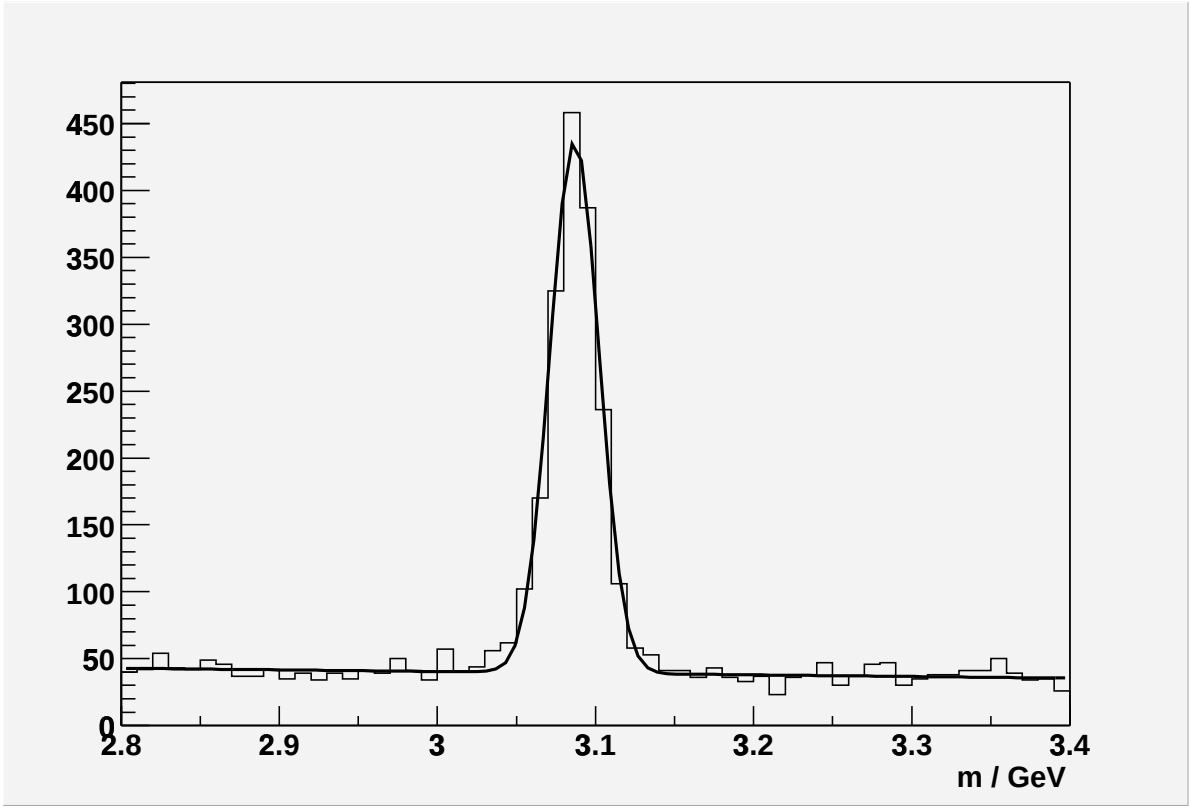

Abbildung 5.6: Der gefittete J/Ψ-Massenpeak bei 3.09 GeV.

ist kinematisch verboten. Negative Zerfallslängen können nur durch Meßungenauigkeiten auftreten.

Die Daten zeigen ein Verhalten wie das Monte Carlo mit prompten J/Ψ; von J/Ψ aus B-Zerfällen ist kaum etwas zu sehen. Allerdings ist deutlich, daß die Verteilung mit 215 μm deutlich breiter ist als die Monte Carlo-Verteilung. Die Rekonstruktion von echten Daten erfolgt also noch nicht mit derselben Präzision wie die von Monte Carlos. Die wahrscheinlichste Ursache hierfür ist eine unvollständige Detektorbeschreibung in der CDF-Software. Dadurch erklärt sich, daß die Rekonstruktion mit echten Daten nicht so gut zurechtkommt wie erwartet; da jedoch Monte Carlos mit derselben Detektorbeschreibung erzeugt werden, treten hier keine Unstimmigkeiten auf, die Rekonstruktionssoftware arbeitet besser. Wahrscheinlich fehlt in der Detektorbeschreibung eine innenliegende Materielage. Die Vielfachstreuung und der Energieverlust an dieser Lage verursachen eine Verschlechterung der L_{xy}-Auflösung [49].

In Abbildung 5.8 ist die Zerfallslänge L_{xy} über der rekonstruierten Masse des jeweiligen J/Ψ-Kandidaten aufgetragen. Die Monte Carlo-Plots zeigen die aus Massenpeak und L_{xy}-Verteilung zu erwartende Verteilung. Interessant ist hingegen die Verteilung bei den Daten (unteres Bild). Hier fällt auf, daß der Untergrund des Massenplots fast ausschließlich prompt ist, während der Massen-Signalbereich (3.10 ± 0.05 GeV) eine Asymmetrie zu positiven L_{xy} zeigt. Allerdings ist die Asymmetrie sehr klein.

Um die J/Ψ aus b-Zerfällen besser hervorzuheben, wird eine Untergrundsubtraktion vor-

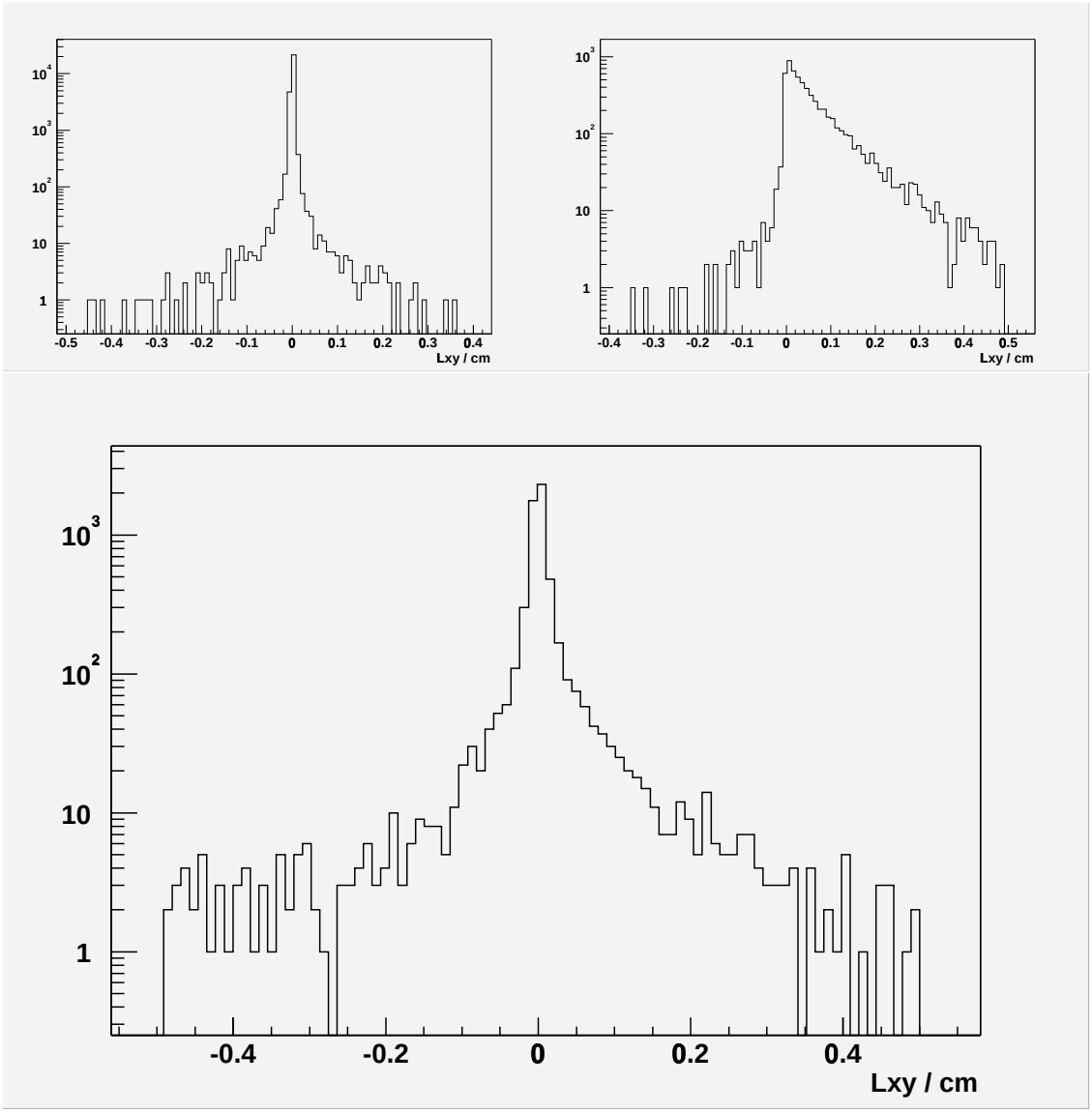

Abbildung 5.7: J/Ψ-Zerfallslänge L_{xy}. Oben Monte Carlo (links prompte J/Ψ, rechts aus b-Zerfällen), unten Daten. Die endliche B-Lebensdauer ist oben rechts deutlich erkennbar.

genommen. Dazu werden die Massenbereiche von 2.8 bis 3.0 GeV und von 3.2 bis 3.4 GeV (die Seitenbänder) als Untergrund betrachtet (vgl. Abbildung 5.6) und die Zerfallslängen der entsprechenden Ereignisse in das Untergrund-Histogramm (Abbildung 5.9 rechts) eingetragen. Die Zerfallslängen der Einträge aus dem Signalbereich 3.05 bis 3.15 GeV füllen das Signal-Histogramm (Abbildung 5.9 links). Der Verlauf des Untergrunds wird für den gesamten Bereich von 2.8 bis 3.4 GeV als linear angenommen. Daraus folgt, daß der zu erwartende Untergrund im Signalbereich gerade $\frac{1}{4}$ des Untergrunds in den Seitenbändern ist. Zur Bestimmung der Untergrund-korrigierten L_{xy}-Verteilung des Signalbereichs wird nun das mit dem Faktor $\frac{1}{4}$ gewichtete Untergrund-Histogramm vom Signal-Histogramm abgezogen. Das

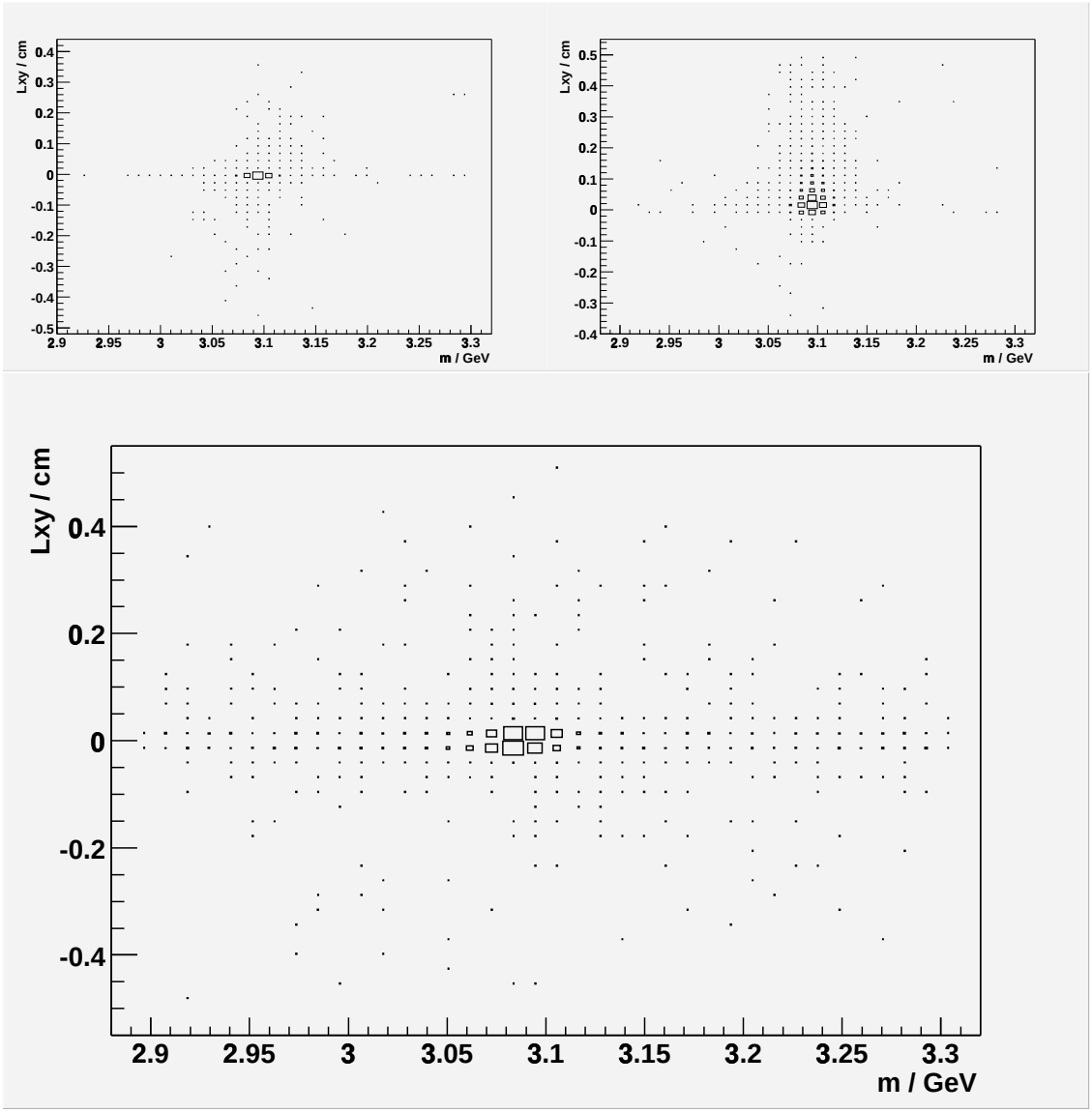

Abbildung 5.8: Rekonstruierte Masse gegen Zerfallslänge. Oben Monte Carlo (links prompte J/Ψ, rechts aus b-Zerfällen), unten Daten.

