

Chapter 14

Distributed Computing



Manuel Delfino

Distributed computing is an established discipline in computer science and engineering. It has evolved over the past 40 years to be one of the most important methodologies for implementing the data processing services needed by almost every activity in society. Distributed computing is still evolving rather rapidly, with major innovations introduced every few years. The aim of this chapter is to introduce the reader to the basic concepts of distributed computing in the context of particle physics, allowing a better understanding of what is behind the scenes when using distributed systems, and providing starting points in order to seek further information on the subject.

14.1 Usage by Particle Physics

Distributed computing, i.e. the coherent use of many computers to accomplish a given task, is extensively used in particle physics. It is particularly suited to simulating and analyzing data from particle collisions, where processing the data of one collision is largely independent from processing data of all other collisions and hence very little communication between the different computers is necessary. This type of distributed computing is called “High Throughput Computing” (HTC),¹ because there is a very large number of quasi-independent tasks to accomplish,

¹Livny, Miron, et al. “Mechanisms for high throughput computing.” *SPEEDUP journal* 11.1 (1997): 36–40.

M. Delfino (✉)

Departament de Física, Facultat de Ciències, Universitat Autònoma de Barcelona, Barcelona, Spain

e-mail: manuel.delfino@uab.cat

and the performance perceived by the user is the rate of completion of tasks, or “throughput”. This type of computing is complimentary to “High Performance Computing” (HPC), i.e. the execution of a single task at maximum speed on a classical supercomputer. The particular case of event or collision processing requiring little inter-processor communication is an example of “loosely coupled parallel computing” or even “embarrassingly parallel computing”.^{2,3} Certain kinds of accelerator simulations, particularly those that use ray-tracing techniques, can also be executed as distributed computing tasks. Multiple computer systems can also be interconnected in “tightly-bound” configurations via low-latency networks,⁴ essentially yielding a supercomputer. Tightly-bound systems are used in particle physics, for example, for solving equations from theories using numerical methods (e.g. lattice-gauge theories); they are mentioned here for completeness and will not be covered further.

From the decade of the 2010s, the boundary between HTC and HPC has started to blur, for a number of disparate reasons:

Multi-core Applications Detectors have become more complex and have more channels. This impacts the memory footprint of reconstruction, simulation and analysis programs, which has grown substantially. In parallel, processors have become multi-core⁵ with shared RAM.⁶ The growth in number of cores per processor has been faster than the drop in price of RAM chips, resulting in an effective memory shortage. This means that running an independent copy of the operating system and the application in each core is not economical. The way out is to implement the safe execution of parallel threads⁷ of a single copy of the program on each multi-core processor. This introduces a number of dependencies, for example competition for RAM and for input–output services, that break the loose coupling and make the applications behave more like HPC programs.

²Wilkinson, Barry, and Michael Allen. *Parallel programming*. Pearson India, 2004.

³Birritella, Mark S., et al. “Intel[®] Omni-path architecture: Enabling scalable, high performance fabrics.” *High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on*. IEEE, 2015.

⁴Pfister, Gregory F. “An introduction to the infiniband architecture.” *High Performance Mass Storage and Parallel I/O* 42 (2001): 617–632.

⁵A core corresponds to the Central Processing Unit (CPU) of a classical computer, which is able to execute a single stream of instructions. Technology allows a growing number of ever smaller transistors to be placed on a single chip. However, profiting from these large number of transistors in a single core design would require prohibitively complex processor designs and impossible to achieve clock speeds. The alternative is to populate the chips with many copies of the same processor core, which work independently except for sharing external connections, leading to multi-core processors. Gepner, Pawel, and Michal Filip Kowalik. “Multi-core processors: New way to achieve high system performance.” *Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006. International Symposium on*. IEEE, 2006.

⁶Random Access Memory, the external solid state memory used by a processor.

⁷A thread is a stream of processing instructions coming from a shared program image which has its own private data instances in processor hardware registers, instruction and data stack and caches, and RAM.

Supercomputers as High Core-Count Clusters The original design of supercomputers was based on a rather small number of the fastest single core processors available at the time, interconnected with custom very low latency links. In addition, the design usually included a much larger amount of RAM than in standard computers. The design of supercomputers has changed completely over the last 20 years.⁸ The main driver of this change is the difficulty in building processors with ever shorter clock cycles. The so-called “clock speed wall” has been hit, and practically all processors operate within a narrow range around a few GHz. Hence, the only way to make a supercomputer faster is through parallelism. Modern supercomputers are in fact huge clusters of relatively standard multi-core processors. Low latency links continue to be used and their hardware costs have become much lower. In addition, “latency hiding” techniques sometimes allow the use of standard networks, such as Ethernet. Finally, the amount of RAM per core does not differ much from standard computers. Hence, executing an HTC workload on a supercomputer targeted for HPC is no longer wasteful. At worst, only the low latency interconnect will be underutilized.

Increase of Workloads with Low Input–Output Particle physics has traditionally been a heavy user of HTC systems because they were the least expensive architecture for executing detector track reconstruction and analysis, applications with a relatively low CPU to input–output ratio. The needs for CPU for reconstruction and analysis have grown, however, as detectors have become much more complex. In parallel, the precision needed in the simulations has vastly increased. These high CPU, low input–output applications currently represent the largest computing demand of a modern particle physics detector. Hence, the global workload profile has moved closer to HPC in the last decades.

14.2 Functional Decomposition of a Distributed Computing Environment

It is useful to introduce a functional decomposition, or reference framework, to discuss distributed computing systems. The decomposition that has dominated particle physics data processing since the 1990s, is the “SHIFT Model”, introduced in the early 1990s by Robertson and collaborators.⁹ It will be described here with some updates of terminology. The basic assumption in the SHIFT framework is that a Local Area Network (LAN) can be built with enough capacity and flexibility so that the rest of the elements in the distributed environment can

⁸Xie, Xianghui, et al. “Evolution of supercomputers.” *Frontiers of Computer Science in China* 4.4 (2010): 428–436.

⁹J.P. Baud, et al. “SHIFT, the Scalable Heterogeneous Integrated Facility for HEP Computing”, *Proc. Conference on Computing in the High Energy Physics, CHEP91, Tsukuba, Japan, Universal Academic Press.*

communicate freely amongst themselves. Furthermore, the environment is loosely coupled and is therefore not sensitive to relatively long round-trip time for network messaging (large network latency). Connected to the network we have the following elements:

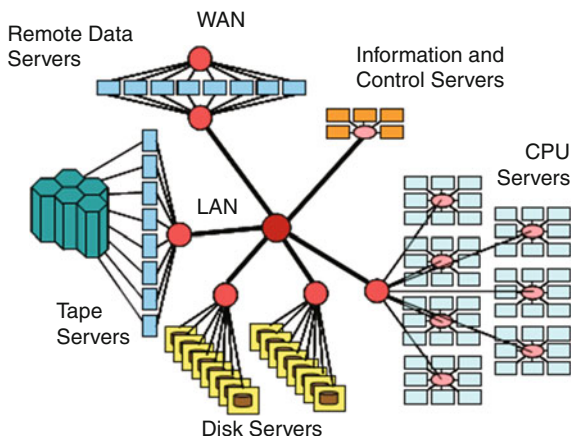
- *CPU servers*: Elements that receive input data, perform calculations and produce output data. They do not implement any permanent storage, although they often provide volatile disk storage used to temporarily store data while executing a task.
- *Disk servers*: Elements that store data in a stable and reliable manner with an access latency (defined as time to open a file and receive the first byte of data) which is relatively low. They provide inputs to and receive outputs from the CPU servers. They can also send and receive data to Tape server elements.
- *Tape servers*: Elements that store data in a stable and reliable manner with an access latency which is relatively high. They send and receive data to Disk server elements.
- *Information servers*: Elements that maintain data in a stable and reliable manner about the status of the various elements of the distributed computing environment.
- *Control servers*: Elements that issue commands to trigger operations in other types of servers and coherently update the relevant Information servers.
- *Remote Data servers*: Elements that send and receive data from other security domains, often to Disk server elements via Wide-Area Network (WAN) connections.

The elements of the functional decomposition presented, illustrated in Fig. 14.1, together with the aforementioned powerful network and the maintenance of coherent security¹⁰ domains, constitute a framework which is sufficient to analyze the most commonly used distributed computing environments.

The computing industry places a lot of emphasis on how storage is attached to the computers: Direct-Attached Storage (DAS) does not use a network, attaching storage hardware via an internal communication bus of a computer; Storage Area Network (SAN) uses a short-distance network (originally Fiberchannel, with an increasing tendency towards Ethernet) to couple storage hardware to computers; and Network Attached Storage (NAS) which uses computers hosting storage devices and connected to a LAN (or WAN configured to behave like a LAN) to present storage services to the other computers on the network. These distinctions are important in environments where few computers are used. In the case of particle physics, DAS and SAN are used to construct disk servers, which in turn are exposed almost exclusively in NAS mode.

¹⁰A more precise term is “Authentication and Authorization Infrastructure”, abbreviated as AAI or AA.

Fig. 14.1 Functional decomposition of a distributed computing environment



14.3 Data Processing Clusters

The most common way to deploy a distributed computing environment is a “cluster”.¹¹ A cluster consists of a number of CPU, Disk and Control servers connected on a LAN¹² and sharing status information through local Information servers while maintaining a *single, coherent, security domain*. A cluster may also implement Tape servers and Remote Data servers.

