Document Name	Storage Services for the Fermilab Grid Facility
Authors	G.Garzoglio
Document #	cd-docdb 3279

Version	**Date**	**Comment**
v0.1-v0.3	Jul 09, 2009	Rearranging interview notes and considerations
v1.0	Jul 29, 2009	First coherent document
v1.1	Aug 19, 2009	Integrated feedback from interviewees
v1.2	Oct 26, 2009	Integrated feedback from Jon Bakken

Storage Services for the Fermilab Grid Facility

Gabriele Garzoglio

Oct 26, 2009

Table of Contents

1. Introduction

In preparation for the FY10 budget, the Grid Facility Department has initiated an investigation on the adequacy of the storage services provided in conjunction with the Fermilab Grid Facilities (FermiGrid). The user communities of FermiGrid have a variety of use cases and access patterns that sometimes strain the storage services currently available. This document describes recently reported problems with storage access from FermiGrid and the current and future use cases of storage access for the FermiGrid user community. This description can be used to allocate the appropriate effort to prepare the facility to support the problematic storage use cases or to work with our stakeholders to modify these problematic use cases. This information will form the basis for a set of recommendations on how stakeholders should use the storage services provided via the Grid.

2. Executive Summary

This investigation reports a series of storage incidents caused by jobs running on FermiGrid. The incidents are typically related to jobs' I/O overloading the BlueArc (BA) servers. We report problematic areas specific to BA; in particular, disk cost vs. performance, data migration policies, lack of monitoring and controls, change in user access patterns, and possible synergies with other storage solutions (sec. 3.1). We also discuss typical uses of dCache from FermiGrid, noting that Kerberos is not supported by policy and describing how X509-authenticated transfers can be used more effectively, by transferring multiple files within a single security session (sec. 3.2).

The report discusses the causes of common storage problems. In particular, uncoordinated concurrent access to storage, lack of data traffic controls for the BlueArc, use of the file system as a file catalogue, running "short" jobs, and users not adhering to recommended best practices (sec. 4). Some classical solutions are also discussed, such as the adoption of a peer-to-peer storage solution, data access queuing, and job throttling (sec. 4).

The document concludes describing the storage solutions for the use cases of some members of the Fermilab community, including DZero, CDF, Minos, CMS, and the future neutrino experiment (sec. 5). These solutions are organized according to activities, such as production processing (sec. 5.1), access to applications (sec. 5.2), and analysis (sec. 5.3).

For FY10, The areas of work uncovered by the investigation are the following:

- Investigating how emerging storage solutions, such as Hadoop [7] or Lustre [10], would serve our use cases, studying the synergy with the BA storage (point 2 and A sec. 4).
- Investigating how peer-to-peer storage solutions adopted in the past, such as DiskFarm [8], meet our needs today (point A sec. 4).
- Investigating how SAM can interface the BA to coordinate access to files for the use cases of the neutrino experiments (point 3 sec. 4).
- Investigating the availability of standard or commercial data queuing services, equivalent to fcp [6] in terms of features, low authentication overhead, etc. (point 1 sec. 4)
- Investigating how to shape traffic to the BA by intercepting I/O directed to it, similarly to Parrot (point 2 sec. 4 and point 5 sec 3.1).
- Implementing ephemeral home areas at the FermiGrid worker nodes, rather than allowing file writing directly to the BA (point 5 sec. 4).
- Continuing the program of reorganizing the BA volumes to optimize rate of access and to compartmentalize disruption of service (point 5 sec. 4).
- Educating users in low-overhead usage of X509-authenticated file transfers (point 2 sec. 3.2).

- Understanding if idle CPUs at CDF grants changing how SAM interfaces to dCache, from using the dcap protocol to using SRM [9] (par. “CDF” sec. 5.1).
- Producing a catalogue of successful use cases and their storage solution (sec. 5).

3. Storage Services and FermiGrid

In the current infrastructure, there are two major storage services accessible from FermiGrid: the BlueArc and dCache systems. FermiGrid relies on other groups for the maintenance and support of these storage systems. BlueArc provides a remote file system mounted to the FermiGrid nodes accessible via POSIX commands; dCache and Enstore provide remote storage service accessible via command line tools pre-installed at the FermiGrid nodes. The current intended usage of these systems is discussed below.