Ergebnis zeigt Abbildung 5.10. Hier sieht man an der Tendenz hin zu größeren positiven Zerfallslängen durch die endliche B-Lebensdauer einen Beitrag von J/Ψ aus b-Zerfällen.

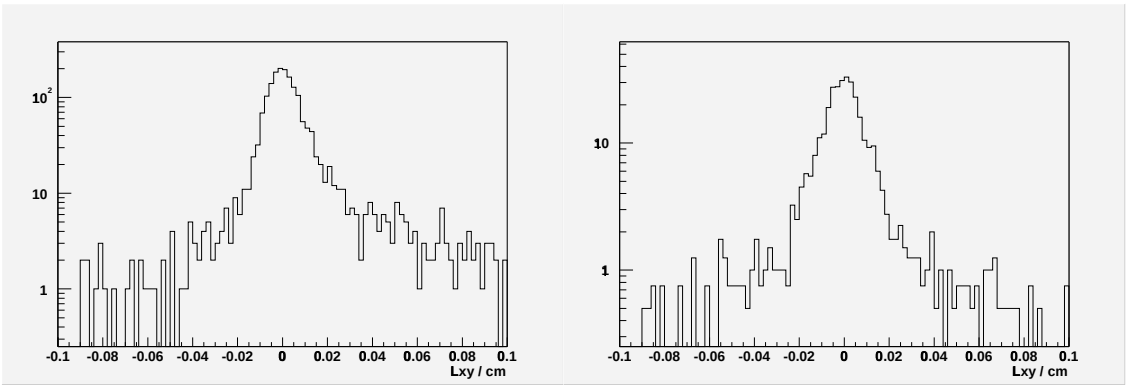

Abbildung 5.9: Zerfallslängenverteilungen für Signal- (links) und Untergrund- (rechts) Massenbereiche.

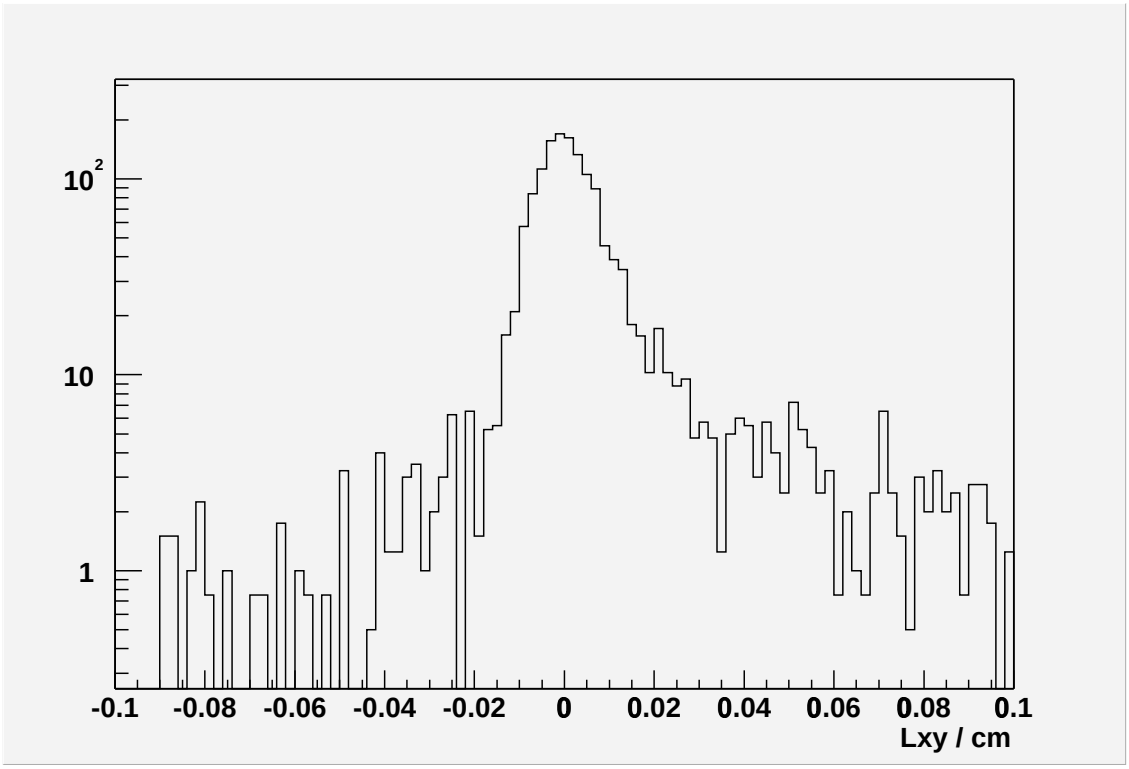

Abbildung 5.10: Untergrund-korrigierte J/Ψ-Zerfallslänge L_{xy}. Die endliche B-Lebensdauer ist deutlich erkennbar.

Kapitel 6

Ausblick

Die vorherigen Kapitel haben die Leistungsfähigkeit sowohl der CDF-Software als auch der angeschafften Hardware demonstriert und gezeigt, daß die Analysekette von CDF einsatzfähig ist und bereit, um komplexere Analysen durchzuführen.

Dennoch ist die Entwicklung nicht abgeschlossen und wird sicher noch mehrere Jahre fortgesetzt. Daher soll dieses Kapitel zum Abschluß noch auf einige zukünftige Entwicklungen hinweisen. Dies betrifft vor allem die Pflege der EKPplus, aber auch das neue Modell für (weltweit) verteiltes Rechnen, das Grid.

6.1 Zukünftige Analysen

Die Beispielanalyse in Kapitel 5 zeigt, daß die CDF-Analysekette funktioniert und bereit ist für umfangreichere, komplexere Analysen wie die in der Einleitung aufgeführten Beispiele aus dem CDF II-Physikprogramm. Noch reicht die aufgezeichnete Luminosität nicht für Präzisionsmessungen aus, doch dies ist nur eine Frage der Zeit. Die Werkzeuge liegen bereit und man kann optimistisch sein, daß CDF II die hochgesteckten Erwartungen bis zum Start des LHC 2007 erfüllt.

6.2 Die (mögliche) Zukunft der EKPplus

Die EKPplus hat sich in der Beispielanalyse und auch sonst sehr gut bewährt. Durch ihr gutdurchdachtes und anpassungsfähiges Gesamtkonzept hat sie sich bereits während der Entstehung dieser Arbeit bei mehreren Umstellungen und Erweiterungen als äußerst flexibel erwiesen, und die Rechenleistung ist im vollen Umfang für Analysen und Monte Carlo-Produktion nutzbar.

Wie eingangs bereits erwähnt, bedarf jedoch *jeder* Rechnercluster der ständigen Erneuerung, um mit dem technischen Fortschritt mitzuhalten. Ohne Erneuerung ist die EKPplus in zwei Jahren veraltet. Schon heute sind die in den Knoten verwendeten Prozessoren (AMD Athlon mit 1400 MHz) nicht mehr im Handel erhältlich. In zwei Jahren werden wahrscheinlich Prozessoren mit 4 GHz erhältlich sein. Auch die Festplattenkapazität hat sich seit der

Anschaffung des Fileservers erheblich vergrößert: heute wären bereits 2.5 TB statt 1.6 TB möglich. In zwei Jahren werden 1.6 TB dem Benutzer ziemlich klein erscheinen.

6.2.1 Neue Prozessoren

Neben den üblichen Steigerungen der Taktfrequenz und Verkleinerung der Leiterstrukturen planen die beiden größten Prozessorhersteller, Intel und AMD, die Einführung neuer Prozessoren, deren Eignung für einen Einsatz in der EKPplus hier kurz beleuchtet werden soll.

AMD Hammer/Opteron

Unter dem Codenamen "Hammer" in den Varianten ClawHammer (für Ein- und Doppelprozessorsysteme) und SledgeHammer (für Vier- und Acht-Prozessorsysteme) bringt der Chiphersteller AMD Ende 2002/Anfang 2003 einen 64 bit-Prozessor mit 2 GHz auf den Markt, der gleichzeitig voll Pentium/Athlon-kompatibel ist (bei gleicher Taktfrequenz jedoch etwa 25% schneller).

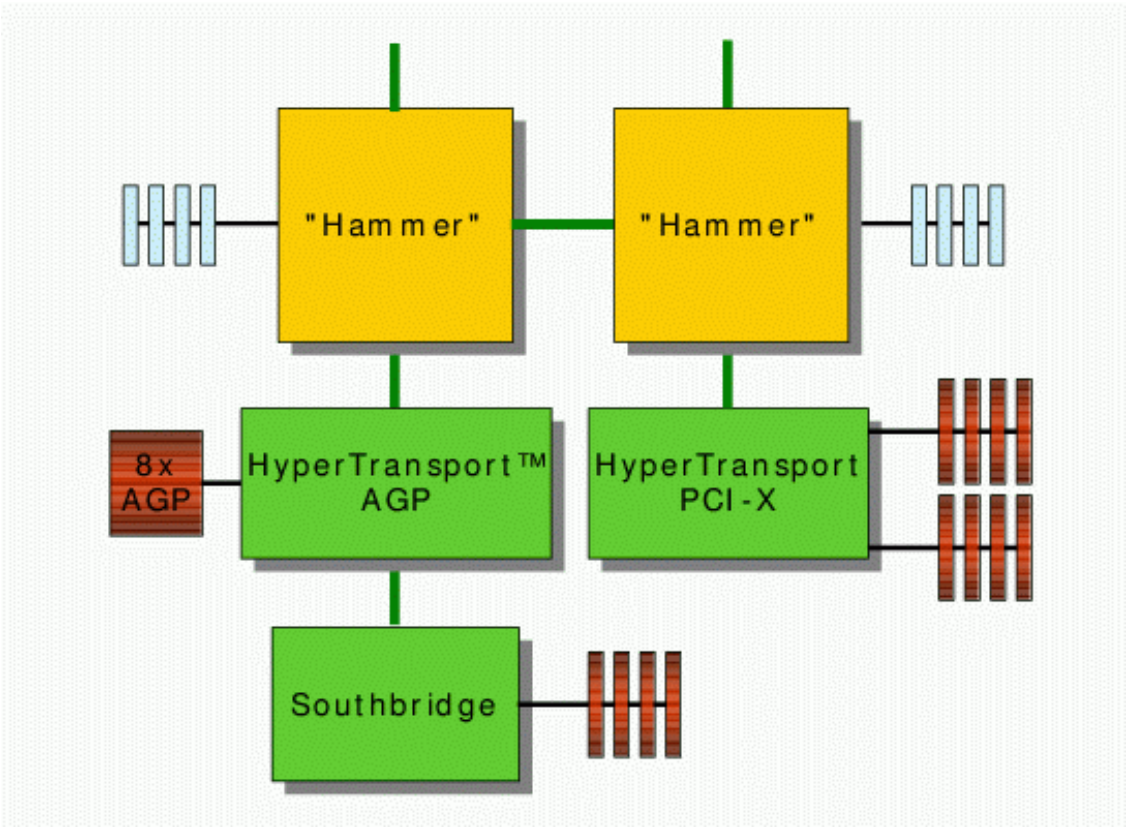

Abbildung 6.1: AMD Hammer als Doppelprozessorsystem.

Der Hammer besitzt zahlreiche neue Merkmale, die eine deutlich höhere Rechenleistung versprechen, unter anderem:

- neue und größere Register: statt der bisherigen 8 32 bit-Register verfügt der Hammer über 16 64 bit-Register. Außerdem wurde die Zahl der 128 bit-SSE-Register auf 16 verdoppelt.
- integrierter Speicher-Kontroller: bis zu acht DDR-RAM-Module mit einer Gesamtspeicherbandbreite von 5.3 GB/s (bei Verwendung von PC2700-Speichermodulen).
- integrierte HyperTransport-Anbindung: das HyperTransport-Protokoll dient der Anbindung an I/O-Busse und weiterer Prozessoren. Ein HyperTransport-Kanal überträgt 2.1 GB/s; je nach Ausführung besitzt der Hammer 1-3 Kanäle (siehe Abbildung 6.1).

Als Ersatz für die derzeit verwendeten Prozessoren ist der Hammer deshalb besonders interessant, weil zunächst einmal aufgrund der Binärkompatibilität keine Umstellung der CDF-Software erforderlich ist. Das ist wichtig, solange der Code nur mit dem KAI C++-Compiler übersetzt werden kann[1]. Ist der Übergang zum GNU C++-Compiler erst einmal geschafft, ist auch der Einsatz als echte 64 bit-Anwendung denkbar. Die zweite Plattform für CDF-Software, Silicon Graphics Workstations, ist schon lange 64 bit-fähig, sodaß hier mit einer nicht allzu aufwändigen Portierung zu rechnen ist. Das größte Problem ist die mit eingebundene kommerzielle Software wie beispielsweise Oracle, da hier die Akzeptanz der neuen Plattform durch den jeweiligen Hersteller abgewartet werden muß.

Als Betriebssystem käme weiter Linux zum Einsatz. Der Kernel ist bereits auf x86-64 portiert; ein erstes Hammer-Testsystem wurde bereits erfolgreich mit einer speziell angepaßten Version von SuSE Linux gebootet [55].

Intel Itanium

Anders als der AMD Hammer ist Intel's 64 bit-Prozessor Itanium eine vollständige Neuentwicklung. Der Bruch mit dem x86-Befehlssatz ermöglicht einige attraktive Neuerungen, deren bedeutendste sicherlich EPIC ist, *Explicitly Parallel Instructionset Computing*. Bisher war das parallele Ausführen mehrerer Befehle *implizit*, das heißt die CPU stellt fest, daß eine Recheneinheit frei ist, und zieht eine entsprechende Anweisung aus dem Programm vor (sogenannte *out-of-order execution*). In Verbindung mit bedingten Sprüngen kann es dabei jedoch zu Problemen kommen, wenn das Sprungziel falsch vorhergesagt wurde (*branch prediction*). Dann muß nämlich die vorgezogene Anweisung rückgängig gemacht werden, was sehr zeitaufwändig ist. Der Ansatz des Itanium ist, diese Form der parallelen Ausführung explizit zu machen, also spezielle Instruktionen dafür einzuführen. Die Compiler sind nun dafür zuständig, den entsprechenden Code zu erzeugen. Die Intelligenz wird gewissermaßen vom Silizium in die Software verlagert, in der Annahme, daß der Compiler mit seinem umfassenderen Kontext (Quellcode) in der Lage ist, fundiertere Entscheidungen zu treffen. Außerdem hat der Compiler mehr Zeit für Optimierungen.

