GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Fakultät für
Physik

**Bachelor's Thesis**

# Hotfile- und Flaschenhals-Erkennung im dCache-System

# Hotfile and Bottleneck Recognition in the dCache System

prepared by

**Christian Georg Wehrberger**

from Bloomington, Indiana (USA)

at the II. Institute of Physics
Georg-August University of Göttingen

| | |
|---|---|
| **Thesis number:** | II.Physik-UniGö-BSc-2011/06 |
| **Thesis period:** | 31st March 2011 until 7th July 2011 |
| **Supervisor:** | Dr. Jörg Meyer, Dr. Pavel Weber |
| **First referee:** | Prof. Dr. Arnulf Quadt |
| **Second referee:** | Prof. Dr. Ariane Frey |

# Abstract

Der ausfallfreie und produktive Betrieb von Grid-Ressourcen für das ATLAS-Experiment am LHC bedingt den Einsatz effizienter und verlässlicher Überwachungssyteme sowie Verbesserungstechniken. Eine dieser Ressourcen stellt dCache, ein explizit für den Einsatz in der Hochenergie-Physik entwickeltes Massenspeicher-Verwaltungssystem, dar. Daher wird im Rahmen dieser Bachelorarbeit die Leistung von dCache exemplarisch am ATLAS Tier-2-Zentrum GoeGrid untersucht. Typische Nutzungsmuster wie Dateigrößen-Verteilungen, aber auch Charakteristika wie Geschwindigkeiten von Datentransfers und Zugriffshäufigkeiten, insbesondere häufig verwendeter Dateien (hotfiles), werden analysiert. Darüber hinaus wird auch eine Flaschenhals-Erkennung durchgeführt, die im GoeGrid Komponenten identifiziert, welche die Leistung des dCache-Systems limitieren. Die Resultate dieser Analysen werden zur Erstellung einer Echtzeit-Überwachungssoftware für GoeGrid eingesetzt.

**Stichwörter:** Physik, Bachelorarbeit, WLCG, ATLAS, Grid Computing, GoeGrid, dCache, HappyFace, Hotfile, Flaschenhals, Bottleneck

The stable operation of Grid resources for the ATLAS experiment at the LHC strongly relies on the deployment of efficient and reliable monitoring systems and optimisation techniques. One of those resources is dCache, a mass-storage management system particularly developed for high energy physics. For this purpose, the performance of dCache is studied exemplary for the ATLAS tier-2 centre GoeGrid in the course of this Bachelor's Thesis. Typical dCache usage patterns like the distribution of file sizes and characteristics like data transfer speeds and file access frequencies, especially for repeatedly accessed files (hotfiles) are analysed. Furthermore, a bottleneck recognition is performed in order to detect the limiting components of dCache at GoeGrid. The results of those analyses are the basis of recently created real-time monitoring software for the use at GoeGrid.

**Keywords:** Physics, Bachelor's Thesis, WLCG, ATLAS, Grid Computing, GoeGrid, dCache, HappyFace, Hotfile, Bottleneck

# Contents

# List of Figures

# List of Tables

# 1 Introduction

In 1994, the CERN [1] Council approved the construction of the *Large Hadron Collider (LHC)* [2], a proton-proton collider with a centre-of-mass energy of 14 TeV. There are four main experiments ALICE [3], ATLAS [4], CMS [5], and LHCb [6], arranged at four collision points, producing experiment and simulation data which have to be processed, analysed, and stored. In the following, the data taking is discussed exemplarily at the ATLAS experiment.

The data flow is generated by a number of sub-detectors, each designed for a specific purpose, arranged in several layers around the beam pipe. Closest to the collision point in the centre of the detector, the *Inner Detector* (*Pixel Detector*, *Silicon Microstrip Tracker*, *Transition Radiation Tracker*) is responsible for tracking charged particles' trajectories. Enclosing the Inner Detector, the calorimetry (electromagnetic and hadronic calorimeters) measures the energy of particles penetrating it. The outermost layer is responsible for registering muons that have most likely bypassed all other detector elements. By the electronics of each of the sub-parts of the ATLAS detector, a data stream is generated, read out over a large quantity of channels. An overview of the number of readout channels is given in table 1.1.

At the LHC, with the bunch crossing rate of 40 MHz about 23 measurable collisions at each crossing are expected for a luminosity of $\mathcal{L} = 10^{34}$ cm$^{-2}$s$^{-1}$. This yields an event rate of $10^9$ Hz [7] at each of the four collision points at one of which the ATLAS experiment is located. Given the event size of $1 - 2$ MBytes, the expected data volume is about $10^{21}$ Bytes per year [7, 8]. The processing of this amount of data is not feasible due to limitations on processing time and storage capacity. Because it is neither necessary nor possible to store all $10^{21}$ Bytes per year, a trigger system is implemented to reduce the initial event rate. It consists of three levels and is designed to reduce the data rate, selecting only interesting events. On *Level-1*, a hardware filter with a very fast selection (below 2.5 $\mu$s latency) already rejects 99.8% of all data using coarse granularity of the detector [8]. The trigger *Level-2* applies full precision readout and more sophisticated

event reconstruction. Thereby, it reduces the data rate by a factor of 100. This trigger level consists of standard Linux PCs with specialized interfaces. The third trigger level called *Event Filter* selects events according to a physics menu list. The utilized computer farm reduces the output rate by a factor of 10, generating data for offline analysis at a rate of about 200 MByte/s corresponding to an event rate of 100 Hz. This results in a data taking rate of about 7 PByte per year. In addition, 3.2 PByte of data per year is produced by simulated events.

| **Detector system** | **Number of active detector elements** | **Number of modules or chambers** |
|---|---|---|
| Pixel detector [9] | 80,363,520 | 1,744 |
| Silicon microstrip tracker [10] | 6,279,168 | about 4,100 |
| Transition radiation tracker [11] | 424,576 | about 240 |
| Liquid Argon calorimeters [12] | 182,468 | 48 |
| Tile calorimeter [13] | 463,500 | 192 |
| Muon system [14–17] | 1,080,608[1] | about 2,000 |

***Table 1.1:*** Active detector elements and modules/chambers in the ATLAS experiment.

For providing quick access, the data is distributed to a world-wide disk and tape storage and decentrally processed with the computing power of about 360,000 today's PCs. This takes place in the *Worldwide LHC Computing Grid (WLCG)* [18]. It is structured in four layers (*tiers*), each of those layers providing specific services. The head of this infrastructure is the CERN Computer Centre (tier-0), passing on the data to tier-1, tier-2, and tier-3 centres. It is also responsible to store an initial copy of the data, for a first reconstruction, and data reprocessing. Eleven national tier-1 centres provide the safe-keeping of a proportional fraction of raw and simulated data, reprocessing, and distribution of

[1] The Muon system consists of four sub-systems: Thin Gap Chambers (321,072 channels), Resistive Plate Chambers (about 350,000 channels), Monitored Drift Tubes (354,240 channels), Cathode Strip Chambers (55,296 channels).

data to tier-2 and tier-3 centres. More than 140 regional tier-2 sites handle certain analysis tasks, produce simulated events, and store a share of reprocessed and simulated data. Some tier-2 centres are also in charge of calibrating data. Tier-3 centres make up the lowest level of the WLCG infrastructure, consisting of local compute resources.

The Grid facility *GoeGrid* [19] contributes to the WLCG as a ATLAS tier-2 centre. Hosted by the *Gesellschaft für Wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)* [20], it is operated mainly by the *II. Institute of Physics* [21] of the *Georg-August University of Göttingen*. The share contribution to WLCG has a *HEPSpec2006* [22] performance of about $25,500$ and provides about half a Petabyte disk storage.

*D-Grid* [23] is the German grid computing initiative. It contains projects like MediGrid [24] (bio-informatics) and TextGrid [25] (human sciences and arts) to which GoeGrid also contributes.

In order to manage the high energy physics data, a scalable mass storage system has to be used. A system designed for this purpose is dCache [26]. Typically consisting of many distributed, heterogeneous elements (storage nodes), it is accessed centrally via different interfaces from all over the world. dCache system also provides the possibility to transfer data from and to tertiary storage.

Mass storage is an essential service for which high performance is required and important for everyone involved in ATLAS analysis and production. In order to optimise the dCache performance for GoeGrid, this Bachelor's thesis deals with several analyses of the dCache system. The optimisation is a multi-step process. It consists of monitoring dCache and identifying possible performance problems. Subsequently, appropriate methods to avoid those problems are developed.

Besides determining and visualizing times of high load, especially files which are accessed significantly more often than others (*hotfiles*) have to be identified as well as whole storage nodes with high access rates. The objective of this thesis is to analyse the dCache performance for GoeGrid and to provide a monitoring tool to identify so-called bottlenecks and hotfiles taking the network topology of GoeGrid into account.

# 2 GoeGrid Tier-2 Resource Centre

GoeGrid is a grid resource centre mainly involved in the WLCG as an ATLAS tier-2 centre, allocating a major part of its resources for the ATLAS experiment. The CPU power is shared by its user groups according to the extent of their contribution as displayed in table 2.1.

| **Contributor** | **Fraction** |
|---|---|
| II. Institute of Physics | 66.8% |
| Institute of Theoretical Physics | 20.7% |
| MediGrid | 10.0% |
| TextGrid | 1.8% |
| GWDG | 0.7% |

***Table 2.1:*** GoeGrid contributors and fractions of usage for scheduling.

All jobs are disposed by a batch scheduler, implementing a *fair share algorithm* to apportion the computing power. Not only Grid jobs are processed, but also local users submit jobs to GoeGrid.

The availability and reliability of GoeGrid and other WLCG sites is regularly tested by *Service Availability Monitoring (SAM)* [27] tests. From January to June 2011 these tests show an average availability of 92.4%. This is about 2% above average of all German WLCG sites. GoeGrid reliability of 94% is also higher-than-average by about 1%.