3.1. BlueArc

FermiGrid provides access to a cluster of worker nodes via Grid interfaces and to an interactive login machine (fnpcsrv1). These nodes mount a series of partitions from the BlueArc system. Permissions are set according to the requirements of the Open Science Enclave baseline [1]. The partitions are accessible by general Grid users and have the following intended usage [2]:

- Home areas: two partitions, one (fermigrid-home) for Grid users to store temporary files from the worker nodes (the area is purged regularly); one (gpfarm-home) only available to the integrative login machine.
- Product areas: two partitions with pre-installed software; one (fermigrid-products) with UPS products available for read and execute from the general purpose grid cluster; one (fermigrid-oseclient) with Grid and Condor software exported to all worker nodes. An additional partition (fermigrid-gceclient) with OSG client software will be exported to the interactive login machine (fnpcsrv1) in the future.
- Application areas: two partitions to make applications available for read and execute to the worker nodes; one partition (fermigrid-app) is for all remote grid users and is writable via grid interfaces (quota is 30GB per VO); the other partition (fermigrid-fermiapp) is for Fermilab-based experiments and is writable from non-Grid Fermilab machines.
- Data area: a partition for storing intermediate results of computation or for pre-staging input data that needs to be analyzed multiple times (default quota is 400GB per VO, but can be extended)
- VO-specific areas: FermiGrid mounts special purpose areas (e.g. extra partitions for data, code, etc.) for some major stakeholders (CDF, MINOS, DES, ILC).

In addition to these shared partitions, worker nodes provide local temporary and scratch space. Additional disk partitions used for administrative purposes are also available.

These are a few issues to consider when serving new disk space through the BA.

1. **Disk cost vs. performance**: an experiment that wants to use disk space from the BA should consider its requirements for both amount of space and rate of access. If the area is going to be made available from a few machines only, then a reasonable choice is buying slow disks (7.2k rpm) with a nominal 75 IO/s and an effective 37.5 IO/s (disks perform 1 read operation every write) for up to $1 per GB. If the area needs to be served from multiple machines, e.g. from a Grid cluster, buying faster disks might be a better option e.g. 10k rpm at an effective 100 IO/s or 15 rpm at an effective 150 IO/sec. These disks, however, are more expensive than $1 per GB. Another alternative for the use case of the Grid is buying a lot of small size disks, so that information can be accessed from multiple disks in parallel.
2. **Data migration policies**: the BA can transparently transport files from disk to disk. In principle, BA allows administrators to express policies on file age, extension, path, etc. This could be useful if an experiment bought some fast and some slow disks, so that depending on the expected access pattern, the BA can optimize the placement of the files. For example, if most people access files no older than 2 weeks, then the BA could migrate files older than 2 weeks to the slow disk and place new ones on the fast disk. In reality, this feature does not seem to work as advertised for some customers, such as US CMS. For US CMS, while migration to slower disks is automated, migration to faster disks is still a manual process. In addition, US CMS is adding more controllers and disks to mitigate BA bottlenecks.
3. **Lack of monitoring and controls**: the current BA system does not have good internal or external monitoring. Internal monitoring shows the status of the storage hardware, such as load on the disk controllers. External monitoring shows what user or process is causing IO operations. Lack of internal monitoring results in long troubleshooting time when users observe degradation of service (e.g. see Common Problems with Storage par 4 point 1.b). Lack of external monitoring requires that the system that mounts the storage (i.e. FermiGrid) limits the storage access, in order to guarantee a certain overall quality of service. In principle, this can be achieved by controlling the number of concurrent IO via job throttling, data queuing, etc. In reality, FermiGrid has no effective controls over user's access to storage (see par 4 point 2).
4. **Change in user access patterns**: when experiments repurpose disk areas on the BA and access patterns change e.g. with increased load, different file size, etc., it is a good practice discussing such changes with the BA administrators. This is especially true when using disks for interactive access, rather than for batch access. Administrators, in fact, can adjust the disks parameters to fit the new needs. For example, while latencies of a couple of seconds would not be acceptable for interactive access, it might be acceptable for batch access.

5. **Synergy with other solutions**: an experiment can use the BA together with other storage solutions. For example, an experiment might keep old data on dCache and new data on BA, or vice versa. Understanding what configuration better serves each use case should be investigated by activities spurred by this document.

3.2.dCache / Enstore

FermiGrid makes available storage client commands at the worker nodes. These include standard tools such as curl, wget, as well as GridFTP client, dcap / SRM [9] clients for dcache, and Enstore[1] commands. This is a list of data transfer options organized per authentication method:

1. **Kerberos**: to adhere to the Open Science Enclave baseline, Kerberos tickets are not available by default at worker nodes[2]. Therefore, transfer mechanisms such as kerberized dcap, are generally not available. For some legacy systems, such as for CDF computing, wavers to this policy have been obtained in the past. In general, commands such as scp / ssh cannot be used. A common benefit of using kerberized data transfer protocols is the low overhead in establishing the security session. Since Kerberos is generally banned, users might need to appropriately use X509-based transfer protocols to reduce their authentication overhead (see below).
2. **X509**: One reported concern from users accessing dCache, especially for small files, is the excessive overhead for establishing the security context for X509 via SRM or GridFTP. In reality, it seems a little known fact that both the SRM and FTP protocols (and their common client implementations: srmcp and globus-url-copy) support the transfer of multiple files at the same time[3]. In particular for GridFTP, only the control channel needs to be authenticated and multiple data channels can then be opened for each of the multiple files. We recommend that this fact be advertised for transfers within Fermilab. As a note, on the WAN, this transfer mechanism may incur in other overheads, since it is preferable to reuse an opened data channel, rather than opening a different channel per file, as the TCP windows takes time to reach an optimal size.
3. **UID/GIS** is a weak authentication mechanism, supported through dcap or encp. To limit the risk of unauthorized access, such mechanism is restricted for usage within Fermilab only for dcap and to nodes that are authorized to mount pnfs for encp. It should be noted that Grid jobs are typically mapped to generic local pool or group accounts i.e. UID/GID cannot be

[1] Enstore commands are available only on nodes that mount the pnfs file system, e.g. D0 CAB.

[2] Among other reasons, this policy prevents that compromised nodes in the Open Science Enclave provide attackers with credentials that can be exploited to compromise nodes in the General Computing Enclave.

[3] This of course assumes that the user does not specifically request to authenticate each data transfer channels, which would defeat this mechanism to minimize the authentication overhead per file.

used in general to access files stored from a personal user account. To circumvent this issue, for many users, communities such as CMS enforce the mapping of Grid credentials to local personal accounts, rather than generic pool or group accounts.

4. Common Problems with Storage and Possible Solutions

This is a list of common problems with storage services, as exposed during this investigation:

1. **Uncoordinated concurrent access to storage**: when accessing files from the BA, especially if the files are small like for some Minos use cases, multiple concurrent access may lead to problems with the storage system. The solution for Minos was queuing (serializing) access with a custom script (cp1). Two *caveat* to this solution:
    a. We think that it should be investigated using fcp servers [6], operated by FermiGrid, instead of a custom script. This would be a more common solution, considering that the fcp software is maintained by the REX department. Fcp servers provide a mechanism to queue up storage access requests. Clients request access to storage pre-pending the fcp command to the regular storage command line tools. The fcp command is configured to contact an fcp server and block until the queued higher-priority requests are finished. We propose that FermiGrid offer fcp servers, for the short term. For the long term, we propose to investigate some standard or commercial data queuing service, equivalent to fcp in terms of features, low authentication overhead, etc.
    b. This solution helped Minos avoid overloading the BlueArc controller. In general, however, this works only if all clients of the BA are properly queued. It seems that a recent incident has been caused by the fact that controllers on the back-end SAN are shared between the DZero databases and the MINOS data: multiple large data transfers overlapped during DZero database back ups. This DZero activity was independent from Minos data access and data streaming was not queued. The automatic load balancer of the controller could not cope with the traffic (see plots in Fig 1). Manual adjustments were necessary to rectify the problem, even if a full resolution may not be reached yet.

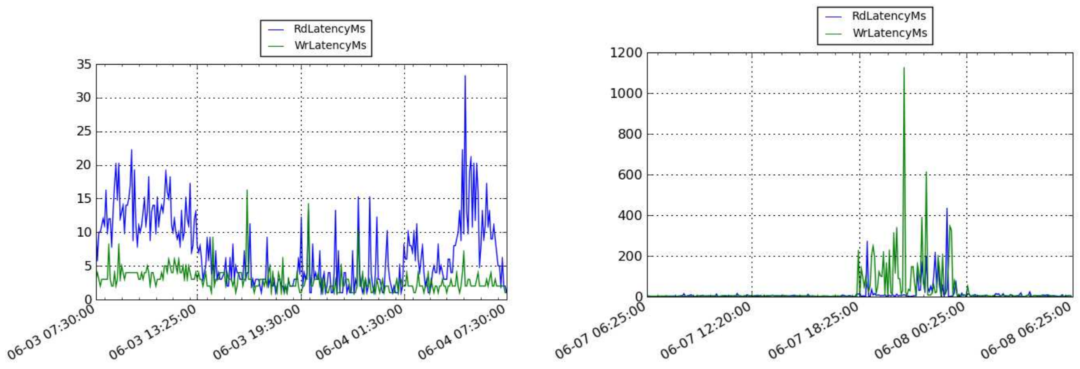

Fig 1: Latencies in the Read (blue line) / Write (green line) access of Minos files from the BA. Typical latencies are 5 – 10 ms. During a problem with concurrent DZero database backups to the BA, there were latencies up of 1 – 2 order of magnitudes. The problem was rectified by manually adjusting the load balancing of the BA controllers (plots courtesy of Art Kreymer)