Die Idee des EPIC stammt aus den 80er Jahren, also aus der Zeit vor *just-in-time*-Compilern und Hotspot-Optimierern, wie sie heute verbreitet sind (besonders durch den Erfolg von Java).

[1] Es wäre denkbar, dem KAI C++-Compiler einen 64 bit-fähigen GNU C-Compiler unterzuschieben und dadurch doch die 64 bit-Erweiterungen des Hammer zu nutzen. Es ist allerdings fraglich, ob dann das Zusammenspiel mit den 32 bit-Bibliotheken des Compilers noch funktioniert.

Es gilt als sicher, daß die Teilchenphysik den Itanium bzw. seinen Nachfolger Itanium II im großen Maßstab einsetzen wird. Das CERN erwägt die Anschaffung von 10000 Itanium-Systemen für den LHC. Als Ersatz in der EKPplus eignet sich der Itanium jedoch nicht besonders, da die EKPplus nicht nur den LHC-Experimenten, sondern auch CDF zur Verfügung stehen muß, also eine einheitliche Plattform darstellen muß. Der Itanium ist zwar nominell x86-binärkompatibel, doch ist die Geschwindigkeit in diesem Emulationsmodus sehr schlecht[2]. Dazu sei jedoch angemerkt, daß eine schnelle x86-Emulation nie Intels Ziel war.

6.2.2 Neue Speicher

Auch der Festplatten-Ausbau ist ein Punkt, an dem die EKPplus schon bald Wachstumsbedarf haben wird. Die derzeit verfügbaren 1.4 TB Plattenplatz werden nicht länger als ein halbes Jahr ausreichen.

Derzeit üblich sind IDE-Platten mit ATA-100-Protokoll, das eine Datenrate von 100 MB/s pro Kontroller ermöglicht, was bedeutet, daß sich bis zu zwei Platten diese Bandbreite teilen. Bei den derzeitigen Plattengeschwindigkeiten von ca. 30 MB/s ist das noch unkritisch. Die verwendeten IDE-RAID-Kontroller [42] stellen ohnehin einen Kanal pro Festplatte bereit, sodaß hier auch nicht so bald ein Flaschenhals entsteht. SCSI-Platten kommen aufgrund des sehr viel höheren Preises für die EKPplus nicht in Frage. Zwei Technologien sind jedoch für den Ausbau interessant:

Network Attached Storage

Bei Network Attached Storage (NAS) handelt es sich um fertig konfigurierte Fileserver, die ihr Innenleben (Betriebssystem, Hardware) vor der Außenwelt verbergen und lediglich über einen Netzwerkanschluß verfügen, über den sie Netzwerklaufwerke über NFS und oft auch Windows SMB zur Verfügung stellen. Meist ist NAS über ein Webinterface konfigurierbar. Derzeit ist es allerdings noch schwierig, NAS-Server mit Gigabit- oder gar Doppel-Gigabit-Anbindung zu bekommen.

Der Vorteil von fertigen NAS-Boxen besteht darin, daß der Konfigurations- und Administrationsaufwand minimal ist. Besonders im kommerziellen Umfeld ist dies wichtig, da dort – anders als im universitären Umfeld – Arbeitszeit Geld kostet. Dennoch ist gerade die einfache Konfiguration auch im universitären Umfeld interessant, da Administratoren meist Doktoranden sind, die eigentlich Physik machen wollen.

FibreChannel

FibreChannel ist eine glasfaser-basierte Technologie zum Aufbau sogenannter *Storage Area Networks* (SAN). Darunter versteht man Netze, die nur zur Bereitstellung von Speichermedien (Festplatten, Bandlaufwerke, Band-Roboter etc.) dienen. Dabei sind alle Elemente eines "normalen" Netzes vorhanden: als Server fungieren die Platten und Bandlaufwerke, Clients sind PCs mit FibreChannel-Karten, und es gibt FibreChannel-Hubs und -Switches (für große

[2] In einem Benchmark mit verschiedenen 32 bit-Applikationen unter Windows 2000 erreichte ein frühes Testsystem mit einem 667 MHz Itanium lediglich eine mit einem Pentium 200 MHz vergleichbare Performance [56].

Switches hat man den Namen Director eingeführt). Das so entstehende Netz nennt man Fabric. Da Glasfasern als Medium verwendet werden, sind die erzielten Reichweiten sehr groß, was das "A" in SAN rechtfertigt.

Die Idee hinter SAN ist es, die Speichereinheiten für mehrere Rechner sichtbar zu machen, und zwar im Gegensatz zu Netzwerklaufwerken auf Geräteebene. Für PCs im SAN sieht es also so aus, als seien die SAN-Platten *in den PC eingebaut*. Dies hat zur Konsequenz, daß es Aufgabe des Clients ist, die Platten zu partitionieren und Dateisysteme anzulegen. Da der Client die Verantwortung für das Dateisystem trägt (anders als zum Beispiel bei NFS: dort ist das Aufgabe des Servers), ist es nicht möglich, mit mehreren Clients gleichzeitig auf dieselbe Platte/Ressource zuzugreifen. Dies schränkt den Nutzen von SAN für die EKPplus insofern ein, als daß hier die Netzwerkbandbreite zu den Platten mit der Kapazität der Platten skalieren sollte.

Sollte sich das Kriterium, daß die Netzwerkbandbreite zu den Fileservern mit deren Größe mitwächst, als nicht zwingend erweisen (zum Beispiel durch Engpässe bei der Anbindung der Knoten selbst, oder auch weil die Jobs auf den Knoten CPU-limitiert sind), so bietet sich eine kombinierte SAN/LAN-Architektur an, wie sie Abbildung 6.2 zeigt. Dabei gibt es das SAN auf der einen Seite (links), das normale Netzwerk (LAN) auf der anderen, und als Mittler einige – wenige – Zugangsrechner. Jeder dieser Zugangsrechner ist für eine Anzahl SAN-Geräte zuständig, die er allein verwaltet. Über eine hinreichend starke Anbindung ans LAN (Gigabit Ethernet, möglicherweise mehrere Karten) werden Platten über NFS an die Clients exportiert. Im Falle des Bandlaufwerk-Zugangsrechners wird nur die Staging-Platte exportiert, auf die Bandinhalte bei Bedarf kopiert werden.

FibreChannel wird in der obengenannten Beispielkonfiguration quasi als Ersatz für IDE oder SCSI benutzt, also lediglich als Anschlußmöglichkeit für Platten und Bandlaufwerke, wenngleich wesentlich mehr Geräte angeschlossen werden können als bei IDE oder SCSI, da dort die Kabellänge einen limitierenden Faktor darstellt (maximal 45 cm bei IDE, 150 cm bei UW-SCSI). Der Hauptvorteil gegenüber der derzeitigen Konfiguration ergibt sich jedoch dadurch, daß bei FibreChannel ein Switch zwischen Speichermedium und verwaltendem Rechner steht. Dadurch erhöht sich die Ausfallsicherheit, denn wenn ein Zugangsrechner ausfällt, kann ein anderer die Platten übernehmen, ohne daß hardwareseitig Veränderungen vorgenommen werden müssen, da die Speichergeräte ja rein softwaremäßig einem Zugangsrechner zugeordnet, nicht aber fest dort eingebaut sind.

Alternativ kann man einen *Meta Data Controller* (MDC) einsetzen, der es in Verbindung mit speziellen Dateisystemen ermöglicht, von mehreren Clients aus gleichzeitig auf eine Platte zuzugreifen. Dabei synchronisiert der MDC die Zugriffe: nur jeweils ein Client darf Blöcke schreiben, und diese Blöcken dürfen während des Schreibvorganges nicht gelesen werden, während ansonsten Lesevorgänge, auch gleichzeitig, kein Problem darstellen. Das bedeutet jedoch, daß sämtliche Schreib- und Lesevorgänge vom MDC autorisiert werden müssen, was einen beträchtlichen Flaschenhals darstellt. Der MDC muß daher auf einer leistungsfähigen Maschine mit besonders starker FibreChannel-Anbindung laufen. Die angebotenen MDC-Lösungen sind alle proprietär und verlangen nach besonderer, teurer Hardware. Sie enthalten auch das Dateisystem, das natürlich mit dem MDC zusammenspielen muß. Ein MDC ist teuer, proprietär, ein Flaschenhals und ein Schwachpunkt hinsichtlich Ausfallsicherheit. Für den Einsatz in der EKPplus ist die erste Lösung sicherlich die sinnvollere.

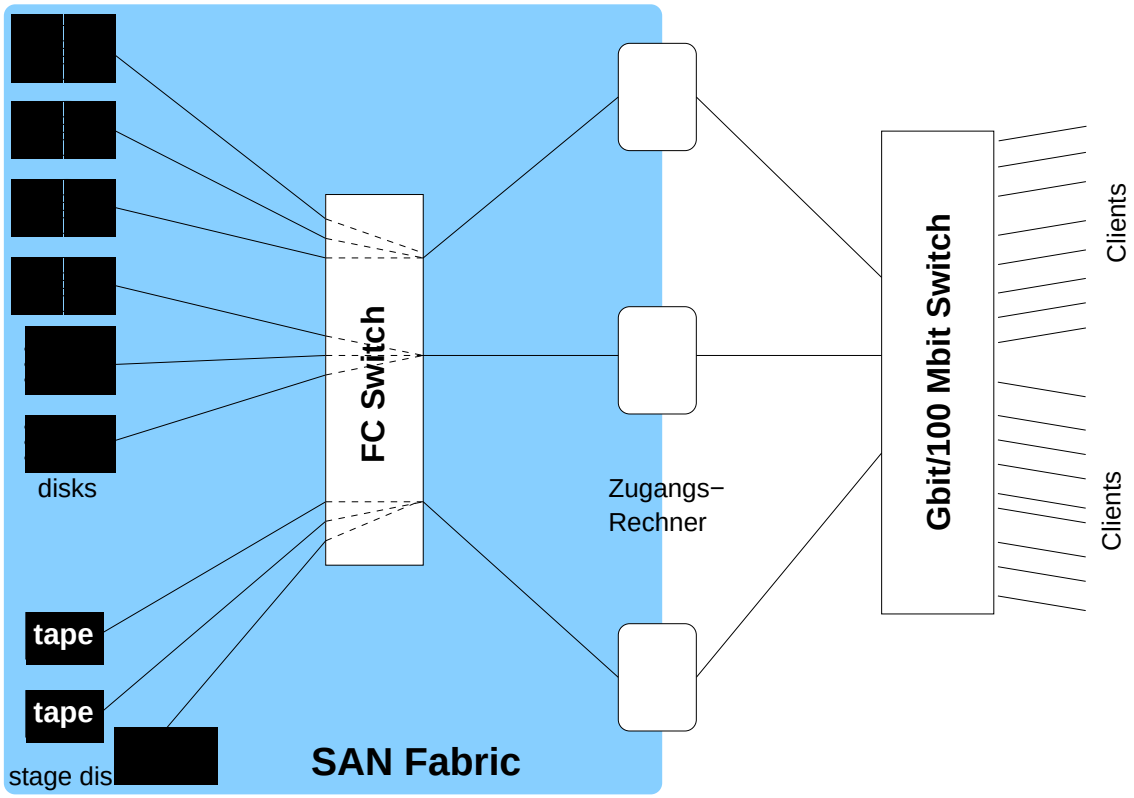

Abbildung 6.2: Mögliche SAN-Infrastruktur für die EKPplus.

6.2.3 Neue Netze

Der wahrscheinlich am schwersten zu behebende Flaschenhals der EKPplus ist das Netzwerk. Das Problem hierbei ist, daß Gigabit Ethernet nur etwa drei bis viermal soviel Durchsatz bietet wie 100 Mbit Ethernet. Dadurch kann es leicht zu Engpässen kommen, etwa wenn mehr als vier Knoten auf eine Gigabit-Karte eines Fileservers zugreifen wollen: statt der dann benötigten ca. 40 MB/s oder mehr stehen lediglich 30 MB/s zur Verfügung.

10 Gbit Ethernet

Im Juni 2002 wurde IEEE P802.3ae verabschiedet, der Standard für 10 Gbit-Ethernet. Anders als 1 Gbit Ethernet handelt es sich dabei um eine reine Glasfaser-Technologie. Eine Kupferversion wird es aller Voraussicht nach nicht geben, da sich bei einem Takt von 10.3125 GHz schon geringe Unterschiede in den Längen der einzelnen Kupfer-Adern negativ auswirken würden.

10 Gbit-Ethernet ist in erster Linie für die Verbindungen des Switches mit den Fileservern interessant. Selbst wenn man von einer ähnlich schlechten Effizient wie bei 1 Gbit-Ethernet ausgeht, kann keiner der derzeit vorhandenen Rechner die etwa 300 MB/s über den PCI- bzw. PCI64-Bus (133 bzw. 266 MB/s) transportieren. Lediglich PCI-X wäre dazu in der Lage (64 bit bei 66 MHz ergibt 528 MB/s – gerade die Hälfe der theoretischen Bandbreite

von 10 Gbit-Ethernet).

In Verbindung mit einer FibreChannel-Lösung wie in Abbildung 6.2 wird 10 Gbit-Ethernet auch für den Einsatz in den SAN-Zugangsrechnern attraktiv. Die vielversprechendste (wenngleich nicht besonders billige) Lösung der Netzwerk- und Speicher-Sorgen der EKPplus führen über das Ersetzen von Gigabit Ethernet durch 10 Gbit-Ethernet.

