

FIFE Architecture Design Report

The FIFE Working Group

Michael H. Kirby, Adam L. Lyon, Keith Chadwick, Gerard Bernabeu Altayo,
Mine Altunay, Dennis D. Box, Joseph B. Boyd, Michael Diesburg, David W. Dykstra,
Lynn A Garren, Gabriele Garzoglio, Oliver Gutsche, Robert A Illingworth,
Arthur E. Kreymer, Robert K. Kutschke, Tanya Levshina, Igor V. Mandrichenko,
Marc W. Mengel, Parag A. Mhashilkar, Andrew John Norman, Ruth Pordes,
Andrew J. Romero, Marko J. Slyz, Frederick D. Snider, Steven C. Timm,
Margaret Votava, Stephen A. Wolbers

4-Feb-2014

Version 1.3

Abstract:

This is the Fabric for Frontier Experiments (FIFE) Architecture Design document. This document is managed by the FIFE Working Group project of the Scientific Computing Division of Fermilab and documents the currently agreed upon designs for the collaborative scientific-data processing solutions to Intensity Frontier and Cosmic Frontier experiments at Fermilab.

This document is derived from the FIFE Architecture Report.

Table of Contents

Executive Summary	4
Document Revision History	6
1. Introduction	7
2. Design Considerations	8
3. FIFE Architecture	9
3.1 Network	12
3.2 Authentication	12
3.3 Central SAN	13
3.4 Central NAS	13
3.5 General-Purpose Cloud Services	13
3.6 Dedicated VM Services	14
3.7 On-Demand Services	14
3.7 Off-site Grid / Cloud Services	15
3.8 User Home Areas	15
3.9 User/Account Information	16
3.10 Database Management Service	16
3.11 Scientific Database Service	16
3.12 Experiment Data Storage	17
3.13 User Data Storage	18
3.14 Data Handling	18
3.15 Processing and Analysis Framework	19
3.16 Experiment and Support Software Access	19
3.17 Software Build System	20
3.18 Web Caching	20
3.19 FermiGrid	21
3.20 FermiGrid Worker Nodes	22
3.21 Pilot Based / Glide-In System	23
3.22 Batch / Job Submission	23
3.23 Interactive Servers	23
3.24 Desktops, Laptops and Mobile Devices	24
3.25 Off-Site Interactive Computers	24
3.26 User Job	24
3.27 User Access to Data	24
4. A Conceptual Design	25
4.1 Central NAS	26
4.2 Dedicated VM Service	27
4.3 User Home Areas	27
4.4 User/Account Information	28
4.5 Scientific Database Service	28
4.6 Experiment Data Storage	28
4.7 User Data Storage	28
4.8 Data Handling	29
4.9 Experiment and Support Software Access	29

4.10	Software Build System	29
4.11	Web Caching	30
4.12	FermiGrid	31
4.13	FermiGrid Worker Nodes	32
4.14	Pilot Based / Glide-In System	32
4.15	Batch Submission	32
4.16	Interactive Servers	33
4.17	Cloud and On-Demand Services	33
4.18	User Access to Data	33
5.	Summary	34
	References	35
	FIFE Architecture Committee Charge	37

Executive Summary

This document is a living document that reflects the currently agreed upon FIFE Architecture Design. The FIFE offline compute system is designed to enable physicists to efficiently develop software, generate events, simulate detector responses, reconstruct physics quantities, select events and analyze data at Fermilab and also from off-site. The architecture is:

- network centric, i.e. an Internet Protocol network binds the components into a system and thus equalizes naturally on-site and off-site activities;
- modular, i.e. switching the implementation of a component will keep the overall architecture intact and thus bring longevity across technology updates;
- scalable, i.e. allows data/CPU intensive components to be implemented as distributed services;
- fault stable, i.e. components have minimal inter-dependency and maximum isolation so failures will stay contained and not escalate to unrelated components/services.

The architecture can be implemented with current technology. Many components exist already in the current FIFE computing setup. With a few evolutionary changes, the current system can become a robust, scalable and fault tolerant system that will be able to serve the experiments needs for a long time..

Central network attached storage, NAS, should not be used to store scientific data, and direct access from the large number of worker nodes should be eliminated. Scientific data should be stored in the mass storage system and accessed through the distributed disk cache system via a data-handling layer. For the conceptual design EnStore, dCache and SAM-Lite were chosen as implementation.

Services should be made independent of the central NAS as much as possible.

Grid computing facilities on-site and off-site provide the computing power for all data processing, selection and analysis. They are accessed via a batch/job submission service that is based on a pilot/glide-in system.

There is an urgent need to upgrade the central user/account information system with experiment membership / affiliation information. The system must be capable of supporting the use case of users with multiple experiment memberships / affiliations.

Database read access should be tiered. We recommend SCD to design and write the database schema and abstraction with the experiments for maximum cache ability and to hide the database implementation.

There are opportunities at both handling of user generated scientific data and experimental software to improve the analysis experience of users. A dCache based user data storage/project space is part of the conceptual design. Experiment software builds are moved from the interactive GPCF machines to a dedicated build system where a large number of processors can be used to build software releases in parallel. Experiment and support software is then distributed and accessed on/off-site, interactively and in batch via CVMFS.

Document Revision History

Date	Version	Author	Comments
29-Jul-2013	V1.0	Stephan Lammel	Initial Committee Report
16-Dec-2013	V1.1	Keith Chadwick	Transfer content into Microsoft Word
7-Jan-2014	V1.2	Keith Chadwick	Change document title to Design Report (from Report), add text from Gabriele Garzoglio regarding "on demand services", revise squid architecture to reflect network architecture, update FermiGrid services description, import revise figures from Microsoft PowerPoint
4-Feb-2014	V1.3	Keith Chadwick	Incorporate final comments, publish latest version.

1. Introduction

The scientific research program of Fermilab includes an increasing number of neutrino and rare decay experiments at the Intensity Frontier (IF) and experiments to probe dark matter/energy at the Cosmic Frontier (CF). While on average those experiments are of smaller scale than Energy Frontier (EF), experiments their total computing needs start to match those of individual EF experiments.

Most of the IF and CF experiments have small offline computing groups, limited manpower, and limited expertise they can devote to computing infrastructure. The Scientific Computing Division, SCD, of Fermilab is providing more complete solutions, like data processing and analysis frameworks and a fabric of integrated services [1] to help the experiments manage, process and analyze their data. This reduces the required offline computing effort for the experiments, brings naturally more sharing but also coupling of experiments. Software features and bugs are not limited to an individual experiment, maintenance downtimes need broad coordination and service outages can have a wider spread impact. This leads to a demand for higher quality and reliability.

A few years ago Fermilab Computing Division set up a general purpose compute facility, GPCF, (for both interactive and batch computing) and a Grid [2] compute cluster, the GP Grid cluster. The setups were evolutions from previous clusters. Both, GPCF and the GP Grid cluster, have been very successful. The GP Grid cluster is an Open Science Grid, OSG [3], facility but looks to Fermilab experiments more like the GPCF batch system than a Grid facility: Experiment filesystems are mounted over the network on the worker nodes. The central Network Attached Storage, NAS, service is provided by a high performance NFS appliance, a BlueArc Titan. The steady increases in disk capacity (and decreases in storage costs) allowed the experiments to store all their scientific data there. Data are accessible at all times from anywhere inside Fermilab and made the usage of GPCF and GP Grid cluster very convenient.

The setup worked well while both scientific data and GP Grid cluster were small. As the GP Grid cluster grew, the probability of jobs simultaneously accessing the NAS filesystems (and thus temporarily overloading the I/O capacity of the NAS server) increased. The first experiments overcame those peak collisions with a simple throttling and queuing mechanism. With the increased number of experiments and an influx of new users, the accidental temporary overloads have become common. In case when the I/O capacity of the NAS server is reached not one or two but all of its filesystems become unresponsive and the impact is widespread [4].

The good network at Fermilab, the high-performance and attractive storage cost of the BlueArc and the possibility to mount filesystems anywhere as needed without

advanced planning triggered a broad adoption. Various scientific services use the central NAS heavily. Neutrino, rare decay and dark matter search experiments have typical particle physics data and workflows: large numbers of independent events or spills in files are being processed, filtered/grouped and analyzed. For the Dark Energy Survey, DES, and other experiments with imaging-type data, files do not consist of independent elements but the complete image data file is needed for processing and images are analyzed individually. The architecture presented here was developed for computing tasks of both types. However, experiments with imaging-type data may require additional data handling features beyond what has been considered here.

Architecture, the art of building, is commonly done in three phases: (1) planning, i.e. thinking through needs, usage and forecasting changes; (2) designing, i.e. creating the architectural blueprint; and (3) construction, i.e. building the components and assembling the structure. In computer science, architecture specifies the components and their relationships. For this report we start with the computer science definition: guiding principles of the architecture, components and their relations.

