Abstract. For over a decade, dCache has been synonymous with large-capacity, fault-tolerant storage using commodity hardware that supports seamless data migration to and from tape. In this paper we provide some recent news of changes within dCache and the community surrounding it. We describe the flexible nature of dCache that allows both externally developed enhancements to dCache facilities and the adoption of new technologies. Finally, we present information about avenues the dCache team is exploring for possible future improvements in dCache.

1. Introduction
For over a decade, dCache [1] has been synonymous with large-capacity, fault-tolerant storage using commodity hardware that supports seamless data migration to and from tape. Over that time, it has satisfied demanding requirements of various scientific communities to store their data, transfer it between sites and provide fast, site-local access.

dCache satisfies these demands by delivering a storage system capable of storing and providing access to immutable files: files that, once written, are not allowed to change. Although the files are immutable, they may still be deleted, renamed, and moved between directories. Updating the contents of a file is possible by deleting the file and creating a new file with the same name, or by uploading the updated contents as a different file and renaming the new file so it replaces the old one.

Access to data is provided through multiple protocols, with each operating on a coherent view of the stored files. This variety allows clients to choose protocols that are best suited for the application; e.g., some protocols are more suited to local access and others for wide-area transfers. Subject to the capabilities of the protocol, clients can also modify the namespace by, for example, renaming, moving or deleting files and directories.

dCache is designed to scale, both in terms of storage capacity (there are multiple dCache instances that have online capacity of over 20 PB) and performance. This level of scalability is achieved, in part, from the separation of namespace from data storage, which allows direct communication between clients and the data servers that host the file’s data, subject to support in the underlying protocol.

dCache also provides a rich set of tools for controlling and managing the data placement within the dCache cluster. Common operations (such as provisioning extra storage or
decommissioning old storage) may be undertaken without interrupting user access to data. dCache also supports several advanced features, including dynamic replication, hot-spot detection with automatic mitigation and periodic background integrity checks of the stored files.

When the dCache project started, the focus was on managing a relatively small disk cache in front of large tape archives. Over the project’s lifetime the storage technology has evolved, driving down the cost-per-GiB of hard-disks and resulting in appearance of huge disk-only dCache installations.

The remainder of this paper is split into five sections. The next section provides an update on various dCache issues, including funding, training activity, links with industry, community building and software license. Section 3 describes some of the changes in dCache code base, including the modularisation of the code and how this is allowing external projects to benefit from dCache code. After this there is a section describing many ways dCache behaviour may be altered by writing customised plugins. This section is followed by a description of future directions: new features that are currently being developed. The paper concludes with a summary of the major points discussed.

2. dCache Community Updates
In this section we summarise some recent activities going on within the dCache team and some noteworthy events within the community of dCache users.

2.1. Funding
The dCache project has existed as a collaboration between DESY [2] and Fermilab [3] for over ten years. In 2007, the NDGF [4] joined, further strengthening the collaboration. Although the dCache.org team accepts external contributions, these three sites provide the bulk of the effort in maintaining dCache.

The WLCG [5] alliance is the group of institutes that provide resources to store and analyse the data taken by CERN’s LHC experiments. Members of the dCache.org collaboration supply dCache as part of their contribution to WLCG. The software is made available to the WLCG community and is used by sites throughout the world. Including all sites, dCache software provides roughly half of the storage capacity available to the CERN LHC experiments.

The EMI project [6] was established in 2010 to create a unified software platform for high performance distributed computing. This has involved the consolidation and harmonisation of four different software stacks so that the software will support communities beyond WLCG.

As services running dCache software provide critical resources within WLCG, it is natural that dCache receives some of EMI’s funding. Such funding has supported work towards solving the catalogue synchronisation problem, integration with the authorisation service ARGUS, providing storage accounting, updating the info-provider to use GLUE2.0 and support for the HTTP, WebDAV and NFS v4.1/pNFS protocols.

The EMI project will end in May 2013. As dCache.org project continues to enjoy support from its principal investors (DESY, Fermilab and NDGF/NORDUnet) the project will continue beyond the end of EMI. In addition, the Large Scale Data Management and Analysis (LSDMA) project is starting as EMI is ending. The LSDMA depends on running dCache services, therefore the project will fund people who will work on dCache. Overall, the end of EMI will result in only a small reduction of effort supporting dCache.