Historically, the first complete and reliable commercial implementation of a cluster was the VAXcluster¹³ (or VMSccluster), implemented by Digital Equipment Corp. starting in 1983 on VAX computers and workstations using the VMS operating system. Today, essentially all clusters used by particle physics are implemented using the Linux operating system¹⁴ as a basis, and adding a number of additional packages for handling authentication and storage.

¹¹Unfortunately, after many decades of using the term “cluster” as in the present text, some commercial firms have taken to using the terms “Grid” or “Cloud” to mean “cluster”. This is considered by the author as incorrect and confusing.

¹²There are also implementations using several LANs transparently interconnected via WANs, but maintaining a single security domain. This configuration is often used in business environments to gain fault-tolerance and disaster recovery. Problems related to performance limitations of the WAN, as well as inter-institutional security management complications, limit the use of this configuration in particle physics data processing. Interconnection of clusters via a Grid or a federated Cloud is used instead.

¹³Kronenberg, Nancy P., Henry M. Levy, and William D. Strecker. “VAXcluster: a closely-coupled distributed system.” *ACM Transactions on Computer Systems (TOCS)* 4.2 (1986): 130–146.

¹⁴Linux (or more precisely GNU/Linux) is mentioned specifically as it has become the dominant Unix-like operating system, and has become better known than the original Unix operating system. Most of the remarks in this chapter, however, are equally applicable to other Unix-like systems, notably macOS and FreeBSD.

14.3.1 *Authentication and User Identification*

The machinery behind authentication and user identification is often poorly understood by users, which leads to usage and security problems. It is important to gain an understanding of the basic concepts, especially as more complex distributed environments, such as Grids and Clouds (see sections below), have come into widespread use.

A number of Information servers are deployed to integrate individual computers to form a cluster.¹⁵ The most crucial ones are the ones that generate a single, coherent, security domain.¹⁶ Security under Linux and most other operating systems is based on the concept of an “account” which, ideally, is used exclusively by a single trusted individual. An account is represented externally by an alphanumeric string (the “username”) and internally by a numeric code (the *userid* or *uid*). Although modern operating systems allow very long usernames, local restrictions may apply in order to maintain compatibility with older system software and utilities (the most common restriction is a maximum length of eight characters). Each account is associated to specific directives (called “rights” or “privileges”) to allow or disallow access to operating system services, and can also be used to control access to files and other resources directly or via Access Control Lists (“ACL”). Since people often work in teams, it is useful to associate an account with a “group” named by an alphanumeric string and represented internally by a numeric code (the *groupid* or *gid*). Certain rights and, very importantly, file access control can thus be quickly managed for all accounts belonging to a group.

The process of authentication involves an exchange of credentials which establishes the identity of the user wanting to access a system which results in the creation of a process running under the corresponding *uid*. Authentication has been traditionally accomplished by a “password”, an alphanumeric string which, ideally, is known only to the individual owner of an account. Password related issues are major contributors to computer security problems. Practically all computers are nowadays connected directly or indirectly to the Internet. An inherent weakness of a cluster (which is more than compensated by the gained functionality) is that having a single security domain means that gaining access to any component of the cluster grants access to all of it. The components of clusters directly connected to the Internet are under constant attacks, the most common being automated attempts to guess passwords (usernames are easily obtained from public information sources, such as Web pages listing email addresses). Attacks from within the cluster must also be considered, especially in clusters with many accounts. A

¹⁵The term Information server is used in a broad manner. For example, Domain Name Service (DNS) servers which translate alphabetic Internet addresses to numerical, and Network Time servers which ensure all cluster elements have their clocks synchronized, are considered Information servers.

¹⁶The actual implementation is a highly technical matter, using NIS, ldap, Microsoft Active Directory or other secure database sharing schemes.

password which is easily guessed is denominated “weak”. Simple measures exist to avoid weak passwords.¹⁷ The password should be as long as possible and contain a mix of numbers and upper and lower case letters. Many installations require users to change their passwords periodically in order to improve computer security. A very dangerous practice which should be completely avoided, but which is unfortunately common practice in the particle physics community, is the “sharing” of personal accounts (and hence, their passwords) or the use of “service” accounts with passwords known by dozens of persons, written in documents and blackboards or even posted on Web sites. Modern operating system features make these practices completely unnecessary, as they can be configured so that each user can first authenticate with their personal account and then gain access to a common environment to perform the tasks required.

Portable devices (“laptops” and “smartphones”) can be deployed so that they are elements of a cluster and therefore part of the security domain. Since the portable computer must remain useable when temporarily disconnected from the network, the operating system will keep a local copy (or “cache”¹⁸) of the security data, including the passwords. This means that a stolen portable computer is a computer security threat. Care should be taken in configuring the “suspended” or “hibernated” modes in portable computers to ask for a password when they are turned back on.

Another major issue with authentication based on usernames and passwords is that each cluster (and each online service such as electronic mail, social networks, and document and file repositories), being a separate security environment, requires a separate username and password and has its own policies for requiring password changes. Users are becoming overwhelmed in keeping track of their usernames and passwords and their reaction is often to use weak passwords or keep a list of passwords in a file which can be stolen. In order to reduce username/password proliferation, many organizations are linking their clusters and online services to shared authentication servers, thus providing a “single-sign-on” environment.

In the 2000s, organizations, most notably the CERN Large Hadron Collider experiments, started to use an alternative authentication and authorization frame-

¹⁷Users should consult their local security rules for specific recommendations on password choices.

¹⁸The term “cache” is used in computer architecture to describe a local copy of a limited amount of data which is normally stored elsewhere. The introduction of this local copy serves to increase the performance and efficiency of use of the element connected to it, by smoothing out peaks and valleys in storing and retrieving information, or avoiding retrieves altogether if the same information is needed repetitively. The cache is transparent to the element accessing the information; if the data needed is in cache, it will be delivered from there (a cache “hit”) whereas if the data is not in cache, the normal access path to the normal storage place will be used (a cache “miss”). The key to successful implementation of a cache is the right choice of algorithm to choose which data to copy and keep in the cache. Examples are high-speed memory caches in CPU chips which hold local copies of instructions and data, and memory caches in disk controllers which hold local copies of disk blocks.

work based on digital certificates.¹⁹ This was one of the pillars for constructing the Worldwide LHC Computing Grid (WLCG), later generalized to the European Grid Infrastructure (EGI), the U.S. Open Science Grid (OSG) and others. The certificates used are based on the X.509 standard, supported by all major Web servers and browsers. These schemes were very successful and are still being used, allowing tens of thousands of computers to provide coherent services to thousands of users. They are, however, extremely difficult to manage and expensive to operate. Hence, the tendency is to abandon these schemes in user-facing services, keeping them mostly for machine-to-machine services.

Industry and academia are putting a lot of effort on the emerging area of “distributed” authentication and authorization (AA) schemes.²⁰ The ultimate aim is to simplify the AA process and make it easier to use, enabling “single-sign-on” to a wide variety of resources, a functionality that users have come to expect from their experience with social networks. Unfortunately, progress has been rather slow and solutions proposed by industry and academia are often incompatible. Part of the problem is that there are two competing standards: SAML²¹ and OpenID.²²

The main problem, however, is that the usage by large collaborations, such as those in particle physics, ideally requires the simultaneous use of many different AA sources (something known as Identity Federation). The largest Identity Federation currently deployed is the *eduGAIN* federation,²³ where the AA sources come from the universities and research institutes who employ the users. *eduGAIN* is modeled after the successful *eduroam* system²⁴ used to grant worldwide access to academic WiFi networks worldwide. Solving the general access problem is much more difficult, however, as it requires the maintenance of many more attributes for each user. For example, each university and research institute would have to include and maintain in their databases which experiment or project each employee is participating in, something which is not practical. In order to solve this issue, hybrid schemes are being worked on, where the authentication would come from *eduGAIN* but the authorization information would come from an attribute server managed and operated by a specific experiment or project. This still leaves the problem of reliably operating a service build from thousands of independently managed AA servers with varying degrees of service quality.

¹⁹Thompson, Mary R., Abdelilah Essiari, and Srilekha Mudumbai. “Certificate-based authorization policy in a PKI environment.” *ACM Transactions on Information and System Security (TISSEC)* 6.4 (2003): 566–588.

²⁰These schemes centralize AA information and then make it available to a distributed set of heterogeneous resources, hence the term “distributed”.

²¹Rosenberg, Jonathan B., and David L. Remy. *Securing web services with WS-security: Demystifying WS-security, WS-policy, SAML, XML signature, and XML encryption*. Sams, 2004.

²²<https://openid.net/developers/specs/>

²³López, D. “eduGAIN: Federation interoperation by design.” *TERENA Networking Conference*. 2006.

²⁴López, Gabriel, et al. “A proposal for extending the eduroam infrastructure with authorization mechanisms.” *Computer Standards & Interfaces* 30.6 (2008): 418–423.

Some interesting alternatives are starting to emerge, which avoid Identity Federation. One approach is to identify a single reliable provider for AA, for example the OpenID service from ORCID²⁵ combined with an appropriate attribute server. Another approach is for a project or experiment to deploy a single AA scheme available to services *for that project* around the world. This can be accomplished in a simple and economical manner by re-using the project's personnel database and exposing it through the Internet in a secure manner using an *ldap* server.²⁶ These type of implementations of single-sign-on are ideal for service providers that serve a single project. On the other hand, they do shift the diversity problem to the service providers, which must locally adjust crucial attributes such as *uid* and *gid* in order to avoid duplications in multi-project or multi-experiment environments.