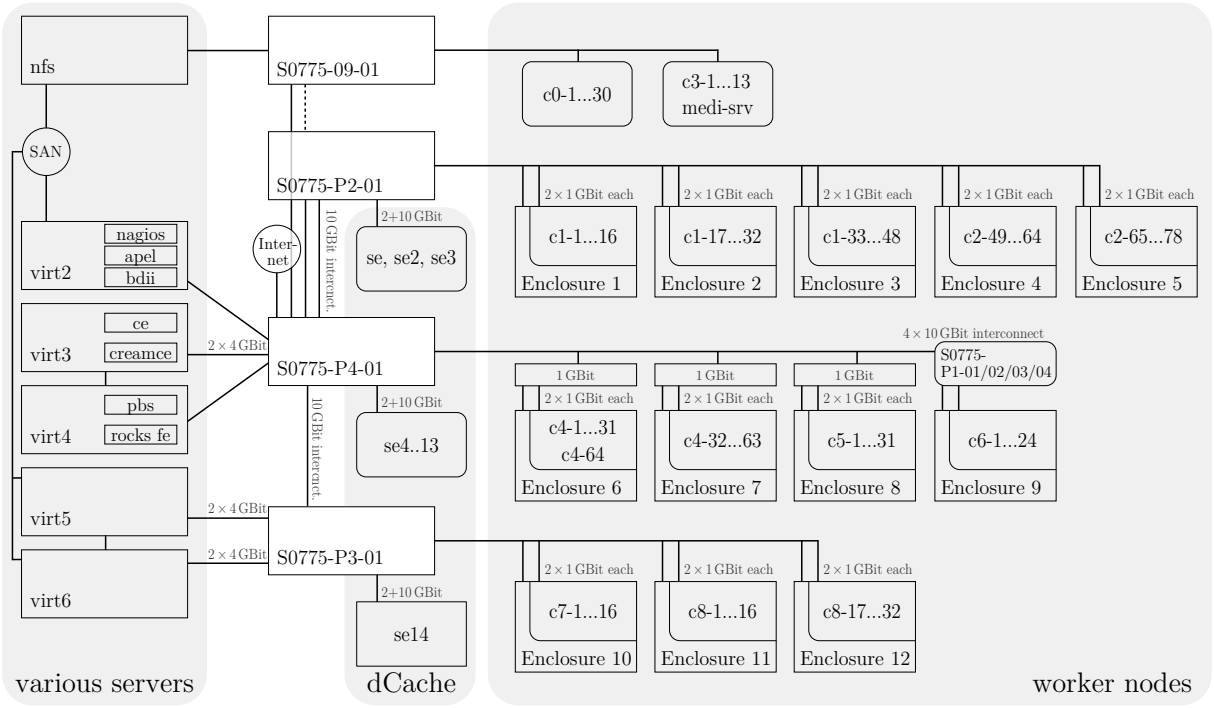


***Figure 2.1:*** GoeGrid network topology.

## 2.1 Hardware Setup

The network topology of GoeGrid in figure 2.1 displays several groups of storage and compute nodes. Typically, one *storage element (se)* contains several RAID systems. All dCache storage nodes add up to a size of approximately 600 TByte. One enclosure contains 16. . . 32 compute elements, each having also a local hard disk. On dedicated servers, ATLAS *glite* [28] middleware, other central services, and local services are hosted. All those components are connected over four central switches with each other and to the Internet and a *Storage Area Network (SAN)*. Due to their historical development, the connection bandwidth within GoeGrid may differ significantly from component to component. While enclosures 6, 7, and 8 are connected via 2 GBit to the central switches, e.g. enclosure 9 is connected via $4 \cdot 10$ GBit. The storage servers are connected with almost equal bandwidths, each with a 1 . . . 2 GBit public network connection and a 10 GBit private network connection.

## 2.2 Software Setup

To ensure the functionality of a resource centre, many services have to be provided.

### Apel

When Monte Carlo simulation data is generated, it is stored into a local database. *Apel (Accounting Processor for Event Logs)* [29] is responsible for publishing the production result and for GoeGrid Monte Carlo simulation accounting.

### BDII

The *BDII (Berkeley Database Information Index)* [30] service collects and publishes site status information. This is static and dynamic information, e.g. space tokens, installed software, and monitoring websites. BDII is structured hierarchically. Each single resource has its *resource BDII*, sending the information to a *site BDII*. A *top-level BDII* gathers its data from the site BDIIs.

### CE and CreamCE

*CE (Computing Element)* [31] is a Grid jobs front-end to the local batch system managed by *Torque*. It is responsible for local user mapping and also publishes accounting information. *Cream CE* is an improved version of CE, using a newer version of *gLite* middleware and more recent Linux kernels.

### Nagios

*Nagios* [32] is a generic open-source monitoring system. In addition to that, it sends alerts in case critical infrastructure fails.

### Torque

*Torque* [33] resource manager, a derivation of *OpenPBS*, provides control of all local batch jobs. It handles the job submission by passing incoming jobs to idle worker nodes, using *Maui* [34] scheduler.

# 3 dCache System

The dCache system is a joint project of *Deutsches Elektronen-Synchrotron (DESY)* [35] Hamburg and *Fermi National Accelerator Laboratory (Fermilab)* [36] Batavia (Illinois) in collaboration with Copenhagen and Linköping University. dCache is a distributed storage solution for storing and transferring large amounts of data. It consists of various services and storage nodes connected in a *Local Area Network (LAN)*. The stored data is managed by a centrally accessible file system, the *purely normal file system (pnfs)* [37]. dCache provides features like scalability, upgradeability, central accessibility and management of large data volumes, data migration, cost metrics to monitor the data flow, user accounting, a web interface for real-time monitoring, and tertiary storage connection (e.g. tape storage). End users can access the pnfs file system without knowledge of its underlying structure. For transferring data, a variety of widely-used protocols is supported, e.g. *dCap* [26], *GridFTP* [38], *GSIdCap* [39], *HTTP*, *SRM* [40], *Web-DAV* [41], and *XRootD* [42].

## 3.1 dCache Components

The base component in dCache is the *cell*, a process within a *Java Virtual Machine (JVM)*, employed by every involved machine except the *pnfs server*. Each dCache cell has a certain role and communicates with other cells via a protocol on top of *TCP/IP* in order to fulfil its specific tasks. Those cells are distributed over a LAN; in most cases, one machine is running several cells. In table 3.1, a list of the most important cells and their functionality is shown [43].

Another component is the dCache *domain*. It is an environment/container for dCache cells being executed in a JVM. There are dCache domain definitions for groups of cells with different tasks to perform. In figure 3.1 the structure of dCache is shown.

| Component name | Functionality |
|---|---|
| I/0-door | data transfers over certain interfaces; one door for each interface: administration, dCap, GridFTP, GSIdCap, HTTP, SRM, WebDAV, and XRootD. |
| pnfs manager (server) | managing the pnfs file system (hierarchy), pnfs database, meta-data |
| pool | data storage |
| pool manager | handling the pool interaction, *transfer request* for each user action (retrieving files from one or more pools), pool replication, tertiary storage management, pool selection unit (finding pools), cost manager (selecting best pool via cost metric) |
| *gPLAZMA* [44] | *Grid-aware PLuggable AuthoriZation Management*: user authorization over various methods: *kpwd*, *Grid-mapfile*, *gPLAZMAlite-VORole-Mapping*, *Sams-VO-mapping* |

***Table 3.1:*** List of dCache cells and corresponding functionality.

## 3.2 pnfs File System

The file system dCache bases upon is pnfs. Each object in the namespace of pnfs is assigned a unique 96 bits ID. Although some bits are used for the identification of the database and other management information, the size of a database is almost unlimited (a database can have up to $2^{77}$ entries). All database entries are stored in a *GNU dbm* [45] database file.

## 3.3 dCache Protocols

dCache provides various protocols to access the stored data as aforementioned. The I/0-doors are implemented as dCache cells and work like plugins: each door can be set up individually and works independent of all other doors.

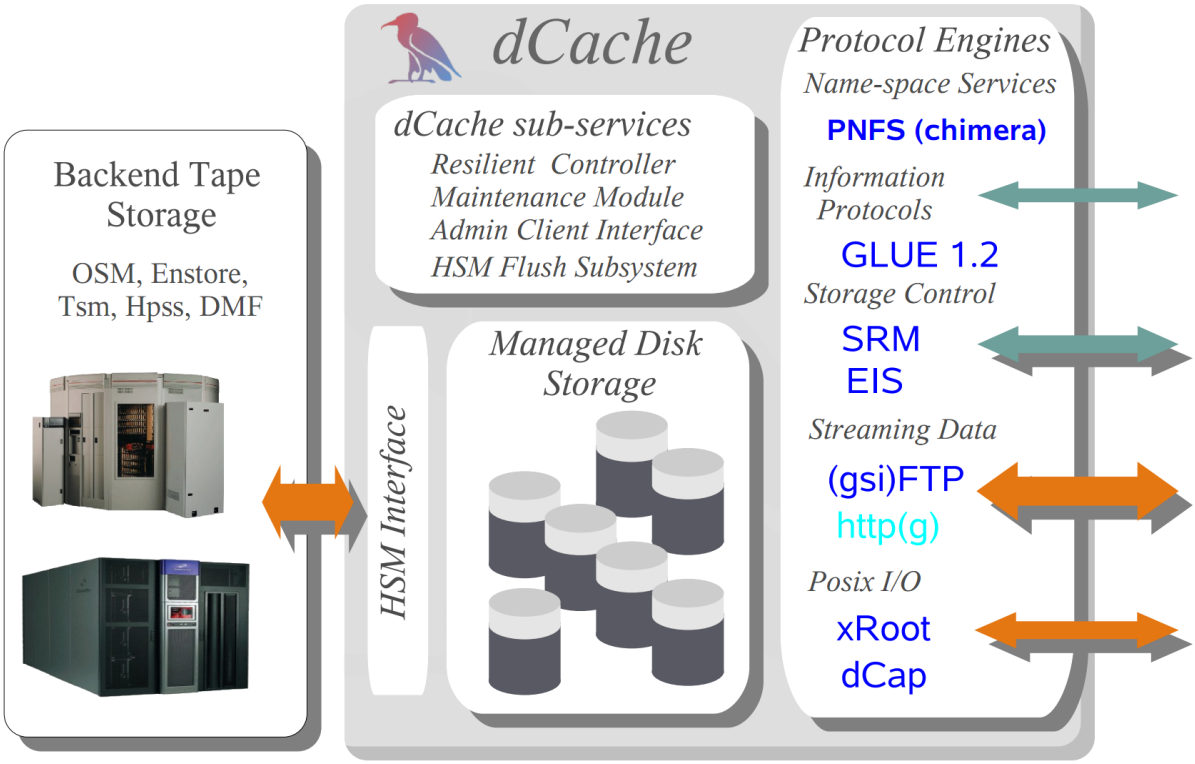


***Figure 3.1:*** dCache structure.

Amongst several protocol for data access in dCache, two are most important for the tier-2 centre GoeGrid: GridFTP and DCap.

### 3.3.1 dCap Protocol

*DCache Access Protocol (dCap)* is the native protocol to access files in dCache. It is able to emulate POSIX access via LAN or WAN and is mainly used for local file transfers. For a certificate authentication, *Grid Security Infrastructure dCap (GSIdCap)* (from the *Globus toolkit* [39] middleware) is available.

### 3.3.2 GridFTP Protocol

Extending the *File Transfer Protocol (FTP)*, GridFTP is part of the Globus toolkit. It is designed for reliable and fast transfers especially of large files and supports the parallel transfer of small parts of (large) files. For authentication and encryption, GridFTP uses the GSI. It also has a built-in automated *Transfer Control Protocol (TCP)* optimisation of e.g. TCP window and TCP buffer size. Furthermore, it is fault tolerant.

## 3.4 dCache Billing Logs and Database

There are two central resources for dCache monitoring: the dCache billing log files and a dCache billing database, both containing the same information. The billing log files are being generated once a day. They hold one entry for each file accessing action in dCache as well as the billing database. An entry contains the information as shown in table 3.2.

The billing database allows faster access of random entries. It is realized as a *PostgreSQL* [46] database and more efficient and sophisticated based on advanced database queries. Furthermore, iterators on queries can be used to access all query results. Another advantage of databases are a large set of built-in functions, enabling to perform e.g. arithmetic operations in database queries, expressed in a slight modification of *Structured Query Language (SQL)*.