6.3 Grid Computing

Die Vision, die auch den Namen des Grid erklärt, stammt aus der Energieversorgung. Der Begriff *Computational Grid* ist als Gegenstück zum *Power Grid*, dem Stromnetz, gedacht. Genau wie beim Stromnetz beauftragt der Benutzer das Computer-Equivalent der Stadtwerke, eine *Certification Authority* (CA), ihm Rechenleistung zu liefern. Wenn der Antrag erst einmal bearbeitet ist, braucht der Benutzer nur noch den (Netzwerk-) Stecker in die Steckdose zu stecken, und die benötigte Rechenleistung und Speicherplatz stehen ihm zur Verfügung, wie Abbildung 6.3 veranschaulicht. Dabei ist der Vergleich mit der Strom-Infrastruktur durchaus aufschlußreich: es gibt Rechenzentren (Kraftwerke), Provider/Endkunden-Anbieter (Stadtwerke), und möglicherweise billigere Nachtrechenleistung, wenn Firmen nicht auf prompte Ergebnisse angewiesen sind[3]. Dabei ist die Verfügbarkeit einer leistungsfähigen und zuverlässigen Netzwerkinfrastruktur essenziell. Kunden werden in erster Linie Firmen und Forschungseinrichtungen mit großem Rechenzeitbedarf sein, zum Beispiel:

- Teilchen- und Astroteilchenphysik
- Meteorologie
- Gentechnologie
- Pharmaindustrie (Entwicklung von Medikamenten)
- Automobilindustrie (Simulation von Crashes, Aerodynamik)
- Hollywood (Rendern von Trickfilmen)

6.3.1 Grid in der Teilchenphysik

Spätestens die kommenden Großexperimente am Large Hadron Collider (LHC) in Genf, die 2007 in Betrieb genommen werden sollen, werden die Möglichkeiten der bisherigen herkömmlichen Rechenzentren sprengen. Mit $O(10^{15})$ Bytes/Jahr (PetaByte, PB) wird es für die einzelnen Institute bzw. Universitäten nicht mehr möglich sein, mehr als nur winzige Auszüge aus den Daten für ihre Mitarbeiter bereitzuhalten.

Um dieses Problem zu bewältigen, ist eine Multi-Tier-Architektur auf der Basis von Grid-Technologie vorgesehen wie sie Abbildung 6.4 zeigt, mit dem CERN-Rechenzentrum als Tier

[3]Möglicherweise wird auch die Zeitverschiebung genutzt, um tageszeitabhängige Überschüsse nach Übersee zu exportieren, doch ob sich das lohnt – die Politik wird vermutlich Einfuhrzölle auf importierte Rechenleistung erheben – wird sich zeigen.

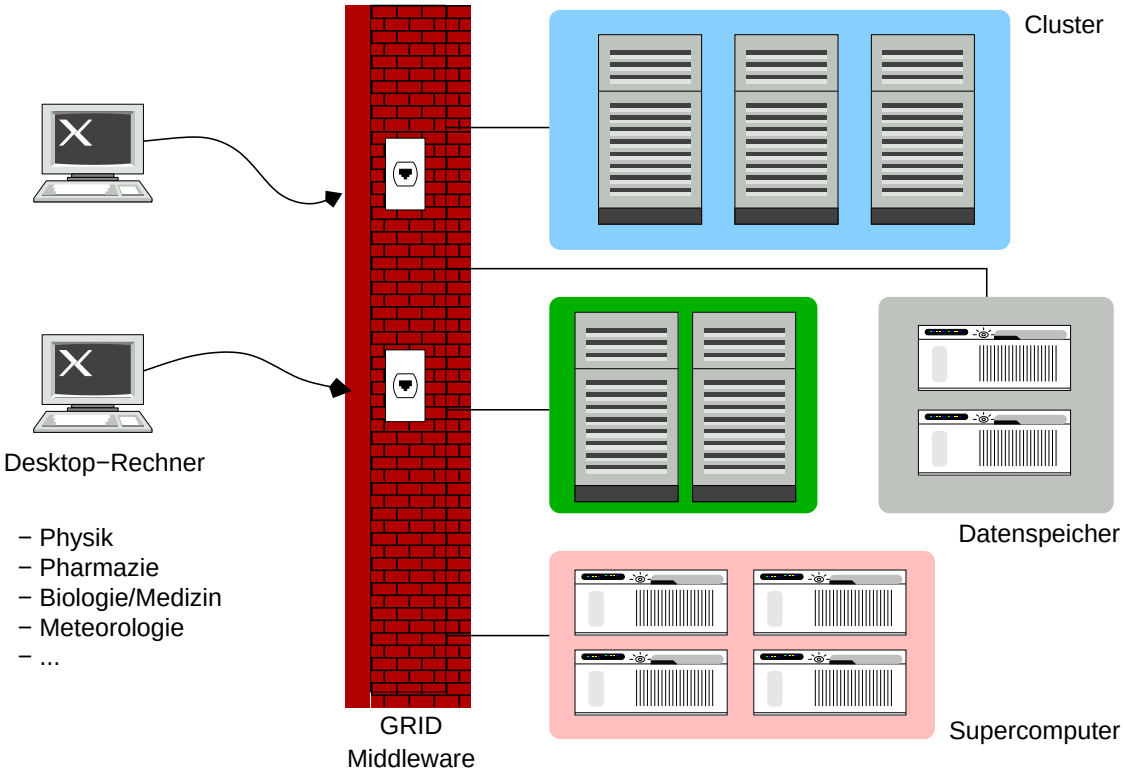

Abbildung 6.3: Ausblick auf die zukünftige Funktion des Grid.

0 im Zentrum [57]. Mit Netzwerkanbindungen im Gigabit-Bereich angeschlossen sind die regionalen Rechenzentren der Teilnehmerländer Deutschland, Italien, Großbritannien, der USA und Frankreich (Tier 1-Zentren), von denen jedes einen anderen Teil der CERN-Daten speichert. Die Universitäten wiederum schließen sich mit eigenen zentralen Clustern (Tier 2) und/oder Institutsclustern (Tier 3) an die regionalen Rechenzentren an. Das deutsche regionale Rechenzentrum wird derzeit am Forschungszentrum Karlsruhe (FZK) aufgebaut.

Über die Grid-Technologie ist es nun Forschern aus aller Welt möglich, sich die Rechenleistung und die Daten der großen regionalen Rechenzentren quasi auf den Schreibtisch zu holen. Ein Beispiel: ein Analysejob wird am Institut L der Universität E entwickelt, ist jedoch zu groß für das dortige Rechenzentrum. Nachdem sich der Entwickler beim Grid authifiziert hat, findet die Grid Middleware heraus, welche Daten der Job benötigt (beispielsweise das ganze J/Ψ-Sample) und wo diese Daten liegen (beispielsweise am FZK). Dorthin wird dann der Job geschickt. Auch andere Szenarien sind denkbar. So kann ein Job, der nur auf ein kleines Datensample zugreifen muß, dorthin geschickt werden, wo derzeit viel Rechenleistung ungenutzt ist, und die Daten werden ebenfalls dorthin kopiert. Welches Vorgehen im Einzelfall günstiger ist, muß die Middleware entscheiden.

Die Rolle der EKPplus wird in diesem Zusammenhang die eines Tier 3-Clusters sein, direkt an das Tier 1-Zentrum des FZK angeschlossen. Damit wird unser Institut in Zukunft über erstklassige Möglichkeiten verfügen, LHC-Daten auszuwerten, was für unser Engagement beim CMS-Experiment von größter Bedeutung ist.

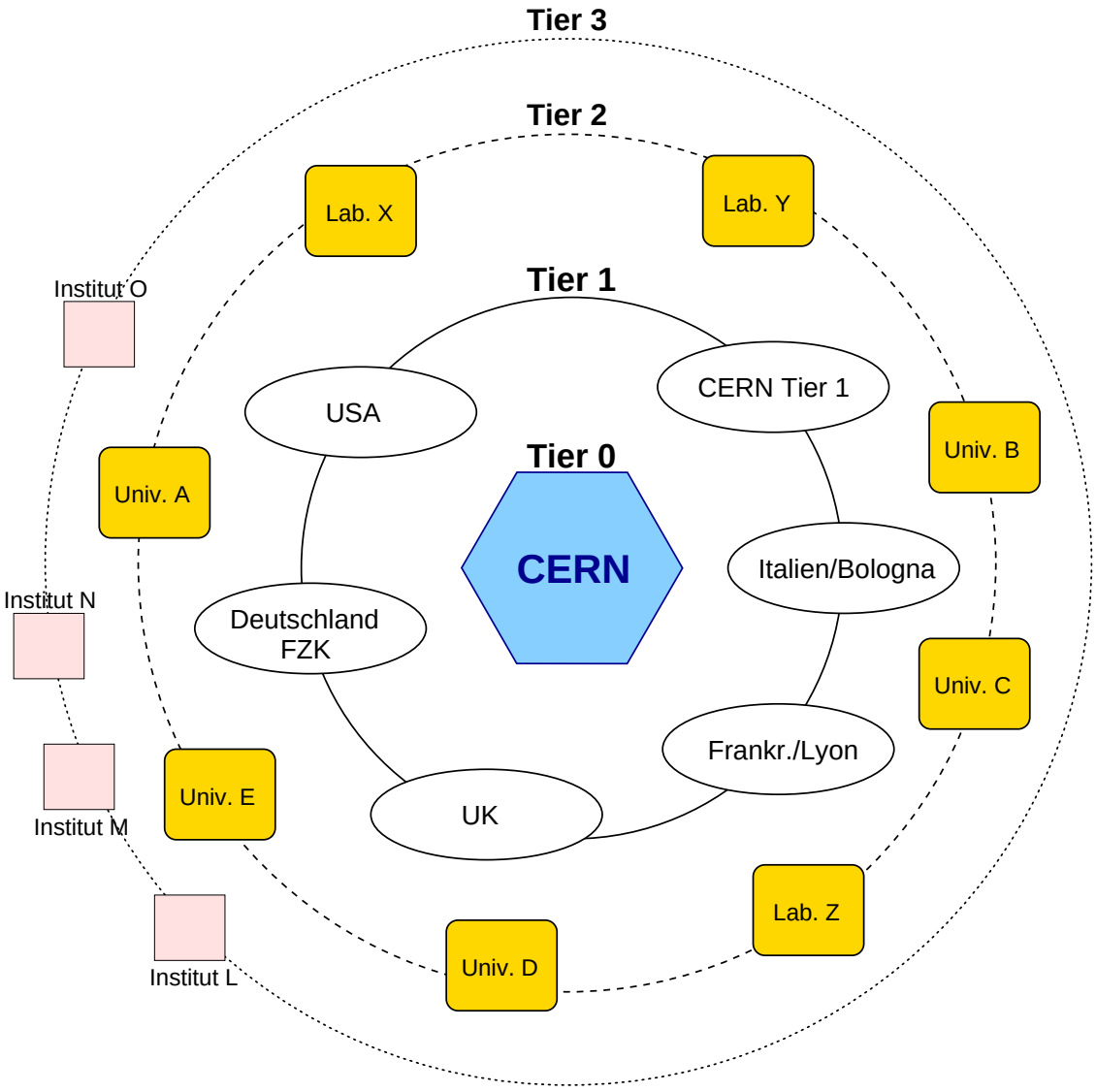

Abbildung 6.4: Multi-Tier-Architektur zur Verteilung der LHC-Daten.

6.3.2 Die Technologie

Um die Vision des Grid umzusetzen, werden zahlreiche Technologien eingesetzt, die aus anderen Bereichen der Informatik stammen, und erstmals in ein umfassendes Projekt integriert. Technische Details und die entsprechende Software ist zum Beispiel bei [58] erhältlich.

Authifizierung

Vor der Nutzung einer Rechenanlage muß sich der Benutzer authifizieren, das heißt, einen Nachweis seiner Identität erbringen. Dazu bietet sich das sogenannte *Public Key*-Verfahren an. Bei diesem Verfahren erzeugt sich der Benutzer ein Schlüsselpaar, bestehend aus einem öffentlichen (also nicht-geheimen) und einem privaten (geheimen) Schlüssel. Zur Verschlüsse-

lung wird der private Schlüssel verwendet, zum Entschlüsseln ist jedoch nur der öffentliche nötig. Abbildung 6.5 verdeutlich die Vorgänge bei der Authifizierung über PKI-Verfahren.

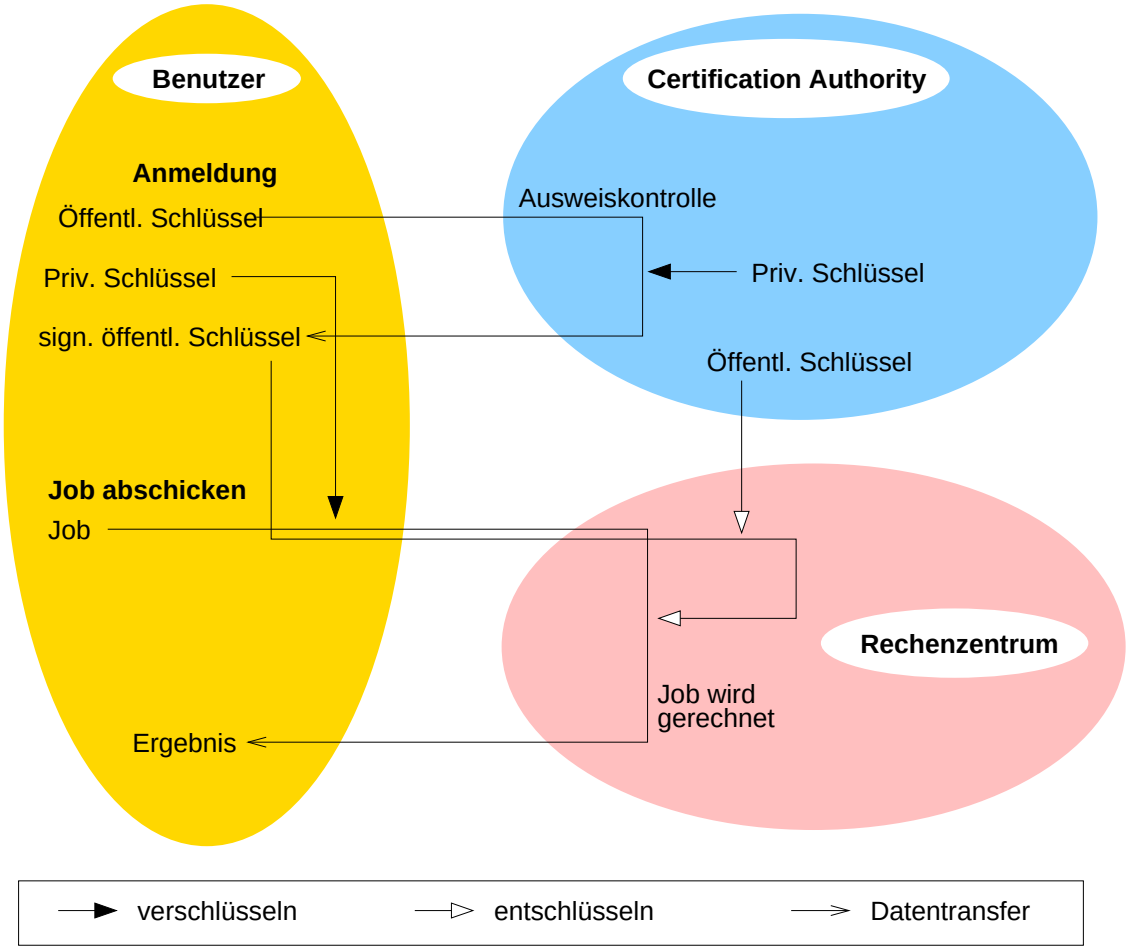

Abbildung 6.5: Authifizierung mittels *Public Key*-Verfahren.

