**Data Architecture for eScience**

The data deluge is changing the nature of computing in Science and the architecture of systems designed to support it. Several important parameters differ between systems designed to support data analysis and those aimed at simulation. These include

* The Ratio of Disk (I/O) bandwidth to instruction execution rate (Amdahl’s I/O number [[1](#_ENREF_1), [2](#_ENREF_2)])
* The bandwidth and connection between source of data and computing system
* The nature of data – size and dynamic structure. Is it an instrument or sensor generating a time series or a repository

A traditional computer system is often organized as shown in figure 1 below.



*Figure 1: Computing System with a three level storage hierarchy supporting multiple clusters where “the work” gets done. Each computer C has its own local disk not shown. S represents a storage node.*

Figure 1 shows a three level data hierarchy with typically temporary data stored on cluster nodes, a shared set of files and a backend archival storage. The shared files are shown in figure as either managed by computers in hosted storage or as dedicated (SAN/NAS/etc.) storage. The shared file system for scientific computing may support high performance distributed file systems such as Lustre or GPFS. This architecture is used for both data and simulation intensive work with good success. There are many attractive features of this architecture including separation of concerns -- storage and its backup are managed separately from the possibly large number of clusters supported, computers and storage can be separately upgraded and a single storage system (and single copy of a data item) supports all computing venues. There is an obvious problem in data intensive applications that the bandwidth between the compute and data system components may be too small. Note clusters typically have bisection bandwidths that are very large scaling up with system size. However the link between storage and compute subsections are typically provisioned with static number of interconnects (perhaps some number of Gigabit or 10Gigabit Ethernet connections). Even simulation systems see the same issues [[3-5](#_ENREF_3)] at the largest scales when programs output data (for visualization) at volumes that overwhelm the connection to shared storage. Note important technologies like MPI-IO are built around this model.

An alternative architecture shown in figure 2 addresses this issue by using “data parallel file systems” DPFS such as Google File System (MapReduce)[[6](#_ENREF_6)], HDFS (Hadoop)[[7](#_ENREF_7)], Cosmos (Dryad)[[8-10](#_ENREF_8)] and Sector [[11](#_ENREF_11)] with compute-data affinity optimized for data processing. This design was motivated by Internet search – perhaps the application with the largest data – but has seen little practical use outside that area.



*Figure 2: Data Parallel File Systems showing disks attached to compute nodes with files broken into blocks and stored across multiple computers with replication for fault tolerance.*

Here we have a simpler architecture with a uniform array of computer nodes with (large) local disks. Adding 12-18TB of disk per node is quite practical these days. User Files are broken up into blocks, which are replicated around 3-6 times and spread across different nodes and different clusters. This architecture allows one to support “bringing the computing to the data” [[12](#_ENREF_12), [13](#_ENREF_13)]. Archival storage is not necessary – all copies can be stored on spinning disks. The copies should be designed that some are near each other to support local computing whereas at least one should be “far off” to provide a safe back up. Note the disks and compute nodes within a cluster are linked to the scalable cluster interconnect and so good performance in fetching data from disk does not require computing to be on node where data stored but rather on a node with a high performance (cluster) interconnect to data.

Figure 3 shows an investigation of the importance of “exactly” co-locating processing and data on FutureGrid with a Hadoop job reading a large distributed data set. The figure compares the execution time for Hadoop I/O job for four scheduling choices: *default*, for Hadoop with compute-data collocated; *random*, compute placed randomly in same cluster as data; *0.5 random*, where the Hadoop and random strategies are chosen with equal probability; and *HPC Style*, which has computing and data on different clusters in Chicago and Bloomington. The slowdown for non-optimized placement of computation is at most 20% and is less for the fastest case with 32 Hadoop mappers on each node. These performance studies need to be extended but suggest that data-parallel file systems will often be used and that computing will be assigned near but not exactly collocated with the data.