## 3.5 dCache Statistics for GoeGrid

The result of a preceding analysis during the *Spezialisierungspraktikum* were statistical information on GoeGrid. The overall transfer size since the first transfers in dCache (2009-07-09) is about 3 PByte. Over the time period from March 2010 until March 2011 the transfer rate results to be about 50 MByte/s. dCache also had only few downtimes which reveals a high availability (about 91% from March 2010 until March 2011).

| **Entry name** | **Example** |
|---|---|
| time and date of access | 07.23 00:00:02 |
| pool name | pool:se6-goegrid_6@se6-goegridDomain |
| pnfs ID | 000077700FBCFB5A4C13B969BE77751F6A6F |
| file size | 221435 |
| storage class | atlas:STATIC@osm |
| transferred bytes | 221435 |
| duration of transfer in ms | 119 |
| read/write flag | false |
| transfer protocol | GFtp-2.0 |
| transaction ID | pool:se9-goegrid_6@se9-goegridDomain:1247177179293-30923 |
| client full qualified name | 192.108.46.83 |
| transfer listen port | 20474 |
| return status | 0 |
| error message | [empty] |

***Table 3.2:*** Information from dCache billing log/database entry with example.

# 4 GoeGrid Monitoring and dCache Analysis

Resource centres as the tier-2 centre at the Georg-August University of Göttingen have to facilitate a large variety of services for diverse users. It is important to provide high availability, performance, and reliability of all services and sub-services. In order to ensure those qualities, as a first step all kind of monitoring is indispensable to fulfil the users' need as well as possible. Requirements of monitoring in general are the following:

- *scalability*: a monitoring system should be able to cope with any size of monitored system efficiently
- *extensibility*: a monitoring system is preferentially extensible to new resources or system components
- *data delivery*: dynamic and static
- *portability*: a monitoring system should be portable to multiple platforms
- *security*: only authorized personnel may access monitoring information

In addition, monitoring systems should display their results in real-time and continuously. In case of a critical problem, they should immediately notify the staff in charge of maintaining the monitored resource and the results of the analysis should be easily accessible.

In the following sections, dCache related analyses are accomplished. These are the basis for monitoring tools described in chapter 5. The presented analysis has not been performed and described by this means until now; the software used was developed in the course of this Bachelor's thesis. Without actively interfering with dCache, the created tools utilize the passively generated billing database.

## 4.1 dCache File Size Distributions

In high energy physics, in this case the ATLAS experiment, very particular file types are employed. Examples of those file types and their typical sizes are:

- *AOD (Analysis Object Data)* files (Monte Carlo simulated and real event data), $10^8 \ldots 10^9$ Byte
- *D3PD (Derived[3] Physics Data)* files (root tuples), $10^8 \ldots 10^9$ Byte
- (analysis) job log files, $10^3 \ldots 10^6$ Byte

When large files are transferred, the reading, switching, routing, and writing processes take more time than for smaller file sizes. This causes a higher system load and can effect the dCache performance, becoming noticeable in lower transfer rates. When too many transfers are requested at the same time, they get enqueued and are not transferred directly; if this results in long waiting times, timeouts abort the transfers completely.

In dCache, two different types of pools exist: data and cache pools. While data pools store regular files, cache pools contain copies of very often accessed files/pools. For both pool types, file size distributions have been investigated.

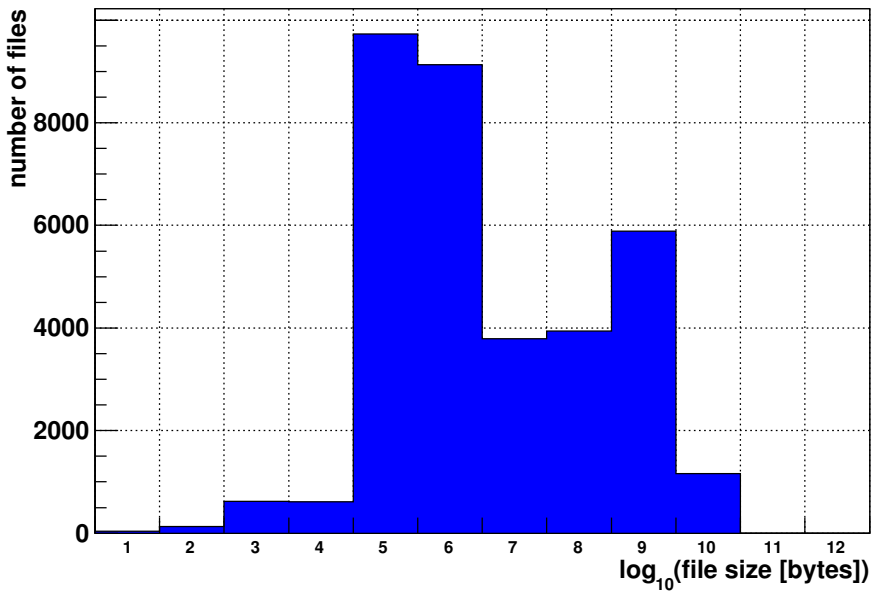


***Figure 4.1:*** File size distribution of pool *se-goegrid_2*.

In figure 4.1 the file size distribution of pool *se-goegrid_2* is shown. Overall, about 35,000 files are stored on this pool. The distribution peaks at a file size of about 100 kByte . . . 1 MByte. In order to find out whether each pool has to be reviewed individually, an overall file size distributions is produced. It is depicted in figure 4.2. The correlation of the file size distribution of pool *se-goegrid_2* with the overall file size distribution is an example for the similarity of the file size distributions on all pools. Nevertheless, in dCache two different types of pools exist: cache and data pools. While data pools store regular files, cache pools contain copies of very often accessed files. In figure 4.4, the overall file size distribution of data pools is shown. It resembles very much the overall file size distribution. In contrast, the file size distribution of cache pools has a completely different shape, peaking at file sizes between 100 MByte and 1 GByte (figure 4.3).

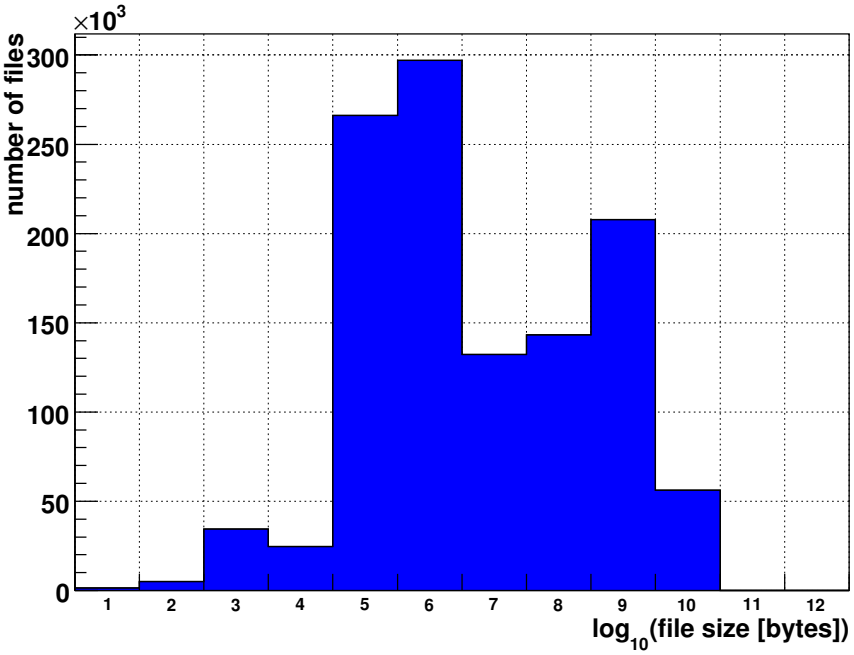


***Figure 4.2:*** File size distribution for all kinds of pools; due to the high number of files on data pools, the shape of the overall distribution resembles the data pool file size distribution. Nevertheless, the number of accesses for file sizes between 100 MByte to 1 GByte (cache pool peak) is significantly contributing to the shape of the overall distribution.

This difference is caused by the way cache pools are generated. When a pool is detected to be *hot*, meaning that it has many accesses, each file requested for transfer is replicated to cache pools. Once a cache pool reaches its size limit, the files least accessed are deleted,

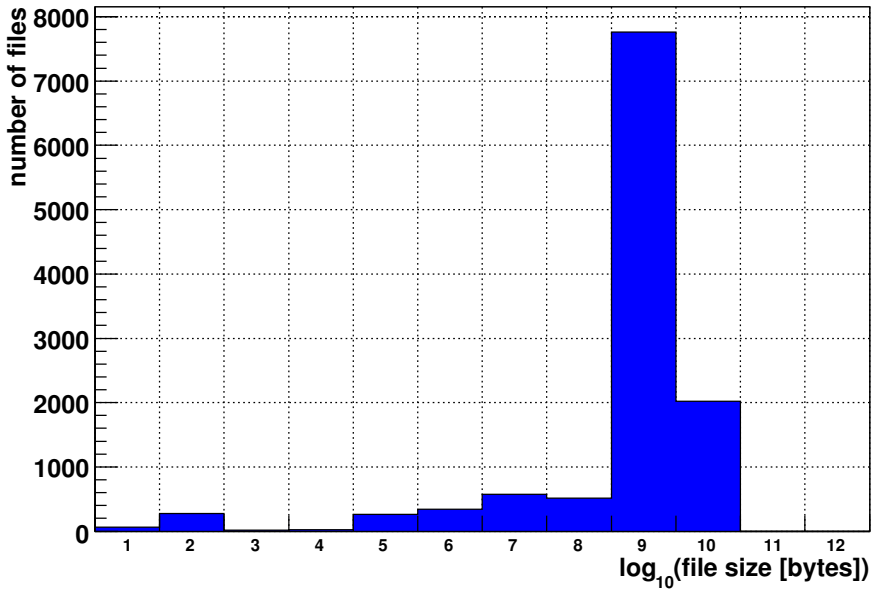


***Figure 4.3:*** File size distribution for cache pools; the maximum between 100 MByte to 1 GByte accounts for more than half of all files on cache pools.

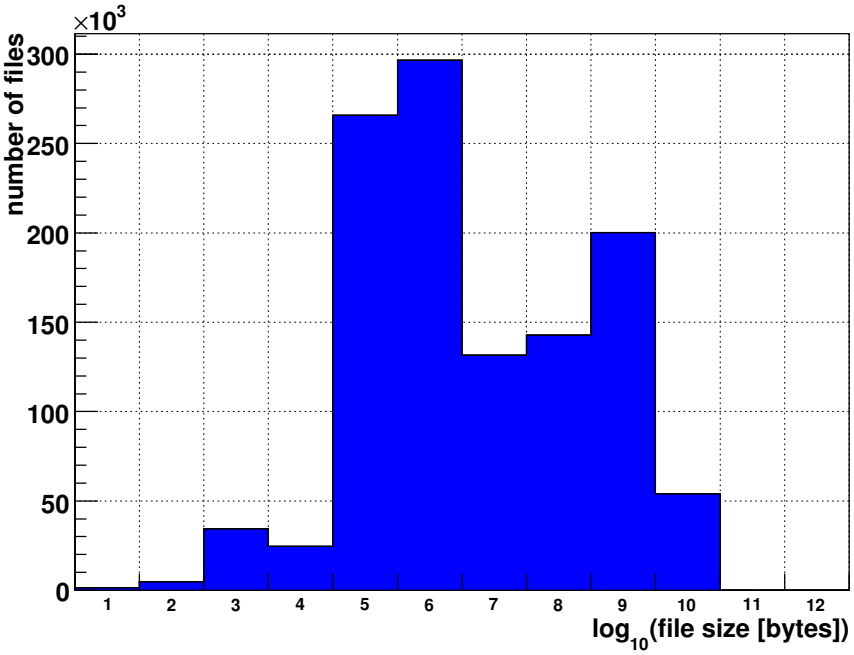


***Figure 4.4:*** File size distribution for data pools; the maximum between 100 kByte and 1 MByte is by far not as dominating as the cache pool maximum.

clearing a space for new files. That is why high access rates especially for the file size peaks in the file size distribution of cache pools are expected.