In a second part we provide a conceptual design of a possible FIFE offline computing system based on this architecture.

2. Design Considerations

The architecture of the FIFE offline compute system is designed to enable physicists to efficiently develop software, generate events, simulate detector responses, reconstruct physics quantities, select events and analyze data at Fermilab and from off-site. The architecture is:

- network centric, i.e. an Internet Protocol, IP, network binds the components into a system and thus equalizes naturally on-site and off-site activities;
- modular, i.e. replacing a component will keep the overall architecture intact and thus bring longevity across technology updates;
- scalable, i.e. allows data/CPU intensive components to be implemented as distributed services;
- fault stable¹, i.e. components have minimal inter-dependency and maximum isolation so failures will stay contained and not escalate to unrelated components/services.

The users and user programs interact with a few high-level services from the interactive servers, desktops, laptops and batch worker nodes at Fermilab and off-site. The high-level services combine the lower level computing software and

¹ Fault stable here means that in case of a failure only the component/service and its dependent components may be

services into scientific services that each addresses a particular task or functionality. As the result, the users need to learn less about computing components, and computer scientist and engineers can design and implement robust, scalable and cost-effective solutions.

3. FIFE Architecture

In this section 3, we describe the FIFE offline computing architecture components and their relations. The following section 4 provides a specific example of the implementation of this architecture. The current architecture of the FIFE compute system has served the IF experiments well so far, even considering the instability during the last year due to accidental user overloads of the central NAS service. Future IF and CF computing requirements are not fully defined.. The estimates of computing resource needs are taken from the 2013 Scientific Project Portfolio Management review [5]. The detailed requirements for batch computing are taken from [6] together with requirements and usage information based on experience from current and past experiments. Over the next three to five years we do not expect revolutionary computing changes or disruptive innovations but evolutionary computing changes and incremental technology updates. The FIFE architecture will allow for technology updates and will consist of components with well-defined functions and interfaces that remain valid in case of an implementation switch. The expectation is that reduced support for customized solutions and a significant need of IF and CF experiments to use other academic, i.e. opportunistic, and potentially commercial computing resources outside of Fermilab. At the highest level there are five groups of components that a user sees: interactive computing, batch computing, storage, software and databases.

Figure 1 shows the five groups and their relationships. The interactive computing accesses storage for management and development/testing/debugging.

The storage system is accessed predominantly by batch computing. Batch jobs are prepared and submitted from the interactive computing and they require access software (libraries) and database information. The access to database and software is also required for interactive computing and is updated/managed from there. The software comprises of analysis software, experiment software, support software, the frameworks with data handling and database interfaces, the system to build libraries and executables, and the system to distribute those files to interactive and batch computing.

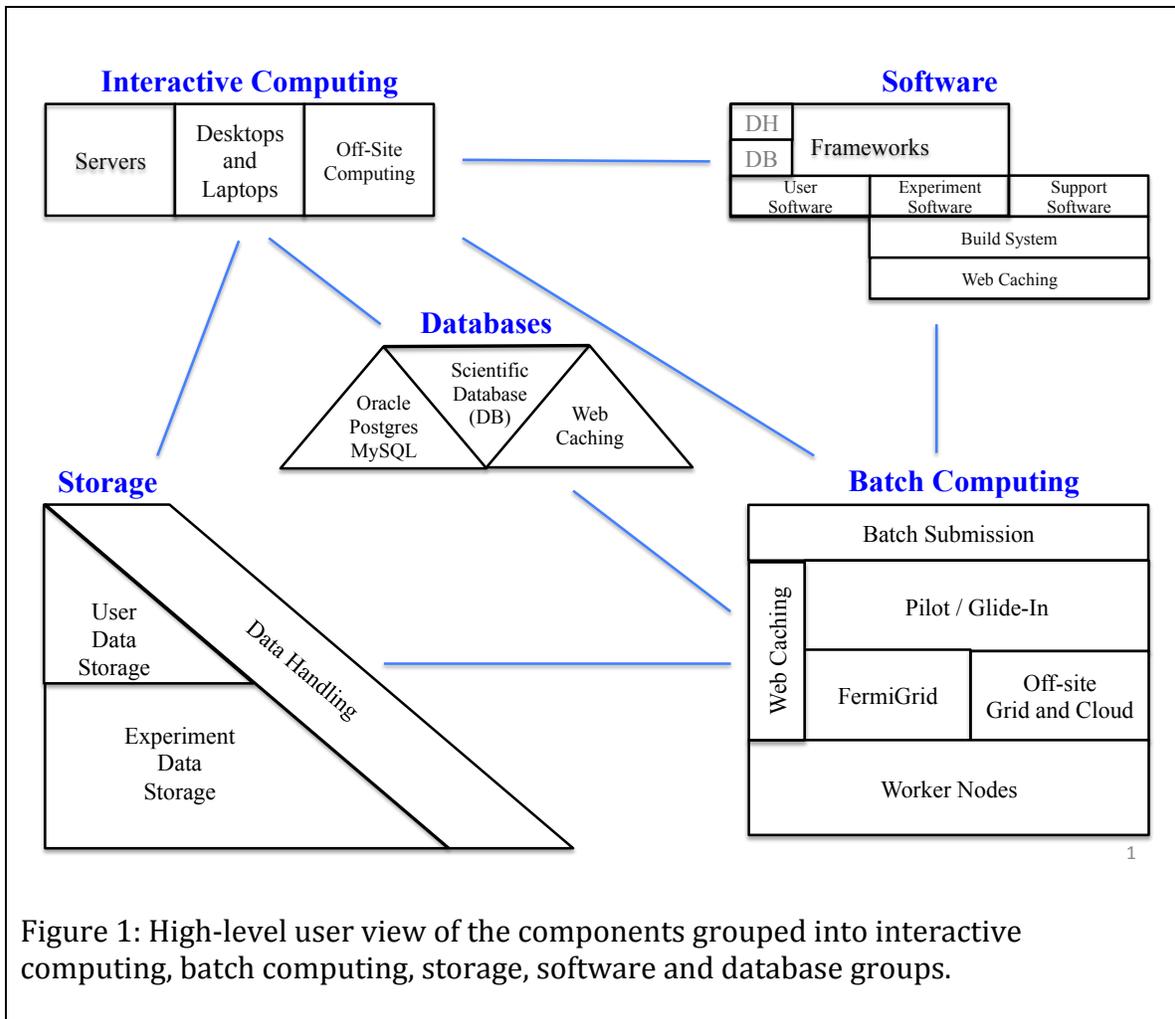


Figure 1: High-level user view of the components grouped into interactive computing, batch computing, storage, software and database groups.

The architecture of the FIFE compute system starts from the current architecture but splits handling of scientific data into several new components based on positive Tevatron and LHC experience. The users and user jobs utilize a data handling system that provides access to user and experiment data stored in the storage components and that provides metadata management. Figure 2 shows all the components of the architecture. The components are layered with well-defined hierarchical relations. Each component addresses a specific task. The components in the lower layers fulfill more basic functions, and components in the higher layers (using components in the lower layers) will provide more complex services. When combined, the system will enable experiments to develop their software, generate events, simulate detector responses, process and select data and analyze them efficiently.