2. **Lack of data traffic control for BA**: because of a lack of external monitoring, the BA relies on the Grid facility to impose limits on the data traffic. On the other hand, FermiGrid has limited control over the data access patterns of user jobs. Job throttling has limited effectiveness, because under high-load conditions, a few misbehaving jobs can overload the BA. Data queuing is a possible solution, but difficult to enforce because it requires all users to change their code (see above). Other solutions should be investigated, including: (a) automatically intercept IO to the BA (as done for accessing applications by Parrot) and queue the requests e.g. via fcp; (b) studying other storage solutions, such as Hadoop, which possibly better shape the network traffic for data access. These data throttling techniques should be configured considering that the bottleneck is not the 10 GBit/s network connection with the BA system, rather the back end SAN.
3. **Use of the file system as a file catalogue**: Attempts by VOs, such as Minos, to use the BA file system as a file catalogue resulted in a particularly high load for the BA. In a reported incident, the file catalogue was created periodically by issuing recursive file list commands (ls -R) on large sections of the BA. The computing division supports data handling solutions, such as SAM, that offer a file cataloguing service. In particular for SAM, the catalogue is used in conjunction with file caching technologies (SAM cache and dCache), so that the catalogue can be kept up to date without the need for recreating it periodically. In principle, SAM could interface the BA storage, either defining a SAM internal cache for it (for a limited number of files) or interfacing it as an external storage, like dCache (for a large number of files).
4. **Short jobs**: Short running jobs that interact with the file system may create a BA overload. In a reported incident, ILC jobs downloaded input files from the BA to the worker node and uploaded the output at the end of the job, as per best practices. The logic of the job, however,

was such that the input file was never processed (it had been processed by other jobs before) and the job lasted about 15 seconds. The resulting intensive IO to the BA caused a disruption of service. In general, “very short jobs” tend to create problems for FermiGrid, as they tend to show characteristics of data intensiveness, not very well supported by computational Grids in general. Problems running very short jobs have been observed in the past also with DES.

5. **Users not adhering to best practices**: When running on FermiGrid, there are accounts of incidents caused by users not adhering to the FermiGrid best practices. Adhering to these best practices does not guarantee freedom from system problems; rather it tends to mitigate potential troubles. These violations include reading input data directly from the BA, instead of making a local copy first; ignoring recommended fair-share quotas on local disk; inappropriate use of home areas, including using BA mounted home areas as local scratch disks. To address some of these issues, FermiGrid is devising a range of solutions, which include creating ephemeral home areas at the worker nodes i.e. moving the content to the user at the end of the job before erasing it from the worker node; reorganizing the BA volumes to optimize rate of access and so that an incident related to an experiment-specific area does not affect other experiments.

In summary, to address overload problems with the file system servers, these are the classical solutions that can be applied to a Grid computing facility:

A. **Adoption of a peer-to-peer storage solution**: files are stored across a large number of nodes e.g. the worker nodes. This way, clients do not all contact and potentially overload a few file servers. Typically, the system can manage dynamic data replication, so that there is no data loss should a node be lost. Examples of these file systems include Hadoop [7] and DiskFarm [8]. Hadoop is an open source project adopted by Yahoo. Single files are striped across multiple nodes. The OSG Storage group at Fermilab and other groups in OSG are studying the properties of this storage solution. DiskFarm was developed and adopted at Fermilab. Individual files are distributed across worker nodes. A criticism of such solutions is that they require the installation and operation of servers at the worker nodes and that they make reinstallation of the nodes difficult if the node holds persistent data. To address the latter, automatic redundant file replication would lift the need to manually restore the status of the persistent data on the node.

B. **Data access queuing**: I/O requests from clients are queued, so that the concurrent requests served do not overload the file server. Data queuing can be implemented by different systems, including fcp and SRM. As noted, this solution depends on clients utilizing the queue, rather than accessing the data transfer servers directly.

C. **Job throttling**: the batch system is notified that the jobs are I/O intensive, so that only a limited number of jobs can be scheduled at one time or on certain nodes. This is a partial solution to the problem, as already a few misbehaving jobs can overload a file server. In any case, it is a solution easy to implement at least on the Condor batch system.

5. Storage Use Cases

This section discusses the storage use cases of major Fermilab VOs. It is not intended to describe in detail the computational model of each VO; rather, to highlight aspects of interest in the usage of storage.