In order to verify this assumption and as a first step of the hotfile analysis, distribution histograms of file size and number of accesses versus number of files are examined. The overall histogram is shown in figure 4.5. In this histogram the shape of the previous one-dimensional histograms is recognisable. Pre-eminent is the long tail of the number of accesses for file sizes between 100 MByte and 1 GByte. It is present in both the two-dimensional distributions for data pools (figure 4.7) and for cache pools (figure 4.6). This tail is consistent with the file sizes of files typically employed by ATLAS analysis and production jobs (AOD and D3PD files). That is why it is not surprising that especially on cache pools files of this size are stored and accessed above average.

Another yield from this analysis is the observation that for all individual data respectively cache pools with reasonably high statistics the shape of the one- and two-dimensional file size distributions is very similar to the average distribution of data respectively cache pools. This also ensures that on average all data (cache) pools are accessed equally often.

## 4.2 dCache Load

The load of GoeGrid is highly varying. For example, it is dependent on ATLAS production, its downtimes, and the local user behaviour which leads to high load in particular prior to high energy physics conferences. There are also active, scheduled monitoring tests which deliberately cause a high capacity utilization. As the performance limit of dCache is only given for high load, it is inevitable to identify such time periods. In order to measure this, the number of transfers and the transfer sizes are taken into account.

At certain times, peak values of dCache usage are observed. Overall transfer sizes for the period of one year (February 2010 until February 2011) are depicted in figure 4.8, showing the usage fractions of different protocols. Most file transfers are handled via dCap protocol and are mainly local transfers within GoeGrid e.g. from storage nodes to compute nodes. Concerning overall dCache transfer sizes, the trend is rising. This is due to a higher LHC luminosity and resulting from this more data to be stored and analysed. Peaks exhibit a daily transfer size of up to 30 TByte.

In the load statistics, a significant change of the protocol usage is observable. From about July 2010, a high GridFTP usage is replaced by a high usage of dCap. This is due

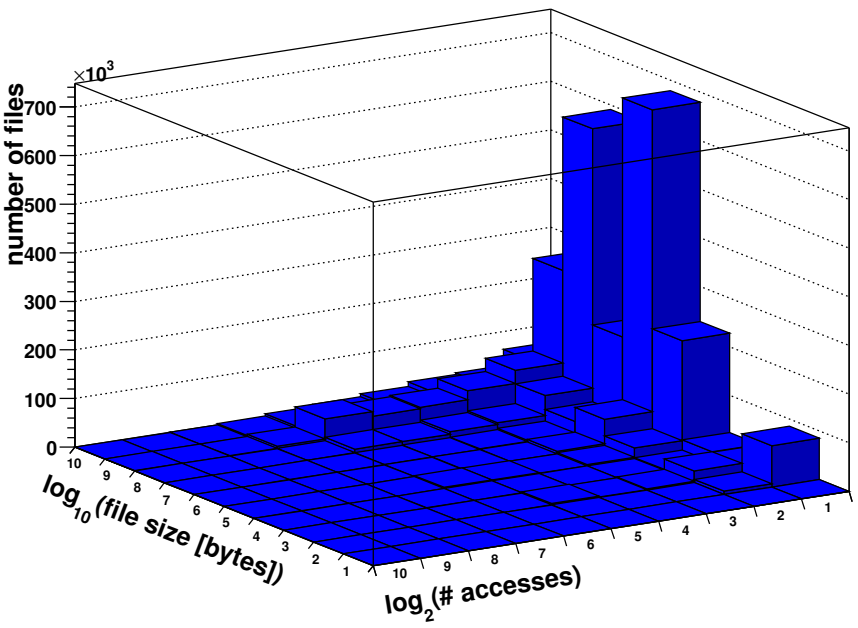


***Figure 4.5:*** Distribution of file size and number of accesses for all kinds of pools; due to the high number of files on data pools, the shape of the overall distribution resembles the data pool file size distribution. Nevertheless, the number of accesses for file sizes between 100 MByte to 1 GByte (cache pool peak) is significantly contributing to the shape of the overall distribution. Especially for file sizes between 100 MByte to 1 GByte (cache pool maximum), the number of accesses is much higher than for all other file sizes combined.

to the change of the default protocol for ATLAS analysis jobs from GridFTP to dCap for performance reasons. dCap allows to open files via network and to access only requested events instead of transferring the whole file via GridFTP.

Due to the fact that peak times are neither avertible nor concentrate on a single resource, one cannot prepare GoeGrid for those time periods. Albeit, they are very useful to analyse the whole system under high load. Especially in terms of a bottleneck analysis, those time periods are examined.

## 4.3 dCache Hotfiles

The file size distribution histograms document that most files are accessed no more than once after being written to dCache. Nevertheless, there are a few files which exhibit a high number of accesses, especially files with sizes 100 MByte to 1 GByte, most likely

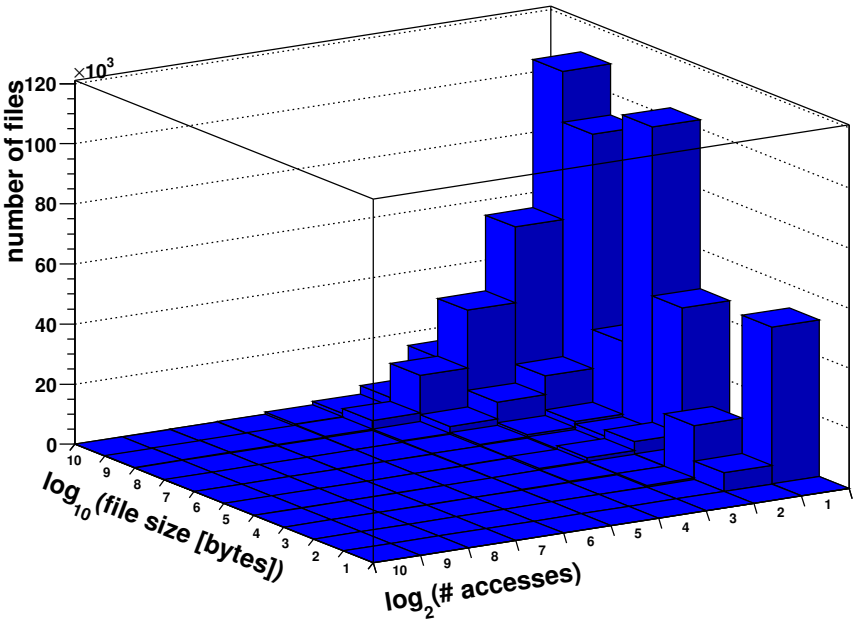


***Figure 4.6:*** Distribution of file size and number of accesses for cache pools; the maximum between 100 MByte to 1 GByte accounts for more than half of all files on cache pools.

AOD and D3PD files, on both types of pools (see section 4.1). This is able to influence the performance of the dCache system due to the high traffic, large numbers of large files transferred cause. In this section, the results of a *hotfile* analysis are discussed.

There are several ways to define a file as *hot*.

1. number of read accesses above certain threshold
2. number of read accesses and file size above certain threshold
3. product of number of read accesses and the size above certain threshold
4. one of the aforementioned criteria and taking also the network connection into consideration
5. one or more criteria above extended by taking the load of the transferring pool into account

The hotfile analysis is performed for all days since the dCache logging for GoeGrid was started at the end of 2008. Hotfile distributions for different criteria and the same

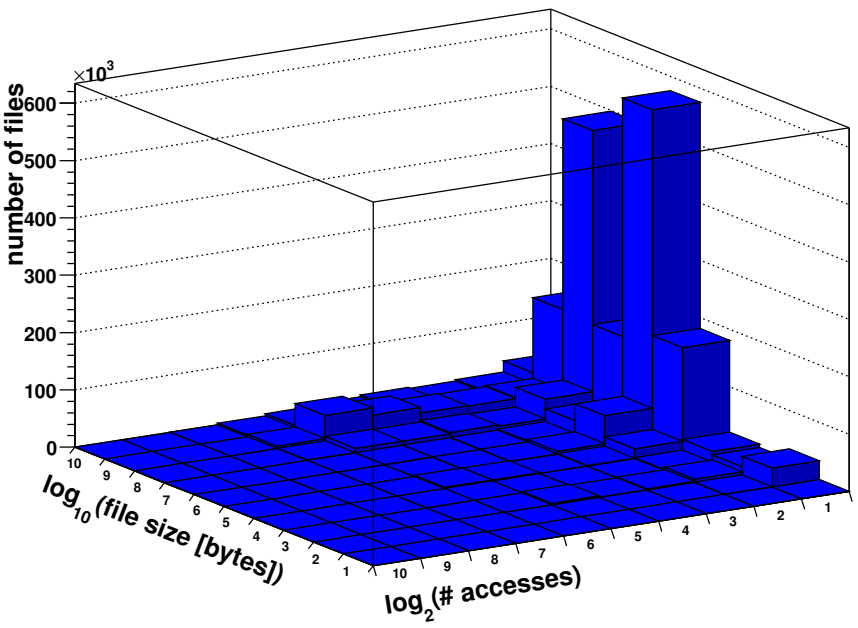


***Figure 4.7:*** Distribution of file size and number of accesses for data pools; the maximum between 100 kByte and 1 MByte is by far not as dominating as the cache pool maximum. Especially for file sizes between 100 MByte to 1 GByte (cache pool maximum), the number of accesses is much higher than for all other file sizes combined.

time interval are shown in figures 4.9 and 4.10, a hotfile distribution for a different day is depicted in figure 4.11.

For the different criteria, the number of hotfiles differs only slightly, depicted in figure 4.9 (criterion 1) and figure 4.10 (criterion 3). In both histograms, the file most accessed is the same. Its resolved file name is *DBRelease-14.2.1.tar.gz* which refers to recent calibration data. It is a hotfile because it is needed by a large number of ATLAS jobs. With a size of about 500 MByte, it contributes to the long tail of accesses in the file size and number of accesses versus number of files distribution.

As to be seen (not representatively) in figure 4.11, the number of hotfiles per day and their number of accesses is highly fluctuative. Days with high dCache load exhibit a larger number of hotfiles.