14.3.2 Processing and Storage

Two configurations of CPU servers are usually deployed:

- “Batch workers”, which execute tasks (jobs) that don't require user interaction and are scheduled using a “Batch” system²⁷ or a more sophisticated resource harvester, such as HTCCondor.²⁸
- “Interactive nodes”, where users can connect (or “log-in”) and perform work that requires interaction via a screen, keyboard and mouse.

Several configurations of Disk servers are usually deployed:

- “Network File servers”, based on industry-standard or widely used protocols, such as NFS,²⁹ smb2³⁰ or http with WebDAV,³¹ are used to provide “home” and “project” directories holding text and binary files of relatively small size. These directories appear in the CPU servers as “mount points” or “network shares” that largely behave as a local, conventional disk resource. The POSIX input–output

²⁵Haak, Laurel L., et al. “ORCID: a system to uniquely identify researchers.” *Learned Publishing* 25.4 (2012): 259–264.

²⁶Johner, Heinz, et al. *Understanding LDAP*. Vol. 6. IBM, 1998.

²⁷See for example Yoo, Andy B., Morris A. Jette, and Mark Grondona. “Slurm: Simple linux utility for resource management.” *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, Berlin, Heidelberg, 2003.

²⁸Fajardo, E. M., et al. “How much higher can HTCCondor fly?” *Journal of Physics: Conference Series*. Vol. 664. No. 6. IOP Publishing, 2015.

²⁹Shepler, Spencer, et al. *Network file system (NFS) version 4 protocol*. No. RFC 3530. 2003.

³⁰French, Steven M., and Samba Team. “A New Network File System is Born: Comparison of SMB2, CIFS and NFS.” *Linux Symposium*. sn, 2007.

³¹Goland, Yaron, et al. *HTTP Extensions for Distributed Authoring – WEBDAV*. No. RFC 2518. 1999.

standard³² is the most often used definition of this behavior. The implementations are far from trivial, but fortunately they are widely used and hence available as packages fully supported by operating systems, or even as part of the operating system. These protocols support operations such as simultaneous access by multiple users in read/write/append mode as well as record-level locking. This forces the deployment in CPU servers of some Control and Information server functionality, with information being constantly exchanged between all nodes accessing a mount point in order to keep file system coherency and distribute lock information. In order to fulfill performance expectations, these Control and Information functionalities must often be implemented as drivers, often executing inside the operating system kernel. The price of this is the risk of all CPU servers becoming inoperative (or “hung”) if a network file server has a fault or the status information becomes corrupted. Some Network File servers, such as the CernVM-FS³³ system used for software installation, serve very specialized purposes. They implement only part of the functionalities, for example serving files only in read-only mode, with the benefit of having a smaller impact on the operating system environment.

- “Database servers”, are disk servers³⁴ which organize their data using database products such as Oracle, MySQL or PostgreSQL. They are mostly used to store and manage detector configuration and calibration data, though on occasion they can store high-level analysis data such as the “Event Tags” used by some experiments.³⁵ CPU servers read these data through the network by issuing database queries and retrieving the results. Database servers must be configured in accordance to the type of most-often used queries (a process called “tuning”) in order to achieve the required performance. Because the overall cost of purchasing and maintaining a Database server is high, situations often arise where the deployed capacity is insufficient to serve all the CPU servers unless specific caches or buffers³⁶ are deployed. An emerging alternative type of database

³²Gallmeister, Bill. POSIX. 4 Programmers Guide: Programming for the real world. “O’Reilly Media, Inc.”, 1995.

³³Blomer, Jakob, et al. “Status and future perspectives of CernVM-FS.” Journal of Physics: Conference Series. Vol. 396. No. 5. IOP Publishing, 2012.

³⁴It may seem odd to classify a database server as a Disk server. It is considered correct in this context, as their usage pattern is unusual in that the record update frequency is very low compared to the record reading frequency. Note that the technical database servers used to implement Data and Tape server features are considered Information servers.

³⁵See for example Cranshaw, Jack, et al. “Event selection services in ATLAS.” Journal of Physics: Conference Series. Vol. 219. No. 4. IOP Publishing, 2010.

³⁶The term “buffer” is used in computer architecture to mean “temporary storage location”. Buffers are used to accumulate information and then transmit it as a block, or to receive information as a block and then distribute it. The difference between a buffer and a cache is that in the case of buffers all the information is temporarily stored, whereas a cache implements an algorithm to store only the most relevant information. In addition, whereas a cache is always transparent, a buffer may be transparent or it may allow explicit buffer manipulation by the application.

servers are those based on “Big Data” tools, such as NoSQL databases³⁷ with underlying *Hadoop* storage management systems.³⁸

- “Data servers”, based on products such as dCache,³⁹ GPFS,⁴⁰ Lustre,⁴¹ EOS⁴² and DPM,⁴³ are used to hold the large-size binary files which contain physics data. They use a set of specific Control and Information Servers to present to the other elements of the cluster, in particular to the CPU servers, an interface approximating a conventional (POSIX-compliant) disk resource, but with some important differences in order to achieve the large capacities and high performance needed for particle physics applications (and data-intensive applications in other fields). Much of the optimization is possible because of the particular usage pattern of particle physics data, which is essentially written once and then read many times.⁴⁴ The key point is to separate⁴⁵ the “namespace” information (directory tree or file folder structures, file attributes such as name, creation date, etc.), the “file storage” information (physical location of the blocks that comprise a file, whether it is open for read, write or append, etc.) and the “file data” information (the actual data blocks that comprise the file). This separation makes it possible to serve the data blocks from a large number of Data servers with a combined capacity far exceeding what can be provided by a conventional file system. In addition, the data blocks of a given file may be stored in multiple servers in order to enhance read performance or provide high-availability (for reading).⁴⁶ The cost of this separation (apart from the complexity of the multiple

³⁷Han, Jing, et al. “Survey on NoSQL database.” Pervasive computing and applications (ICPCA), 2011 6th international conference on. IEEE, 2011.

³⁸Shvachko, Konstantin, et al. “The hadoop distributed file system.” Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on. Ieee, 2010.

³⁹Millar, Paul, et al. “Storage for advanced scientific use-cases and beyond.” Parallel, Distributed and Network-based Processing (PDP), 2018 26th Euromicro International Conference on. IEEE, 2018.

⁴⁰Schmuck, Frank B., and Roger L. Haskin. “GPFS: A Shared-Disk File System for Large Computing Clusters.” FAST. Vol. 2. No. 19. 2002.

⁴¹Schwan, Philip. “Lustre: Building a file system for 1000-node clusters.” Proceedings of the 2003 Linux symposium. Vol. 2003. 2003.

⁴²Peters, A. J., E. A. Sindrilariu, and G. Adde. “EOS as the present and future solution for data storage at CERN.” Journal of Physics: Conference Series. Vol. 664. No. 4. IOP Publishing, 2015.

⁴³Alvarez, Alejandro, et al. “DPM: future proof storage.” Journal of Physics: Conference Series. Vol. 396. No. 3. IOP Publishing, 2012.

⁴⁴Aside from the write-once/read-many access pattern, particle physics datasets almost never have data appended to them nor have their already existing records updated. For an extreme contrast, consider the access pattern of bank account datasets, which constantly have data appended and updated.

⁴⁵Conventional file systems, whether local or networked, tightly bind the management of the three types of information described in order to optimize file open/close operations and implement sharing of files in read/write/append mode.

⁴⁶This is conceptually similar to the “striping” of files across many disks performed by RAID controllers on DAS devices.

Information Servers) is a relatively high overhead and latency for file open/close operations and the loss of full POSIX compliance. In some cases, applications have to use input–output methods specific to the data server product, such as *Xrootd*,⁴⁷ whereas in other cases standard protocols, such as NFS or http or subsets of POSIX I/O can be used. Historically, namespaces have been confined to single clusters. There are, however, some interesting implementations of global namespaces, using for example a hierarchy of distributed name servers coupled through redirection⁴⁸ techniques offered by various protocols such as *Xrootd*, WebDAV or http.⁴⁹ Many commercial systems use similar implementations, but with different design criteria, for example the requirement of serving hundreds of thousands of users at relatively low performance in a social network.

Tape servers are hidden from the user,⁵⁰ except in very special cases such as data recording of experimental data. Nevertheless, users and system managers should be vigilant, as the user may generate access patterns which make extremely inefficient use of the underlying tape cartridge system. Tape servers and their related tape cartridge systems are quite complex technologically. Exploring this is beyond the scope of this chapter, but a few remarks are appropriate. The main motivation for introducing Tape servers is that they provide storage which is less expensive than Disk servers, with additional features such as guaranteed read-only access, protection against erroneous deletion by users, lower probability of data loss due to hardware failure and lower power usage. As their name implies, tape servers have been historically implemented using storage on magnetic tape cartridges. Implementations for particle physics are of the “Active Tape” nature, using specific software to manage the tape in a much more agile manner than traditional tape backup systems, as well as providing file by file access. Different computer centers use different software packages for Active Tape management. Examples are HPSS⁵¹ and Tivoli Storage Manager,⁵² both developed by IBM, Enstore⁵³ developed by Fermilab and Castor2⁵⁴ developed by CERN. Decreasing prices for commodity disks, however, are driving an emerging area where Tape

⁴⁷Dorigo, Alvise, et al. “XROOTD-A Highly scalable architecture for data access.” WSEAS Transactions on Computers 1.4.3 (2005).

⁴⁸For example, redirection is a standard feature in the http protocol used for the World Wide Web.

⁴⁹See for example Bloom, Kenneth, and Cms Collaboration. “Cms use of a data federation.” Journal of Physics: Conference Series. Vol. 513. No. 4. IOP Publishing, 2014.