5.1. Production Processing

Storage problems with production processing tend to be more easily understood, as they happen on dedicated facilities, accessible by a few power users, through well-understood applications. These are a few highlights of how different communities use storage for production

Minos: Minos keeps files from raw data to farm output to official root-tuples, both in dCache and in enstore for production and analysis. These files are catalogued via SAM. File delivery to jobs is regulated via SAM mechanisms (projects), even if the actual data transfer is still responsibility of the job. Raw data files range from 70 MB to 250 MB, produced at the rate of 1 file per hour. Raw data is concatenated monthly into file sizes better suited for tape (~ 7GB) and written to a separate set of tapes at Fermilab. Additional copies of the raw data are also kept in Minnesota and Texas. The full reconstruction output files are similar in size to the input files, but these are rarely accessed. Analysis is performed from much smaller root-tuple files, which are concatenated to a size of roughly 1 GB per day. All official files are available from dCache. Files are stored on 20TB of disks on read/write pools, so that stored data is immediately available, without first going to tape. This prevents the system to mount tapes unnecessarily. Data of current interest is also kept on the BA, mainly for analysis. Roughly, there are about 20 TB DCache disk, 90 TB BA disk, and over 300 TB on tape.

DZero: DZero production uses SAM-Grid [3] for both data processing and montecarlo. SAM-Grid provides access to all data through SAM via a dedicated SAM station, with several SAM caches and disks. Files are large by design; raw files are around 1 GB. Fig 2 shows the distribution of file sizes for all reconstructed, thumbnail, and root-tuple files from Jan 2003 to Mar 2005 (note that the diagram does not include raw files) [4].

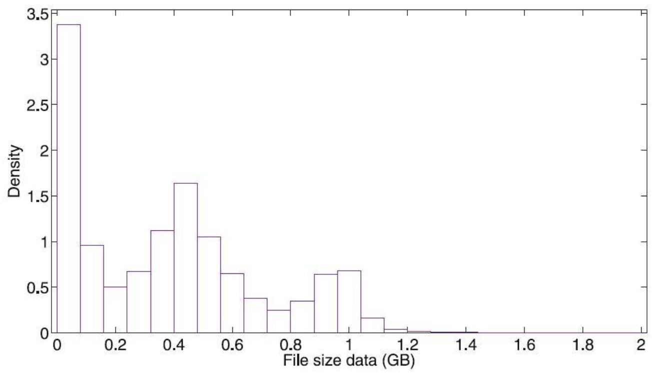

Fig 2: A diagram of the relative abundance of DZero files of a given size. The diagram includes 996,227 files of reconstructed, thumbnail, and root-tuple data tiers. The mean and median size is 0.4 GB.

For some production activities, such as montecarlo generation, the output file sizes is maintained around 1 GB by merging an appropriate number of smaller files. These smaller files are produced by individual jobs and are temporarily stored to “durable” SAM location. Durable locations consist of dedicated disk servers accessible by GridFTP and maintained at an optimal operational load by fcp; data is automatically cleaned periodically after merging by resident cron jobs interfaced with SAM.

CMS: CMS production activities at Fermilab are run on the “OSG” cluster, which consists of 1,200 nodes and 6000 cores. The cluster is accessible from the Grid via 4 Computing Elements (CE). Jobs are managed via the GlideIn WMS (pilot-based) system. The bootstrapping code of the pilot jobs is transferred from the CE to the WN via condor internal transfer mechanisms (see also par “Other communities” in sec. 5.2). The GlideIn system (condor) manages the transport of the stdout, stderr, and log files from user jobs. The worker nodes do not mount any shared file system for data transfer (see also par “CMS” in sec. 5.2). Data is staged in and out of the worker nodes from dCache via X509-authenticated SRM / GridFTP transfers. Weakly-authenticated dcap transfers from dCache to the worker node are also allowed if the unix file permissions are satisfied (i.e. the user has been previously strongly authenticated). In comparison to the general access dCache pool (fndca), the CMS pool has order of magnitude more cache disk. Because of the size of the general access dCache pool, experiments such as Minos cannot hold all of their dataset on disk anymore, as they did earlier, to allow quick data lookup from cache.

CDF: CDF has a processing workflow similar to CMS and DZero. Similarly to CMS, some user access files in dCache directly, after having selected these files through the SAM metadata catalogue. Similarly to DZero, some other user rely on SAM for all their data handling needs. Dissimilarly to DZero, however, SAM does not manage its own cache; rather, it interfaces to dCache through the dcap protocol. This deployment choice has the consequence that SAM cannot synchronize with the information on the availability of the files from dCache; therefore, SAM cannot order file delivery according to waiting time (e.g. first deliver files in the cache, while fetching files from persistent storage). Hence, a fraction of the jobs may unnecessarily wait idle for file delivery from tape. To address this problem, CDF overbooks its cluster by 20%, considering that one job out of five will be waiting for files at any one time. To better handle this situation, improvements to the batch system have been requested to the condor development team.