For the hotfile recognition, the dCache billing database is analysed. The information on the number of accesses per file is not directly contained by the database; for each pnfs

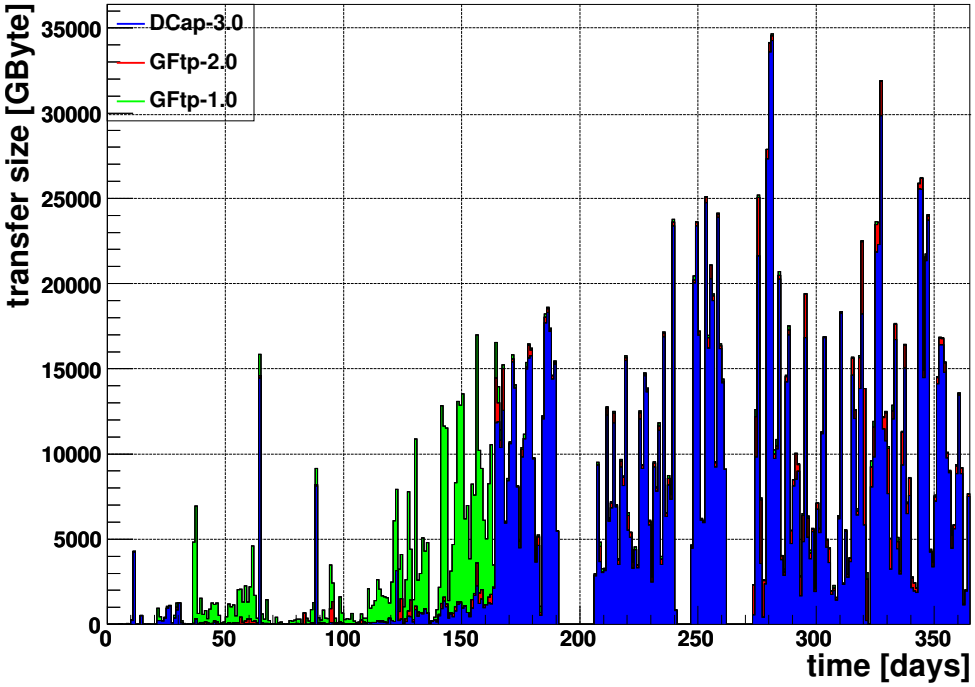


***Figure 4.8:*** Transfer sizes for February 2010 until February 2011.

ID the number of accesses (entries in the billing database) has to be counted. An efficient analysis cannot be done by storing all pnfs IDs read out in one list; for typically about $n = 10^5$ entries per day, this takes too long to be processed. That is why the pnfs ID and the corresponding number of accesses is stored in a binary tree structure; adding, accessing and deleting an entry into a binary tree needs a runtime of $\mathcal{O}(\log(n))$ in comparison to $\mathcal{O}(n)$ for lists.

Files are *hot* over very different periods of time. In figure 4.12 the number of accesses for a specific hotfile is visualized time-resolved. Typically, a hotfile is *hot* for several days, in most cases no longer than two weeks.

Due to the fact that hotfiles are excessively accessed for only short periods of time, the performance gain of a hotfile reproduction is not given by all means. Under the assumption that a hotfile is accessed 5000 times a day and it takes 10 s to transfer (reading or writing) this file, the probability a hotfile is transferred twice at the same time is approximately $\left(1 - (1 - \frac{10\,\mathrm{s}}{24 \cdot 60 \cdot 60\,\mathrm{s}})^{5000}\right)^2 \approx 19.3\%$ when two transfers are statistically independent and all transfers of this file are equally distributed over one day. In this calculation, a hotfile reproduction generates most likely a performance loss since the reproduction itself needs one read and one write access. The tacit assumption that one pool may be requested to

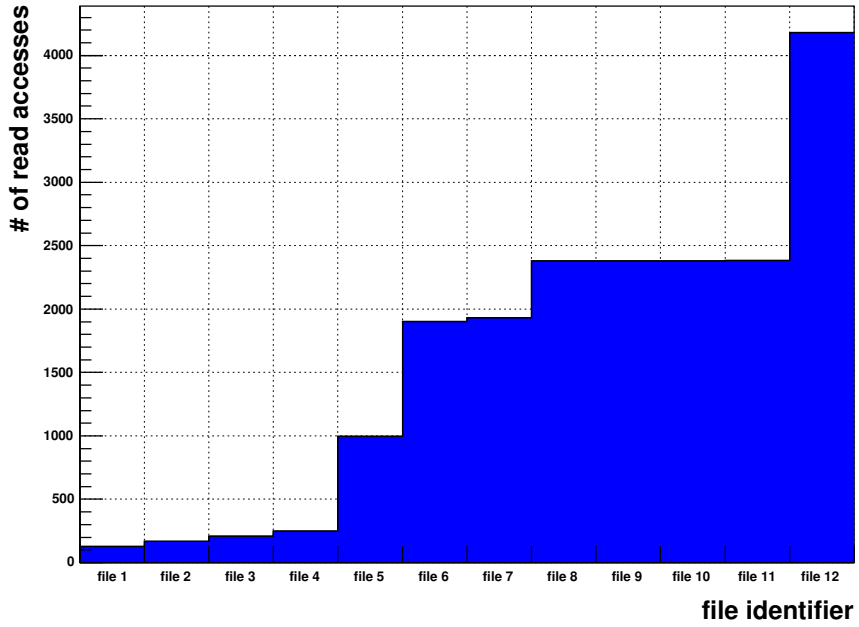


***Figure 4.9:*** Hotfiles for 2011-02-24 (criterion: number of accesses $\geq$ 100); each bin represents one file (pnfs ID).

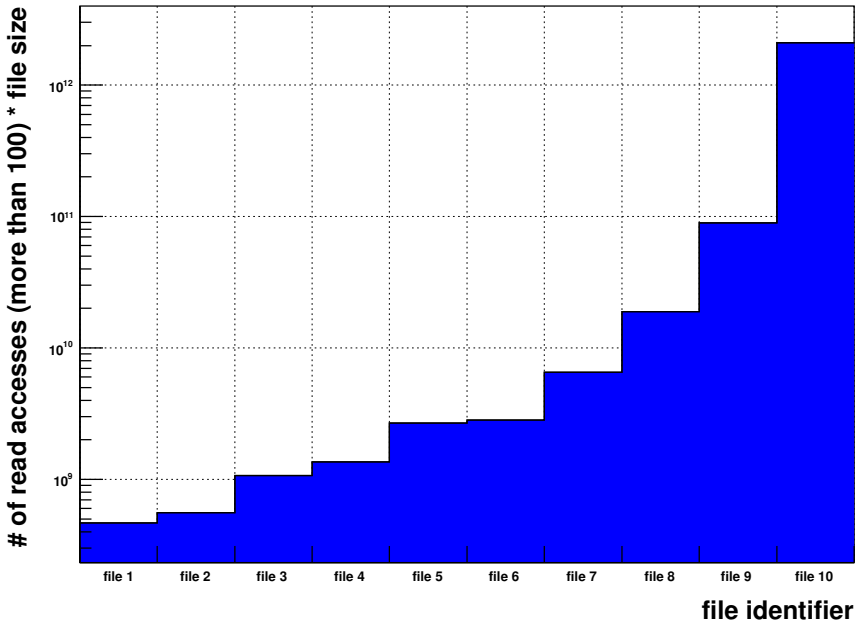


***Figure 4.10:*** Hotfiles for 2011-02-24 (criteria: file size $\geq$ 1 MByte and number of accesses $\geq$ 100); each bin represents one file (pnfs ID).

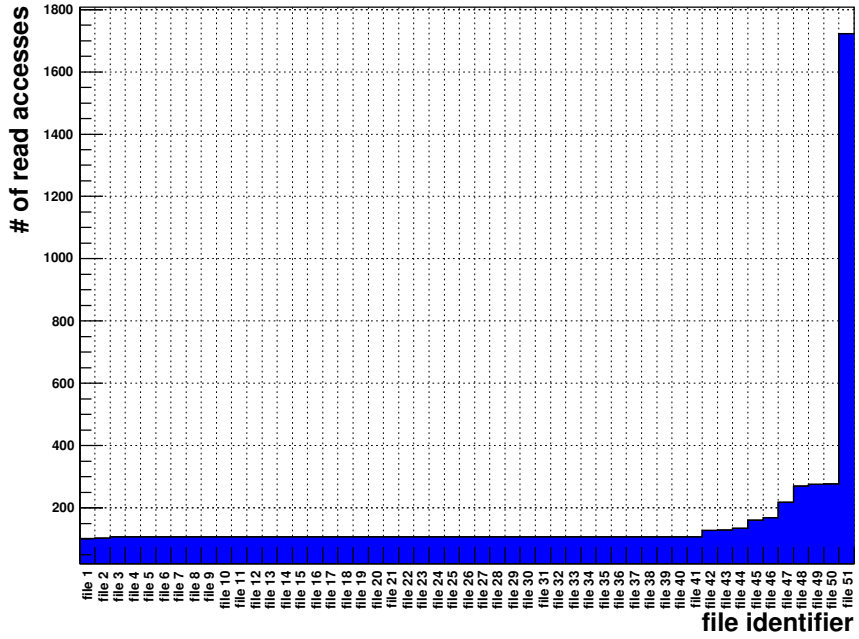


***Figure 4.11:*** Hotfiles for 2011-02-22 (criterion: number of accesses $\geq$ 100); each bin represents one file (pnfs ID).

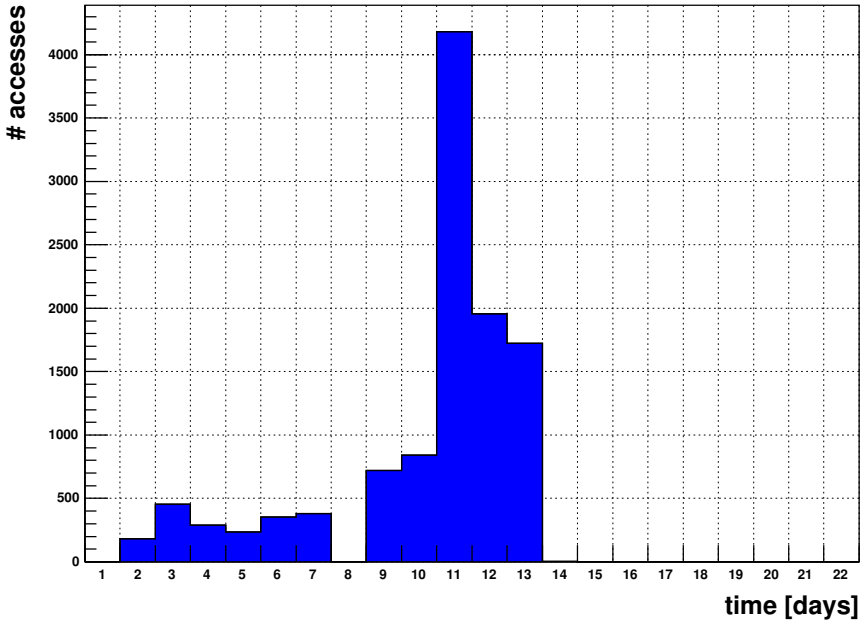


***Figure 4.12:*** Hotfile accesses for pnfs ID *00003586A306686441FAAAE5FC10B680F3F5* (file size about 500 MByte).

transfer other files during the considered hotfile transfers is justified by the fact that the average transfer rate of the whole dCache system is about 50 MByte/s. Nevertheless, in a real situation, the transfers of a hotfile are much more concentrated. The probability of one hotfile transferred twice at the same time calculates to about 80% if the hotfile accesses take place concentrated over six hours. In this case, there is most likely a performance gain. Furthermore, it prevents dCache deficiencies in case a pool, containing a file needed by almost any job processed on GoeGrid, fails.