The architecture of figure 2 supports several data management systems including both databases and the important NOSQL developments [[14](#_ENREF_14), [15](#_ENREF_15)] constructs such as Bigtable [[16](#_ENREF_16)], SimpleDB [[17](#_ENREF_17)] in Amazon and Azure Table [[18](#_ENREF_18)] while databases (for example SciDB and GrayWulf projects [[2](#_ENREF_2), [19](#_ENREF_19)]) essentially collocate data and associated processing. NOSQL technologies emphasize distribution and scalability while their support of simple tables in interesting given that tables are clearly important in science as illustrated by the VOTable standard in astronomy [[20](#_ENREF_20)] and the popularity of Excel. However, there does not appear to be substantial experience in using tables outside clouds. It seems likely that tables will grow in importance for scientific computing, and academic systems could support this using two Apache projects: Hbase [[21](#_ENREF_21)] for BigTable and CouchDB [[22](#_ENREF_22)] for a document store. Another possibility is the open source SimpleDB implementation M/DB [[23](#_ENREF_23)].

a)

b)

*Figure 3: Dependence of Hadoop runtime (plotted is runtime relative to default with 8 mappers per node) on placement of computation in FutureGrid India cluster using gigabit interconnect. The dataset is 400 gigabytes in a) and 100 gigabytes in b). The graph marked “HPC Style” in b) corresponds to 10 data nodes on Hotel and 10 compute nodes on India*

There are research issues both in data parallel file systems themselves and in their integration with programming models [[24](#_ENREF_24)] and runtimes like Twister/MapReduce[[25](#_ENREF_25)]. A key change – illustrated by the Purlieus [[26](#_ENREF_26)] project – is that the scheduling problem now is one of both data and computing rather the usual approaches which just schedule computing tasks. Here there is an important issue about locating user files. In the architecture of figure 1 one only needs to place files in the shared file system to allow access for all clusters and applications. In figure 2, one has to be more careful and place the data on or “near” (in terms of scalable connectivity) the compute systems that will be allocated to users of the files. There does not seem any experience in supporting the architecture of figure 2 in a complicated heterogeneous multi-user environment. The problem is easier if one just has a few very clusters as is in fact used in clouds used for Internet search. Data parallel file systems on a grid of many (small) clusters seem difficult to use. More study is clearly needed here for multi-user environments in real data centers using data parallel file systems with multiple clusters.

Although DPFS originated in the cloud (internet search) arena, commercial clouds tend to use a rather different object store model seen in Amazon, Azure and the open source OpenStack system. Here one assigns a set of nodes to be storage servers as in the top of the middle layer of figure 1 but rather than the full distributed file semantics of Lustre, one supports a simple object model. Objects have containers and metadata with operations such as get, put, update, delete, and copy objects. Again there is little experience with this in scientific computing arena. The Simple Cloud APIs [[27](#_ENREF_27)] for file storage, document storage services, and simple queues could help in providing a common environment between academic and commercial clouds. There is also some engineering-oriented work [[28](#_ENREF_28)] involved relating these different file systems, so one can run applications – possibly with performance degradations – however the data is stored. This is closely related to data movement as one needs to change between storage modes and possibly use a replica system as one does for data grids [[29](#_ENREF_29)].

A traditional approach to scientific data establishes a repository which stores the data and meta-data of a given experiment or set of experiments. This has and will play a critical role but is often inadequate for the common case of enormous amounts of data requiring enormous computing. We really need the data archives to be attached to an appropriate compute resource as it is often impractical for individual researchers to download their data to home compute resources. Genomic data is an important example as Blast and related processing can be very compute intense [[30-33](#_ENREF_30)]. Note that NIH recently announced closure of a petabyte database [[34](#_ENREF_34)] as they could not support it.

**References**

1. Gordon Bell, Jim Gray, and Alex Szalay, *Petascale Computational Systems: Balanced CyberInfrastructure in a Data-Centric World (Letter to NSF Cyberinfrastructure Directorate).* IEEE Computer, January, 2006. 39(1): p. 110-112. <http://research.microsoft.com/en-us/um/people/gray/papers/Petascale%20computational%20systems.doc>

2. Yogesh Simmhan, Roger Barga, Catharine van Ingen, Maria Nieto-Santisteban, Lazslo Dobos, Nolan Li, Michael Shipway, Alexander S. Szalay, Sue Werner, and Jim Heasley, *GrayWulf: Scalable Software Architecture for Data Intensive Computing*, in *42nd Hawaii International Conference on System Sciences*. 2009, IEEE. Waikoloa, Big Island, Hawaii. pages. 1-10. DOI: <http://doi.ieeecomputersociety.org/10.1109/HICSS.2009.750>.