Allowing SAM to interface to dCache via the SRM interface would mitigate this efficiency problem. Using SRM, in fact, SAM can ask the storage to schedule the delivery of a limited fraction of the dataset (look-ahead number), if the data is not available in the cache. For request units (projects) containing “a lot” of files, however, SAM only issues a limited number of look-ahead requests to SRM at any one time, to avoid flooding the system. In this case, some files may be in cache, but SAM might know only at a future round of SRM requests. The number of look-ahead requests to SRM can be configured and in general is proportional to the number of jobs. It should be noted that this solution is better than using dcap, where there is no knowledge of cached files, but it is not as good as using SAM internal caches, where there is complete knowledge of file statuses. It should be noted that the conversion from using dcap to using SRM would require a reorganization of the dCache pool, which today relies heavily on Kerberos doors and does not support X509 doors. This conversion might also have implications on the CDF data handling model.

For some data, such as the stnptuple, in addition to using SAM catalogue, CDF catalogues its data in text files containing filenames, made available through a web server. Such cataloguing method is custom and potentially error prone.

Future Neutrino Experiments: The Fermilab scientific program at the intensity frontier includes several experiments, some of which, such as Minos, that have already started and some that will start in the next few years. These experiments are Minos, MiniBooNE, NOvA, Minerva, MicroBooNE, ArgoNeut, Mu2e, and DUSEL. Fermilab will provide some computational and storage facilities for these experiments; the projected needs for computing and storage, including disk vs. tape, is described elsewhere [5]. This section highlights some of the storage requirements, as they are currently understood.

For Minerva, the current prototype detector is 20% of its final size; data handling considerations below assume the full detector. The data handling needs are built around the concept of “spill” i.e. one injection of beam that generates events in the detector. The relevant data tiers for production processing are *Raw* data from the detector, *Pedestal* files, *CalDigit* (raw data calibrated with pedestals), *ReconData* (reconstructed data from CalDigit input), *DST* (processed data from ReconData). Their storage needs are discussed below. The estimated processing time from raw to DST formats is about 1 sec / spill, even if this may grow in the future.

- The *Raw* data files will be in root format and expected at 1GB in size. One file contains an hour of data (called a subrun) or 1500 spills. 10-24 files are expected in a day (called a run). 3.3 TB of raw data will be produced per year. The data needs scratch disk space for processing and will be store on tape permanently.
- *Pedestal* files will be 400 MB in size and will be acquired every few hours. This will generate about 500 GB / year of data. Like for raw files, pedestal files need scratch disk space and permanent tape storage.
- *CalDigit* files are estimated at 85MB in size, generating about 300 GB / year. Recent versions of *CalDigit* data will need to be available on disk for user processing and all data will need to be stored on tape.
- *ReconData* files are about 100 MB in size and 450 GB / year. Several recent versions of *ReconData* will need to be stored on disk for user processing. All the data will need to be on tape.
- *DST* files resulting from processing subruns are about 50MB each file. Since *DST* are the main input of analysis jobs, they will be merged together by run, yielding an estimated 500-1.2 GB per file. *DST* will consist of about 200 GB / year and will need to be available on disk for user analysis and tape for permanent storage.

All production applications, including montecarlo, need access to a database, even if it is believed that this can be reduced to accessing local information, at least for offsite usage.

For montecarlo, 60 million simulated spills (events) will be produced per year (10 times the amount of real spills acquired). Since the size of an event is similar to the size of the raw data for a spill, the amount of montecarlo data produced is 33 TB / year. The estimated generation time is 10 sec per event. All montecarlo data will also need to be processed through the chain of data tiers as the raw data. Montecarlo jobs require access to *Raw* or *CalDigit* files, an important constraint for data handling needs.

For NOvA, the processing chain is still being designed and it is going to be similar to the one of Minos. It is estimated that NOvA will have a total dataset roughly three times the size of Minerva. The size of the data resulting from the processing chain has not been fixed yet. For montecarlo, however, it is expected that jobs running 1 day will produce files of a few 100s of MB; after reconstruction, these will be 1-2 GB per file. NOvA is considering the BlueArc for data access on FermiGrid, even if people understand the potential need for more scalable solutions.

5.2. Access to Applications

A job running on FermiGrid can access its application by different means. This section describes how this is done by different communities.