⁵⁰Conventional backup to tape of files in home and project directories is usually performed as part of the general operation of clusters using standard commercial or open-source packages. This is an important function but, being completely conventional, is not considered further in this context.

⁵¹Teaff, Danny, Dick Watson, and Bob Coyne. “The architecture of the high performance storage system (hpss).” (1998).

⁵²IBM Corporation, Tivoli Storage Manager; see <http://www.ibm.com/software/tivoli/products/storage-mgr/>

⁵³Bakken, Jon, et al. “Enstore Technical Design Document.” Fermilab-JP0026 (1999).

⁵⁴Presti, Giuseppe Lo, et al. “CASTOR: A Distributed Storage Resource Facility for High Performance Data Processing at CERN.” MSST. Vol. 7. 2007.

servers are implemented on inexpensive commodity computers with large disks. This is in turn stimulating multi-layer implementations, for example the D2D2T (disk-to-disk-to-tape) scheme, where Disk servers store and retrieve data on disk-based pseudo-tape servers, which in turn store and retrieve data on “real” magnetic cartridge Tape servers. Both disk-based tape servers and D2D2T schemes are important because the large growth in data volumes in particle physics is the result of a very modest growth in file size multiplied by a large growth in the number of files, resulting in a situation which can be very inefficient for traditional tape systems (the so-called small file problem in tape cartridge storage).

A “Hierarchical Storage Manager” (HSM) is implemented between the Data servers and Tape servers using specific Control and Information servers. HSM systems are not a common feature of conventional computing environments. HSM systems that fulfill the needs of particle physics data processing are even less common. Furthermore, the interaction between Data servers and Tape servers can be rather complex and no clean, practical interfaces are available. Therefore, each data center implements its own customized solution using a combination of community-written and commercial software. Examples of Data server/HSM implementations are dCache interfaced to Tivoli or Enstore; GPFS interfaced to IBM HPSS; and CERN’s Castor-2 which has its own data server and HSM implementations. Depending on the solution, the HSM system may be more or less intertwined with the Data server software, although the tendency is to separate as much as possible the HSM and Data server implementations.

Experiment specific Information servers are deployed in order to manage and make accessible a view of all their files containing custom details about their characteristics, for example trigger conditions, detector configuration or beam type. This is often called a “File Catalog” and may in itself be a quite complex database.

14.4 The Main Workflows of Data Processing in an Experiment

Most experiments implement two “organized” data processing workflows: one for data coming from the actual detector (awkwardly referred to as “real” data) and another for simulations resulting from the techniques described in the contribution “Detector Simulation” (awkwardly referred to as “Monte Carlo” data). An additional very important “organized” workflow derives calibration constants from detector data taken either in normal mode or in special conditions for calibration (cosmic rays, single beam, sending pulses to the front-end electronics, etc.). Altogether, these three workflows are referred to as “production” and constitute the process by which “raw” data is transformed into “analysis datasets”. Workflows related to the use of these analysis datasets by individual physicists are described in a later section.

The production workflows involve a number of tasks,⁵⁵ such as event reconstruction, simulated event production, data reduction, event filtering, and calibration data processing.

14.4.1 Event Reconstruction

Data arriving from an experiment's data acquisition system is called "raw" data. It is organized in "event records", each record corresponding to a particle interaction "event" which is relevant in the context of the experiment. An event record may or may not correspond to a single collision between particles, though particle physicists often refer to data arising from a single collision as "an event". In a collider, an event record may correspond to data collected from a single beam crossing, which depending on the beam type, luminosity and angular coverage may contain data arising from several particle collisions.⁵⁶ In a fixed target experiment an event record may correspond to data collected over a full beam spill. In an astroparticle experiment an event record may correspond to data collected starting from a trigger signal for a certain length of time. Furthermore, due to performance requirements of data acquisition systems, raw data is often compressed, encoded or packed into rather complex data structures, which are often not very convenient for direct use in reconstruction programs. Therefore, a raw event record is usually read from disk and immediately "unpacked" into simpler but larger data structures where the digitized signals are represented by standard integer or floating point numbers.

The output of the most complete simulations (often called "full" simulations, see below) can be stored in "packed raw" format identical to the detector data, or in "unpacked raw" format in which case the unpacking step is omitted in reading simulated event records.⁵⁷ Thereafter, simulated event records are treated identically to detector "real" data.

A number of event records are written by the data acquisition system to a raw data file, which is the main data item delivered from the "online" to the "offline" environment. The organization of event records into raw data files is very often unrelated to the experimental conditions and simply reflects technical constraints such as the size of disk buffers in the online and offline clusters.

⁵⁵The names of these tasks are not standardized. Therefore readers will have to identify the corresponding task names for their respective project.

⁵⁶In electron-positron collisions there is a single collision most of the time, but there may be occasionally an annihilation event together with a two-photon collision, for example. In proton-proton collisions there are on average 20 collisions per beam crossing at the nominal luminosity of the original LHC configuration.

⁵⁷Even though it would be more economical to store simulated data in "packed raw" format, this is often not possible. For example, the simulations may be needed before the detector data acquisition system is fully designed and implemented, and therefore the exact format for packed raw data will not be known.

The output of the reconstruction arises from the application of the techniques described in the contribution “Patter Recognition and Reconstruction” and consists of data structures containing tracks, clusters, etc. These data structures are written as event records, which may or may not have a one-to-one correspondence to the event records read on input. For example, the reconstruction program may disentangle data from different collisions contained in a single raw data event record and write the corresponding reconstruction output as separate event records for each collision.

The reconstruction program needs detector configuration and calibration data that often varies with time, though reasonably slowly compared to the event rate.⁵⁸ Therefore, the reconstruction program accesses a “conditions database” in order to fetch the configuration and calibration data matching the conditions under which a particular event was acquired. Alternatively, if stable and reliable calibration data are available at the time of acquisition, it may be stored within the raw data files, interspersed with the event records.

Event records are, in principle, all independent from one another and therefore suitable for loosely-coupled parallel processing. Early implementations⁵⁹ would process one raw data file at a time, in essence treating the raw data file as a buffer and implementing an “event server” to distribute individual events via the network to CPU servers. This methodology has been largely abandoned for event reconstruction, although it remains a good choice for parallel processing of analysis data sets. Modern event reconstruction clusters or “farms” are configured so that each CPU server executes jobs that process one or more complete data files at a time. The CPU servers read raw and calibration data and write reconstructed data in an autonomous fashion to the Disk and Database servers. Hundreds or even thousands of jobs can run simultaneously in a single cluster.

Care must be taken to handle aspects that may introduce coupling and break the highly parallel processing scheme. Consider, for example, one of the most common pitfalls that breaks parallelism: simultaneous access to the same file by many jobs. For example, the executable image of the reconstruction program is the same for all jobs and may be quite large. Therefore, each job that starts must read the image file from a Network File server. In an environment where thousands of jobs are executed, the number of jobs simultaneously reading the image file may be

⁵⁸There are exceptions to this rule, especially for neutrino beam or non-accelerator events, where calibration data may in fact be updated faster than the event rate.

⁵⁹Event processing in parallel dates back to the 1970s when physicists used their experience in building digital electronic boards for detectors to design inexpensive processor boards, called emulators, which executed the instruction set of the IBM System 370 mainframe computer and interfaced to its input-output channel by emulating a tape drive. Starting in the mid-1980s, parallel event processing was mostly done using specially configured clusters (called “farms”) of scientific workstations with Reduced Instruction Set CPUs (RISC) running the VMS operating system from Digital Equipment Corp. and a number of variants of Unix, interconnected by proprietary networks or by daisy-chained Ethernet networks. Starting in the mid-1990s to date, essentially all reconstruction is run on clusters of widely marketed, industry standard “commodity” computers using the x86 instruction set supported by Intel and Advance Micro Devices CPU chips and interconnected by switched Ethernet networks.

high enough to overload the Network File server, therefore creating a performance bottleneck.⁶⁰ Caching of the image file on a local disk of each CPU server can be used to restore parallelism more economically compared to deploying a more powerful Network File server. A similar situation arises from the access by all jobs to a unique conditions database. Parallelism is harder to restore in this case and requires tuning of the database server and the caching algorithm, or deployment of very powerful (and expensive) replicated database servers.

Raw data files must often be retrieved from tape, and parallelism may be broken from the fact that a single tape cartridge is used to store multiple raw data files. Many HSM systems are unable to handle multiple, asynchronous, closely spaced requests for files on the same tape. They queue the requests and mount and dismount the tape cartridge in order to read each of the files, a slow procedure as it involves mechanical actions. As individual raw file sizes have historically increased more slowly than total raw data volume, the number of files per tape cartridge has increased and parallelism breakage by tape access has become quite common. In order to circumvent these problems, production managers run special jobs to “pre-fetch” in an efficient way lists of specific datasets from tape to Disk servers. Efficient pre-fetching requires detailed knowledge of the HSM software and the contents of the tape cartridges in order to minimize the number of tape mounts.

Most modern experiments strive to run a production workflow for reconstructing the raw data that keeps up with the data acquisition, called “quasi-online” or “prompt” reconstruction. The quality of the calibration constants which are available for this workflow is often limited and therefore the reconstructed output may be of limited value for final physics results; the output is very valuable, however, to monitor detector performance and to derive more definitive calibration constants.

Frequently, reconstruction of a particular raw data file must be repeated a few times, in order to incorporate improvements or correct errors in calibration constants and reconstruction algorithms. This is called “reprocessing” and applies to both detector and simulated data.