3. C.S. Chang. *Needs for Extreme Scale DM, Analysis and Visualization in Fusion Particle Code XGC*. [accessed 2011 July 23]; Center for Plasma Edge Simulation CPES at Exascale data management, Analysis, and Visualization, Feb. 22-23, 2011, Houston, TX, Available from: <http://www.olcf.ornl.gov/wp-content/uploads/2011/01/CSChang_Fusion_final.pdf>.

4. *Scientific Grand Challenges: Fusion Energy Sciences and the Role of Computing at the Extreme Scale*. [accessed 2011 July 15]; U. S. Department of Energy, March 18-20, 2009. Washington DC Available from: <http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Fusion_report.pdf>.

5. Jacqueline Chen and John Bell. *Combustion Exascale Co-Design Center*. [accessed 2011 July 15]; 6th International Exascale Software Project Workshop, San Francisco, CA, April 6-7, 201 Available from: <http://www.exascale.org/mediawiki/images/3/38/Talk29-Chen.pdf>.

6. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, *The Google file system.* SIGOPS Oper. Syst. Rev., 2003. 37(5): p. 29-43. DOI:10.1145/1165389.945450

7. *Hadoop Distributed File System HDFS*. 2009 [accessed 2009 December]; Available from: <http://hadoop.apache.org/hdfs/>.

8. Yu, Y., M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P.K. Gunda, and C. J., *DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language*, in *Symposium on Operating System Design and Implementation (OSDI)*. December 8-10, 2008. San Diego, CA.

9. Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly, *Dryad: distributed data-parallel programs from sequential building blocks.* SIGOPS Oper. Syst. Rev., 2007. 41(3): p. 59-72. DOI:10.1145/1272998.1273005

10. Jaliya Ekanayake, Thilina Gunarathne, Judy Qiu, Geoffrey Fox, Scott Beason, Jong Youl Choi, Yang Ruan, Seung-Hee Bae, and Hui Li, *Applicability of DryadLINQ to Scientific Applications*. January 30, 2010, Community Grids Laboratory, Indiana University. <http://grids.ucs.indiana.edu/ptliupages/publications/DryadReport.pdf>.

11. Gu, Y. and R. Grossman, *Sector and Sphere: The Design and Implementation of a High Performance Data Cloud.* Crossing boundaries: computational science, e-Science and global e-Infrastructure I. Selected papers from the UK e-Science All Hands Meeting 2008 Phil. Trans. R. Soc. A, 2009. 367: p. 2429-2445.

12. Gray, J., *Jim Gray on eScience: A transformed scientific method*, in *The Fourth Paradigm: Data-intensive Scientific Discovery.* 2009, Microsoft Reasearch. p. xvii-xxxi.

13. Jim Gray, Tony Hey, Stewart Tansley, and Kristin Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. 2010 [accessed 2010 October 21]; Available from: <http://research.microsoft.com/en-us/collaboration/fourthparadigm/>.

14. NOSQL Movement. *Wikipedia list of resources*. 2010 [accessed 2010 June 5]; Available from: <http://en.wikipedia.org/wiki/NoSQL>.

15. NOSQL Link Archive. *LIST OF NOSQL DATABASES*. 2010 [accessed 2010 June 5]; Available from: <http://nosql-database.org/>.

16. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and R.E. Gruber, *Bigtable: A Distributed Storage System for Structured Data*, in *OSDI'06: Seventh Symposium on Operating System Design and Implementation*. 2006, USENIX. Seattle, WA. <http://labs.google.com/papers/bigtable-osdi06.pdf>.

17. Amazon. *Welcome to Amazon SimpleDB*. 2010 [accessed 2010 June 5]; Available from: <http://docs.amazonwebservices.com/AmazonSimpleDB/latest/DeveloperGuide/index.html>.

18. Jai Haridas, Niranjan Nilakantan, and B. Calder. *WINDOWS AZURE TABLE* 2009 May [accessed 2010 June 5]; Available from: <http://go.microsoft.com/fwlink/?LinkId=153401>.