## 4.4 dCache Bottlenecks

If one specific part or resource of a system limits its overall performance, this part is called *bottleneck*; by increasing only the performance or capacity of this single part or resource, the whole system's performance is augmented.

In figure 2.1, the GoeGrid network topology is shown. The average bandwidth is smallest for enclosures 6 to 8.

Possible bottlenecks of dCache for GoeGrid in general can be of different types:

- storage nodes: RAID write/read speed
- dCache services
- network bandwidth between groups of hosts: network connection to central switches
  - enclosures 6, 7, and 8 (each containing $2 \cdot 16$ storage nodes; groups of 16 nodes are connected via a 1 GBit ethernet connection)
  - se, se2 ... se14 (each containing several storage nodes)

For the bottleneck analysis, transfers from and to computes nodes in enclosures 6, 7, and 8 are investigated. One's attention should be turned to peaks in the traffic concerning especially the aforementioned enclosures. That is how possible bottlenecks in the network bandwidth are identified.

The result of the bottleneck analysis is depicted in figures 4.13, 4.14, and 4.15. The time period (one day with a resolution of two minutes) analysed was selected using the results of the dCache load analysis, choosing the time of highest dCache load. In each of the figures, the network bandwidth is shown as a horizontal line. Other services than dCache use this connection as well; nevertheless, dCache accounts for almost all traffic generated.

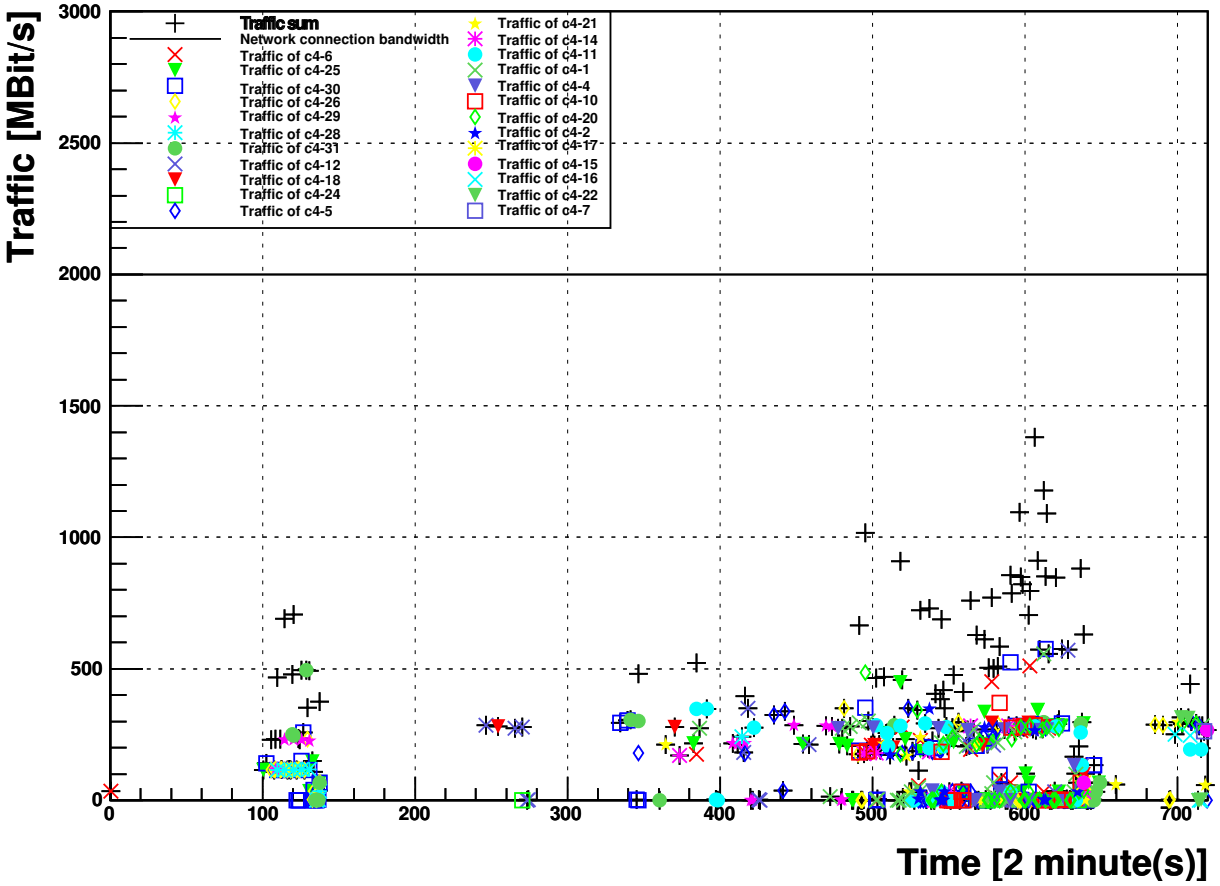


***Figure 4.13:*** Bottleneck recognition histogram for enclosure 6 (2011-02-09).

It is clearly visible that the transfer sum (black cross) for the considered groups cuts across the connection bandwidth only a few times making up a very low fraction below one percent. If the network bandwidth of those enclosure were the bottleneck, one would expect a saturation of the traffic just below the capacity. The reason that the traffic can exceed the limit given by the network bandwidth is the way the bars of the histograms are filled: Each bar contains all transfers started in the time interval it represents. It is not taken into consideration the transfer time period might be shorter than the chosen bin width.

From this it can be concluded that there is no bottleneck due to the network bandwidth. It also excludes that the storage nodes themselves account for the bottleneck, assuming a write/read speed of more than 100 MByte/s for each RAID system.

As there is no bottleneck observed from the network setup, the limitations remain to be found within dCache itself. The speed of dCache transfers is caused by the dCache

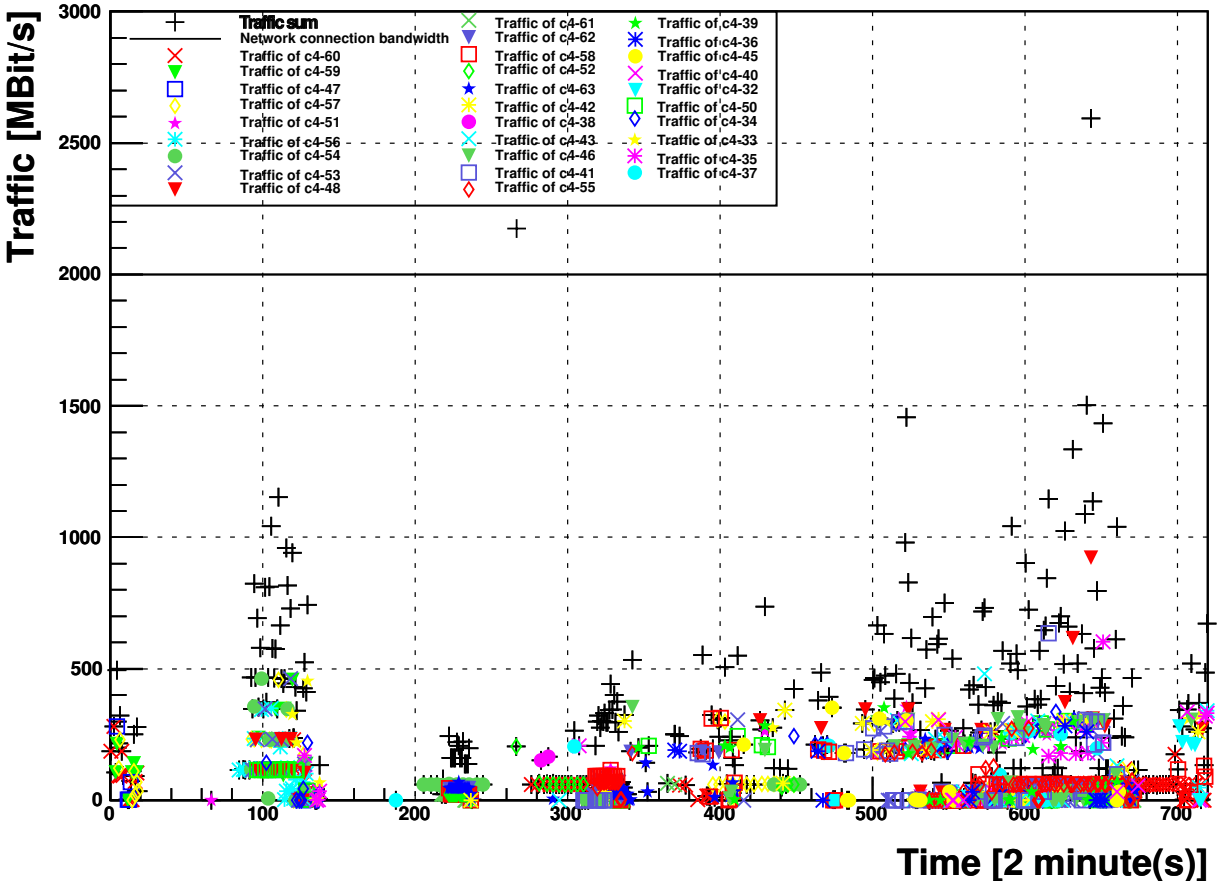


***Figure 4.14:*** Bottleneck recognition histogram for enclosure 7 (2011-02-09).

services; services to be considered further are:

- pnfs server
- pool manager
- I/0-doors

The pnfs server manages all accesses and resolves the file name according to the pnfs ID (and reverse). Due to the fact that in most cases there are not many but large files transferred in the dCache system for GoeGrid, the pnfs manager has to handle no more than typically one lookup per second. Therefore, the pnfs server is probably not the limiting component. The same considerations account for the pool manager in charge of finding the appropriate pool for each file requested: each action of the pool manager is instructed by an action of the pnfs manager.