Zunächst muß der Benutzer seinen öffentlichen Schlüssel zertifizieren lassen. Dies übernimmt die *Certification Authority* (CA). Dort wird der vorliegende öffentliche Schlüssel einer tatsächlichen Person zugeordnet. Dies geschieht, indem die CA den öffentlichen Schlüssel des Benutzers (nach Prüfung dessen Personalausweises) mit ihrem privaten Schlüssel verschlüsselt[4]. Damit ist der öffentliche Schlüssel des Benutzers beglaubigt.

Um nun einen Job abzuschicken, verschlüsselt der Benutzer den Job mit seinem privaten Schlüssel, und übermittelt den so verschlüsselten Job einem Rechenzentrum zusammen mit dem eigenen, beglaubigten öffentlichen Schlüssel. Das Rechenzentrum überprüft zunächst die Beglaubigung, indem es sie mit dem öffentlichen Schlüssel der CA entschlüsselt. Gelingt dies, so ist die Beglaubigung echt, der Benutzer gilt als authifiziert. Mit dem so erhaltenen öffentlichen Schlüssel des Benutzers wird der Job entschlüsselt, wodurch gleichzeitig sichergestellt wird, daß der Job auch tatsächlich vom angegebenen Benutzer stammt, da nur dieser

[4]Der private Schlüssel der CA unterliegt strengsten Sicherheitsvorschriften.

den entsprechenden privaten Schlüssel hat.

Prämisse dieses Verfahrens ist, daß die privaten Schlüssel geheim bleiben. Stiehlt jemand den privaten Schlüssel eines Benutzers, so kann der Dieb Jobs im Namen des Benutzers abschicken. Schlimmer noch der Diebstahl des privaten Schlüssels der CA: mit ihm kann der Dieb beliebige eigene Schlüsselpaare selbst zertifizieren. Die Grundlagen des *Public Key*-Verfahrens werden in [59] erläutert.

Ressource Broking

Um einen Job möglichst effizient abarbeiten zu können, ist es wichtig, die folgenden Ressourcen gegeneinander abzuwägen und den günstigsten Arbeitspunkt zu finden:

- Rechenzeit – Wieviele CPUs stehen zur Verfügung? Wie lange ist die Warteschlange?
- Datenbestand – Sind alle benötigten Daten vor Ort vorhanden, oder müssen sie herkopiert werden?
- Übertragungszeit – Wie lange dauert es, benötigte Daten herzukopieren? Wie lange, die Ergebnisse abzuliefern?
- Speicherplatz – Ist genügend Platz für die Ergebnisse? Für eventuell herzukopierende Daten?

So hilft es beispielsweise wenig, einen Analysejob auf einen freien Cluster zu schieben, wenn die zu analysierenden Daten dort nicht zur Verfügung stehen und ihre Übertragung länger dauert als die Wartezeit in einem anderen Cluster, der die Daten bereits hat. In diesem Fall hätte man Bandbreite belegt, ohne eine frühere Abarbeitung des Jobs zu erreichen. Andererseits ist es wahrscheinlich lohnend, einen sehr CPU-intensiven Job zu den stärksten freien CPUs zu schieben, auch wenn dort zu wenig Speicherplatz für die Ergebnisse vorhanden ist und diese über das Netzwerk woanders gespeichert werden müssen, da sich die Übertragungszeit hinter der CPU-Zeit verstecken läßt.

Diese und ähnliche Abwägungen sind Aufgabe des Ressource Brokers. Ein Job wendet sich mit seinem Anforderungsprofil an den Ressource Broker, und dieser findet die geeignetsten Maschinen, indem er ihre Leistungsprofile abfragt, was über einen Verzeichnisdienst – eine Art Mini-Datenbank – geschieht, der von dem Maschinen ständig aktualisiert wird[5]. Hat der Broker die am besten geeignete Maschine gefunden, so übergibt er den Job an den Scheduler des dortigen Batch-Systems[6]. Der Ressource Broker ist also gewissermaßen ein Meta-Scheduler.

Datenübertragung

Zur Übertragung der Daten, die ein Job eventuell benötigt, und auch zur Übertragung von Ergebnissen wird ein parallelisiertes FTP-Protokoll verwendet. Dabei werden mehrere Verbindungen zwischen den beteiligten Rechnern gleichzeitig aufgebaut; jede Verbindung

[5] Das Globus-Toolkit verwendet hierzu OpenLDAP [60], eine freie Implementation des *Lightweight Directory Access Protocol*.

[6] Beispielsweise das auf der EKPplus verwendete und in 4.2.4 beschriebene OpenPBS [37].

überträgt einen Teil der Daten. Dieses Verfahren beruht darauf, daß es mehr als eine physikalische Verbindung zwischen den Rechnern gibt, beispielsweise mehrere transatlantische Kabel, oder zumindest mehrere Kanäle auf dem Kabel; es handelt sich also um eine Art Kanalbündelung, wie sie von ISDN-Adaptern her bekannt ist, wo ebenfalls zwei 64 kbit-Kanäle zu einem 128 kbit-Kanal zusammengefaßt werden können.

Dieses Verfahren bringt natürlich nur dann Vorteile gegenüber dem herkömmlichen FTP-Protokoll, wenn tatsächlich mehrere Leitungen über den Engpaß im Netz zur Verfügung stellen. Über eine simple 1 Gbit-Ethernet-Leitung dagegen bringt es keine Vorteile. Die Datenübertragung basiert wie das ganze Grid-Konzept auf der Verfügbarkeit von Höchstleistungsnetzwerken.

Herausforderungen

Die größte technische Herausforderung im Rahmen des Grid-Projekts ist wahrscheinlich die extreme Inhomogenität der verwendeten Rechnerumgebungen. Es gibt zahlreiche inkompatible Hardware-Architekturen, so etwa Intel x86 vs. Intel Itanium vs. Alpha vs. IBM Power4 etc. Damit nicht genug, gibt es auf jedem der genannten Systeme mehrere Betriebssysteme, in unterschiedlichen Versionen. Schon heute ist die Linux-Distributionsvielfalt ein ernsthaftes Problem bei der Wartung der EKPplus, dabei sind hier lediglich drei Experimente vertreten (CDF mit Fermi Linux 7.1.1; CMS hätte gerne Redhat 6.2, was aber auf moderner Hardware nicht läuft, deshalb wird SuSE 7.2 verwendet; und DELPHI, das glücklicherweise mit beiden auskommt). Wenn sich nicht einmal drei Experimente aus dem selben Forschungsbereich auf eine Linux-Distribution einigen können, ist der Ausblick für eine interdisziplinäre Einigung (mit eventueller Beteiligung der freien Wirtschaft) nicht unbedingt vielversprechend.

Es bleibt daher zu hoffen, daß die United Linux-Initiative [61], die im Mai 2002 von den Linux-Distributoren Caldera, Conectiva, SuSE und TurboLinux gegründet wurde, die Situation etwas entschärft. Die erste Version dieser einheitlichen Basis-Distribution (mit herstellerspezifischen Erweiterungen) ist für Ende 2002 geplant. Allerdings ist mit Redhat der größte Linux-Distributor nicht in der United Linux-Initiative vertreten.

Zu den Herausforderungen zählen sicher auch die im folgenden beschriebenen ethischen Aspekte des Grid.

6.3.3 Ethische Überlegungen

Das Grid ist weit mehr als ein technisches Hilfsmittel, um an mehr Rechenleistung oder Speicherplatz zu kommen. Das Grid ist das Code gewordene Bestreben nach Zusammenarbeit. Grundlage ist die Bereitschaft, fremden Benutzern die eigene Rechenanlage zu überlassen. Dabei werden sowohl den Benutzern als auch den Betreibern von Rechenanlagen ethisches Verhalten abverlangt. Dabei sind die folgenden Überlegungen auch gültig, wenn die *Übertragung* der Daten völlig sicher ist.

Ethik des Benutzers

Vom Benutzer wird erwartet, daß seine Jobs keinen Schaden anrichten und den Betrieb der Anlage nicht gefährden. Dazu gehört unter anderem, keine “bösartigen” Programme (Viren,

Würmer, trojanische Pferde, *denial of service*-Tools, Rootkits etc.) auszuführen. Da man sich vor Benutzung der Anlage zu authifizieren, also seinen virtuellen Ausweis vorzulegen hat, sind solche Störaktionen eher unwahrscheinlich — es sei denn, jemand benutzt eine gefälschte oder gestohlene Zugangsberechtigung. Unglücklicherweise sind Schutzmaßnahmen, wie etwa das Binden einer Zugangsberechtigung an einen bestimmten Desktop, eine starke Einschränkung – auf Konferenzen oder bei Aufenthalten in anderen Forschungslabors wäre damit kein Zugriff mehr möglich. Der einzige Weg führt über die Schärfung des Sicherheitsbewußtseins der Benutzer.

Ethik des Betreibers

Doch auch die andere Seite, die des Betreibers, birgt Versuchungen, insbesondere die der Spionage. Simuliert beispielsweise ein Pharmakonzern die Funktionsweise eines neuen Wirkstoffes, so liegen alle relevanten Daten auf den Rechnern des ausführenden (möglicherweise kommerziell betriebenen) Rechenzentrums. Das bedeutet, der Benutzer vertraut seine eventuell hochsensitiven Daten einem Rechenzentrum an, das möglicherweise in einem Land liegt, in dem Datenschutzverletzungen und die unberechtigte Nutzung von geistigem Eigentum als Bagatelle gesehen werden.

Schlußbemerkung

Mit dem *Kaltrak*-Paket wurde eine leistungsfähige, flexible und dennoch wartbare Spurrekonstruktionssoftware entworfen. Die Leistungsfähigkeit der Software wurde unter anderem in der durchgeführten Analyse $J/\Psi \to \mu^+\mu^-$ unter Beweis gestellt. Auch die Werkzeuge zur Überwachung der Online-Monitoring-Programme, der *Consumer*, haben sich als alltagstauglich erwiesen.

Die EKPplus hat sich in der Beispielanalyse bestens bewährt. Auch für zahlreiche andere Analysen am EKP stellt sie ein unverzichtbares Werkzeug dar. Allerdings besteht bereits Bedarf nach einem Ausbau: der 1.4 TB Fileserver ist zu über 95 % belegt. Dank der gutdurchdachten Netzwerkinfrastruktur ist es aber ein leichtes, weitere Fileserver zu integrieren.

Die Analyse zum Zerfall $J/\Psi \to \mu^+\mu^-$ zeigt die Leistungsfähigkeit der entwickelten Software. Der J/Ψ Massenpeak wurde gefittet und anhand der Zerfallslänge der rekonstruierten J/Ψ wurde neben den prompten J/Ψ auch ein Beitrag von J/Ψ aus b-Zerfällen nachgewiesen. Der nächste Schritt wäre eine Lebensdauer-Messung mit einer größeren Datenmenge; die hier verwendeten 240 nb^{-1} reichen dazu nicht aus.

Die Rolle der elektronischen Datenverarbeitung in der Teilchenphysik ist schon heute bedeutend; in Zukunft jedoch wird sie *die* Technologie sein, die den Fortschritt auf diesem Gebiet der Physik, und auf vielen anderen Gebieten auch, bestimmt und ermöglicht. In diesem Zusammenhang ist es wichtig zu erkennen, daß das Grid keine *optionale* Angelegenheit ist. Vielmehr sind die LHC-Experimente ohne das Grid nicht durchführbar. *Ressource pooling* ist die einzige Möglichkeit, mit den enormen Datenmengen fertig zu werden.

Anhang A

Anhang

A.1 Flexible Parameter mit const-Unterscheidung

Das folgende Code-Fragment zeigt, wie es möglich ist, in C++ vererbte Funktionen mit variablen Parametern zu implementieren. Die `process()`-Methode der Basisklasse `Algorithm` hat einen Parameter, der die *eigentlichen* Parameter beinhaltet. Diese Lösung hat immer noch den Nachteil, daß erst zur Laufzeit vom jeweiligen Algorithmus festgestellt werden kann, ob die tatsächlichen Parameter korrekt und vollständig sind, wie hier in `MyTracking::process()` zu sehen ist.