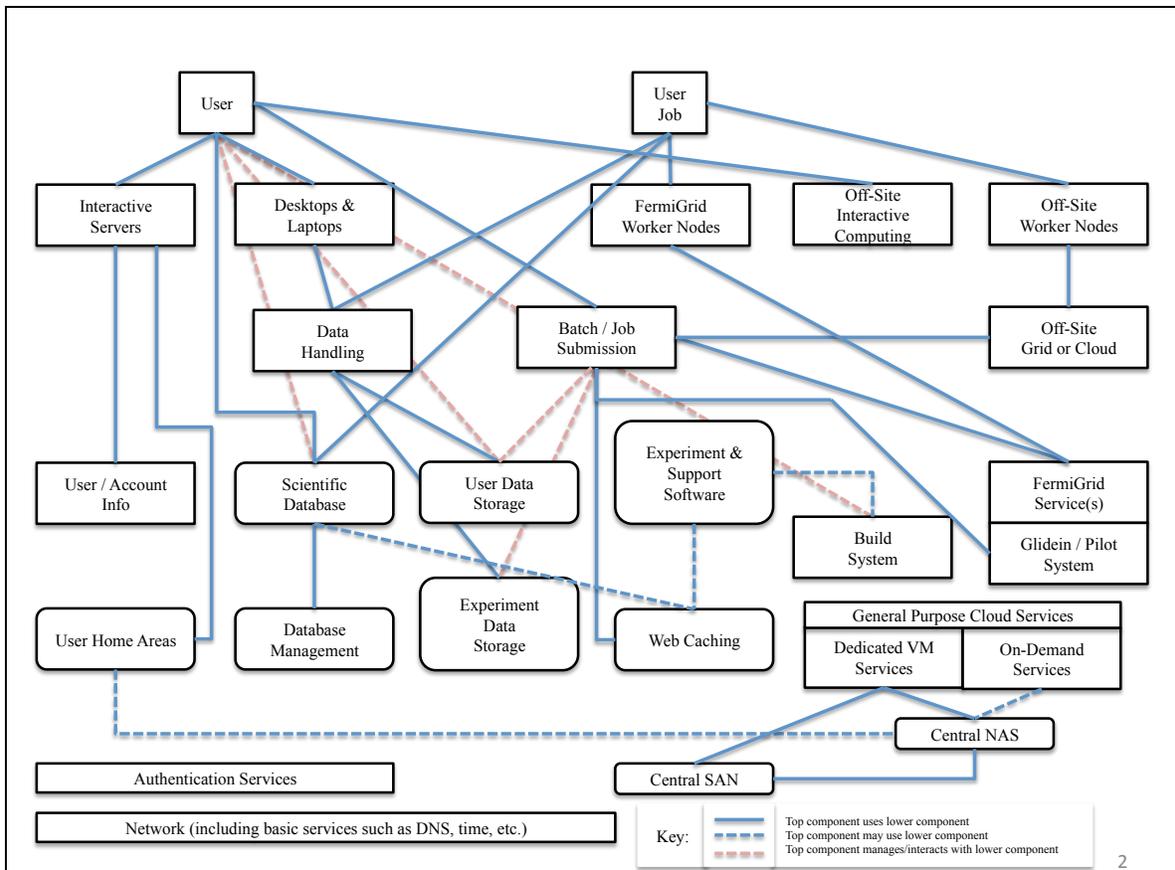


Figure 2: Components of the FIFE architecture organized in layers and with relations of the high-level components shown. Components depend only on components in lower layers.

The overall architecture is network centric. The IP network with its core services provides the lowest layer and binds most components. Scientific data flows over the network to the interactive and batch nodes that process and analyze the data. Virtual Machine, VM, cloud, storage and database services are in the second layer. Most services will be VM based (and test instances of them cloud based) in the future. The central VM service for scientific computing needs to be extremely robust and should have multiple partitions (within the same cluster or multiple independent instances) for the various service groups². The storage and databases are less likely to become virtualized (except for administrative nodes, testing instances, archival, etc.) and, if virtualized, will likely use dedicated, vendor-recommended virtualization, i.e. internal to the component. The main scientific services are realized in the next two layers: user information and scientific database management services, software support, data handling and batch/job submission

² Multiple partitions means locating independent services on different hardware, i.e. different physical servers and SAN storage, so they stay independent. A failure of a server impacts then only services of one group. The GPCF and CDF VM clusters are one current such example.

service. Interactive and batch computing will be the customers of the scientific services. With the IP network being the mortar/glue of the architecture, the inclusion of resources located off-site becomes natural. For this, the component interfaces need to be designed with the Wide Area Network, WAN, in mind and to be scalable to high load and high latency, i.e. need to be Grid ready.

3.1 Network

The current IP network at Fermilab is of modern modular design with core, distribution and access layers. The data center module in the distribution layer provides the backbone for the scientific data flow in the six computer rooms of the Feynman, Grid and Lattice Computing Centers. It is currently implemented via four interconnected Nexus 7010 switches. Most access layer switches uplink to two of the Nexus. The setup has been robust and reliable. The network bandwidth was specified for the final stage of Tevatron Run II experiments, i.e. has sufficient capacity to handle IF and CF needs over the next few years. The architecture scales with Ethernet physical layer technology upgrades.

Locating dependent components/services in close proximity on the network is a good practice. This includes both physical, i.e. connection on the same switch, and address space location, i.e. same Virtual Local Area Network, VLAN. However, at Fermilab considerations of space, cooling and electrical power in the computer rooms may take precedence over reduced network dependency when deciding on its physical location.

Similar proximity considerations apply to network services (VLAN routers, Domain Name Service, DNS, time service, etc.) or their secondary/slave servers.

3.2 Authentication

In the late 90s Fermilab decided to use Kerberos to protect access to network services. The authentication service is still used for all interactive access. For non-interactive access public key certificates are also used and the de-facto standard to access Grid resources.

Public key certificate based authentication should expand further. New services, access methods and protocols developed in the Fermilab Computing Sector should include public key certificate access. With Kerberos tickets easily convertible into X509 certificates (signed by the Key Distribution Center, KDC) there appears to be no need to change the current setup.

User requirements call for a single sign-on. With the login service at Fermilab being Kerberos based, the client commands and utilities need to accept Kerberos tickets and should derive public key certificates as needed.

3.3 Central SAN

The central storage area network is very robust and reliable, like the IP network. It is a simple fabric with two redundant Fibre Channel (FC) switches at the core. The setup should be kept simple³.

The infrastructure, i.e. the fabric/FC switches, should be sized generously to avoid any bottlenecks as it is shared by all components and services using the central SAN and thus brings unwanted coupling. On the storage side, i.e. the disk/RAID arrays, independent components/services should be kept separate and should use different storage hardware. The extra money that this may cost is well worth the independence gained.

Similar proximity considerations as for the IP network apply.

3.4 Central NAS

The central network attached storage has been a great success and also the area of serious problems. The central NAS provides filesystem-based storage to other machines on the network. It uses the central SAN for block-level storage. Use of a standardized network filesystem enables distributed computing and a heterogeneous computing environment. This is important for scalability and allows implementation changes within the overall architecture.

The storage of scientific data and storage necessary to host services⁴ should not be mixed. The scientific data should be off-loaded from the central NAS. This is likely to be a multi-year project. Neutrino experiments that currently collect and analyze data will not want to switch to a data handling system with different data access protocols. Services that don't require shared access to a common area should be decoupled from the central NAS. The use (or decoupling) of central NAS in each of the components and services will be discussed below together with each component/service.

3.5 General-Purpose Cloud Services

The cloud service in this document means infrastructure as a service, i.e. the most basic cloud services. The cloud service should also be used to provision shared Grid worker node resources in case experiments require incompatible configurations.

³ If a feature, connectivity or other needs would increase complexity of the central SAN one (or more) additional instances should be considered.

⁴ This includes interactive services to develop and test software, submit jobs, manage an experiment, analyze data (other than the data itself) and publish results.

The setup of the cloud service at Fermilab should follow industry lead on both hardware and software side. Application storage space should come from the central NAS (even if provided for block access), providing that the central NAS is sufficiently robust and redundant. Using the leading cloud interfaces will facilitate the use of commercial cloud services in the future.

Fermilab should complete development of an on-site cloud computing strategy, specifying both level and quality of service, so that the service can be engineered and used appropriately. The FermiCloud project is an encouraging first step toward cloud technology inside Fermilab.

Both VM and Cloud service provision virtual machines. There is an opportunity to share management tools. In the future, VM services could also be realized as a special instance or partition of the cloud service. At the present time, critical services should not be deployed in (public) cloud infrastructures but possibly in the future. Similar separate instance consideration as for VM service applies.

3.6 Dedicated VM Services

Server virtualization decouples the application host machine from the physical hardware. It allows isolating applications and/or to improve resource utilization, and reduces the number of physical servers. I.e., it leads to a greener computing environment, increases uptime and provides good support for disaster recovery. Fermilab has used server virtualization for many years. The SCD operates several VM clusters including GPCF and FermiGrid services.

Virtualization shall be used by all services that don't require large CPU / bandwidth / memory resources or tight resource control. A disadvantage is that virtualization introduced a new coupling of components/services. To minimize any adverse impact, the VM service must be very robust. Multiple partitions (within the same cluster or multiple independent instances) should be used to isolate unrelated and unconnected components/services. Disk space needed by the applications that require only local access, especially space for running services, should be based on the central SAN. This will provide performance equivalent to the local disks of a server and will also not couple the service internally to the IP network. The SAN storage, i.e. the RAID unit(s), should be separate for each partition/instance.

3.7 On-Demand Services

On-demand services are a platform to enable the dynamic instantiation of scientific services (data transfer, worker nodes, batch systems, etc.) depending on the resource needs. This elasticity of service deployment is typically built on top of Infrastructure as a Service (IaaS), see section 3.6 above, whereby IaaS provides on-demand resource provisioning and the layer above the coordination of the services to present themselves and function as an ensemble. Such on-demand services layer can be implemented as a Platform-as-a-Service (PaaS) or as Software-as-a-Service

(SaaS) layers on top of an existing Infrastructure-as-a-Service layer. Alternate names for dynamically deployed worker node services include Grid-bursting and Cloud-bursting depending on the mechanisms utilized and the interfaces presented to the user.