Minos: Minos jobs access their application through Parrot. The system makes the application distribution appear as local. The application can be a set of executables and dependent libraries or an archive. When a job tries to access a file in the distribution, Parrot intercepts the I/O request and issues a transfer command for that file to a web server. The web server can be made more scalable using a web proxy (squid.fnal.gov). The web proxy ends up caching the most popular versions of the application. Once the application file is transferred to the worker node, it is accessed by the job transparently. This mechanism has the advantage that the data access traffic can be shaped / queued through the web proxy. In addition, jobs running the same application on the same worker node can be configured to transfer the application only once. Using Parrot has been very useful for initial deployment at Fermilab and at the TACC TeraGrid site. For full scale production, however, Minos is considering installing all software locally, to avoid Parrot's overheads.

Parrot is typically used only for applications because it does not offer file selection through a metadata catalogue; rather, the job needs to know the location of its application files in the distribution. However, for simple data access use cases, this solution could be more scalable than direct access to a remote file system (e.g. the BA) and should be investigated.

CDF: like Minos, CDF also uses Parrot to access application at clusters in Europe. At the CDFGrid cluster, applications are accessed through a code server. At the NAMGrid cluster, applications are made available as archives and downloaded by the job at startup. Access to these archives is made scalable through the use of web server proxies.

DZero: DZero production access job applications from SAM. Applications are stored as archives and each has associated metadata in SAM. Once SAM makes available the application to the job, the application is typically transferred to the worker node via GridFTP / fcp.

For analysis, the DZero offline code-base (executables and libraries) is available at all CAB worker nodes through NFS. The code can be used either as is (e.g. to run the reconstruction application) or by linking some of its libraries to code supplied by the user. For the two main analysis use cases (analysis of a thumbnail and of a root-tuple), DZero makes available job submission tools that orchestrate the delivery of the user-supplied code to the worker nodes. Before submission, the user makes a private code release on a disk area and builds an executable: this release is the user-supplied code. The users submit their jobs using the appropriate submission tool, specifying the location of their release area and the name of the executable. The tool makes an archive of that area and submits a job bootstrapping script. At the worker node, the bootstrapping script copies the archive and runs the specified executable. The transfer protocol for the copy depends on the location of the code release. If it is on the user's machine, then it is copied with rsync via ssh; if it is on a project disk (e.g. bluearc), then it is copied over the remote file system with cp.

CMS: the CMS facility makes available applications to its two clusters (OSG and LPC) via a read-only NFS file system. The system consists of 4 servers (1 master and 3 slaves) and has a capacity of about 40 MB/s.

Future Neutrino Experiments: the characteristics of each application depend on the specific experiment. For Minerva, the code consists of a 6GB build area. Generating a standalone executable is considered difficult and is deferred to a later time. The current approach is distributing the code via NFS. NOvA also is planning on a similar approach on FermiGrid. They want to distribute the application as a tar ball to exploit opportunistic resources.

Other communities: other communities directly send the application to the worker node via condor mechanisms. On FermiGrid, condor is configured to stage the application in the user’s home area, to make it available to the worker nodes. Since the home areas are mounted from the BA, this might in principle create problems of concurrent access, especially if the application archive is large. In practice, to date, there are no BA incidents related to application transfer. Despite the ability of condor to transfer the application using internal mechanism (i.e. without relying on a shared file system), this configuration is not the default option on the Grid, because the official condor job manager does not support it. VDT makes available a job manager (jobmanager-nfslite) that supports the condor internal transfer mechanism. Some facilities, such as CMS have used this job manager with good results. Drawbacks of the job manager-nfslite are that it only works with the condor batch system. Also, testing showed that it does not work for jobs of certain non-HEP VO's that rely on workflow managers such as Pegasus. In any case, for security considerations, it is preferable to transfer the application (as well as the user proxy and stdout/err) via internal batch system mechanisms. FermiGrid, therefore, is considering adopting the job manager-nsflite.

5.3.Analysis Activities

Analysis activities are typically uncoordinated and tend to cause most of the problems when accessing storage. These are the main mechanisms used to access data for analysis.

DZero: DZero analysis jobs are mostly run on the CAB cluster, using local batch system interfaces for job submission. Opportunistic FermiGrid resources are also used depending on the need. To give a sense of scale, the amount of computation performed monthly on CAB is around one million CPU hours. Access to official data is controlled via two SAM stations, distinct from the one dedicated for production, managing about 750 TB of cached data. The system has successfully transferred more than 200 TB of data per day. An additional 300 TB of disk space are available from the BA (the so-called project disks). Some of these disks are mounted to the worker nodes. Users typically use these disks to store results of a data skim or to process an intermediate input root-tuple, not available from SAM. Input data is generally copied from the project disks to the worker nodes using kerberized cp. Some of these transfers are authenticated through Kerberos keytab files installed at the worker nodes. Like other communities, also DZero has faced problems when transferring in an uncoordinated fashion too much data (order of 100GB) from the BA.