Im vorliegenden Code ist zu beachten, daß es Ein- und Ein-/Ausgabe-Parameter, jedoch keine reinen Ausgabe-Parameter gibt. Dies widerspiegelt die Tatsache, daß die Ausgabeparameter-Klasse `AlgParameterOut` von der Eingabeparameter-Klasse `AlgParameterIn` erbt: ein Ausgabeparameter ist immer auch ein (potentieller) Eingabeparameter.

```
#include <cstring>
#include <iostream>
#include <list>


class DataSet {
public:
  virtual char* name () const = 0;
};


class KalSiHitSet : public DataSet {
public:
  char* name () const { return "KalSiHitSet"; }
  // ... more hit set related stuff
};
```

```
class KalTrackSet : public DataSet {
public:
  char* name () const { return "KalTrackSet"; }
  // ... more track set related stuff
};


class AlgParameter {
public:
  virtual char* name () const = 0;
};


class AlgParameterIn : public AlgParameter {
private:
  const DataSet* d;
public:
  AlgParameterIn (const DataSet& data) { d = &data; }
  char* name () const { return d->name(); }
  const DataSet* data () const { return d; }
};


class AlgParameterOut : public AlgParameterIn {
private:
  DataSet* d;
public:
  AlgParameterOut (DataSet& data) : AlgParameterIn(data) { d = &data; }
  char* name () const { return d->name(); }
  DataSet* data () const { return d; }
};


class AlgParameterSet {
private:
  typedef list<const AlgParameter*> ParaSet;
  ParaSet paraset;
public:
  AlgParameterSet () { }
  AlgParameterSet (const AlgParameterSet& rhs) { add(rhs); }
  AlgParameterSet (const AlgParameter& rhs) { add(rhs); }
  void add (const AlgParameter& p) { paraset.push_back(&p); }
  void add (const AlgParameterSet& ps) {
    for (ParaSet::const_iterator i = ps.paraset.begin();
```

```
         i != ps.paraset.end(); i++)
      paraset.push_back(*i);
  }
  const AlgParameter* find (const char* name) const {
    for (ParaSet::const_iterator i = paraset.begin();
         i != paraset.end(); i++)
      if (strcmp((*i)->name(),name) == 0) return (*i);
    return 0;
  }
};


AlgParameterSet operator+ (const AlgParameter& a, const AlgParameter& b) {
  AlgParameterSet s (a);
  s.add(b);
  return s;
}

AlgParameterSet operator+ (const AlgParameterSet& a, const AlgParameter& b) {
  AlgParameterSet s (a);
  s.add(b);
  return s;
}

AlgParameterSet operator+ (const AlgParameterSet& a,
                           const AlgParameterSet& b) {
  AlgParameterSet s (a);
  s.add(b);
  return s;
}


class Algorithm {
public:
  virtual bool process (const AlgParameterSet& parameter) = 0;
};


class MyTracking : public Algorithm {
public:
  bool process (const AlgParameterSet& p) {

    const AlgParameterIn* inHitSet =
      dynamic_cast<const AlgParameterIn*>(p.find("KalSiHitSet"));
```

```
    if (!inHitSet) {
      cerr << "No input parameter KalSiHitSet!" << endl;
      return false;
    }
    const KalSiHitSet* siHitSet =
      dynamic_cast<const KalSiHitSet*>(inHitSet->data());
    cout << "Have input hit set" << endl;

    const AlgParameterOut* outTrackSet =
      dynamic_cast<const AlgParameterOut*>(p.find("KalTrackSet"));
    if (!outTrackSet) {
      cerr << "No output parameter KalTrackSet!" << endl;
      return false;
    }
    KalTrackSet* trackSet =
      dynamic_cast<KalTrackSet*>(outTrackSet->data());
    cout << "Have output track set" << endl;

    // actual algorithm here...

    return true;
  }
};


int main ()
{
   KalSiHitSet siHitSet;
   KalTrackSet trackSet;
   MyTracking tracker;
   tracker.process(AlgParameterIn(siHitSet) +
                   AlgParameterOut(trackSet));
   return 0;
}
```

Versucht man beispielsweise, im Hauptprogramm den Parameter `trackSet` von `AlgParameterOut` auf `AlgParameterIn` umzustellen, wird der Compiler dies nicht beanstanden; zur Laufzeit hindoch wird `MyTracking::process()` feststellen, daß kein gültiges `KalTrackSet` unter den Parametern ist, und abbrechen.

A.2 ARP Race Conditions

Auf unterster Ebene, den sogenannten *Frames*, arbeitet Ethernet auf der Basis von MAC-Adressen, also weltweit eindeutigen Hardwareadressen der Netzwerkkarten. Will ein Rechner

ein IP-Datenpaket an einen anderen Rechner schicken, braucht er zunächst die MAC-Adresse des Empfängers. Dazu erzeugt er eine *ARP who-has*-Anfrage[1], in der die gesuchte Ziel-IP-Adresse steht. Diese Anfrage wird durch das ganze Netz verschickt (*broadcast*). Der Zielrechner empfängt die ARP-Anfrage, stellt fest, daß er gemeint ist, und antwortet gezielt dem Ursprungsrechner mit seiner MAC-Adresse. Dieser kann nun ein Frame mit den eigentlichen Nutzdaten konstruieren, das dann gezielt an den Zielrechner verschickt wird. Durch die vorherige ARP-Anfrage wissen Switches, die das Datenpaket weiterleiten, bereits wo (an welchem Port) der Zielrechner zu erreichen ist – die Switches haben die ARP-Kommunikation ausgewertet und zwischengespeichert. Aber auch der Linux-Netzwerk-Stack speichert die MAC-Adressen und über welche Netzwerkkarten sie zu erreichen sind. Dadurch ist nicht für jedes Paket eine neue ARP-Anfrage nötig, denn die Zuordnung von Ziel-MAC-Adresse und eigener Netzwerkkarte ändert sich ja nur, wenn irgendwo dazwischen Kabel umgesteckt werden.

Bei zwei (oder mehr) Netzwerkkarten in einem Rechner A, die beide an denselben Switch angeschlossen sind, aber in verschiedenen IP-Subnetzen stehen, kann es zu folgendem Problem kommen: Rechner A will eine IP-Verbindung zu Rechner B aufbauen. Rechner A wird also eine *ARP who-has*-Anfrage abschicken – über *beide* Netzwerkkarten. Rechner B sieht also zwei ARP-Anfragen und wird beide beantworten – wobei die Reihenfolge der beiden Anfragen bzw. die der Antworten nicht definiert ist! *Wahrscheinlich* wird die erste Ethernetkarte, eth0, zuerst senden und auch zuerst eine Antwort erhalten – sicher ist dies jedoch nicht. Im ARP-Cache von Rechner A sind jetzt zwei verschiedene Einträge für Rechner B, einer von jeder Netzwerkkarte. Bei der Konstruktion des eigentliche IP-Datenpakets wird die erste im ARP-Cache mit Rechner B assoziierte Netzwerkkarte zum Versenden des Pakets verwendet. Wenn Rechner B im selben IP-Subnetz wie die assoziierte Netzwerkkarte in Rechner A, ist alles klar und das Paket wird seinen Weg machen. Wenn jedoch die "falsche" Netzwerkkarte zuerst im ARP-Cache auftaucht, ergibt sich ein Widerspruch zur IP-Routing-Tabelle von Rechner A, es kommt zu Übertragungsfehlern.

In unserem Fall trat diese ARP *race condition* bei den zwei mit Gigabit Ethernet ausgestatteten Portalen `EKPCMS1` und `EKPDELPHI` auf. Hier hat in 85% der Fälle die "falsche" Netzwerkkarte, nämlich die für die Außenanbindung zuständige Fast Ethernet-Karte, zuerst auf die ARP-Anfrage geantwortet. Dies hatte zur Folge, daß die meisten Knoten nicht auf die NFS-Laufwerke der genannten Portale zugreifen konnten. Nachdem das Problem erkannt war, wurde die physikalische Trennung der beiden Teilnetzte (öffentliche und private IP-Adressen) vorgenommen, was das Problem behoben hat.

Das geschilderte Verhalten kann als ein Fehler im Linux-Netzwerk-Stack angesehen werden. Die ARP *who-has*-Anfrage hätte nur auf der Netzwerkkarte, die vom Routing her für die Ziel-IP zuständig ist, einen Broadcast durchführen dürfen. Damit wären alle obengenannten Probleme vermieden worden, denn es hätte nur *eine* Antwort und nur *einen* ARP-Cache-Eintrag gegeben. Eine Untersuchung von OpenBSD und Digital UNIX hat ergeben, daß diese Betriebssysteme im ARP-Cache ausschließlich Paare von MAC- und IP-Adressen speichern, ohne Information über die verwendete Netzwerkkarte; das Problem der ARP *race conditions* ist also Linux-spezifisch.

[1] ARP steht für Address Resolution Protocol, Adressauflösungsprotokoll.

A.3 Hardwareprofil der EKPplus

Es folgt eine detailliertere Übersicht über die Hardware der Rechner der EKPplus inklusive Herstellerangaben.

Rechner	`ekpcdf1`	`ekpcdf2`
CPU	2x 1400 MHz Athlon	2x 1400 MHz Athlon XP1600+
Board	Tyan ThunderK7 [43]	Tyan ThunderK7 [43]
RAM	2048 MB ECC-DDR-RAM	2048 MB ECC-DDR-RAM
Platten	1x 40 GB (IBM), 4x 80 GB IDE (Maxtor) RAID0	1x 40 GB (Maxtor), 4x 80 GB IDE (Maxtor) RAID0
Netzw	4x 100 Mbit (2 onboard)	3x 100 Mbit (2 onboard)
Gehäuse	19" 4 HE	19" 4 HE
Hersteller	Kircher EDV, Karlsruhe	Kircher EDV, Karlsruhe
Sonst.	3ware Escalade 7410 [42] RAID0	3ware Escalade 7410 [42] RAID0

Rechner	`ekpgrid`	`ekpdelphi`
CPU	1400 MHz Athlon	1400 MHz Athlon
Board	Asus A7S-VM	Asus A7S-VM
RAM	1024 MB SD-RAM	512 MB SD-RAM
Platten	120 GB IDE	120 GB IDE
Netzw	1x 100 Mbit, 1x Gigabit D-Link DGE-550T	1x 100 Mbit, 1x Gigabit 3COM 996T
Gehäuse	19" 2 HE	19" 2 HE
Hersteller	Kircher EDV, Karlsruhe	Kircher EDV, Karlsruhe
Sonst.	–	Conrad Digital Thermometer 304 an serieller Schnittstelle

Rechner	`ekpplusctl`	`ekpfs1`
CPU	800 MHz Duron	2x 1200 MHz Pentium III
Board	Asus A7S-VM	ServerWorks
RAM	512 MB SD-RAM	2048 MB SD-RAM
Platten	2x 40 GB IDE (Maxtor und IBM)	16x 100 GB IDE (Western Digital) RAID5
Netzw	1x Gigabit D-Link DGE-550T	1x 100 Mbit, 2x Gigabit D-Link DGE-550T
Gehäuse	19" 2 HE	19" 4 HE (16x *Hot Swap*-Rahmen)
Hersteller	Kircher EDV, Karlsruhe	FMS Conputer, Nephten
Sonst.	–	2x 3ware Escalade 7850 [42] RAID5

Rechner	`ekpcms1`	`ekpplusnat`
CPU	2x 1733 MHz Athlon MP 2100+	1800 MHz Pentium 4
Board	Tyan Tiger MPX	Intel D845GRGL
RAM	2 GB DDR-RAM	512 MB DDR-RAM
Platten	1x 40 GB, 8x 120 GB (Maxtor)	2x 40 GB (Maxtor)
Netzw	1x 100 Mbit, 1x Gigabit Intel Pro/1000	1x 100 Mbit, 2x Gigabit Intel Pro/1000, 1x Gigabit SK-9D21 (SX)
Gehäuse	19" 5 HE	19" 2 HE
Hersteller	FMS Computer, Nephten	FMS Computer, Nephten
Sonst.	–	–

Rechner	`ekpplus001-005`	`ekpplus006-018`
CPU	1400 MHz Athlon XP1600+	1400 MHz Athlon
Board	Asus A7S-VM	Asus A7S-VM
RAM	512 MB SD-RAM	512 MB SD-RAM
Platten	40 GB IDE (IBM)	40 GB IDE (Maxtor)
Netzw	1x 100 Mbit (onboard, bootfähig)	1x 100 Mbit (onboard, bootfähig)
Gehäuse	19" 2 HE (gute Luftführung!)	19" 2 HE
Hersteller	FMS Computer, Nephten	Kircher EDV, Karlsruhe
Sonst.	–	–

A.4 Übertragungsraten

Hier sind einige Übertragungsraten für Daten innerhalb des EKPplus-Clusters angegeben. Dabei handelt es sich um Messungen auf Dateiebene: es wurden Dateien mit dem `dd`-Kommando erzeugt bzw. gelesen, und die Zeit mittels vorangestellem `time`-Kommando gemessen. Beim Schreiben auf lokalen Platten wurde die Zeit für das physikalische Schreiben ebenfalls mit berücksichtigt, um Cache-Effekte zu unterdrücken (`time sync`); die effektiven Übertragungsraten (ohne `sync`), die ein Benutzerprogramm sehen würden, sind auch angegeben, denn schließlich ist der Cache eine bewußt eingesetzte Maßnahme zur Verbesserung des Durchsatzes. Wichtig ist schließlich die Leistung des Gesamtsystems (Platte *und* Cache), nicht die der Einzelkomponenten.

A.4.1 3ware Escalade 7410 (RAID0)

Rechner: `ekpcdf1` (lokal)
Dateigrößen: 100, 250, 500, 1000 MB
Übertragungsdauern (lesen): 3.9, 3.0, 4.7, 7.8 s
Übertragungsrate (lesen): 25.6, 83.3, 106.4, 128.2 MB/s
Mittlere Übertragungsrate (lesen): 85.9 MB/s
Übertragungsdauern (schreiben): 1.0+6.5, 2.4+2.7, 5.5+3.4, 12.6+3.8 s
Übertragungsrate (schreiben): 100.0, 104.2, 90.9, 79.4 MB/s

Mittlere Übertragungsrate (schreiben): 93.6 MB/s
Übertragungsrate (schreiben mit sync): 13.3, 49.0, 56.2, 60.9 MB/s
Mittlere Übertragungsrate (schreiben mit sync): 44.9 MB/s

Die mit RAID0 erreichten Übertragungsraten sind sehr überzeugend und bis auf den Ausreißer beim synchronen Schreiben vom 100 MB weitgehend konstant.

A.4.2 3ware Escalade 7850 (RAID5)

Rechner: `ekpfs1` (lokal)
Dateigrößen: 100, 250, 500, 1000 MB
Übertragungsdauern (lesen): 2.1, 3.6, 9.1, 16.3 s
Übertragungsrate (lesen): 47.6, 69.4, 54.9, 61.3 MB/s
Mittlere Übertragungsrate (lesen): 58.3 MB/s
Übertragungsdauern (schreiben): 1.0+5.6, 2.7+13.0, 4.8+26.7, 33.7+40.4 s
Übertragungsrate (schreiben): 100.0, 92.6, 104.2, 29.7 MB/s
Mittlere Übertragungsrate (schreiben): 81.6 MB/s
Übertragungsrate (schreiben mit sync): 15.2, 15.9, 15.9, 13.5 MB/s
Mittlere Übertragungsrate (schreiben mit sync): 15.1 MB/s

Beim Schreiben auf RAID5 spielt der Cache eine große Rolle, da das Erzeugen der Prüfsummen recht aufwändig ist. `sync` macht diesen Aufwand sichtbar.