The whole traffic is routed by the dCache I/O-doors. In GoeGrid, several doors per protocol manage all transfers. According to the load statistics, the three dCap doors for

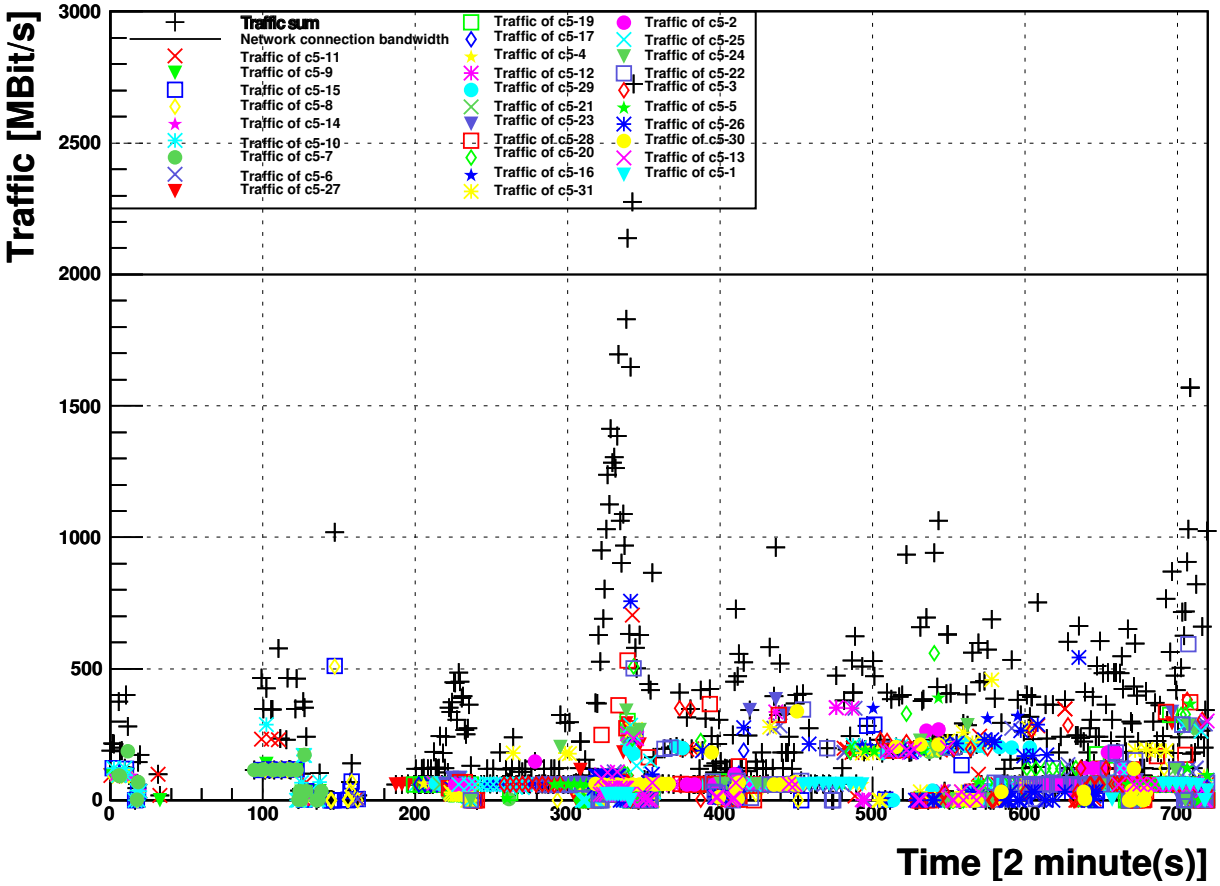


***Figure 4.15:*** Bottleneck recognition histogram for enclosure 8 (2011-02-09).

GoeGrid are used most frequently. Despite the fact there are three dCap doors, the load for those doors is unbalanced: When using dCap directly, the selection of the door is the client's choice. A high load for one of the doors can result in a temporary limit for the system's transfer speed, a bottleneck. Furthermore, it has been observed that the transfer speed of dCache transfer may be limited by the protocol itself, leading to a certain saturation of transfer speeds.

Since all other hardware and software components have been excluded to be the bottleneck in most cases, the only component remaining to be the bottleneck are the dCap I/O-doors.

It is a very inexpensive way in terms of effort and costs to establish additional dCap doors on dedicated servers, linked to the central network infrastructure with high bandwidth connections. Easily, the efficiency of this approach can be tested by the consideration of this bottleneck analysis. In addition to that, newer versions of dCap and dCache

can also contribute to a performance gain.

# 5 HappyFace Meta-Monitoring

A tier-2 centre like GoeGrid utilizes many monitoring systems at the same time to supervise all its software and hardware running; external, ATLAS- and WLCG-wide monitoring resources are also processed. That is why virtually an information flood is created, displayed and stored at different places, accessible via different interfaces and showing the monitoring results in different ways. For this reason, *meta-monitoring* is necessary to focus the data flow.

## 5.1 Meta-Monitoring Requirements

As described in [47], a meta-monitoring system fulfils the following requirements:

- *flexible and generic*, being adaptable to all sites
- *single point of access*, showing all relevant monitoring information e.g. on one website
- *up-to-date monitoring information*, in the best case real-time
- *history functionality*, giving its users the possibility to review previous monitoring results
- *fast accessibility*, letting the users access the monitoring information quickly
- *comfortable*, giving its users easy access to the monitoring outputs
- *simple warning system*, notifying immediately the responsible persons and displaying the status information unmistakeably and simple
- *customizable and extendible*, matching perfectly the site's monitoring requirements

## 5.2 HappyFace

*HappyFace* meta-monitoring system matches the above quoted requirements. It is a modular monitoring system, combining and correlating all monitoring system information and displaying the gathered information on a single website. All detailed data is not discarded but still stored either by HappyFace in a database or by the systems HappyFace gathers its information from (monitoring resources), enabling the user to browse a history. HappyFace basically consists of the HappyFace core *HappyCore* and specialized *test modules*.

The HappyCore is responsible for the execution of all modules. Furthermore, it also manages the database accesses and generates the HappyFace web page. This web page mainly consists of the specialized test modules' outputs. They represent the monitoring of a resource which may be an output generated by the module itself or an external resource as e.g. another monitoring system (log file, database, web page etc.). When periodically called, the modules' course of actions is as described in the following. First of all, an initialization takes place. The module then collects the information, processes it, stores the results in a database and carries out other database accesses. It now assigns a rating value, evaluating the monitoring output in terms of criticality. This value can be weighted by user-defined functions in HappyCore. In the end, the test module generates a dynamic web page section for the HappyFace website output. For each module, two configuration files exist (default and local settings) [47].

The HappyFace web page consists of categories which are defined by each resource centre individually. In these categories, the output of the corresponding test modules is displayed. At the tier-2 centre at the Georg-August University of Göttingen the categories are *SAM tests*, *Nagios*, *Hardware*, *Production*, *dCache*, *Cloud status* and *Monitoring links*, partly giving a rating output. In case such an output is generated, it is represented e.g. by a smiley. Looking happy, even-tampered or sad, the smiley indicates the test modules results' status.

Running under Linux systems, HappyFace requires the execution of *cron jobs*, Python v2.5.2 or higher, SQLite2, Python SQLObject package and PHP v5. Several resource centres avail themselves of HappyFace: University of Hamburg, DESY Hamburg, RWTH Aachen, Karlsruhe Institute of Technology (KIT), and Georg-August University of Göttingen. Each of these sites develops its own modules, sharing them with other sites via an *SVN* repository. At GoeGrid, HappyFace monitors amongst others local hardware, local

**dCache Hotfile Detection**

Tue, 28. Jun 2011, 11:31 - Show module information

Show current hotfiles (9)

| File | Pool | Access Count | Filesize |
|---|---|---|---|
| 0000D02E52083FD8488BAA80B84BA3823EC7 | se10-goegrid_13 | 1045 | 331.969 MB |
| 0000F5F91BBBE0514984B9BD8B1AF715F415 | se8-goegrid_14 | 464 | 407.301 MB |
| 00003586A306686441FAAAE5FC10B680F3F5 | se7-goegrid_9 | 381 | 479.699 MB |
| 000028498532309748719653155573 6BA3AF | se8-goegrid_10 | 362 | 0.015 MB |
| 0000368AE9F90C8443ABA4F45865EDA65C58 | se10-goegrid_10 | 362 | 1.076 MB |
| 00001725D17567A64C7F87232FAECE5C81A4 | se4-goegrid_17 | 361 | 0.544 MB |
| 00004BA5AE45531149E0B758A21D72417782 | se-goegrid_3 | 361 | 0.029 MB |
| 0000DE2F38BB55824A7091145A9BCC75A2A0 | se-goegrid_1 | 307 | 8.562 MB |
| 000028090771ECF749DAB16B654D9D659684 | se-goegrid_1 | 225 | 479.452 MB |

***Figure 5.1:*** Hotfile module for HappyFace.

(infrastructure) services and ATLAS job submission and execution [48].

## 5.3 Hotfile and Bottleneck Recognition Modules

Due to the fact that HappyFace is used for GoeGrid monitoring, all monitoring software is preferentially implemented for HappyFace. Gathering its information from tools developed and made use of in this thesis, the modules on the recognition of hotfiles and bottlenecks are updated in intervals of 15 minutes. Neither the results of this analysis, nor the modules developed were available before. Without many customisation necessary, they can be employed by other resource centres.

### 5.3.1 Hotfile Module

As hotfiles might influence the performance of GoeGrid, a HappyFace module to detect recent hotfiles has been developed [49], depicted in figure 5.1. It outputs each hotfile's pnfs ID and resolves its storage pool displayed in a *mouseover effect*. In two further columns of the table generated, the number of accesses and the file size is given. Finally, the results are ordered descending by the number of accesses for each file. The module's output can be used for a manual hotfile reproduction in a testing phase on the effectiveness of hotfile reproduction in GoeGrid.

### 5.3.2 Bottleneck Module

Another module is responsible to display the results of the bottleneck analysis. It shows the figures discussed in 4.4 and assigns a rating value according to the occurrence of bottlenecks in GoeGrid. A fine grained, tunable time resolution allows the recognition of bottlenecks and limitations on variable time scales which is not possible with existent network monitoring in GoeGrid.

# 6 Conclusion and Outlook

## 6.1 Conclusion

In a complex system like GoeGrid, monitoring is inalienable to observe problems in time and to avoid performance losses or even downtimes. This Bachelor's thesis yields not only observations on dCache behaviour but also produces monitoring software that is used hereafter to supervise dCache at GoeGrid. Furthermore, it provides solution statements for possible performance losses due to hotfiles and bottlenecks.

dCache file size distributions have shown to be very specific for the user group of GoeGrid which is mainly the ATLAS collaboration storing simulated events and real event data. On the one hand, there are high access rates for file sizes of 100 MByte − 1 GByte (typically AOD, D3PD files). High traffic is mainly caused by those large files; they are mostly the hotfiles. On the other hand, there are many *cold* files accessed rarely. Less *cold* files are on cache pools, where more than half of all files have a size between 100 MByte and 1 GByte and are accessed relatively often. Nevertheless, there are no *cold* pools.

Load statistics for dCache are important for overall performance checks; its problems will appear in these histograms. Therewith, the usage fractions for different dCache protocols are monitored. It is also possible to detect production downtimes as well as dCache and GoeGrid downtimes.

There are many different criteria to identify hotfiles, all of them more or less producing the same results. Hotfiles are confirmed to be of file sizes between 100 MByte and 1 GByte as already presumed in 4.1. The number of hotfiles per day varies strongly. In order to increase dCache performance and load reliability, a hotfile reproduction can be considered.

Bottlenecks of the transfer speed in dCache are located only exceptionally in the network bandwidth, several enclosures are attached with to a central infrastructure. In most cases, the dCache services are the limit. It is very probable that the I/O-door for dCap

constitutes the transfer speed limitation, as this is the protocol most used (see section 4.2) and the three dCap doors existing in the dCache system are not employed equably. Finally, one must acknowledge that none of the bottlenecks is striking. Much more limiting are downtimes that have to be minimized under all circumstances.

HappyFace has been chosen for the implementation of a hotfile and a bottleneck module not only because of its usage in GoeGrid, but also because it fulfils monitoring requirements perfectly. With those modules, problems can be recognized promptly and reacted on quickly.

## 6.2 Outlook

The work described and documented in this Bachelor's thesis has drawn a wide range of dCache related monitoring and statistics. Nevertheless, there is a need for further research related to dCache.