2.2. International dCache Workshops
In April 2012, the German dCache support team organised the “6th international dCache workshop” in Zeuthen, near Berlin. The workshop was well attended; indeed, the interest in workshop was so overwhelming that organisers had to turn some applications down due to
physical limitations of the venue. The workshop attendees included representatives from all the dCache Tier-1 centres in addition to many of the smaller sites. Overall, the attendees reflected dCache’s international deployment, with people coming from 13 countries covering roughly 1/3 of the world.

Before the main workshop, there was a half-day tutorial for people who had no previous experience with dCache. This tutorial provided a short introduction to dCache itself while the programme of the workshop proper covered more advanced topics. The tutorial had both introduction talks and practical, hands-on experience. During these hands-on section, the attendees were able to install and configure a working dCache within minutes.

Each dCache workshop has a main theme that guides the choice of talks. For the most recent workshop, the focus was the new identity management component: gPlazma 2. This is a redesigned system, allowing dCache to overcome some previous limitations, such as inability to easily combine dCache with site-local authentication mechanism. Section 4.3 will provide a more detailed description of gPlazma.

The workshop included talks explaining design and work-flow of gPlazma 2. Talks also described the set of gPlazma 2 plugins supplied with dCache. This information gave administrators an insight into how gPlazma 2 works and what are its benefits over the existing system. The workshop also included a hands-on session providing practical experience in setting and running dCache with gPlazma 2.

Informal feedback from the workshop attendees suggests that many sites anticipate switching from gPlazma 1 to gPlazma 2. The administrators anticipate the new gPlazma will be useful to them and will plan to migrate their sites soon.

2.3. Links with industry
DESY is one of Germany’s principal sites for investigating high-energy particle physics and has been the location of several particle-physics facilities. The experiments from such facilities have stored large amounts of data and it is this experience that lead to the formation of the dCache.org collaboration with Fermilab, a site with similar experience in handling large amounts of data. This need to handling large amounts of data has continued at DESY with XFEL (the X-ray Free-Electron Laser) facility [7], which is currently under construction. Once completed, the facility can generate multi-petabytes of data per month, depending on the operational mode.

At CeBIT 2012, DESY announced [8] a strategic partnership with IBM in tackling Big Data problems. This agreement cited DESY’s experience with large data and, in particular, the dCache project as motivation for forming the partnership.

2.4. Community support
Increasingly, software projects seek to blur the sharp boundary between those that provide the software and those that make use of it. In particular, it is important to reduce (or ideally, eliminate) barriers that might exist preventing external members from contributing to dCache, in whatever form their contribution might take.

One method of providing a more open environment is the use of social media sites. In particular, Stack Exchange is widely used to provide a forum for different communities to ask a question and receive answers to that question. The site already provides forums for many high-level software components and dCache.org and the community of dCache administrators are establishing a forum [9]. Stack Exchange has a formal process through which new forums are established; this is designed to ensure the underlying community is big enough and active enough to maintain the forum. It is currently unclear if the dCache community will satisfy their requirements.

Separate from Stack Exchange, the dCache.org team has introduced new avenues for keeping dCache administrators up-to-date. In addition to the existing mailing list (the “user forum”),
information is also made available through Google+ and Twitter.

2.5. Software License
Traditionally, dCache used a custom license on the code that allows access to the source-code and modification of the code, but that did not allow redistribution of such modified versions.

Since version 2.2, dCache has adopted a new software license: AGPLv3 [10]. This is similar to the GPL license but includes a requirement to make the source code available to users of the software. In practice, this means that sites are allowed to modify dCache to provide additions but they must make such modifications public.

3. Technical changes
There have been some interesting changes to the dCache code-base recently.

3.1. Modularisation
There has been recent activity within the dCache team on modularising parts of dCache. This allows reuse of components outside of dCache in new and novel locations. This is obviously beneficial for the projects using dCache components; however, it also benefits the dCache project by providing feedback, such as bug-reports and patches.

The following components now also enjoy life as separate projects.

Chimera This is the new namespace implementation used within dCache. It has replaced the older PNFS, which suffered from performance limitations in terms of throughput and scalability.