A.4.3 100 Mbit Ethernet

Rechner: `ekpcdf1` und `ekpplusctl` (onboard 100 Mbit)
Dateigrößen: 100, 250, 500, 1000 MB
Übertragungsdauern (lesen): 9.8, 26.5, 49.7, 100.0 s
Übertragungsraten (lesen): 10.2, 9.4, 10.1, 10.0 MB/s
Mittlere Übertragungsrate (lesen): 9.9 MB/s
Übertragungsdauern (schreiben): 9.4, 23.2, 46.6, 102.3 s
Übertragungsraten (schreiben): 10.6, 10.8, 10.7, 9.8 MB/s
Mittlere Übertragungsrate (schreiben): 10.5 MB/s

Die mit Fast Ethernet erreichten Übertragungsraten liegen sehr nahe an den theoretisch erreichbaren Werten. Die Übertragungsraten sind im gesamten Bereich unabhängig von der Dateigröße, also vom Cache einer der beiden Maschinen.

A.4.4 Gigabit Ethernet

Rechner: `ekpdelphi` und `ekpfs1`
Dateigrößen: 100, 250, 500, 1000 MB
Übertragungsdauern (lesen): 4.2, 10.9, 20.2, 44.2 s
Übertragungsraten (lesen): 23.8, 22.9, 24.8, 22.6 MB/s

Mittlere Übertragungsrate (lesen): 23.5 MB/s
Übertragungsdauern (schreiben): 4.9, 10.1, 19.2, 53.7s
Übertragungsraten (schreiben): 20.4, 24.7, 26.0, 18.6 MB/s
Mittlere Übertragungsrate (schreiben): 22.4 MB/s

Die Übertragungsraten von Gigabit Ethernet sind insgesamt enttäuschend; sie liegen lediglich bei 20-25% des theoretischen Wertes.

A.5 Arbeitsweise von RAID-Controllern

"Redundand Arrays of Inexpensive Disks" (RAID) sind zusammengeschaltete Platten, die einander auf je nach RAID-Level unterschiedliche Weise ergänzen. Die gebräuchlichsten RAID-Level sind:

- RAID 0 *Striping*: die Daten werden gleichmäßig auf N Platten verteilt: bei einer Datei von 4 MB Größe beispielsweise wird auf ein 4 Platten-Array je ein MB pro Platte geschrieben. Bei N baugleichen Platten ver-N-facht dies die Datentransferrate, jedoch sind bei Versagen nur einer einzigen Platte alle Daten verloren, da grob gesagt jeder Datei ein N-tel fehlt. RAID 0 wird eingesetzt, wenn die Übertragungsrate von großer Bedeutung ist und ein externes Backup existiert. Die volle Kapazität der N Platten steht zur Verfügung.

- RAID 1 *Spiegeln*: zwei (gleichgroße) Festplatten mit exakt demselben Inhalt. Alle Daten werden gleichzeitig auf beide Platten geschrieben. Fällt eine Platte aus, ist eine vollständige Kopie vorhanden und sofort einsatzbereit. Durch Spiegelung steht nur die Hälfte der Plattenkapazität zur Verfügung; die Übertragungsrate entspricht der einer einzelnen Platte. Durch die nur 50%ige Kapazitätsnutzung wird RAID 1 meist nur bei zwei Festplatten angewandt; bei vier oder acht Platten empfiehlt sich der Einsatz von RAID 5.

- RAID 10 *Striping und Spiegeln kombiniert*, auch als RAID 1+0 bezeichnet: hierbei spiegeln sich zwei RAID 0-Verbände mit je N Platten. Bei einer 50%igen Kapazitätsnutzung ergibt sich so eine N-fache Übertragungsrate bei voller Redundanz. RAID 10 wird angewandt, wenn hohe Datenraten bei hoher Ausfallsicherung gefordert werden – und der Preis keine Rolle spielt.

- RAID 5 *verteilte Prüfsummen*: von den Daten wird eine Prüfsumme berechnet und gespeichert, sodaß von N Platten die Kapazität von $N-1$ Platten zur Verfügung steht, während die Kapazität einer Platte zur Speicherung der Prüfsummen verwendet wird, mit deren Hilfe beim Ausfall einer der N Platten die Originaldaten rekonstruiert werden können. Beim selten verwendeten RAID-Level 4 werden alle Prüfsummen auf einer Platte gespeichert, bei RAID 5 speichert jede Platte ein Ntel der Prüfsummen, aus denen bei Versagen einer Platte deren Daten rekonstruiert werden können. Theoretisch ist beim Beschreiben von RAID 5-Verbänden die $N-1$-fache Datenrate einer einzelnen Platte erreichbar, da $N-1$ Platten unabhängig voneinander Daten schreiben können. In

der Praxis jedoch bleibt die Datenrate weit hinter der eines RAID 0-Verbundes mit $N-1$ Platten zurück, da das Berechnen der Prüfsummen sehr CPU-intensiv ist. Aus diesem Grund müssen RAID 5-Controller über leistungsstarke Prozessoren verfügen, was sie verhältnismäßig teuer macht. Beim Lesen ist der Einfluß der Prüfsummenberechnung weniger stark. RAID 5-Verbände bestehen üblicherweise aus 4 oder 8 Platten und werden eingesetzt, wenn große Datenmengen ausfallsicher gespeichert werden müssen.

RAID-Controller gibt es für SCSI- und IDE-Platten. Viele Controller bieten die Möglichkeit des *hot swapping*, also des Wechselns defekter Platten im laufenden Betrieb ohne Abschalten des Rechners. Manche bieten für die RAID-Level 1 und 5 auch die Möglichkeit einer *hot spare*-Platte an, also einer im Normalfall gänzlich unbenutzten Platte, die beim Versagen einer Platte bis zu deren Ersetzung automatisch deren Funktion übernimmt.

A.6 Filterregeln der Firewall

Hier ist das komplette Regelwerk der Firewall/NAT-Maschine mit kurzen Kommentaren zur Bedeutung der jeweiligen Regeln. Die Syntax der Regeln, ebenso wie zahlreiche Tips und Beispiele, finden sich auf [62].

```
#
# Packet Filter Configuration /etc/pf.conf
#
# Normalize: reassemble fragments and resolve or reduce traffic ambiguities
#
scrub in all

#
# Translate packets from EKPplus internal network
#
nat on gx1  from 192.168.0.0/16 to any -> 129.13.102.40
nat on fxp0 from 192.168.0.0/16 to 129.13.102.0/26 -> 129.13.102.40

#
# RULES FOR OUTBOUND INTERFACE (gx1)
#
# Block stray traffic. Pass only packets to EKP/1 and EKP/2 subnets.
block in log on gx1 from any to any
pass in on gx1 from any to 129.13.102.0/26
pass in on gx1 from any to 172.22.9.0/26

# Block bad packets from the outside.
block in log quick on gx1 proto tcp from any to any flags FUP/FUP

# Block obviously spoofed packets.
```

```
block in log quick on gx1 from 129.13.102.0/27 to any
block in log quick on gx1 from 129.13.102.32/32 to any
block in log quick on gx1 from 129.13.102.33/32 to any
block in log quick on gx1 from 129.13.102.34/32 to any
block in log quick on gx1 from 129.13.102.35/32 to any
block in log quick on gx1 from 129.13.102.36/32 to any
block in log quick on gx1 from 129.13.102.37/32 to any
block in log quick on gx1 from 129.13.102.38/32 to any
block in log quick on gx1 from 129.13.102.39/32 to any
block in log quick on gx1 from 129.13.102.40/32 to any
block in log quick on gx1 from 129.13.102.41/32 to any
block in log quick on gx1 from 129.13.102.42/32 to any
block in log quick on gx1 from 129.13.102.43/32 to any
block in log quick on gx1 from 129.13.102.44/32 to any
block in log quick on gx1 from 129.13.102.45/32 to any
block in log quick on gx1 from 129.13.102.46/32 to any
block in log quick on gx1 from 129.13.102.47/32 to any
block in log quick on gx1 from 129.13.102.48/32 to any
block in log quick on gx1 from 129.13.102.49/32 to any
block in log quick on gx1 from 129.13.102.50/32 to any
block in log quick on gx1 from 129.13.102.51/32 to any
block in log quick on gx1 from 129.13.102.52/32 to any
block in log quick on gx1 from 129.13.102.53/32 to any
block in log quick on gx1 from 129.13.102.54/32 to any
block in log quick on gx1 from 129.13.102.55/32 to any
block in log quick on gx1 from 129.13.102.56/32 to any
block in log quick on gx1 from 129.13.102.57/32 to any
block in log quick on gx1 from 129.13.102.58/32 to any
block in log quick on gx1 from 129.13.102.59/32 to any
block in log quick on gx1 from 129.13.102.60/32 to any
block in log quick on gx1 from 172.22.9.0/27 to any
block in log quick on gx1 from 172.22.9.32/32 to any
block in log quick on gx1 from 172.22.9.33/32 to any
block in log quick on gx1 from 172.22.9.34/32 to any
block in log quick on gx1 from 172.22.9.35/32 to any
block in log quick on gx1 from 172.22.9.36/32 to any
block in log quick on gx1 from 172.22.9.37/32 to any
block in log quick on gx1 from 172.22.9.38/32 to any
block in log quick on gx1 from 172.22.9.39/32 to any
block in log quick on gx1 from 172.22.9.40/32 to any
block in log quick on gx1 from 172.22.9.41/32 to any
block in log quick on gx1 from 172.22.9.42/32 to any
block in log quick on gx1 from 172.22.9.43/32 to any
block in log quick on gx1 from 172.22.9.44/32 to any
```

```
block in log quick on gx1 from 172.22.9.45/32 to any
block in log quick on gx1 from 172.22.9.46/32 to any
block in log quick on gx1 from 172.22.9.47/32 to any
block in log quick on gx1 from 172.22.9.48/32 to any
block in log quick on gx1 from 172.22.9.49/32 to any
block in log quick on gx1 from 172.22.9.50/32 to any
block in log quick on gx1 from 172.22.9.51/32 to any
block in log quick on gx1 from 172.22.9.52/32 to any
block in log quick on gx1 from 172.22.9.53/32 to any
block in log quick on gx1 from 172.22.9.54/32 to any
block in log quick on gx1 from 172.22.9.55/32 to any
block in log quick on gx1 from 172.22.9.56/32 to any
block in log quick on gx1 from 172.22.9.57/32 to any
block in log quick on gx1 from 172.22.9.58/32 to any
block in log quick on gx1 from 172.22.9.56/32 to any
block in log quick on gx1 from 172.22.9.57/32 to any
block in log quick on gx1 from 172.22.9.58/32 to any
block in log quick on gx1 from 172.22.9.59/32 to any
block in log quick on gx1 from 172.22.9.60/32 to any
block in log quick on gx1 from localhost to any
block in log quick on gx1 from 0.0.0.0/32 to any
block in log quick on gx1 from 255.255.255.255/32 to any

# Pass ports above 1023 except some services. Port 6000/6001: X login
block in log quick on gx1 proto { tcp, udp } from any to any port sunrpc
block in log quick on gx1 proto { tcp, udp } from any to any port nfsd
block in log quick on gx1 proto tcp          from any to any port mysql
block in log quick on gx1 proto tcp          from any to any port 6000
block in log quick on gx1 proto tcp          from any to any port 6001
pass  in     quick on gx1 proto { tcp, udp } from any to any port 1024 >< 65535

# Pass selected privilegued ports
pass  in     quick on gx1 proto { tcp, udp } from any to any port ntp \
   flags S/SA keep state
pass  in     quick on gx1 proto { tcp, udp } from any to any port domain \
   flags S/SA keep state
pass  in     quick on gx1 proto tcp          from any to any port ssh \
   flags S/SA keep state

# Pass mail and web server
pass  in     quick on gx1 proto tcp from any to 129.13.102.59 port smtp \
   flags S/SA keep state
pass  in     quick on gx1 proto tcp from any to 129.13.102.59 port www \
   flags S/SA keep state
```

```
# Pass in ICMP
pass  in     quick on gx1 proto { icmp } all

# Block the rest.
block in log quick on gx1 all

#
# RULES FOR DESKTOP/NOTEBOOK/PORTAL NETWORK (fxp0)
#

# Pass most ports
pass  in           on fxp0 all

# Except NFS, BOOTP and DHCP
block in log quick on fxp0 proto { tcp, udp } from any to any port sunrpc
block in log quick on fxp0 proto { tcp, udp } from any to any port nfsd
block in log quick on fxp0 proto { tcp, udp } from any to any port bootps
block in log quick on fxp0 proto { tcp, udp } from any to any port bootpc
block in log quick on fxp0 proto { tcp, udp } from any to any port 546
block in log quick on fxp0 proto { tcp, udp } from any to any port 547
block in log quick on fxp0 proto { tcp, udp } from any to any port 647

#
# RULES FOR EKPPLUS NETWORK (gx0)
#

# Pass most ports
pass  in           on gx0 all

# Except NFS, BOOTP and DHCP
block in log quick on gx0 proto { tcp, udp } from any to any port sunrpc
block in log quick on gx0 proto { tcp, udp } from any to any port nfsd
block in log quick on gx0 proto { tcp, udp } from any to any port bootps
block in log quick on gx0 proto { tcp, udp } from any to any port bootpc
block in log quick on gx0 proto { tcp, udp } from any to any port 546
block in log quick on gx0 proto { tcp, udp } from any to any port 547
block in log quick on gx0 proto { tcp, udp } from any to any port 647
```

A.7 Monte Carlo Generator-Einstellungen

Hier angegeben sind die Konfigurationsparameter des Pythia-Generators bei der Erzeugung des Monte Carlo-Samples, aufgeteilt in prompte J/Ψ und solche aus b-Zerfällen. In beiden Fällen wird der Zerfall von $J/\Psi \to \mu^+\mu^-$ erzwungen, die anderen Kanäle auf 0 gesetzt

(Parameter `mdme`). Auch wurde die Pseudorapidität der direkt erzeugten Teilchen auf ± 1 beschränkt (Parameter `ckin`).