14.4.2 Event Simulation

Production of simulated events usually starts with the generation of physics final-states using an “event generator”. The CPU time needed for this step is usually

⁶⁰To understand how parallelism is broken, consider that the Network File server handles read requests in parallel up to a certain limit and will serialize into a queue all requests exceeding this limit, breaking parallelism. To understand how coupling is introduced, consider that the event records are coupled through the fact that they are all being reconstructed using the same reconstruction program image file. This can be exacerbated by additional coupling at the hardware level, for example when many cores compete for the same Ethernet network interface in modern multi-core computers.

small, as is the size of the output file (which is often called a “four-vector file”).⁶¹ A single (non-parallel) job can usually produce enough four-vectors for a whole simulation campaign. The situation is completely different in the next step, where the four-vectors go through a full detector simulation which is very CPU time consuming. Many jobs must be run in parallel reading four-vectors and writing simulated event records⁶² to Disk servers. The data rate per job is not very high, but the integrated output rate can be substantial since there are many CPU servers running simultaneously, an issue that should be taken into account. Simulation is often run on many separate clusters and the oversight of simulation productions can consume substantial amounts of time from a large number of physicists. Deployment of Grid interfaces on clusters (see below) has simplified these tasks somewhat. A recent tendency is to use “production management systems” based on databases to keep track of simulation jobs. The most tedious part, however, is the detection and correction of errors coming from the execution of the simulation program itself, which currently is not automated in part due to lack of a systematic approach to error signaling.

14.4.3 Data Reduction and Event Filtering

Appropriate data representations for physics analysis are quite different from the raw data format. They can also be quite different from the reconstruction output format for several reasons, some related to enabling limited reprocessing capabilities from the reconstruction output itself, while others are related to signal-to-noise issues. Reconstruction output may have a format optimized for limited reprocessing without access to the raw data, for example recalculating vertices from tracks or track and calorimeter cluster matches. This format may not be optimal for analysis. A simple example can illustrate this: a typical analysis condition is to access all tracks that have a given minimum number of points (or “hits”); hence, the optimum data structure is a list of tracks, each track having the property “number of points”. The reconstruction output, however, will most likely be stored as a list of points, each point having a property of belonging to a track, a data structure optimized for reprocessing the assignment of points to tracks, but completely “backwards” from the one needed for optimal analysis. Reconstruction output will often have additional information stored, related to enabling reprocessing, which makes it too bulky for practical use in analysis. It is often organized keeping a

⁶¹ A single generation step is described for simplicity. Note that in some cases the output of an event generator may need to be post-processed by another physics simulation program, for example to turn quarks and gluons into jets.

⁶² A single step is described for simplicity. Note that in some cases the output of the detector simulation may have to be post-processed to generate simulated event records, for example by adding simulated pile-up events for proton–proton collisions in the LHC.

one-to-one relationship to raw data files, which in turn may have little to do with physics analysis. Furthermore, it usually has output for all raw data records, which depending on detector triggering and physics signal-to-noise conditions may be very large compared to the optimal for physics analysis.

These considerations lead to the definition of “reduced” or “analysis” datasets along two complementary lines. Data reduction takes the reconstruction output and generates data structures optimized for analysis by removing information related to reprocessing and “turning around” data structures related to tracks, vertices, calorimeter clusters, etc. to be optimal for analysis. Particle identification by correlation between different detector elements is often done (or re-done) during data reduction and, when combined with Event Filtering, may be optimized for particular physics processes (heavy quark identification, rejection of electrons, etc.). The historical name for the output of data reduction is “Data Summary Tape” or “DST” and successively smaller outputs have been called “mini”, “micro” or even “nano” DSTs.⁶³ A nano-DST may simply consist of a list of “reconstructed particles”, each stored with a few properties such as particle ID probabilities and momentum components. A nano-DST is in some sense the experimental counterpart to the event generator output.

DST files often do not mirror the raw and reconstructed file organization scheme. At a minimum, DST files contain the DST records that correspond to many raw files, in order to ease the management of large numbers of datasets and avoid handling very small files. In accelerator experiments, DSTs may be organized by running conditions. In fixed-target experiments where on-beam and off-beam data are alternatively taken, separate DSTs for each may be produced. In astroparticle experiments, which often behave like telescopes, DSTs will often be organized by observed source, merging data from many observation days or nights. These reorganizations of data are special cases of event filtering.

Event filtering⁶⁴ may be performed before, after or during production of any DST-like output. The purpose is to group together in particular files the DST output most relevant for a particular analysis, thereby avoiding repetitive reading of the rest of the data by analysis programs. Event filtering can be done according to many criteria, but is most often performed as a pre-selection for particular physics analyses, especially in situations of unfavorable signal-to-noise ratio such as the LHC experiments. Output datasets are often referred to by the name of the physics process pre-selected and are written in an identical format as the original DST-like datasets.

⁶³The names of Data Reduction output datasets are not standardized. Therefore readers will have to identify the corresponding dataset names for their respective project.

⁶⁴Event Filtering in this context should not be confused with the actions taken by high-level trigger “Event Filters”.

14.4.4 Processing of Calibration Data and Calculation of Calibration Constants

Reconstruction of data from detectors requires a large amount of calibration constants, many of them time-varying. Raw calibration data, such as temperatures, pressures and voltages are recorded periodically during data taking. Calibration data related to response and alignment of different detector elements can be obtained by generating artificial events with specialized calibration devices, for example pulsed lasers. Alternatively, they can be calculated from the raw physics data itself, using better understood reactions or additional events occurring in parallel to the physics, such as the passage of cosmic rays.

Data input to calibration procedures often comes from diverse sources, and handling it poses challenges that are often underestimated. For example, laser pulses for calibration may be recorded in parallel to the physics data and stored by the online system interspersed in the raw data files, thereby requiring the offline to access all raw data files in order to process the laser pulses, a rather tedious procedure. Conversely, sometimes calibration data are stored in a *potpourri* of small files and databases of diverse formats, introducing unnecessary complexity in the handling of these data.

There is no standardized framework to handle the output of all calibration procedures, although the recent trend is to centralize it into a single calibration (or “conditions”) database that is accessed through heavily cached methods by instances of the reconstruction, DST generation and analysis programs. Care must be taken to serve this database with adequate resources, and to ensure efficient and cached access to it in order not to break parallelism as described earlier.

The flow of raw and derived calibration data and the related data processing tasks should be studied with equal care as that of the physics data, and solutions should be implemented to ensure efficient and agile data access for calibration purposes.

14.5 Interactive Analysis Using Clusters

The last stages of physics analysis are performed by individual users using interactive tools such as ROOT,⁶⁵ Jupyter Notebooks⁶⁶ or SWAN.⁶⁷ They apply

⁶⁵See <http://root.cern.ch>

⁶⁶See <https://jupyter.org/about>. Although Jupyter Notebooks are language agnostic, they are closely associated with the Python language. They are increasingly popular in particle physics and many other scientific disciplines, given the rise of the use of Python by them. Jupyter Notebooks must be combined with suitable data-access modules. For example, the SWAN project at CERN combines Jupyter Notebook with ROOT input–output modules to create an interactive analysis platform.

⁶⁷Danilo Piparo, Enric Tejedor, Pere Mato, Luca Mascetti, Jakub Moscicki, Massimo Lamanna, SWAN: A service for interactive analysis in the cloud, *Future Generation Computer Systems*, Volume 78, Part 3, 2018, Pages 1071–1078, ISSN 0167-739X, <https://doi.org/10.1016/j.future.2016.11.035>

selection criteria to events in DST-like files or in personal or group files derived from them and written directly in the format of the analysis tools. The basic record in these files is often a collection of ordered lists of properties, or “n-tuples”.

Distributed computing is used for preparing the input files for the interactive tools, using the same methodologies described earlier for running reconstruction and data reduction jobs. In addition, since the application of selection criteria and the generation of histograms from n-tuple files can often be mapped to highly parallel tasks (if each n-tuple record is independent of any other), efforts have been made to provide parallel “back-ends” to interactive analysis tools, for example the PROOF back-end to ROOT or the use of *Hadoop* in conjunction with Jupyter Notebooks. The idea is simple in principle: the user issues a command that specifies some actions to be performed on all records; the back-end automatically generates and executes in parallel a number of tasks reading the data and writing partial result files; when all parallel tasks have finished, the partial results are merged to a global result which is presented to the user via the interactive front-end. Speed is gained, in principle, by using many CPUs and, more importantly, many input data channels in parallel. In addition, since the interactive user pauses to examine the results, resources can be used more efficiently by serving many users from a single parallel back-end. Practical implementation with good performance, however, can be extremely complex, as they involve running a cluster at very large peak input-output rates and clever scheduling of bursts of parallel tasks.

14.6 Multiple Sites and Grid Computing

Grids were defined in the late 1990s as a new manner of integrating services from clusters across multiple sites.⁶⁸ Practical development and deployment of Grids was originally led by the particle physics community, especially the LHC collaborations. Grids are convenient for two main purposes: data distribution to many end-users who are geographically dispersed, and data processing load sharing across many computing centers (“resource” centers).

The main idea of a Grid is to abstract the most commonly used interfaces for authentication, access control, job submission and data access into a unique “meta-interface”, which is the only interface exposed to end-users. Therefore, multiple sites are integrated for the end-user under the illusion of a single “meta-cluster”. Specialized Grid Information and Control servers are deployed in order to translate from the abstract meta-interface into the specific interface used by each cluster. Grid interfaces and protocols have not been standardized. After an intense period of development during the 2000s, a handful of partially compatible Grid schemes are now deployed. Many claims of success in providing resources on the Grid

⁶⁸“The grid: blueprint for a new computing infrastructure”, Ian Foster and Carl Kesselman, editors. San Francisco: Morgan Kaufmann Publishers, c1999. ISBN 1558604758.

come from providing simple CPU-intensive services with little input or output. Providing reliable data-centric services has proven to be much more difficult, both from the point of view of the operation of resource centers themselves and of the Information and Control servers which provide the Grid interface for each center. Nevertheless, the CERN Computer Center, the WLCG Tier-1 centers and a subset of the WLCG Tier-2 centers do provide the needed reliable data-centric services needed for nucleating the rest of resources (see below).