In addition to being used within dCache, Chimera is now also used by Enstore. Enstore is the tape access system developed by Fermilab and used both there and at the Spanish WLCG Tier-1 facility, located at PIC.

jrpc dCache supports the NFS v4.1 protocol using a pure Java implementation of the protocol developed at DESY. This library, in turn, depends on a pure Java ONC RPC dispatch library, which was also developed at DESY.

This ONC RPC library is being used by BACNET and a Swiss bank. There may be other users that we are unaware of.

xrootd4j xrootd is a proprietary protocol developed at SLAC used primarily by the particle physics community. As support for xrootd is mandated by WLCG, dCache provides xrootd protocol which is implemented as stand-alone Java library. This library allows others to implement pure Java support for xrootd.

The WLCG ALICE experiment is currently investigating using this library for their next-generation management tools.

3.2. NFS v4.1
One of the key features of dCache is its support for NFS v4.1 along with the pNFS extension. NFS 4.1 is the latest version of the long standing NFS protocol, yet it has marked differences from earlier versions, such as sessions, allowing call-backs from the server to the client and allowing different data and control channels.

The pNFS extension allows clients to perform parallel I/O by sending or receiving data directly to the server node on which the data is stored. This removes the congestion otherwise created on the single machine providing the mount-point, so eliminating a performance bottleneck.

HEP storage systems have long required more storage than can fit on a single server. To provide sufficient bandwidth, such systems require a protocol that supports parallel data
transfers. The lack of an industry solution has lead to the HEP community developing proprietary protocols. These are problematic since they increase the support load and impede other communities from using HEP-developed storage systems.

With the development of NFS v4.1/pNFS, it becomes practical to use an industry standard protocol for HEP work-flows. It removes the need to have custom client software and allows other communities to use the same software, providing an economy of scale.

The following list provides the current status [11] of the different hardware vendor’s support for NFS v4.1/pNFS:

- NetApp Supported since OnTap v8.1,
- Panasas Support coming “in 2012” [11],
- IBM “will be key part of SONAS Active Cloud Engine” [11],

Here is the current status of client support for NFS v4.1/ pNFS in various operating systems:

- RHEL supported since 6.x, 5.x requires installing an Oracle kernel and a custom nfs-utils package.
- Fedora supported since version 15,
- Debian supported since 7.0 (“Wheezy”),
- Ubuntu supported since 9.10 (“Karmic Koala”),
- Windows support since Windows 8, an open-source driver from CITI (University of Michigan) adds support for Windows 7,
- Solaris see Oracle’s statement above.

NFS v4.1/pNFS has been supported in dCache since 2009. It has been deployed in production (at DESY) for over a year without any problems.

dCache’s implementation of pNFS was evaluated by Femilab’s REX department for their “Intensity Frontier” experiments. They concluded that the “results look promising, throughput scales well with number of pool nodes.”

dCache NFS supports both trusted-host and GSS authentication. The GSS library implements Kerberos support.

When using GSS (such as Kerberos), three security modes are available: authentication, integrity and privacy. dCache supports all three modes. With authentication, the sender is identified with cryptographic assurance but no guarantee is made that the data has not been modified. In addition to the guarantees of authentication, integrity guarantees that the data has not been modified. Finally, privacy provides cryptographically strong assurance of the sender’s identity, that the data was unmodified and that it was not read when sent over the network.

4. Flexibility

As dCache has evolved over the many years it has been developed, many aspects have been improved. When some aspect of dCache was modified, it is desirable to allow sites to run either the old or new behaviour. This allowed sites to thoroughly test new features before deploying them in production environments without halting their ability to receive new versions of dCache.

Therefore, as dCache evolved, extension points were added to the original code-base. An extension point is a place within dCache’s logical structure that allows an administrator to provide an alternative, custom behaviour. Typically, this is achieved by providing plugins. By configuring dCache to use a different plugin, the behaviour of an extension point may be modified.

As we will show in 4.2, there are several extension points where the behaviour of dCache may be modified.
4.1. Who should be interested

There are three main groups for whom dCache’s flexibility is of interest.

**developers** Supporting plugins provides developers with some key benefits. By retaining the existing plugin as the default configuration, plugins allows additional functionality to be rolled out without immediately affecting the site. Then, sites can test the new feature and provide feedback to dCache without the new plugin affecting unprepared sites. This provides an additional layer of testing and safe-guard when rolling out new features.