Concerning the results of the hotfile analysis, the *dCache Migration Module* can be used to automatically reproduce files, triggered by the output of the developed HappyFace module for hotfiles. This requires on all accounts an extensive manual testing phase and a verification of a performance gain. Extending the storage capacity would allow to reproduce hotfiles continuously. The reproduction then can take place on cache pools which are specifically chosen by their high connection bandwidth and the performance of the utilized RAID system.

In addition to that, *cold* files can be identified and displayed in a HappyFace module. Via DDM, the local deletion of those files can be requested. Another approach is to move such files on pools with older hardware.

For the prevention of bottlenecks, one can set up the dCap I/O-door most used on a dedicated server which is not employed as a storage node simultaneously. By all means, enhancements for dCache and especially dCap will be made by updates and a steady advancement process.

Finally, more generic tools and monitoring software for HappyFace and Nagios can be created. This will ensure the results of this Bachelor's thesis are made available to other resource centres and generic network topologies in order to increase their performance.

# Bibliography


[1] *European Organization for Nuclear Research (CERN)*, URL `http://public.web.cern.ch/public/`

[2] L. Evans, *The large hadron collider project*, CERN Reports pages 275–286 (1997)

[3] ALICE Collaboration, *ALICE Technical Design Report*, CERN/LHCC 2001-021 (2001)

[4] ATLAS Collaboration, *ATLAS: technical proposanal for a general-purpose pp experiment at the large hadron collider at CERN*, CERN/LHCC pages 171–173 (1994)

[5] CMS Collaboration, *CMS - Technical Proposal*, CERN/LHCC/94-38 (1994)

[6] LHCb Collaboration, *LHCb Technical Proposal*, CERN/LHCC 98-4 (1998)

[7] C. Lefevre, *LHC: the guide (english version)* (2008), `http://cdsweb.cern.ch/record/1092437/files/CERN-Brochure-2008-001-Eng.pdf`

[8] *ATLAS fact sheet*, URL `http://atlas.ch/pdf/atlas_factsheet_all.pdf`

[9] G. Aad, M. Ackers, F. Alberti, M. Aleppo, G. Alimonti, J. Alonso, E. Anderssen, A. Andreani, A. Andreazza, J. Arguin, et al., *ATLAS pixel detector electronics and sensors*, Journal of Instrumentation **3**, P07007 (2008)

[10] A. Ahmad, Z. Albrechtskirchinger, P. Allport, et al., *The silicon microstrip sensors of the ATLAS semiconductor tracker*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **578(1)**, 98 (2007)

[11] T. Åkesson, F. Anghinolfi, E. Arik, O. Baker, S. Baron, D. Benjamin, H. Bertelsen, V. Bondarenko, V. Bytchkov, J. Callahan, et al., *Status of design and construction of the Transition Radiation Tracker (TRT) for the ATLAS experiment at the LHC*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **522(1-2)**, 131 (2004)

[12] N. Buchanan, L. Chen, D. Gingrich, S. Liu, H. Chen, J. Farrell, J. Kierstead, F. Lanni, D. Lissauer, H. Ma, et al., *Design and implementation of the Front End Board for the readout of the ATLAS liquid argon calorimeters*, Journal of Instrumentation **3**, P03004 (2008)

[13] The Tile Calorimeter Group of the ATLAS Collaboration, *The Production and Qualification of Scintillator Tiles for the ATLAS Hadronic Calorimeter*, Technical Report ATL-TILECAL-PUB-2007-010. ATL-COM-TILECAL-2007-026, CERN (2007)

[14] K. Nagai, *Thin gap chambers in ATLAS*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **384(1)**, 219 (1996)

[15] A. D. Simone, *The calibration of the resistive plate chambers of ATLAS*, Journal of Physics: Conference Series **219(3)**, 032036 (2010), URL `http://stacks.iop.org/1742-6596/219/i=3/a=032036`

[16] M. Livan, *Monitored drift tubes in ATLAS*, Nuclear Instruments and Methods in Physics Research-Section A Only **384(1)**, 214 (1997)

[17] P. O'Connor, V. Gratchev, A. Kandasamy, V. Polychronakos, V. Tcherniatine, J. Parsons, W. Sippach, *Readout electronics for a High-Rate CSC detector*, in *Fifth Workshop on Electronics for LHC Experiments, Snowmass Colorado* (1999)

[18] M. Delfino, L. Robertson, *Solving the LHC Computing Challenge: A Leading Application of High Throughput Computing Fabrics combined with Computational Grids*, CERN-IT-DLO-2001-003 (2001)

[19] J. Meyer, A. Quadt, P. Weber, *ATLAS Tier-2 at the Compute Resource Center GoeGrid in Göttingen*, (accepted for) Journal of Physics: Conference Series (2011)

[20] *GWDG*, URL `http://www.gwdg.de/`

[21] *II. Institute of Physics, Georg-August University Göttingen*, URL `\http://physik2.uni-goettingen.de/`

[22] S. Jarp, A. Lazzaro, J. Leduc, A. Nowak, C. G. I. Department, *Evaluation of the Intel Nehalem-EX server processor*, CERN Openlab (2010)

[23] *D-Grid*, URL `http://www.d-grid.de/`

[24] *MediGRID*, URL `http://www.medigrid.de/`

[25] M. W. Küster, C. Ludwig, H. Neuroth, A. Aschenbrenner, *e-Humanities for Digital Eco-systems: A Social, Cultural, Economic and Political Agenda*, IEEE DEST, Cairns, Australien, SPECIAL SESSION 3 (2007)

[26] M. Ernst, P. Fuhrmann, M. Gasthuber, T. Mkrtchyan, C. Waldman, *dCache, a distributed storage data caching system*, CHEP2001 Conference Notes pages 152–156 (2001)

[27] *Availability and Reliability Monthly Statistics*, URL `https://wiki.egi.eu/wiki/Availability_and_reliability_monthly_statistics`

[28] E. Laure, F. Hemmer, F. Prelz, S. Beco, S. Fisher, M. Livny, L. Guy, M. Barroso, P. Buncic, P. Kunszt, et al., *Middleware for the next generation Grid infrastructure*, Computing in High Energy Physics and Nuclear Physics (CHEP 2004) (2004)

[29] M. Göhner, C. Rückemann, *Accounting-Ansätze im Bereich des Grid-Computing*, D-Grid, Fachgebiete Monitoring, Accounting und Billing im D-Grid-Integrationsprojekt (2006)

[30] *Berkeley Database Information Index (BDII)*, URL `https://twiki.cern.ch/twiki/bin/view/EGEE/BDII`

[31] C. Aiftimiei, P. Andreetto, S. Bertocco, S. Fina, S. Ronco, A. Dorigo, A. Gianelle, M. Marzolla, M. Mazzucato, M. Sgaravatto, *Job submission and management through web services: the experience with the CREAM service*, Journal of Physics: Conference Series **119**, 062004 (2008)

[32] *Nagios Open Source Monitoring*, URL `http://www.nagios.org/`

[33] *Torque Batch System*, URL `http://www.clusterresources.com/products/torque-resource-manager.php`

[34] *Maui Batch Scheduler*, URL `http://www.clusterresources.com/products/maui-cluster-scheduler.php`

[35] *Deutsches Elektronen-Synchrotron (DESY)*, URL `http://www.desy.de/`

[36] *Fermi National Accelerator Laboratory (Fermilab)*, URL `http://www.fnal.gov/`

[37] P. Fuhrmann, *A Perfectly Normal Namespace for the DESY Open Storage Manager*, in *Conference on Computing in High Energy Physics, Berlin* (1997)

[38] W. Allcock et al., *GridFTP Protocol Specification*, Global Grid Forum Recommendation GFD.20 (2003)

[39] I. Foster, C. Kesselman, *Globus: A Metacomputing Infrastructure Toolkit*, International Journal of Supercomputer Applications **11**, 115 (1997)

[40] A. Shoshani, A. Sim, J. Gu, *Storage resource managers: Middleware components for grid storage*, in *NASA Conference Publication*, pages 209–224 (2002)

[41] J. E. James Whitehead, Y. Y. Goland, *WebDAV - A network protocol for remote collaborative authoring on the Web*, European Computer Supported Cooperative Work (ECSCW'99) conference (1999)

[42] A. Dorigo, P. Elmer, F. Furano, A. Hanushewsky, *XROOTD - A highly scalable architecture for data access*, WSEAS (2005)

[43] M. de Riese, P. Fuhrmann, T. Mkrtchyan, M. Ernst, A. Kulyavtsev, V. Podstavkov, M. Radicke, N. Sharma, D. L. andTimur Perelmutov, T. Hesselroth, G. Behrmann, T. Zangerl, P. Millar, O. Syngea, A. Petersen, *The dCache Book for 1.9.12-series* (2011), URL `http://www.dcache.org/manuals/Book-1.9.12/Book-a4.pdf`

[44] Abhishek Rana et al., *Introducing Advanced Fine-grained Security in dCache-SRM for PetaByte-scale Storage Systems on Global Data Grids: gPLAZMA 'grid-aware PLuggable AuthoriZation MAnagement System'*, IEEE Nuclear Science Symposium (2006)

[45] *GNU database management*, URL `http://www.gnu.org/software/gdbm/`

[46] *PostgreSQL database*, URL `http://www.postgresql.org/`

[47] V. Büge, V. Mauch, G. Quast, A. S. an dArtem Trunov, *Site specific monitoring of multiple information systems - the HappyFace Project*, Journal of Physics **Conference Series 219, 062057** (2010)

[48] *GoeGrid HappyFace Monitoring Web Page*, URL `http://happyface-goegrid.gwdg.de/newhappy/`

[49] G. Jahn, *Meta-Monitoring of the Grid Resource Centre GoeGrid with HappyFace, Bachelor's Thesis*, Georg-August University Göttingen (2011)

# Acknowledgements

Firstly, I like to take this opportunity to thank Prof. Dr. Arnulf Quadt who gave me the opportunity to write this thesis and offered me a warm welcome in his research group. I would also like to show my gratitude to Prof. Dr. Ariane Frey for the time and effort she puts into this thesis as a referee.

I am indebted to Dr. Jörg Meyer and Dr. Pavel Weber for their support and advice indispensable for the successful completion of my Bachelor's Thesis. They have made available their support in a number of ways and are responsible for the very yielding topic of dCache analysis.

My sincere thanks go to my dear friend and colleague Georg Jahn. It was a great pleasure to work and discuss with him. Finally, I want to express my profound relationship, friendship, and exaggerated gratitude to him, Miriam Reuter, Lucas Lang, Christina Heinicke, Friedrich Bös, and Thilo Müller von der Grün. I lost the game.

# Eigenständigkeitserklärung - Statement of Authorship

**Erklärung** nach §13(8) der Prüfungsordnung für den Bachelor-Studiengang Physik und den Master-Studiengang Physik an der Georg-August Universität Göttingen:

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe.

Darüberhinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, im Rahmen einer nichtbestandenen Prüfung an dieser oder einer anderen Hochschule eingereicht wurde.

**Declaration** according to §13(8) of the Examination Regulations for the Bachelor's degree and the Master's degree of the Georg-August University of Göttingen:

I declare that this document and the accompanying code has been composed by myself, and describes my own work, unless otherwise acknowledged in the text. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Furthermore, this thesis has not been accepted in any previous application for a degree neither at this university nor at any other.

Göttingen, 06.07.2011

(Christian Georg Wehrberger)