On-demand services would require integration with other services, such as job management, data management, etc., in order to project the need for resources and trigger their provisioning through IaaS. Such layer becomes particularly relevant in two cases: (1) at times of resource constraints, when static allocation of resources to dedicated services becomes cost prohibitive; (2) to automate the management of scientific services deployment, when manual dynamical allocation of services to resources become labor intensive.

3.7 Off-site Grid / Cloud Services

There are many Grid sites around the world. Most computing resources of the particle physics community are available in Grid facilities. Universities participating in a Fermilab IF or CF experiment may want to dedicate some of their Grid resources to FIFE. Most Grid sites allow opportunistic use when the primary application is not running. Computing clouds can also be used to provision additional Grid worker nodes on a temporary basis.

Off-site Grid facilities, direct and cloud based, academic and commercial, will become an ever more important component of FIFE resources.

The off-site resources could contribute close to half the CPU resources within a few years. Therefore the scientific applications and services need to be Grid ready.

3.8 User Home Areas

The user home areas provide login space for the collaboration members and support people to store source code, configuration files, analysis information and documentation. The same area should be available on all interactive servers (and desktops). Experience has shown that one home area solution for all IF and CF experiments is appropriate. While a separate, independent area for each experiment would reduce experiment interconnection, the connection is considered small and most relevant when coordinating maintenance. A significant fraction of users will also be in more than one experiment and want a unified home area. In case experiments are grouped, multiple areas could become appropriate. At the present time, the experiments have not presented any requirements that would make sharing of a common home area undesirable.

The user home areas should support a distributed, heterogeneous, changing environment. The implementation must support user and also experiment access protections and user quotas. Those should be used and set for each user. The area should be backed up daily (or at least a filesystem snapshot taken). At the present

time the AFS based user home areas are currently accessible off-site. At the present time neither the FIFE architecture nor the experiments require this.

3.9 User/Account Information

At Fermilab, the computer username of a person is allocated centrally, and is assigned a numeric user identification, UID, and recorded in a database. The database entries are used to create user accounts on the computers, i.e. the password records on UNIX type operating systems. The records, however, contain no password information as authentication is provided via Kerberos. Experiment membership is set up at each computer and for the Grid in the Virtual Organization Membership Service, VOMS, of the experiment or in the experiment group of the Fermilab VOMS.

Experiment membership on the computers are not well managed and often a relic of a user's first experiment.

Fermilab should upgrade its central user/account information system to include experiment membership information. The system should allow experiments to manage this membership directly. VOMS information should be based on the central user/account information system. User home area location (based on cluster/home area service) should be integrated. Renewal in case of expiration is an important aspect that should be well planned and implemented!

The user/account information should be served to computers on-site (and potentially trusted ones off-site) via a distributed, tiered service to achieve the required scalability and robustness.

3.10 Database Management Service

The detector geometry, alignment, calibration and condition data of the experiments need to be stored, potentially updated, managed and accessed by the data processing, simulation and/or analysis jobs. Metadata and data flow status⁵ information from the data handling system have similar needs.

A database management system, DBMS, with support for transaction processing, live snapshots, backup and replication is used by all larger experiments.

The database management service currently uses the central SAN for storage and is implemented using both commercial and freeware DBMSs.

3.11 Scientific Database Service

The experiments need to read the geometry, alignment, calibration and conditions data to simulate, reconstruct and analyze the scientific data. With many such jobs running simultaneously on the Grid and each potentially being split into many

⁵ In the Sequential Access Model, SAM, data handling system this would be the project status information.

sections executing in parallel, direct database access is not an option⁶. Instead, a tiered service for read access is needed. The service should provide one-to-many distribution and throttle peak access before it reaches the database. Limited manpower and expertise of the IF and CF experiments makes it unrealistic to expect from the experiments to design and implement their database schema and/or program efficiently in Structured Query Language, SQL. Middleware providing an abstraction layer is needed. Such a layer would also decouple experiments from the DBMS used in the database management service, i.e. allow implementation switches without impacting the experiment's offline software. The database information is commonly accessed at the beginning of a job or a processing segment.

Especially during analysis, the processing segments can be small and the on-demand database access can slow down execution if reading from the database has a significant latency [7]. When jobs run at off-site Grid facilities, accessing a Fermilab resident database can lead to large inefficiencies. Caching of the database information can help here if the lifetime of the database information is known a priori and large compared to the time between accesses.

Database abstraction and tiering should be provided by a scientific database service. The SCD should provide this middleware layer to the experiments. It is understood that the middleware may contain custom written code for each experiment but expects very substantial sharing, i.e. this being a worthwhile investment in the direction of integrated services.

The experiments at Fermilab currently use services based on Apache/WSGI, CherryPy, Tomcat/Java Servlets [8] and the Frontier Distributed Database Caching system [9] for offline database access. All use the HyperText Transfer Protocol (HTTP), and have a Representational State Transfer (REST) style architecture.

The scientific database service will consist mainly of designing and building the offline database access (and perhaps also the high level trigger access) of the experiments and operating the tiered database access system (or the tier 0 of the system if the web cache service is used for the higher tiers).

3.12 Experiment Data Storage

The scientific data of the experiments need to be stored for later processing and analysis, and preserved for future research. The data of an experiment is not limited to the raw detector information acquired by the online data acquisition system and the data of reconstructed physics quantities, but includes all simulated, processed, selected or derived data generated by the experiment or one of its organizational

⁶ Direct database access might be feasible for an individual smaller experiment with well-organized data and analysis. However, with the database management service shared among the experiments and potentially other customers it will lead to overloads as encountered by experiments that started with direct database access.

units. The storage system for the scientific data of experiments consists commonly of a mass storage system and a disk cache component. The experiment data storage and worker nodes component provide the computing resources for the data processing and analysis. The two components need to be the most scalable. The experiment data storage should be decentralized, i.e. have distributed components for receiving data for storage, serving stored data back to users and migrating data between different storage media, to provide the necessary scalability and fault tolerance.

The users are not expected to interact directly with the experiment data storage system. Instead the data will be stored and retrieved via the intermediate handling component. The direct access to the storage system will be reserved for administration and data management.

The experimental data storage system does not depend on components other than networking and authentication (outside the experimental data storage component itself). It should be kept this way.

3.13 User Data Storage

The experiments typically divide their data into multiple primary datasets, either at the trigger level or after processing. These datasets are often large; accessing them is slow and only a small subset is needed for an analysis. Physicists select a secondary dataset applying rough selection criteria, study it, and select refined subsets from it until the selected dataset is small enough to analyze quickly. The selection process is paramount for effective analysis. Physicists need a convenient way to store those secondary and tertiary datasets, i.e. have access to user data storage/project areas. The difference of user data storage to experiment data storage is two fold: (1) most user data are of limited, short lifetime while experiment data is normally persistent⁷ and (2) any decision about storage and removal is made by an individual user (or small group).

The data rates from the user data storage system are expected to be significantly smaller than from the experimental data storage system. The user programs are not expected to interact directly with the user data storage system but via the data handling component. For development, debugging and data management, the users may occasionally interact directly with the system.

3.14 Data Handling

The Tevatron Run II and LHC experiments use a middleware layer to store, manage and access their data. The layer allows users to deal with physics datasets instead of lists of files. It eliminates the need to specify physical location and access protocol

⁷ Processed experiment data is superseded infrequently by newer versions of processed data. Experiments often recycle the storage media containing old versions of processed data.

for files inside programs and scripts that can now use logical names and thus become storage service (and thus site) independent. The layer also enables consistent error handling, retry, and failover to an alternate service or switching to a better access protocol.

Metadata of both user and experiment data should be stored and managed in a data-handling component. The software framework should be augmented to make this convenient for individual users with frequent and repeated selections. The data handling component will allow experiments to store metadata for files, group files into datasets, associate metadata with such datasets, translate datasets into file lists, and translate filenames into physical location and access information. It will interact with the user and experiment data storage systems to access files.

The experiments do not plan to distribute or replicate a significant amount of data to off-site locations at this time. We expect home institutions to store selections, though. Off-site storage locations need to be supported.

3.15 Processing and Analysis Framework

Processing and early-state analysis programs consist of experiment and user specific modules, classes and functions, and common software. The framework integrates the various software components and controls their execution. The software is written in a high-level computing language. This provides a good balance between performance and productivity.

The subsequent analysis is shifted toward managed or interpreted programming languages and scripting. A different analysis framework is used for this. Both frameworks can read the scientific data and should have interfaces to the scientific database and data handling service.

The frameworks should foresee efficient multi-core support that is easy to use by non-expert programmers.

3.16 Experiment and Support Software Access

Experiment and support software like the data handling package, scientific database software, etc. need to be accessible on the machines executing the processing/analysis jobs and machines used for development and management. Most of the data processing and analysis is based on tested, released and stable software sets. The release intervals vary between days to several months. For development, more unsettled software sets are used. These sets are commonly built nightly.