CDF: Similarly to DZero, CDF officially uses SAM to handle input data to analysis jobs. Many users, however, access data directly from dCache, without relying on SAM. This access mode does not conserve a record of the dataset popularity, a valuable piece of information to understand the community data access patterns. Coordinating the transferring of the output is responsibility of individual groups. A successful configuration consists in deploying a group-maintained file server that provides a GridFTP interface, queued by FCP. Periodically, data of general interest, such as montecarlo files, is stored to SAM using cron jobs.

Minos: Analysis jobs typically access input data from the BlueArc, ideally to minimize delays in the access. All worker nodes mount 90 TB of dedicated disk, as the quota on the public areas is considered too small. This disk contains the two most recent cycles of reconstructed data, which consist of root-tuples of 1-2 GB in size (the result of about 1 day of concatenation). Other input data consists of condensed root-tuples from different analysis groups, index of montecarlo events, etc., all files very small in size and difficult to concatenate.

Future Neutrino Experiments: Minerva has provided some considerations on their storage needs for analysis. The main input for analysis jobs is DST files (sec. 5.1), consisting of about 200 GB / year. In addition, the collaboration plans to eventually use microDST, a summary data format anticipated at 1-2 kB / spill in size i.e. 6-12 GB / year. The envisioned processing pattern consists in physicists running on the order of 50 parallel jobs, each accessing 1 GB files to optimize throughout. Analysis may need access to a subset of a database, but this may be reduced to accessing local information only. With respect to DZero or CDF data, Minerva data will be more I/O bound, because it requires less CPU time per event. This will be especially true in the early days of the experiment, where users may be running over the full sample and rejecting 90% of the events almost instantaneously. There are currently no estimates on the required number of jobs per day, but in terms of number of people, the size of the collaboration is about one tenth of the size of DZero or CDF.

CMS: US CMS runs most of its analysis at Fermilab at the LHC Physics Center (LPC) cluster, which consists of about 240 nodes and 2000 cores. The LPC cluster is partitioned in two sections: (1) is used for interactive login to a small set of load-balanced nodes; (2) is used for batch processing and it is managed by condor. Home areas for the interactive section are served by the BlueArc, which implements a quota system. The batch section also mounts the BlueArc areas, but its usage is discouraged. User jobs in the batch section are managed by the GlideIn WMS system, which uses condor internal mechanisms to transfer the bootstrapping script and stdout, stderr, and log files. The application is accessed from an NFS read-only file server (sec. 5.2). Data input and output is stored in dCache and accessed via the dcap protocol.

6. Acknowledgments

This document could not have been written without the patience and availability of the people that I have interviewed. Their input on has been of fundamental relevance. In particular, I want to thank Adam Lyon, Qizhong Li, Robert Illingworth, Rick St. Denis, Lee Lueking, Heidi Schellman, Art Kreymer, Ray Pasetes, Matt Crawford, Ian Fisk, Steve Timm, and Keith Chadwick.

7. References

[1] Open Science Enclave Baseline: CD DocDB 2573

[2] FermiGrid Bluearc NAS Storage: http://fermigrid.fnal.gov/fermigrid-bluearc.html

[3] G. Garzoglio, "A Globally Distributed System for High Energy Physics Computation", published by Verlag Dr. Müller, Saarbrücken, Germany, 2009. ISBN 978-3-639-16506-7

[4] A. Iamnitchi, S. Doraimani, G. Garzoglio, "Workload characterization in a high-energy data grid and impact on resource management", Journal of Cluster Computing, Jan 2009, DOI 10.1007/s10586-009-0081-3

[5] Fermilab Neutrino Program Computing Resource Needs Assessment: CD DocDB 3205

[6] FCP: Farm Remote Copy Utility: http://www-isd.fnal.gov/fcp/

[7] Hadoop: http://hadoop.apache.org/

[8] Disk Farm: http://www-ccf.fnal.gov/dfarm/

[9] I. Bird, B. Hess, A. Kowalski, D. Petravick, R. Wellner, J. Gu, E. Otoo, A. Romosan, A. Sim, A. Shoshani, W. Hoschek, P. Kunszt, H. Stockinger, K. Stockinger, B. Tierney and J. Baud, "SRM (Storage Resource Manager) Joint Functional Design", Global Grid Forum Document, GGF4, Toronto, Feb. 2002

[10] Lustre File System: http://www.lustre.org