The European Grid Infrastructure (EGI) program,⁶⁹ financed in part by the EU, has deployed the largest production Grid Infrastructure to date in collaboration with the Worldwide LHC Computing Grid (see next section). Other Grid Infrastructures provide support for particle and astroparticle physics, for example the Open Science Grid (OSG) in the USA. The EGI Grid will be described as an example of a working Grid used for daily work by particle physicists. EGI defines a Grid Infrastructure as a set of services deployed over Internet that allows a large number of Resource Centers, each with a different security and management domain, to be used in a coherent fashion by a large number of users from different institutions grouped in virtual organizations. Usually, a virtual organization corresponds to a project.

The basic services offered by the EGI Grid Infrastructure are based on the UMD middleware distribution,⁷⁰ which is deployed in a managed way, including monitoring and fault detection, with the support of a distributed operations organization provided by resource centers, many of which are associated to LHC computing or national Research and Education networks. Resource centers deploy Remote Data Servers using a variety of Disk Servers (possibly with magnetic tape back ends) which expose a uniform interface to the Grid via UMD, whose instances are called “Storage Elements”. Authorization and access control is accomplished using the certificate method described earlier with tokens generated by the Virtual Organization Membership Service (VOMS) upon presentation of X.509 certificates issued by national research agencies and declaration of the virtual organization (e.g. project) under which the user wishes to be authorized. The actual data flow is predominantly through the *ftp* protocol, wrapped in tools such as *gridftp* (which incorporates certificate authentication and multiple parallel transmission streams).⁷¹

Each resource center implements a “Computing Element” (CE) service, a set of Information and Control Servers which map a specific batch system to an abstract

⁶⁹See <http://www.egi.eu>. EGI operates the international coordination structure for Grid resource centers across Europe. EGI also coordinates the maintenance and security support for the Grid software originally developed in the 2000s by the EU Enabling Grids for E-science (EGEE) series of projects and the EU DataGrid project.

⁷⁰The Unified Middleware Distribution (UMD) maintained by EGI is based on software developed in the EU European Middleware Infrastructure (EMI) project. The EMI software is the result of integration and compatibility work done on the *gLite* middleware, developed within the EGEE Project, the ARC middleware, developed by NorduGrid, and various other packages. See <https://wiki.egi.eu/wiki/Middleware>

⁷¹A more sophisticated Grid data access tool known as *srnm* was used in the past but is largely being abandoned due to maintenance and management issues.

Grid batch system. Directives may be given to the CE specifying job requirements such as minimum amount of memory, length of execution, etc. A challenge in configuring very large Grids is to reach agreement on the normalizations used to measure cluster resources, for example the speed of processors. Users submit their jobs through an additional server, called a Resource Broker, which accesses information in tables maintained by all available CEs and submits the jobs as appropriate to balance the load amongst the multiple sites conforming the Grid. Large projects, such as the LHC experiments, have developed job management services which in practice replace the Resource Brokers, interacting directly with CEs.

The Grid data access services described above offer functionality similar to the low levels of operating systems, but with the great advantage of generating coherent behavior across many Resource Centers. In practice, these services are often too low-level for practical use by applications. Hence, additional software, services and APIs are provided by middleware distributions at a higher level. One example is the File Transfer Service (FTS),⁷² designed to support applications which require automated management, monitoring and error correction of replication of large numbers of files and large volumes of data, such as the LHC experiments. It uses a database back end to store replication requests and instances of finite state machines to reliably accomplish the corresponding data replications, including error correction and reporting. In addition, FTS implements the concept of a channel (a virtual path between two Resource Centers) which may be managed by operators, adjusting parameters such as the number of simultaneous files being transferred, the maximum bandwidth used, etc.

Large scientific projects, such as particle physics experiments, can have specific needs that go beyond the scope or functionality of basic middleware (such as UMD) and basic Grid services (such as those offered by EGI or OSG). In these cases, the projects themselves provide their own enhancements.⁷³

14.7 The Worldwide LHC Computing Grid (WLCG)

The largest data processing capacity deployed so far using clusters linked via Grid infrastructures is the Worldwide LHC Computing Grid (WLCG or LCG).⁷⁴ WLCG links some 170 clusters located in 42 different countries into a single

⁷²Andrey Kiryanov, Alejandro Alvarez Ayllon and Oliver Keeble, “FTS3/WebFTS – A Powerful File Transfer Service for Scientific Communities”, *Procedia Computer Science*, Volume 66, 2015, Pages 670–678, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2015.11.076>

⁷³Readers should inquire within their own projects about these enhancements. They usually fall into two categories: Job management enhancements, for example the DIRAC package used by the LHCb experiment, and File or Dataset management enhancements, such as the AAA package used in CMS and the Rucio package used by ATLAS.

⁷⁴See <http://www.cern.ch/lcg>

Table 14.1 Capacity deployed in the tiers of the LHC computing grid in 2018

		2018
CPU (KHS06 ^a)	Tier-0 ^b	1272
	All Tier-1	2061
	All Tier-2	2562
Disk (TeraBytes)	Tier-0	90700
	All Tier-1	192214
	All Tier-2	185455
Tape (TeraBytes)	Tier-0	284700
	All Tier-1	461050

^aThousands of HEPSPEC06 units. HEPSPEC06 is a measure of CPU speed maintained by the particle physics community. See <http://w3.hepfix.org/benchmarking.html>. As a very rough guideline, a current 2 GHz CPU delivers about 15 HS06

^bIn addition to the Tier-0, CERN also deploys an Analysis Facility (CAF) which is not included in this table

coherent distributed system which supports all data processing operations of the LHC experiments. WLCG defines a global service supported in part by Grid Infrastructure services in various countries and regions, for example EGI in Europe and OSG in the USA. The large amount of data handled by WLCG (over 100 Petabytes per year exchanged between sites) requires the sites to be organized in three “Tiers”. Centers belonging to a given Tier are dedicated to perform certain specific tasks within the computing model of a given LHC experiment. There is one Tier-0 center at CERN, which handles Data Recording, Prompt Reconstruction and Remote Data Distribution (mostly to the Tier-1 centers). There are 13 Tier-1 centers, which perform Data Recording (between them they hold a second copy of all raw data stored at the Tier-0), Reprocessing, Data Reduction and Event Filtering. Hundreds of Tier-2 centers receive from the Tier-1 centers DST-type files (reduced and filtered) which physicists then use for analysis. Tier-2 centers also run event simulation, transmitting the outputs to Tier-1 centers for data recording and reconstruction. FTS, coupled to some experiment specific tools, is used to automate all data transmission.⁷⁵ The capacities deployed are quite large: about 400 thousand computing cores, 250 Petabytes of disk and nearly an Exabyte of tape. Table 14.1 shows the capacities deployed in 2018. These capacities have grown at an impressive rate of about 37% year-to-year since 2008. An even larger growth is expected for Run-3 of the LHC starting in 2021.

⁷⁵The Wide Area Network requirements between the Tier-0 center and the 13 Tier-1 centers are so demanding that a special network based on point-to-point 10 Gbps links has been deployed (the LHC Optical Private Network, or LHCOPN). As of this writing, the LHCOPN links are being upgraded to 100 Gbps.

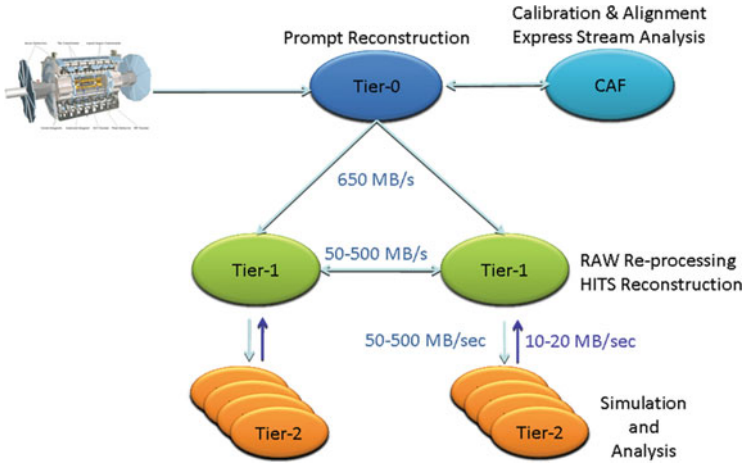


Fig. 14.2 Schematic view of the original ATLAS computing model

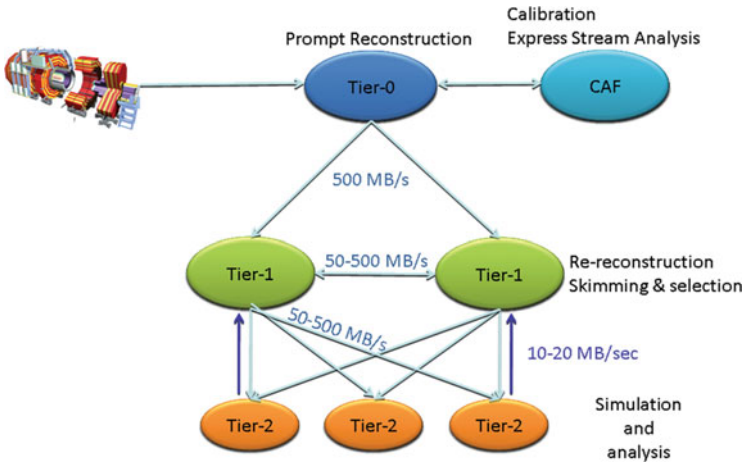


Fig. 14.3 Schematic view of the original CMS computing model

As mentioned earlier, the detailed breakdown of tasks performed at each Tier depends on the computing model of each experiment. On the other hand, the Tier-0 and most Tier-1 sites serve multiple experiments, whereas most Tier-2 sites only serve a single experiment. This leads to quite complex optimization issues, especially in regards to data transmission, which depending on the experiment may involve few or many site pairings. For example, Figs. 14.2 and 14.3 show schematically the original computing models for the ATLAS and CMS experiments, and the different connectivity patterns between Tiers can clearly be seen. Originally, ATLAS closely linked a given Tier-1 to a group of Tier-2 sites, and hence the data flows in a more “hierarchical” fashion. The original CMS model had a focus on

coherent operation of all Tier-1 sites and allows data to flow between any pairing of Tier-1 and Tier-2 centers. The actual situation is even more complex, as there can be large size differences between Tier-1 or Tier-2 centers at different sites.