There is a large and ever increasing number of worker nodes. A distributed service is required to provide access to this software. Worker nodes are geographically

dispersed and the service should thus be tiered to handle the high latency WAN without inefficiency. The web caching service can be used for tiering.

3.17 Software Build System

The data processing and analysis jobs use executables and shared libraries. Those are either prebuilt by the experiment or custom built by users. The compilation and linking of the high level computing language into machine code is CPU and I/O intensive and if done sequentially can take hours.

A dedicated build system should be deployed for each experiment to assemble and build their software releases. There are significant CPU and time savings between incremental builds and full builds. In case of a build failure, the software management team of the experiment will need to examine the detailed results and logs of the build attempt. It is thus best to preserve experiment build areas and not to attempt to reuse them among experiments. The nightly development builds are expected to be a more constant load and the system must be specified for it. The building of stable releases will be a less-predictable, fluctuating and likely daytime activity. Separate SAN storage (due to the significant I/O) and dedicated physical servers (due to the high interface utilization) for the VMs (to time share between experiments and for OS flexibility) should be used.

Planning of the software build system has started recently.

The build releases will be provided to the experiment and support software component for use on interactive and batch machines.

User software builds will mostly be partial builds and done on the interactive machines.

3.18 Web Caching

Many services need a tiered architecture to distribute static information to worker nodes for cost-effective scaling to thousands of clients and to avoid inefficiencies from the high-latency WAN networking. If the lifetime of the information is known, it can be cached⁸ and this off-loads the server further.

The services that need tiering and/or caching but don't have this feature built in have two choices: (1) develop and build a dedicated tiered distribution layer or (2) develop an HTTP protocol implementation and use the web cache service at each site to distribute static information. Based on several factors (availability,

⁸ Caching here means caching for hours and days as oppose to the storing during tiered distribution, which we assume, is in the range of seconds.

performance, scaleability, etc.) the latter solution is recommended. The web cache service has to be very robust as it serves multiple services. The lower levels should also be over-proportioned to sustain failures and fall-throughs to them.

The service providers will interact with the web caching system to manage cached data of their service.

3.19 FermiGrid

FermiGrid, aka the Fermilab Campus Grid, is the umbrella term used for the Grid site services of Fermilab and the Grid compute clusters of Fermilab. In addition to the GP Grid cluster there are CMS, CDF and D0 Grid clusters. All of the clusters use Open Science Grid (OSG) middleware. A Grid cluster, a.k.a. a Grid compute element, is a set of services that provides access for Grid jobs to the local batch system running on a cluster of worker nodes.

At the present time, the FermiGrid shared site and experiment services are implemented using the FermiGrid-HA2⁹ infrastructure, and are listed in Table 1 below:

Table 1 – FermiGrid Shared Services		
Service	HA	Description
VOMS	Active-Active	Virtual Organization Management Service
GUMS	Active-Active	Grid User Mapping Service
SAZ	Active-Active	Site AuthoriZation Service
Squid	Active-Active	Web caching service
CrlSquid	Active-Active	A separate instance of squid to manage caching of x.509 certificate revocation lists (CRLs)
MyProxy	Active-Standby	
Database	Active-Active	Circularly replicating MySQL databases that support the operation of the VOMS, GUMS and SAZ services.

The shared site and experiment services are operated in a highly available configuration¹⁰ with copies deployed in the FCC-2 and GCC-B computer rooms and are supported under a 24x7 level SLA.

The individual cluster FermiGrid services are listed in Table 2 below:

⁹ <http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=3739>

¹⁰ There is no connection between the deployment of the FermiGrid shared site and experiment services and the BlueArc central NAS service.

Table 2 – Individual Cluster Services		
Service	HA	Description
Globus Gatekeeper	No	Low level job submission interface between Grid clients and the underlying batch system
GridFTP	No	File Transfer Protocol interface between Grid storage accesses and the underlying storage system
HTCondor Batch Masters	Active-Active	The HTCondor batch system ¹¹
PBS Batch Master(s)	None	The Torque batch system ¹²
Worker Nodes	No	Worker nodes associated with a particular Grid cluster. Typically the number of batch job slots on any given worker node will correspond to the number of physical cores present on the worker node hardware.

If particular services support a highly available configuration, then copies of the service are deployed in the FCC-2 and GCC-B computer rooms, otherwise the services are distributed across the various computing sector computer rooms to allow for reduced capacity operation in the event of an outage¹³. The individual cluster services are operated under a 9x5 level SLA.

3.20 FermiGrid Worker Nodes

The worker nodes provide all of the CPU for the data processing and analysis jobs. They belong to the Grid facility, i.e. FermiGrid or the off-site Grid service. We call them out here because of their importance and also because of the architectural change involving them.

FermiGrid worker nodes should not use the central NAS service and to have no tight coupling to other components except FermiGrid. Jobs on worker nodes should be kept isolated as much as possible. Virtualizing worker nodes is not envisioned (except in the case of experiments requiring incompatible configurations as described in Sec. 3.6). Grid middleware provisioning on-demand VMs on worker nodes would allow better isolation of jobs and provide more flexibility during OS upgrades. If this becomes available in the future it should be considered.

¹¹ <http://research.cs.wisc.edu/htcondor/>

¹² <http://www.adaptivecomputing.com/products/open-source/torque/>

¹³ At the present time, the individual Grid cluster Globus Gatekeeper services are implemented via shared home areas on the BlueArc central NAS, there is ongoing work to eliminate this connection.

3.21 Pilot Based / Glide-In System

The use of diverse Grid compute resources at many independent sites, which utilize various batch systems, brings complexity to job scheduling and monitoring. As such it is not advisable to have the user interact with the Grid site batch systems directly. A pilot based submission system should be used for FIFE job routing and distribution. One or more glide-in systems will submit pilot jobs to the various Grid sites. When scheduled, the pilot job validates the computing environment on the worker node and presents the resource to the FIFE batch system that schedules a user job there. This allows applying experiment resource quotas and user priorities uniformly across all Grid sites.

3.22 Batch / Job Submission

Users will interact with a batch/job submission service (jobsub)¹⁵ to run jobs on the computing Grid. The submission service accepts jobs from the users, prioritizes and schedules them, handles job configuration input files, (i.e. parameter files, scripts, executables, etc.) applies resource quotas and throttling in case of congestion, matches the job to a suitable worker node for execution, initiates experiment and data handling specific initializations for the job, provides end-to-end job monitoring, and transfers output log and status files¹⁰ back to the user. Experiments are recommended to use the jobsub service rather than directly submitting jobs to the Grid.

The batch submission system uses the pilot based glide-in system to acquire computing resources on FermiGrid and off-site Grid facilities. It will use the web caching service to distribute job configuration input files and will interact with data handling and/or data storage services.

3.23 Interactive Servers

Most physicists' time is spent on interactive computing, to develop software, assemble jobs, visualize, analyze and manage data. The location of interactive computing is driven mainly by network latency and support of computing platforms by experiments/applications. CPU is no longer a limiting factor.

The current approach of providing experiment-specific servers for interactive work is a good strategy. The computing environment on the interactive servers should match closely the environment encountered by batch jobs. Interactive servers will use the user home areas, batch submission and central SAN for scratch space.

¹⁵ Jobsub is the Grid computing interface provided by the Fermilab Computing Sector to the various Frontier experiments. Jobsub includes user-friendly tools that are designed to manage the details of the underlying Grid interfaces.

Experiment and support software, data handling and scientific database¹⁶ access should all be identical to that of FermiGrid worker nodes. Access to the data storage components for management and central NAS for special interactive areas¹⁷ is needed.

3.24 Desktops, Laptops and Mobile Devices

Desktop computing is playing a lesser role and its CPU contribution is no longer a vital analysis contribution. An important consideration is connectivity of the physicist together with support for mobility. As desktop usage shifted to laptops during the past years, usage is expected to shift to personal mobile devices in the future. Thin clients and multi-platform support for interactive tools will be paramount for productivity.

Desktops and laptops will use the experiment and support software, scientific database, data handling, batch submission, data-storage and user home areas.

3.25 Off-Site Interactive Computers

Off-site interactive computing consists of servers and desktops at home institutions, laptops and personal mobile devices anywhere in the world. The computers should access experiment and support software, the batch system, scientific databases and data handling in the same way as on-site computers. For some devices and locations (without a web cache service) interactive work may be limited to reduced functionality.

Good interactive working conditions at home institutions are paramount for successful collaborative data analysis work. Fermilab needs to support its computing platform for off-site installation in both physical and virtual machines.

3.26 User Job

User and experiment event generation, detector simulation, data processing, selection and analysis jobs will execute on worker nodes and occasionally for debugging on interactive machines. The jobs will use mainly the CPU of the worker node, the scientific database, the data handling, and the experiment and support software components.