**administrators** dCache is deployed in a variety of different environments. While dCache features a rich configuration system, sometimes it is desirable to have dCache integrated with a site-local service (e.g., a database of users). If this service is bespoke, developed for some specific set of users, then it may be difficult for the generic dCache to provide sufficient flexibility through configuration. By providing extension points, dCache allows an administrator to write the site-specific code needed to integrate dCache with their site-local, bespoke system. They can also maintain this plugin independently of dCache’s update release process. This allows them to adjust the integration plugin in a timely fashion, matching changes to their bespoke system.

**communities** Sometimes a user-community will devise some special behaviour that they expect storage systems, like dCache, to support. An example of this is namespace mapping. This is where a file is accessed with a world-wide common path for the file and the storage system automatically converts this path to the location on the site where the file is actually stored. Rather than the dCache core team adding support for this behaviour, providing extension points allow the community to provide the necessary behaviour by writing a plugin. The distribution of this plugin can be independent of dCache, allowing the user-community to keep their plugin up-to-date independent of dCache’s update process.

In practise, all three groups have made use of dCache’s extension points to extend the standard support.

4.2. Overview of plugins

In this section we give a brief overview of the different areas that dCache’s behaviour may be modified by configuring extension points to use different plugins, along with examples of the kind of additional plugins that would make sense.

Each section describes a dCache service, which is the finest grain component that may be deployed within a dCache cluster.

4.2.1. **PoolManager** For each read and write request, the pool-manager makes a decision about which pool is to be used. It has two extension points: *pool-selection* and *partition*.

The pool-selection extension point describes the process of choosing which set of pools is eligible for a particular operation. There is currently a single plugin that selects pools based on dCache configuration (links and units). One example plugin would add a naming convention; for example, all pools with names that start with `atlas` are used for requests from users within the ATLAS virtual organisation.

The partition extension point describes the process of choosing the best pool, from the set of eligible pools. There are currently three plugins available: **Traditional**, **WASS** and **Random**. **Traditional** uses a simple model of disk performance and chooses the best pool based on a configured weighting. **WASS** provides better choice of pool when uploading data but behaves as Traditional when selecting the pool for downloading. **Random** selects a pool for data upload randomly, but the pool chosen for reading is as for Transitional. For more details on the **WASS** algorithm, see [12].
The current plugins assume that all pools have equivalent network connectivity for the client. For distributed systems, this may not be true. A Partition plugin could be written that chooses the best pool based on the expected latency and bandwidth between the client and candidate pools.

4.2.2. Namespace The namespace provides a mapping between paths and unique IDs, fine grain authorisation for the available operations and knowledge of where data for files is located within the dCache cluster.

The main extension point is namespace-provider. There are currently two plugins: PNFS (providing support for the PNFS namespace) and Chimera (providing support for storing information in a relational database). A plugin could be written to support an overlay namespace, allowing multiple namespaces to coexist.

Within the Chimera plugin there is a further extension point: filesystem-provider. This controls how basic information is stored. There is a single plugin for this extension point: JdbcFs. This uses a relational database to store the information and includes support for connecting to databases in an efficient fashion. A FileSystemProvider plugin could be written that allows the storage of namespace information within the Hadoop’s HDFS [13] filesystem namespace, or within a distributed database, like Cassandra [14].

The JdbcFs plugin itself has a further extension point: FsSqlDriver. This allows customisation of the SQL used when JdbcFs is talking to the relational database. There are four plugins available: a generic SQL, a PostgreSQL plugin (that pushes work into the database, where it is handled more optimally), HyperSQL and H2. Plugins could be written to support Oracle, MySQL or to try experimental improved optimisation for an existing database.

4.2.3. Pool The pool stores the data from files. It contains two extension points: the file store and the metadata store.

The file-store is responsible for storing a file’s data. There is currently a single plugin that stores all data in a single directory. A plugin could store data hierarchically (to avoid storing a large number of files in a directory), or might include optimisations for the local filesystem. It could store data using a cloud provider, allowing the pool to act as a proxy to a cloud provider.