The computing models of the LHC experiments have undergone substantial evolution in the last 10 years, influenced by the operational experience that has been gained. The most important lesson is that reliably managing storage at sites is a difficult task, requiring specialized skills not always available at resource centers. In addition, the management of data and replica placement by the central production teams of each experiment requires substantial human resources, and the effort needed scales with the number of sites as opposed to the overall amount of storage. Hence, a model where a limited number of large sites provide highly reliable 10–100 Petabyte storage platforms, with the rest of the sites providing CPU power with input–output via Remote Data Servers, is globally more manageable and economical.

The original, static Tier hierarchy was motivated in part by the high costs and technical difficulties of deploying Gigabit per second (Gbps) WAN networking. However, networking in general and WAN in particular have had the fastest evolution of all computer industry technologies. In addition, Research and Education Networks provide a wider variety of connection options, including 10 and 100 Gbps links for specially demanding sites. This means that currently some Tier-2 sites have equal networking capabilities to a Tier-1 site, whereas they may or may not achieve the reliability levels of a Tier-1 site. It is convenient, then, to use these sites in a more flexible manner through a more dynamic hierarchical model.

Hence, the WLCG model has evolved towards differentiating two types of sites: Nucleation sites, which have high reliability and provide Petabyte-level storage, and satellite sites, which are mostly expected to deliver CPU capacity. WAN based input–output is accomplished through the custom *xrootd* or the standard *http* protocols, and the storage is remotely managed by the central experiment project teams via standard *WebDAV* protocols interacting with the Remote Data Servers at nucleation sites. Sites are continuously monitored and their status as nucleation points is defined dynamically according to their current reliability. In this model, most of the Tier-1 sites and larger Tier-2 sites act as nucleation sites. A site with reliability or availability problems will be temporarily removed as a nucleation site and restored when monitoring reveals the issues have been resolved. Tier-1 sites continue to have the differentiating feature of operating the robotic magnetic tape libraries used for bulk-archiving of the data.

14.8 Current State-of-the-Art Technologies

The computing industry continues to evolve at a rapid pace. The decade of the 2010s has seen the consolidation of a number of technologies and the emergence of an important change in business models for the provision of computing services. These are being integrated into the computing environments for particle physics.

14.8.1 *Multi-core CPUs and Virtualization*

Multi-core CPUs have been discussed in Sect. 14.1. Ideally, all applications would by now be multi-threaded and capable of efficiently utilizing the full capacity of the cores in a CPU. Unfortunately, thread-safe programming is a specialized skill and the vast majority of programs used in the world are not thread-safe. Hence, a different approach is needed to fully use multi-core CPUs.

In addition, applications running in distributed computing environments often have intricate dependencies on the underlying libraries and on the operating system. Continuously porting applications to newer versions of the underlying environment and verifying their proper operation requires much larger human resources than can be afforded by scientific communities (and even by many business communities). Hence, deployment of applications in these environments enters into conflict with system administration good practices, which require deploying new operating system versions soon after they become available.

These two factors have led to the re-emergence of two techniques from the past which enable operating system virtualization:⁷⁶ hypervisors and virtual machines (first deployed in the 1970s by IBM in its VM operating system⁷⁷), and containers (also known as jails, partitions or zones, first deployed commercially in the early 2000s by Sun Microsystems⁷⁸).

Virtual machines are implemented by deploying over the hardware a special type of operating system called a hypervisor. The hypervisor is designed to efficiently run multiple copies of a special type of application called a virtual image. In addition, newer versions of hardware implement special features to accelerate virtual image execution. The virtual image contains one or more applications together with the full operating system and library environment required by them. In this way, every time the hypervisor starts running an image it effectively bootstraps a new virtual computer, which is referred to as a virtual machine. Each image can have a different operating system and different libraries. If properly deployed, virtual machines allow to run older, less secure operating systems without compromising security, as the hypervisor can limit access to hardware and file system resources.

Virtual machines allow the implementation of many interesting features. One of the best known is live-migration,⁷⁹ which combines virtual machines with checkpoint/restart and high performance multi-connection file servers to be able

⁷⁶For a short, rather technical overview, see “Virtualization and Containerization of Application Infrastructure: A Comparison”, Thijs Scheepers, 21st Twente Student Conference on IT June 23rd, 2014, Enschede, The Netherlands. Available from the author at <https://thijs.ai/papers/scheepers-virtualization-containerization.pdf>

⁷⁷“The Origin of the VM/370 Time-Sharing System”, R. J. Creasy, IBM J. Res. Develop., Vol. 25, No. 5 (September 1981).

⁷⁸Price, Daniel, and Andrew Tucker. “Solaris Zones: Operating System Support for Consolidating Commercial Workloads.” LISA. Vol. 4. 2004.

⁷⁹Sapuntzakis, Constantine P., et al. “Optimizing the migration of virtual computers.” ACM SIGOPS Operating Systems Review 36.SI (2002): 377–390.

to move virtual machines between different hardware units. This can be useful for providing uninterruptable services or for saving energy in environments with large load variations.

Traditional offline batch computing in particle physics, however, predominantly uses virtualization for simply packaging together an application and its environment. This reveals a weakness of traditional virtualization, as the many copies of operating systems in the virtual machines require large amounts of memory (RAM) to execute efficiently, increasing the costs of the computing clusters. This has recently been circumvented by a new virtualization package, *Singularity*,⁸⁰ which has been rapidly adopted by the scientific community, and large portions of WLCG now support the execution of *Singularity* images. An additional advantage of *Singularity* is that it can be easily supported on supercomputers.

Containers, in their present form, represent a slightly different approach to virtualization. The current de-facto standard is *Docker*,⁸¹ which is widely used in commercial applications. Containers are closely associated to Linux operating systems and to Cloud Computing environments (see below). Containers implement virtual machines which are less hermetic than traditional ones and which have certain limitations. On the other hand, containers can be rapidly created and destroyed, and packages exist, such as *Kubernetes*,⁸² for orchestrating a set of containers which together can implement sophisticated applications using micro-service architectures.

In a somewhat unexpected way, *Singularity* and container environments have recently emerged as the leading approach for preserving data processing environments in order to ensure reproducibility of scientific results.

14.8.2 *Cloud Computing and the Use of Commercial Data Processing Services*

Cloud Computing⁸³ is the currently used term⁸⁴ to describe the deployment of computing resources and higher level computing services in a shareable, user configurable way. It is the new paradigm that replaced client-server computing in the

⁸⁰Kurtzer GM, Sochat V, and Bauer MW (2017) Singularity: Scientific containers for mobility of compute. PLoS ONE 12(5): e0177459. <https://doi.org/10.1371/journal.pone.0177459>

⁸¹Boettiger, Carl. "An introduction to Docker for reproducible research." ACM SIGOPS Operating Systems Review 49.1 (2015): 71–79.

⁸²Bernstein, David. "Containers and cloud: From lxc to docker to kubernetes." IEEE Cloud Computing 3 (2014): 81–84.

⁸³Mell, Peter, Grance, Tim, "The NIST Definition of Cloud Computing", Special Publication 800-145, National Institute of Standards and Technology, U.S. Department of Commerce (September 2011) <https://doi.org/10.6028/NIST.SP.800-145>

⁸⁴Regalado, Antonio. "Who coined 'cloud computing'." Technology Review 31 (2011). <https://www.technologyreview.com/s/425970/who-coined-cloud-computing/> Retrieved 3 February 2019.

2000s, starting a new cycle in the development of outsourced computing services. The first large scale commercialization was done in 2006, when Amazon launched the Elastic Compute Cloud (EC2) service. The first large-scale production cloud services in the academic domain were NASA's OpenNebula⁸⁵ and the services deployed in the RESERVOIR EU-funded project.⁸⁶

Cloud computing is still under heavy evolution, and a full description of the technology is beyond the scope of this work. It deploys interfaces and toolkits that allow users to configure virtual computers, storage servers, clusters and even networks. The ultimate goal is to use software to define the characteristics of a distributed computing service and then automatically map it to the needed hardware. This process is called provisioning.