3.27 User Access to Data

Scientists, engineers and administrators will use the interactive servers, desktops, laptops and off-site computers to manage experiment data, develop software, and process and analyze data. They will mainly use user home areas, the experiment and

¹⁶ Scientific data should be handled through the data-handling component and will not be transferred by the batch submission system.

¹⁷ Experiment management areas, user web areas, etc. are examples of such interactive areas.

support software, batch submission, data-handling and scientific database components. Some users will also interact with the build system and the two data storage components.

4. A Conceptual Design

The architecture described above can be implemented with current technology. Many components exist already in the current FIFE computing setup. The conceptual design provided here is a possible implementation of the architecture.

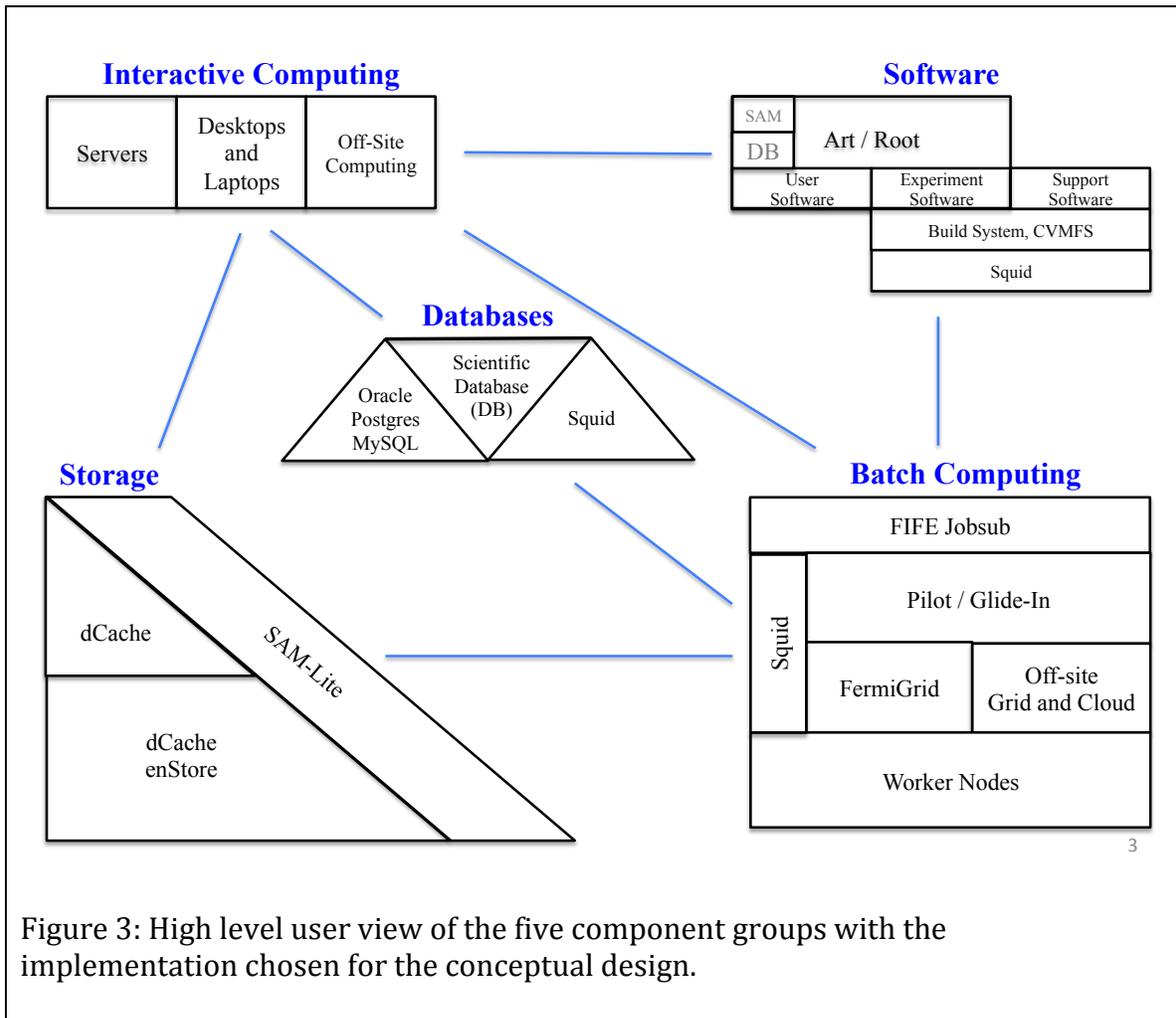


Figure 3: High level user view of the five component groups with the implementation chosen for the conceptual design.

Figure 3 shows the implementations selected for the major components for the conceptual design grouped into the same five component groups as in Fig. 1. The overall design considerations do not impact only the architecture but also the implementation.

For some components this is more important than for others. While the conceptual design shows one possible implementation, some features/details are paramount and should be present in any chosen implementation of the component.

The most significant changes with respect to the current setup are in the area of scientific data handling. User and experiment data are no longer stored on the central NAS and accessed directly from the large number of worker nodes. They are catalogued, managed and accessed via the data-handling layer. Services will be made more independent and decoupled from each other, especially from the central NAS.

To achieve the required scalability (after the bottleneck of scientific data access from the central NAS is eliminated) several services will be upgraded to a tiered read access.

4.1 Central NAS

The central NAS currently serves many important functions for FIFE experiments and FermiGrid: It holds and serves experiment and user data; it is used to distribute job configuration input files to the FermiGrid worker nodes; it is used to collect job output log and status files; it serves experiment and support software to interactive and FermiGrid worker nodes; it has the FermiGrid home areas; and it provides Grid application space for the non-Fermilab experiment. Two BlueArc Titan instances, one for core-IT applications and one for scientific applications, make the central NAS. All of FIFE and FermiGrid filesystems are on the scientific instance.

In the conceptual design the BlueArc setup continues to provide the central NAS. However, the functionality of the central NAS changes as outlined in the architecture section above to disentangle components. User and experiment scientific data will be off-loaded as described in the sections below.

Job configuration input files will be collected in user or scratch space on the interactive machines and handed to the batch/job submission system (that will distribute them to the worker nodes via the tiered web cache setup). Job output log and status files will be collected as part of the batch/job submission system and returned to the user. Experiment and support software will be accessed via a caching system based on the web-caching component as described below:

- FermiGrid home areas for batch jobs should be migrated to local worker node disk space (this was implemented in July 2013).
- Grid application space for Fermilab experiments/VOs should be encouraged to migrate to CVMFS.

- Grid application space for non-Fermilab experiments/VOs should be phased out, and they should be encouraged to utilize the OSG provided OASIS service.

In the conceptual design the central NAS will provide space for a cloud service, the user home areas and experiment management areas. The machines providing cloud service and the interactive machines using the user home and experiment management areas will have network interfaces of similar I/O bandwidth as the central NAS. It is thus important to have host or user based I/O quotas or scheduling priority to avoid unintended overloads from a single application.

The central NAS should always use a standard, widely supported protocol like NFS or CIFS to allow a heterogeneous computing environment and easy use/migration to new computing platforms.

4.2 Dedicated VM Service

In the conceptual design all components that don't require large CPU, memory, or network bandwidth resources or tight control over them will use the VM service. The VM service for central services (i.e. user/account information, experiment and support software, build system, FermiGrid site and experiment services, pilot/glide-in system, data handling, batch submission and interactive servers) uses SAN based storage for both OS and application filesystems. For FermiGrid farm services (i.e. HTcondor services, squid, etc.) a VM service in close proximity to the worker nodes, using local storage, is desirable.

4.3 User Home Areas

The current home area implementation is based on AFS, the filesystem of the Andrew Project. AFS support is declining and Fermilab plans to phase out its AFS service. The most natural replacement would be to base the home areas on the central NAS. Given the importance of interactive work the home areas should not be hosted on the central NAS instance (or at least server) that serves scientific data or is mounted on FermiGrid worker nodes. Using the core-IT central NAS instance or freeing up one of the servers/heads of the scientific central NAS are then the options.

The home areas will be NFS mounted on interactive servers and other trusted hosts using protocol v3. Desktop personal computers could also get access but via protocol v4 which provides additional authentication and security. Off-site machines will not have access.

Implementations that provide host and/or user based I/O quotas or scheduling priority should be considered for future upgrades when available. Interactive servers will be VM based with the physical machines that host the VMs having a

network interface as capable as the user home area server, i.e. without I/O quotas the VMs could easily cause a server overload.

4.4 User/Account Information

Network Information Service, NIS, is the implementation used for user/account information service for most SCD clusters. NIS setups at various sites have been replaced due to stronger security and authentication requirements. With Fermilab having a separate, Kerberos based authentication service this is no concern here. In the conceptual design membership is managed in the central user/account information system and propagates from there to VOMS and cluster NISes. If this cannot be accomplished promptly (as this is outside SCD's responsibility), then SCD should consider enhancing the current NIS based setup for FIFE.