The file-metadata stores information about the files stored in the pool. Currently there are two plugins: one that stores data using simple files; the other stores the information in a Berkeley database. A plugin could be written that stores information in a PostgreSQL database, or using a distributed database such as Cassandra.

4.2.4. Billing Billing records user-triggered activity within dCache. It has an extension point describing where the logging information is stored. There are currently two plugins: a plugin that records data in log files and another that stores the information in a relational database. Since the billing information may be stored in a single table, it is suitable for storing and querying using a NoSQL databases. Therefore, a plugin could be written to write billing data into MongoDB [15], Hadoop [16] or CouchDB [17].

4.2.5. xrootd door The xrootd door provides the initial entry-point into dCache for the xrootd clients. It has two extension points: AuthN and AuthZ.

The AuthN extension point is responsible for determining the identity of the dCache user. The xrootd protocol allows for an arbitrary handshake between the client and server, which allows the client to assert its identity to the server.

dCache currently provides two xrootd AuthN plugins: one that expects no handshake and one that allows X.509 certificate exchange. The latter plugin then uses gPlazma to map the certificate to a specific dCache user.
The AuthZ extension point provides two functions. It allows a plugin to provide a custom authorisation decision, where a plugin can decide to disallow particular operations. The extension point also allows the file’s path to be modified. This allows a client to use a global path (the same path for the same file stored on different storage elements). An AuthZ plugin would convert the global path to the site-specific path either algorithmic or by calling out to some external service that provides the conversion.

4.3. gPlazma

The new gPlazma-2 provides a pluggable mechanism for authentication and identity management. There are two main operations that make use of plugins: logging in and identity mapping.

The login process takes user-supplied credentials and attempts to convert them to a user-ID (uid), a set of group-IDs (gids) and some basic information, such as a home directory and the filesystem root for that user. The process is applied when a user first attempts to use dCache. The process may fail, which results in the user failing to login. Such failures will result in either the user being rejected or treated as an anonymous user, depending on the access protocol. Depending on configuration, an anonymous user may still be allowed to do certain operations.

The identity mapping is the process of mapping a uid or gid to zero or more principals, or vice versa. It is used when discovering information about other users of a dCache instance; for example, when generating a detailed listing of a directory or authorising a user by adjusting an ACL.

4.3.1. Login

There are four successive phases when processing a login request: auth, map, account and session. Each phase has a chain of zero or more configured plugins. The plugins in a chain are executed to try to satisfy a particular goal. Here are descriptions of the four phases along with the aim of each phase.

auth The auth phase is concerned with mapping credentials (information that proves identity) to principals (names or identifiers without any proof). An X.509 certificate is an example of a credential and the corresponding Distinguished Name is an example of a principal.

map In the map phase, the principals are converted to other principals, aiming to produce a uid and set of gids.

account The account phase checks if the user may use the system. It is most commonly used to prevent certain users from using the system, for whatever reason.

session The final phase is session. In this phase, information such as the user’s home directory and filesystem root directory are set.

When executed, a configured plugin may succeed or fail. For a phase, the success or failure of a plugin, along with the plugin’s configuration, determines whether the next plugin in the chain is executed or the chain finishes.

Each phase of the login may succeed or fail. A phase’s success depends on the success of the chain of plugins combined with the plugins’ configuration. gPlazma is commonly configured so that the success or failure of the last executed plugin determining the success or failure of a phase.

If any of the four phases fail then the login request fails and the user is anonymous. The information provided by the login process is also checked for certain requirements. If these checks fail then the login also fails.

A custom plugin may be written for any combination of phases; for example, a plugin might provide support in the auth phase to check username and password against information stored in a site’s local account database, support in the account phase to check for banned users, and support in the session phase to add home directory information.
4.3.2. Identity Mapping  
An identity mapping plugin provides the ability to map a uid or gid to some corresponding set of principals. The plugin must guarantee that each of the returned set of principals corresponds the supplied uid or gid and no other.

As with the login processing, the configured identity mapping plugins form a chain of configured plugins; however, unlike with login processing, identity mapping has a single chain with more restricted set of behaviour. If a plugin succeeds then the chain always terminates. The identity mapping process is successful only if the last executed plugin was successful.

A custom plugin may be written to provide custom mapping between uid and username, or between gid and groupname, based on a site-local database of user accounts.