A.7.1 Prompte J/Ψ

Aktivierte Prozesse (Parameter `msub`):

- $gg \to J/\Psi g$ (Prozess 86)
- $gg \to J/\Psi \gamma$ (Prozess 106)
- $g\gamma \to J/\Psi g$ (Prozess 107)
- $\gamma\gamma \to J/\Psi \gamma$ (Prozess 108)

```
mod enable Pythia
talk Pythia
   PythiaMenu
      msel set 0
      commonMenu
         # Prompt J/Psi processes (Pythia manual pg. 414, Subprocess
         # summary table
         set_msub -index=86 -value=1
         set_msub -index=106 -value=1
         set_msub -index=107 -value=1
         set_msub -index=108 -value=1
         # decay into mu only.
         # suppress decay into e+ and rndmflavbar; allow mu (859) only
         set_mdme -channelIndex=858 -decayType=0
         set_mdme -channelIndex=859 -decayType=1
         set_mdme -channelIndex=860 -decayType=0
         # limit pseudo rapidity to +-1
         set_ckin -index=13 -value=-1.0
         set_ckin -index=14 -value=1.0
         set_ckin -index=15 -value=-1.0
         set_ckin -index=16 -value=1.0
         show
      exit
   exit
   show
exit
```

A.7.2 J/Ψ aus b-Zerfällen

Aktivierter Prozess (Parameter `msub`):

- $gg \to Q\bar{Q}$ (Pythia-Prozess 82)

Zusätzlich wird als zu erzeugendes Quark b angegeben (`mstp[7] = 5`).

```
mod enable Pythia
talk Pythia
   PythiaMenu
      msel set 0
      commonMenu
         # heavy quark pairs
         set_msub -index=82 -value=1
         # heaviest: b
         set_mstp -index=7 -value=5
         # suppress decay into e+ and rndmflavbar; allow mu (859) only
         set_mdme -channelIndex=858 -decayType=0
         set_mdme -channelIndex=859 -decayType=1
         set_mdme -channelIndex=860 -decayType=0
         # limit pseudo rapidity to +-1
         set_ckin -index=13 -value=-1.0
         set_ckin -index=14 -value=1.0
         set_ckin -index=15 -value=-1.0
         set_ckin -index=16 -value=1.0
         show
      exit
   exit
   show
exit
```

A.8 Spurrekonstruktions-Einstellungen

Hier findet sich die Konfiguration der Silizium-Spurrekonstruktion, die für die Daten und Monte Carlos verwendet wurde. Gegenüber der Voreinstellung wurde lediglich das Outside-In Tracking auf die *Kal*-Variante umgestellt.

```
talk PVFinder
  # use quick z vertex finder
  SAS3DPVFinder set t
  TrueVertices set f
exit

talk SiPatternRecModule
   missingLayer set 0
   PerformOutsideInTracking set f
   AddZToOI set f
```

```
   # perform both Kaltrak OI and Kaltrak SVX standalone, nothing else
   PerformOIZTracking set f
   PerformStandaloneTracking set f
   PerformSvxStandaloneTracking set f
   PerformKalSvxStandaloneTracking set f
   PerformKalOISvxStandaloneTracking set t
   PassAllCandidates set f
   KalStandaloneOff set t
   MinimumPt   set 0.0
   useG3XIntegrator set f
   show
exit
```

Literaturverzeichnis

[1] James O. Coplien. Multi Paradigm Design for C++. Addison-Wesley, 1999.

[2] Bjarne Stroustrup. The C++ Programming Language. Addison-Wesley, 3rd edition, 1997.

[3] CDF Collaboration. The CDF II Detector Technical Design Report. Fermi National Accelerator Laboratory, 1996.

[4] F. Abe et al. Observation of Top Quark Production in $\bar{p}p$ Collisions. Phys. Rev. Lett. 74, 2626, 1995.

[5] F. Abe et al. Observation of the B_c^+ in $p\bar{p}$ Collisions at $\sqrt{s} = 1.8$ TeV. Phys. Rev. Lett. 81, 2432, 1998.

[6] F. Abe et al. Comprehensive measurement of the CP-violation parameter $\sin 2\beta$ in $B^0/\bar{B}^0 \to J/\Psi\ K_s^0$ decays. CDF Note 4855, 1999.

[7] CDF Collaboration. Proposal for Enhancement of the CDF II Detector: An Inner Silicon Layer and A Time of Flight Detector. P909, 1998.

[8] James O. Coplien. Multi Paradigm Design. 2000. Vortrag am Forschungszentrum Informatik in Karlsruhe.

[9] René Brun et al. The ROOT System. Webseite. http://root.cern.ch/.

[10] Peter Wilson et al. CDF II Trigger. Webseite. http://www-cdf.fnal.gov/internal/upgrades/daq_trig/trigger.html.

[11] M. Shimojima et al. Consumer-Server/Logger System for the CDF Experiment. IEEE Proceedings, 1999.

[12] Wolfgang Wagner et al. Online monitoring in the CDF experiment. EPS HEP Proceedings, 2001.

[13] Hartmut Stadie. Online Monitoring des CDF-Detektors in Run II. DPG-Vortrag, 2002.

[14] Koji Ikado. CDF II Consumer Display. Webseite. http://www-cdfonline.fnal.gov/consumer/framework/display/my.html.

[15] Patrick Schemitz. Consumer Database Browser. Webseite.
http://www-cdfonline.fnal.gov/clservlet/servlet/CdfConsumerServlet.

[16] Sun Microsystems. Java. Webseite.
http://java.sun.com/.

[17] Jason Hunter und William Crawford. Java Servlet Programming. O'Reilly, 1998.

[18] Cay S. Horstmann und Gary Cornell. Core Java 2 – Advanced Features, volume II. Sun Microsystems Press/Prentice Hall, 2000.

[19] James Rumbaugh et al. Object-Oriented Modeling and Design. Prentice Hall, 1991.

[20] Erich Gamma et al. Design Patterns. Addison-Wesley, 1995.

[21] Patrick Schemitz. Consumer Database Browser CVS Repository. Webseite.
http://cdfcodebrowser.fnal.gov/CdfCode/source/ConsumerFramework
/ErrorReceiver/ConsumerServlet/.

[22] Patrick Schemitz. Consumer Histogram Browser. Webseite.
http://www-cdfonline.fnal.gov/~schemitz/cgi-bin/chb.cgi.

[23] ImageMagick Studio. ImageMagick. Webseite.
http://www.imagemagick.org/.

[24] Michael Feindt et al. TrackingKal – A Tracking and Alignment Software Package for CDF II Taking Advantage of a Fast Kalman Track-Fitter. CDF Note 5387, 2000.
http://www-ekp.physik.uni-karlsruhe.de/pub/CDF/TrackingKal/cdf5387_trackingkal-pkg.ps.

[25] Paul Avery. Applied Fitting Theory. CBX 91-73, 1991.

[26] Patrick Schemitz. Object-Oriented Design and Algorithms for CDF II. Diplomarbeit, 1999.
http://www-ekp.physik.uni-karlsruhe.de/~schemitz/Diploma/.

[27] Kurt Rinnert. ROOT-based Visualization Techniques in the CDF II Software Environment. Diplomarbeit, 1999.
http://www-ekp.physik.uni-karlsruhe.de/~rinnert/Diploma/diploma.html.

[28] Rob Kennedy et al. Event Data Model 2. Webseite.
http://www-cdf.fnal.gov/upgrades/computing/projects/edm/edm.html.

[29] Andrew Hunt und David Thomas. The Pragmatic Programmer. Addison-Wesley, 2000.

[30] Free Software Foundation. GNU Compiler Collection. Webseite.
http://gcc.gnu.org/.

[31] Stephanie Menzemer. TrackingKal – a Tracking Software Package for CDF Run 2. Doktorarbeit, 2003.

[32] Claudia Lecci et al. Pre-Tracking Primary Vertex z Finder. CDF Note 5988, 2002.

[33] S. Dell'Agnello et al. A Primary Vertex Finding Package. CDF Note 1789, 1992.

[34] Hartmut Stadie. Vertexrekonstruktion in der CDF Run II Softwareumgebung. DPG-Vortrag, 2000.

[35] CMS Collaboration. Compact Muon Solenoid. Webseite. http://cmsdoc.cern.ch/cms/outreach/html/index.shtml.

[36] Tom Christiansen. CSH programming considered harmful. Webseite, 1996. http://www.faqs.org/faqs/unix-faq/shell/csh-whynot/.

[37] Veridian Systems. OpenPBS. Webseite. http://www.openpbs.org/.

[38] Free Software Foundation. GNU's Not Unix. Webseite. http://www.gnu.org/.

[39] Patrick Schemitz. Fleximon Flexible Monitoring Framework. Webseite. http://fleximon.sourceforge.net/.

[40] Joanna Weng. Study of the Higgs $H \rightarrow 4\mu$ Decay within the Object-Oriented Framework of the CMS Experiment. Diplomarbeit, 2003. http://www-ekp.physik.uni-karlsruhe.de/~weng/Diploma/.

[41] Thomas Allmendinger. Fleximon-Plugin für Conrad Digitalthermometer 304. Webseite. http://www-ekp.physik.uni-karlsruhe.de/~clusmon/download/download.html.

[42] 3ware. Escalade 7000 Kontroller Produktbeschreibung. Webseite. http://www.3ware.com/products/Escalade700DS-7.4.pdf.

[43] Tyan. ThunderK7 Produktbeschreibung. Webseite. http://www.tyan.com/products/html/thunderk7.html.

[44] Theo De Raadt et al. OpenBSD. Webseite. http://www.openbsd.org/.

[45] William V. Courtright II et al. RAIDframe: A Rapid Prototyping Tool for RAID Systems. Webseite. http://www.cs.cmu.edu/Web/Groups/PDL/RAIDframe/.

[46] Allied Telesyn. AT-8224XL Produktbeschreibung. Webseite. http://www.allied.com/products/at8224xl.html.

[47] Hewlett-Packard. HP ProCurve Produktbeschreibung. Webseite. http://www.hp.com/rnd/products/switches/switch4100GLseries/summary.htm.

[48] Thomas Allmendinger. Raumtemperatur EKPplus. Webseite. http://www-ekp.physik.uni-karlsruhe.de/~clusmon/.

[49] K. Anikeev et al. Calibration of Energy Loss and Magnetic Field using J/Ψ Events in Run II. CDF-Note 5958, 2002.

[50] Bruce Knuteson und Mel Shochet. Datasets and Raw Datastreams for Run II. CDF Note 5565, 2001.

[51] Kirsten Tollefson. Entity Diagram for Trigger Database. Webseite, 2001. http://www-cdf.fnal.gov/internal/upgrades/daq_trig/level3/filter/ERD/ERD.html.

[52] Torbjorn Sjostrand et al. PYTHIA 6.2: Physics and Manual. 2001. http://www.thep.lu.se/~torbjorn/Pythia.html.

[53] CERN Computing and Networks Division. Geant 3. Webseite. http://wwwinfo.cern.ch/asdoc/geantold/GEANTMAIN.html.

[54] Torbjorn Sjostrand et al. Pythia Manual, Subprocess Summary Table. Webseite, 2001. http://www.thep.lu.se/~torbjorn/Pythia.html.

[55] SuSE Linux AG. Pressemitteilung: SuSE Linux unterstützt x86-64 Technologie. Webseite, März 2002. http://www.suse.de/de/press/press_releases/archive02/x86_64.html.

[56] Wouter Tinus. Intel Itanium sneak preview. Webseite, Januar 2001. http://print.tweakers.net/?reviews/204.

[57] LCG Group. LHC Computing Grid Project. Webseite. http://lhcgrid.web.cern.ch/LHCgrid/.

[58] Ian Foster und Carl Kesselman. The Globus Project. Webseite. http://www.globus.org/.

[59] Bruce Schneider. Angewandte Kryptographie. Addison-Wesley, 1996.

[60] Kurt Zeilenga et al. OpenLDAP. Webseite. http://www.openldap.org/.

[61] Caldera/Conectiva/SuSE/Turbo Linux. United Linux. Webseite, 2002. http://unitedlinux.com/.

[62] OpenBSD. IP-Filter Frequently Asked Questions. Webseite. http://www.openbsd.org/faq/faqipf.html.

Literaturangaben mit Internetadressen (URLs) beziehen sich, soweit nicht explizit angegeben, auf das Erscheinungsdatum der Arbeit.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die (direkt oder indirekt) zum Gelingen dieser Arbeit beigetragen haben. Dazu gehören vor allem meine Kollegen am EKP, die für eine lockere, produktive Arbeitsatmosphäre gesorgt haben. Besonderer Dank gilt außerdem:

- Prof. Dr. Michael Feindt, meinem Doktorvater, der mir bei der vorliegenden Arbeit enorme Freiheiten eingeräumt hat und dennoch immer ansprechbar war.
- Prof. Dr. Günter Quast, der das Projekt "EKPplus" iniziiert hat und auch das Korreferat übernommen hat.
- Meinem Bürogenossen Marcel Stanitzki, der sich nicht nur für sich selbst, sondern auch für andere einsetzt, zum Beispiel als "Geheimrat" im GK.
- Der CDF-Softwaregruppe des EKP, insbesondere Dominic Hirschbühl, Claudia Lecci, Stephanie Menzemer, Kurt Rinnert, Alexander Skiba und Hartmut Stadie – (mindestens) einer von ihnen weiß immer Rat.
- Meinen Sysadmin-Kollegen (zu verschiedenen Zeiten) Dr. Thomas Allmendinger, Yves Kemp und Ulrich Kerzel, die mit mir für das Wohl unserer Rechner kämpften und kämpfen.
- Meinen Rezensenten Marcel Reich-Stanitzki, Yves Löffler-Kemp und Kurt Rinnert-Karasek, die im "Literarischen Trio" diese Arbeit auseinandergenommen und neu zusammengesetzt haben.
- Edeltraud Haas, ohne deren Engagement und Hilfe der Anschaffungsprozess der EKPplus nicht zu bewältigen gewesen wäre.
- Dem Land Baden-Württemberg und dem Graduiertenkolleg "Elementarteilchenphysik an Beschleunigern", die die ersten zwei Jahre dieser Arbeit finanziert haben.
- Meinen Freunden, meiner Familie und ganz besonders Corry, die mich immer wieder daran erinnert haben, daß es auch ein Leben *außerhalb* von Physik und Computern gibt und ohne die ich den Bettel mehr als einmal hingeschmissen hätte.