It is noted that account information services based on the Lightweight Directory Access Protocol, LDAP, are gaining traction.

4.5 Scientific Database Service

In the conceptual design scientific databases use the squid service for tiered read access of static information. The database interface for each experiment, current and future, is designed professionally to hide DBMS implementation and maximize cacheability.

4.6 Experiment Data Storage

In the conceptual design all the scientific data of the experiments are stored in Enstore [11] and accessed via dCache [12]. The general-purpose dCache and Enstore instances are used for this. Both have been running successfully for many years. To replace the central NAS with dCache the size of the disk cache and the number of pool servers needs to be upgraded. The Tevatron Run II and CMS experiences with the system have been very good. Both CDF and CMS experiments have production tested the system to the scalability that FIFE needs. The current dCache setup should be augmented with an xrootd layer, especially for off-site data access. Enhancing dCache to a more dynamic access throttling, as to its current static mover protocol limits, is desirable.

4.7 User Data Storage

The conceptual design uses a resilient dCache to implement the user data storage component. The experiment data storage uses dCache already and the additional resilient instance is thus the most simple and manpower/cost-effective implementation. User quotas are currently being implemented and are paramount to a successful, i.e. manageable and low-maintenance, setup.

The NFS v4 support of dcache allows the user data storage to be mounted directly and files accessed via standard Linux commands and APIs. The conceptual design makes use of this for interactive servers. This allows convenient data access for management, development and to test/debug new programs.

If NFS becomes the protocol of choice for the on-site data handling based access then worker nodes should be setup such that direct access without going through the data handling layer is blocked.

Without such a blocking FermiGrid worker nodes would become “special” again and the on-site versus off-site equality broken.

4.8 Data Handling

The implementation of the data handling component in the conceptual design uses SAM-Lite, a simplified version of SAM [13]. The SAM to dcache interface is currently being upgraded to include cache file inventory information and will then provide a more efficient data flow when processing a large number of files. Enhancing SAM to automatically select between dcache access protocols based on file read patterns and job location would be desirable. SAM services are mature and have been robust. With additional SAM-Lite development and more sharing among experiments it could be worthwhile to make the services redundant.

The ART framework has already a SAM interface. A SAM interface needs to be written for the Root framework. SAM-Lite should also be upgraded so frequently accessed static information can be read via HTTP, can be cached and can use the tiered squid service. The interface to declare and store new files, both for user and experiment storage, should be reviewed for convenience.

4.9 Experiment and Support Software Access

CERN developed a filesystem, CVMFS [14], as part of their virtual analysis environment for LHC data processing and analysis, CernVM [15]. For read-only, quasi-static filesystems CVMFS solves the problem that a very large number of clients/worker nodes present to a central file server. Experiment and support software is the ideal application for CVMFS and CVMFS the natural choice for the FIFE implementation of it. The first IF experiments have already begun to test it. CVMFS will use the squid-based web caching service described below.

4.10 Software Build System

The most simple, cost-effective and yet upgradable implementation for the build system would be one (or more) large multi-core server(s) with virtual machines for each platform and experiment, dedicated storage on the central SAN and scheduled nightly build slots. The VMs should have many cores and large memory to allow for quick, O(15 min), software builds. The server(s) hosting the VMs should have two or

more times as many cores as the largest VM. The machine(s) need to be dedicated as the FC interface to the SAN is expected to be quite busy. The server(s) may be heavily over-subscribed. This is not expected to be a problem since:

1. Nightly rebuilds can be scheduled with specific time slots,
2. Stable releases are built less predictably but also less frequently during the day,
3. The server could support two or more builds simultaneously,
4. Any overload would not be fatal but result in slower builds,
5. New features available in virtualization packages allow the amount of cores and memory per virtual machine to be dynamically allocated without rebooting the virtual machine, this could allow various virtual machines to coexist in the same virtual hardware without overbooking.

The SAN storage will provide the good I/O performance required and allow the storage to be persistent, i.e. enable builds to be faster update-builds. The final step of the software build is publishing the version to CVMFS.

A first prototype software build system was recently setup for NOvA.

4.11 Web Caching

The web cache can be implemented easily with existing and additional squid servers deployed as part of the existing highly available FermiGrid squid service. A second level hierarchy of squid servers closely associated with individual Grid clusters could also be deployed if performance so dictates.

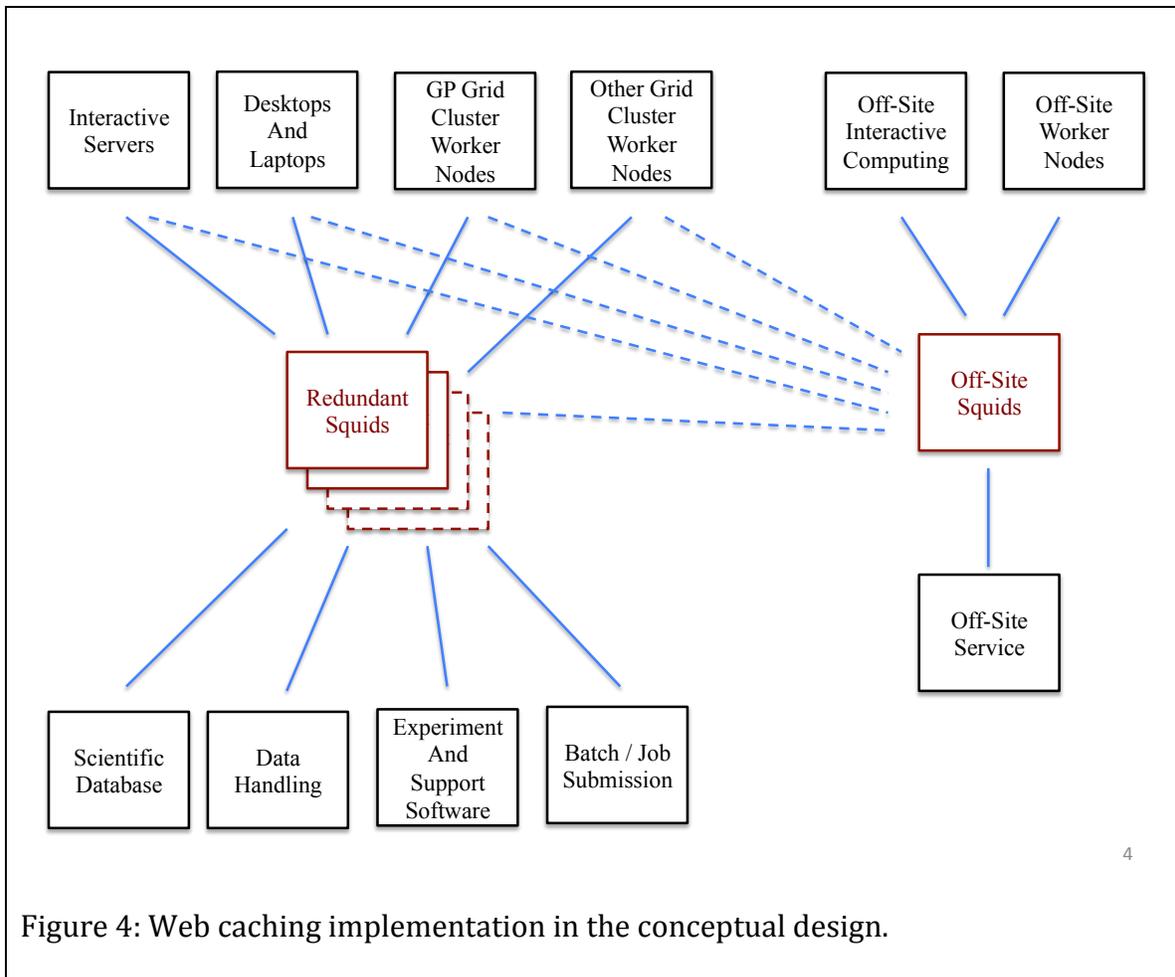


Figure 4: Web caching implementation in the conceptual design.

4.12 FermiGrid

In the conceptual design the FermiGrid site and experiment services remain virtualized and continue their redundant deployments. Non-Fermilab experiments/VOs will be encouraged to use CVMFS service provided by OSG and application space on the central NAS phased out¹⁹.

The individual Grid clusters will continue to be free-standing with all the cluster specific services in close proximity to the worker nodes. The GP Grid cluster should provide single-core slots and multi-core slots to enable and encourage multi-threaded programming by experiments.

Multiple GP Grid clusters would allow for less-interruptive upgrades and configuration changes.

¹⁹ If there are agreements or other reasons preventing application space from being abolished, it should become a FermiGrid internal service so there are no central NAS mounts on worker nodes and any overloads by worker nodes are contained.