Cloud computing can be used to implement private research data centers. CERN has migrated essentially all of its data center platforms to be managed by the *OpenStack*⁸⁷ cloud management system. However, the benefits of introducing cloud-style management in smaller data centers are not evident at present, as standard clusters are probably sufficient to fulfill the needs. In addition, there is at present a lack of personnel trained in cloud computing deployment and operation.

Cloud computing technologies started in the academic world, but are now driven by the cloud computing industry, which has grown in a decade to have sales of over 200 G\$/year. Cloud infrastructure companies deploy data centers which are much larger than academic computing centers, with the exception of a few academic supercomputing centers. Computer room floor areas above one hundred thousand square meters, power feeds above 50 MW⁸⁸ and capabilities for hosting tens of thousand of servers and close to a million cores are typical.

These huge deployments bring with them large economies of scale, making them attractive for scientific computing. In some cases, U.S. funding agencies have started to give grants for the purchase of commercial cloud computing services, replacing funding for purchasing research computing clusters. Gateways have been developed to include commercial clouds as part of WLCG, especially for CPU intensive tasks such as simulations. In the European Union, the HNSciCloud⁸⁹ pre-commercial procurement project in 2016–2018 was aimed to stimulate industry developments to enable hybrid clouds between research and commercial data centers capable of

⁸⁵Nebula Cloud Computing Platform (20 November 2012) <https://www.nasa.gov/open/nebula.html>

⁸⁶Rochwerger, Benny, et al. "The reservoir model and architecture for open federated cloud computing." *IBM Journal of Research and Development* 53.4 (2009): 4–1.

⁸⁷<https://www.openstack.org/software/>

⁸⁸For comparison, CERN's Meyrin data center power was 3.5 MW in 2018. "Data Centre: Key Info & Numbers". http://information-technology.web.cern.ch/sites/information-technology.web.cern.ch/files/CERNDDataCentre_KeyInformation_November2018V1.docx.pdf. Retrieved 3 February 2019.

⁸⁹Helix Nebula – The Science Cloud with Grant Agreement 687614 is a Pre-Commercial Procurement Action funded by H2020 Framework Programme. More information at <https://hnscicloud.eu>

executing data-intensive tasks. A similar project named ARCHIVER⁹⁰ will explore in 2019–2021 the commercial provision of cloud mass storage services for scientific data archiving and preservation.

Cloud computing deployment, especially on commercial services, is quite complex at the infrastructure and platform level. Fortunately, past investments in Grid computing can be re-used in order to hide these complexities from the vast majority of users. Job and data management packaged used for Grid computing have already been deployed on private and commercial clouds, as well as portals for analysis using higher level tools such as *Jupyter* notebooks and *Hadoop* data servers.

14.9 Future Challenges and Directions

Particle physics has historically been at the forefront of innovation in the adaptation of computing systems to fulfill its growing needs, as well as to accommodate tight budgets. It is rare, however, that the configurations developed in the particle physics context can directly be taken over by commercial systems. They should be considered as “precursors” or “early adoptions” of advances to come into general use. This situation has become a challenge with the growing time spans for experiments. If not properly managed, it may result in huge (often hidden) costs for maintenance and the lack of adoption of other innovations.

A first consideration for the future development is the number of scientific projects requiring wide-area distributed processing. In the short term, LHC will continue to produce unprecedented amounts of data and will be joined by a few experiments with similar needs, such as the Square Kilometer Array (SKA)⁹¹ radiotelescope. However, many other experiments, such as neutrino detectors and astroparticle experiments, may produce orders of magnitude smaller data volumes, due to their own instrumental nature or to advances in data reduction within the data acquisition platforms. For example, even though 100 Gbps WAN will become common place by the 2020s, the baseline design of the Cherenkov Telescope Array (CTA)⁹² requires only a 1 Gbps network interface for the sites (a capacity already available to home users in many countries). This is because CTA plans to use powerful clusters housed in compact containerized data centers placed at the sites to reduce the data at the instrument, thus avoiding the recording of large amounts of

⁹⁰ARCHIVER – Archiving and Preservation for Research Environments project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 824516. More information at <https://archiver-project.eu>

⁹¹Dewdney, Peter E., et al. “The square kilometre array.” *Proceedings of the IEEE* 97.8 (2009): 1482–1496.

⁹²Actis, M., et al. “Design concepts for the Cherenkov Telescope Array CTA: an advanced facility for ground-based high-energy gamma-ray astronomy.” *Experimental Astronomy* 32.3 (2011): 193–316.

raw data. Developments in the LHC domain, such as the LHCb turbo stream,⁹³ may also reduce the need for large-scale wide-area distributed processing. In parallel, as discussed below, data centers, particularly commercial ones, are growing in capacity. It is quite conceivable that the future needs of many experiments could be serviced by just a few data centers.

Another issue is the foreseen disappearance of magnetic tape technology. Tape has already disappeared from the commodity market, driven by ever lower disk prices and alternative backup media such as DVDs. If electrical energy considerations are excluded, the overall cost of high-end tape, associated robotic mechanisms and periodic migration to new media is becoming less attractive. Nevertheless, as of this writing, tape storage continues to be more economical than disk storage, especially if electrical costs are taken into account. The prospects are worrisome, however, as fewer companies continue to do R&D in tape technology. Hence, the particle physics community should be preparing for a hypothetical tape-less future by adapting its data management and cluster architectures as described above, allowing for multiple disk copies of the same dataset on a cluster or across a Grid or in several Clouds.

Grid and Cloud computing have brought along a resurgence of “timesharing”,⁹⁴ with research data centers deploying large clusters which give services to many projects of which only a minority are particle or astroparticle physics projects. The early adoption of Grids by particle physics, however, has resulted in development and deployment of a number of cluster and Grid tools which require extremely specific, non-commercially supported operating system or data server configurations which are of no interest to other projects or to the managers of general purpose clusters or Clouds. This does not mean that the developments led by particle physics are not worthwhile or of good quality; it simply means that further steps must be taken outside the particle physics scope to ensure a sustainable future and general applicability. Therefore, the particle physics community has to share knowledge about its developments, but also continuously evaluate alternative solutions. A case in point may be the use of non-standard data serving packages by LHC sites: it is perceived as the best solution (especially for LHC Tier-1 centers), but smaller particle or astroparticle projects with much lower data volumes often prefer what they perceive as a simpler, more standard solution such as NFS. Standard deployment of NFS on a cluster with thousands of CPU cores will however not be favored by system managers or commercial providers for a number of practical operational reasons, and they will propose newer technologies such as Ceph⁹⁵

⁹³Benson, Sean, et al. “The LHCb turbo stream.” *Journal of Physics: Conference Series*. Vol. 664. No. 8. IOP Publishing, 2015.

⁹⁴The term “timesharing” was coined in the 1960s to describe the simultaneous use of a single “mainframe” computer for multiple batch or interactive tasks by multiplexing processes into the mainframe CPU.

⁹⁵Weil, Sage A., et al. “Ceph: A scalable, high-performance distributed file system.” *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX Association, 2006.

or OpenStack swift⁹⁶ Object Storage. Therefore, a likely scenario is that current storage access techniques used by particle physics will become unsupported in the future. Of course, virtualization and Cloud techniques will make it possible for the particle physics community to deploy these legacy technologies themselves, with the corresponding expenses in personnel. A better way forward is to collaborate with industry, and between various Resource Centers, to test alternative tools and methods. Realistic evaluations are far from trivial, however, as they may require substantial investments by all parties, for example to provide UMD Grid or Cloud interfaces.

Another looming issue is the efficiency of use of the CPUs under data-intensive conditions. One basic assumption of the distributed architecture used by the particle physics community, as described above, is the existence of an infinitely powerful network connecting all elements, which in effect assumes infinitely powerful data servers as well as data ingestion by CPUs. Ever increasing data processing requirements and more and more powerful multi-core CPU nodes, however, are revealing data serving performance bottlenecks in clusters coming from the network and disk and CPU server limitations. Removing these bottlenecks may require more sophisticated architectures to be deployed.

A much more severe issue is related to the simplistic, often naïve, manner in which large particle physics projects pretend to use Grids and Clouds. They essentially desire them to behave as a very large cluster perfectly tuned to their needs, ignoring that imposing unreasonably high requirements in peak network and data serving rates within and between clusters can be very expensive. A Grid would better be viewed as a loosely connected federation of largely autonomous, self-sufficient clusters, with some of these clusters possibly hosted on commercial services.

These and other future challenges are being addressed in a coherent manner by the particle physics community through the HEP Software Foundation^{97,98} and will certainly require a vigorous new cycle of research and development, in collaboration with computer scientists and engineers, the supercomputing and HPC communities and industry.

⁹⁶Arnold, Joe. Openstack swift: Using, administering, and developing for swift object storage. "O'Reilly Media, Inc.", 2014.

⁹⁷Alves Jr, Antonio Augusto. A Roadmap for HEP Software and Computing R&D for the 2020s. No. HSF-CWP-2017-001; HSF-CWP-2017-01; FERMILAB-PUB-17-607-CD; arXiv: 1712.06982. Fermi National Accelerator Lab. (FNAL), Batavia, IL (United States); Brookhaven National Laboratory (BNL), Upton, NY (United States); Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States); SLAC National Accelerator Lab., Menlo Park, CA (United States); Thomas Jefferson National Accelerator Facility (TJNAF), Newport News, VA (United States); Argonne National Lab. (ANL), Argonne, IL (United States), 2017.

⁹⁸Slides from the Computing in High Energy Physics 2018 conference (proceedings to be published) https://indico.cern.ch/event/587955/contributions/3012294/attachments/1681524/2708636/CHEP18_-_CWP_Lessons_and_Future_Work.pdf

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