5. Future directions
In this section we describe some of the future plans of dCache; the direction along which dCache is evolving to satisfy future demand.

5.1. HTTP and WebDAV: supporting non-HEP users

dCache is deployed to provide substantial service for HEP users. Sites are increasingly interested in using dCache to support users from other disciplines, such as astronomy and photon-sciences. Such users often have little interest in using HEP-specific protocols. Instead, they would prefer to use industry standards, such as HTTP.

dCache has long supported HTTP as a protocol for fetching files. With HTTP support, obtaining data becomes accessible to anyone since all modern computer platforms have a web-browser.

With the recent addition of support for WebDAV (an extension to HTTP that allows filesystem-like semantics) dCache provides a standards-based access for uploading files and common metadata operations, like browsing directories, renaming files and deleting files and directories.

The industry standard SSL protocol allows secure communication between client and server, where both the client and server may assert their identity.

dCache’s HTTP and WebDAV support has been deployed in production at many sites, including DESY, PIC and BNL.

dCache team will continuing this work by improving the HTTP support; for example, by updating the user interface, providing support for federated identity systems and allowing a dCache installation to be used as cloud storage.

5.2. Federated storage

The workshop at Lyon to facilitate federated storage came up with the following definition of federated storage:

“Collection of disparate storage resources managed by co-operating but independent administrative domains transparently accessed via a common name space.”

Although the Lyon workshop focused on the xrootd protocol, the same functionality may be provided using a standard protocol: HTTP. The benefits of using HTTP include the existence of high-performance clients (maintained by others) and many server and caching software packages (Apache HTTP server [18], Squid [19], Varnish [20], …).

There are two phases to providing an HTTP federated service. First, providing a web front-end to the existing catalogues (e.g., LFC). Second, providing a service that discovers which data is available dynamically by querying storage systems. This work was conducted in collaboration with CERN and, for further details, see [21].
5.3. Missing files
It is possible that a user requests a file that does not exist in a dCache instance. This can happen if the file was never written into dCache or if the file was written into dCache but the file has been subsequently lost (e.g., a pool is offline).

It is possible that dCache should do something in these cases. It could report the missing file to the file catalogues. It could attempt to fetch the missing file from a remote site and store the file. The initial effort in this direction will report the file’s absence using the Syncat notification system. For more details, see [22].

5.4. SSDs and 3-tier model
dCache was initially developed to improve access to tape by caching files that were requested. Over time, it evolved so that, for some instances, there is no tape system and dCache serves files only from disk.

The main characteristic differences between disk and tape is latency and cost. Tape is more cost effective than disk but fetching a file has much higher latency, due to the delay in loading the tape and, once loaded, seeking to the correct location. Given the amount of data a large site needs to store, it is still cost-effective to store the bulk of the data on tape and have only the subset that is currently being processed on disk.

With the availability of Solid State Disks (SSDs), there is a new generation of storage technology. The distinction between SSD and magnetic disks is similar to the distinction between disk and tape: SSDs have a high performance profile (especially with random-access analysis-style access patterns) but are much more expensive than magnetic disks.

As with disk-tape, it is often too expensive to store all data on SSDs. Instead, dCache is investigating adding support for SSDs as a third tier. When requesting a file for reading, dCache will use the file stored on SSD if it is available. If it is not already on SSDs then dCache would make a policy decision whether to replicate the file to an SSD or to make the file available from the slower magnetic disk media.

An initial investigation into whether such an approach makes sense has been conducted. For further details, see [23].

6. Summary
In this paper, we have described how dCache, a contribution to WLCG from several institutes, has funding that is secure after the end of the EMI project.

Recent activity within the dCache community was reported, including a recent training event that attracted record attendance.

We have described how dCache contains numerous places where additional functionality may be added, both to integrating dCache into site-local infrastructure and to adopt new technologies, to remain agile.

We have also given an overview of some of the future directions that the dCache.org team are investigating.

Acknowledgments
This work was partially funded by the EMI project under European Commission Grant Agreement INFSO-RI-261611

References
[22] Furano F SYNCAT — Storage Catalogue Consistency poster at CHEP-2012
[23] Ozerov D and Fuhrmann P Evaluation of benefits of a three tier data model for WLCG analysis oral presentation at CHEP-2012