4.13 FermiGrid Worker Nodes

GP-farm worker nodes have no more central NAS mounts. They have separate, standardized, local scratch space for each job, local CVMFS cache, use assigned UIDs and correct GIDs for jobs or UID pools for unregistered users.

4.14 Pilot Based / Glide-In System

The conceptual design uses the glide-in WMS [16] system based on HTCondor [17] to route jobs to appropriate, available compute resources. OSG operates glide-in WMS factories in two locations, the University of California at San Diego, UCSD, and Indiana University, IU. Using them to access off-site Grid resources will save operations and support effort. A redundant glide-in WMS factory service at Fermilab is needed for robustness, considering the large on-site computing resources²⁰. In addition to the factory, glide-in WMS front-ends are required for each experiment/VO. A redundant service should be setup with FIFE experiment/VOs sharing the front-end setup.

4.15 Batch Submission

The batch submission system in the conceptual design uses HTCondor [17] with a client-server wrapper [18] for user convenience and to hide condor implementation details. The jobsub wrappers of the IF experiments are currently being rewritten to add requested features (like saving and transferring input configuration files to worker nodes and handling output log and status files) and to reorganize it into an easy to install, OSG and condor independent command for users and a server that interacts with glide-in WMS and Grid sites/middleware.

FIFE jobsub supports common workflows of particle physics and has an interface to SAM. An interface to dCache should be added so disk cache congestion can be considered in the scheduling.

Buffer space for output log and status files should be set up.

²⁰ Fermilab may want to negotiate with OSG to operate the on-site glide-in WMS factories as part of the Grid Operations Center.

4.16 Interactive Servers

The current GPCF experiment-specific servers work well for the experiments. In the conceptual design users will build executables and assemble jobs on the interactive machines in “local” scratch space that comes from the central SAN. With the space no longer being shared among interactive servers of an experiment, it is thus desirable to have fewer and larger VMs per platform/OS version (versus the many small VMs in GPCF currently). This will simplify user login load balancing too. Scratch space should have user quotas and can be oversubscribed if regularly cleaned up.

A separate VLAN for interactive work (servers, user home areas, etc.) should be considered to allow easy prioritization/quality or service separation in the future.

4.17 Cloud and On-Demand Services

The implementation of on-demand services, dynamically instantiated and acting as an ensemble, depends on the characteristics of the specific services. The conceptual design of this capability will be developed by first implementing specific higher-priority on-demand services (e.g. data movement on demand), then abstracting supporting services for the on-demand function of potential reuse, such as a common registry service, log gathering, security context, etc.

For the case study of data movement on-demand, currently all the intensity frontier experiments run one gridFTP server with specific local privilege (UID/GID) mapping. This static allocation of resources (1 gridFTP server) is bound to encounter scalability issues as more and more IF communities move their computational activities to the OSG. On the other hand, computational campaigns on the Grid tend to be cyclic in nature, therefore engineering the number of gridFTP servers needed for the peak of the data transfer need is sub-optimal. On-demand services will enable the deployment of multiple gridFTP servers based on need, so that a contained pool of resources shared among the IF experiments will be able to be dynamically allocated depending on demand. The resources could be presented as an ensemble by registering each new instance of a gridFTP server to a common SRM portal. One could envision to run a similar on-demand service for other data transfer implementations, such as xrootd.

4.18 User Access to Data

Users will declare all scientific data to SAM.

5. Summary

The current FIFE compute system has so far allowed the experiments to execute the planned scientific program. The architecture has decoupled components, is modular and scalable. The architecture can be implemented by evolving the current implementation and with existing technologies. Migrating experiment data access to go through a data handling layer and setup of a high-performance distributed cache disk based storage component should have highest priority. SCD should design and write the database layer for experiments. Ongoing SAM-Lite and FIFE jobsub developments should be completed quickly and assistance provided to experiments to adopt them.

References

- [1] M. Kirby et al., FabrIc for Frontier Experiments, FIFE, <https://sharepoint.fnal.gov/org/scd/fife/SitePages/Home.aspx>
- [2] Grid computing federates computing resources in multiple locations via the Internet to work on a common problem as if they were a single powerful computer. F. Berman, G. Fox and A. J.G. Hey (editors), Grid Computing: Making the Global Infrastructure a Reality, John Wiley & Sons, (2003).
- [3] The Open Science Grid Consortium is an organization that develops software, provides services and operates a worldwide Grid of computing resources for scientific research, called the Open Science Grid, <http://www.opensciencegrid.org/>.
- [4] K. Chadwick et al., Continued User Overloads of Central NAS Service, CS-doc-4995 (2013), <http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=4995>.
- [5] B. Boroski et al., Scientific Computing Project Portfolio Management 2013 Review, <https://sharepoint.fnal.gov/project/sppm/SitePages/Home.aspx>
- [6] M. Kirby and St. Lammel, Requirements for the Batch Submission System for Frontier Experiments, CS-doc-5115 (2013), <http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=5115>.
- [7] I. Mandrichenko, FIFE Architecture and Scientific Databases, CS-doc-5156 (2013), <http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=5156>
- [8] I. Mandrichenko, Redundant Web Services Infrastructure for High Performance Data Access, CS-doc-5081 (2013), <http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=5081>.
- [9] The Frontier distributed database caching system distributes data from central databases to many clients around the world. The name comes from “N Tier” where N is any number and tiers are layers of locations of distribution. More information can be found at <https://cdcvs.fnal.gov/redmine/projects/fermitools/wiki/Frontier> and <http://frontier.cern.ch/>.
- [10] Squid is a proxy server and web cache service, <http://www.squid-cache.org/>.
- [11] Enstore is the mass storage system implemented at Fermilab as the primary data store for large data sets, <http://www-ccf.fnal.gov/enstore/>.
- [12] dCache is a system for storing and retrieving huge amounts of data, distributed among a large number of heterogenous server nodes, under a single virtual filesystem tree with a variety of standard access methods. More information can be found at <http://www.dcache.org/>.
- [13] The Sequential Access Model, SAM, is data handling software originally designed for Run II of the D0 experiment at the Fermilab Tevatron, <http://d0dbweb.fnal.gov/sam/>.
- [14] The CernVM File System, CVMFS, is a network file system based on HTTP and optimized to deliver experiment software in a fast, scalable, and reliable way, <http://cernvm.cern.ch/portal/filesystem>.
- [15] <http://cernvm.cern.ch/portal/>
- [16] <http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/>

[17] <http://research.cs.wisc.edu/htcondor/>

[18] D. Box and St. Lammel, Batch Submission System For Intensity Frontier Experiments, CS-doc-4789 (2012), <http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=4789>.

FIFE Architecture Committee Charge

Charge to review and amend the scientific computing system architecture for Frontier Experiments at Fermilab

2013-Feb-27 draft

Dear FIFE Architecture Committee,

A problem investigation was set up in December to understand the many overloads of our central Network Attached Storage, the BlueArc system, during the past year. One of the recommendations from the committee was to review and revise the architecture of the offline computing systems we currently provide to Frontier experiments. The current architecture was built from available components. Various components have been extended and adapted to meet experiment needs during the past five years, but the overall architecture has not been re-evaluated.

The sum of the hardware resource needs has now reached Run II scale. We are currently using solutions that are more costly than alternatives that are now available. Components of some systems have reached scalability limits. Third generation neutrino experiments, NOvA and MicroBooNE, are being commissioned; Intensity Frontier muon and fourth generation neutrino experiments are being built; and more Cosmic Frontier experiments are being supported by the laboratory. Therefore, now is an opportune time to review and re-architect the offline compute systems for those frontier experiments. The outcome will provide a foundation for our new activity to provide and support integrated offline systems - FabrIc for Frontier Experiments (FIFE) - whose goal is to provide collaborative scientific-data processing solutions for these experiments.

I would like the committee to review the current IF offline architecture and redesign it taking into account experience from IF, Run II and CMS. The new architecture should emphasize robustness, scalability, long-term viability and cost. Fermilab and the experiments must be able to migrate from the existing architecture in a cost-effective way.

I am asking you to provide a conceptual design of the scientific computing architecture that we want to migrate to. I anticipate migrating to this architecture during the next few years. The architecture should not focus on implementation but assume technology updates and changes during the lifetime of the FIFE project. (Where new technology can simplify scientific computing, and we should tightly follow a development, I expect you to point this out.)

The scope of the system architecture should include support for software development, data processing, simulation and analysis activities (excluding data acquisition, detector control and monitoring) at Fermilab or via a Fermilab front-

end, in the data center, on the desktop or laptop (i.e. include the use of off-site Grid and Cloud facilities and mobile client devices). The committee should foresee means to test system and component implementation and for the experiments where appropriate.

I ask the committee to complete its work by May 1st and document the proposed architecture. I know that this is tight, but we will rely on this to guide the FY-13 purchases.

Thank you for serving on this important committee!