19. *SciDB: Open Source Data Management and Analytics Software for Scientific Research*. [accessed 2011 May 25]; Available from: <http://www.SciDB.org>.

20. International Virtual Observatory Alliance. *VOTable Format Definition Version 1.1*. 2004 [accessed 2010 June 5]; Available from: <http://www.ivoa.net/Documents/VOTable/20040811/>.

21. Apache. *Hbase implementation of Bigtable on Hadoop File System*. 2010 [accessed 2010 June 5]; Available from: <http://hbase.apache.org/>.

22. Apache. *The CouchDB document-oriented database Project*. 2010 [accessed 2010 June 5]; Available from: <http://couchdb.apache.org/index.html>.

23. M/Gateway Developments Ltd. *M/DB Open Source "plug-compatible" alternative to Amazon's SimpleDB database*. 2009 [accessed 2010 June 5]; Available from: <http://gradvs1.mgateway.com/main/index.html?path=mdb>.

24. Zach Hill and Marty Humphrey, *CSAL: A Cloud Storage Abstraction Layer to Enable Portable Cloud Applications*, in *Second International Conference on Cloud Computing Technology and Science (CloudCom)*. November 30-December 3, 2010, IEEE. Indianapolis. pages. 504-511. <http://www.cs.virginia.edu/~humphrey/papers/CSAL.pdf>. DOI: <http://dx.doi.org/10.1109/CloudCom.2010.88>.

25. Patrick Donnelly, Peter Bui, and Douglas Thain, *Attaching Cloud Storage to a Campus Grid Using Parrot, Chirp, and Hadoop*, in *Second International Conference on Cloud Computing Technology and Science*. November 30-December 3, 2010, IEEE. Indianapolis. pages. 488-495. [http://www.cse.nd.edu/~ccl/research/papers/chirp+parrot+hdfs.pdf](http://www.cse.nd.edu/~ccl/research/papers/chirp%2Bparrot%2Bhdfs.pdf). DOI: <http://dx.doi.org/10.1109/CloudCom.2010.42>.

26. Balaji Palanisamy. *Purlieus: Locality-aware Resource Allocation for MapReduce in a Cloud*. 2011 [accessed 2011 June 12]; Available from: <http://www.cc.gatech.edu/~pbalaji/purlieus/>.

27. Zend PHP Company. *The Simple Cloud API for Storage, Queues and Table*. 2010 [accessed 2010 June 1]; Available from: <http://simplecloud.org/>.

28. Xiaoming Gao, Yu Ma, Marlon Pierce, Mike Lowe, and Geoffrey Fox, *Building a Distributed Block Storage System for Cloud Infrastructure*, in *CloudCom 2010*. November 30-December 3, 2010. IUPUI Conference Center Indianapolis. <http://grids.ucs.indiana.edu/ptliupages/publications/VBS-Lustre_final_v1.pdf>.

29. Sudharshan Vazhkudai, Steven Tuecke, and Ian Foster, *Replica Selection in the Globus Data Grid*, in *Proceedings of the 1st International Symposium on Cluster Computing and the Grid CCGrid*. 2001, IEEE Computer Society. pages. 106.

30. Stein, L., *The case for cloud computing in genome informatics.* Genome Biology, 2010. 11(5): p. 207. <http://genomebiology.com/2010/11/5/207>

31. Elizabeth Pennisi, *Will Computers Crash Genomics?* Science, February 11, 2011, 2011. 331(6018): p. 666-668. DOI:10.1126/science.331.6018.666. <http://www.sciencemag.org/content/331/6018/666.short>

32. Schatz, M. *Cloud Computing and the DNA Data Race*. 2011 [accessed 2011 June 26]; Keynote Presentation at 3DAPAS/ECMLS workshops at HPDC June 8 2011 Available from: <http://schatzlab.cshl.edu/presentations/2011-06-08.HDPC.3DAPAS.pdf>.

33. NIH. *DNA Sequencing Costs: Data from the NHGRI Large-Scale Genome Sequencing Program*. [accessed 2011 May 20]; Available from: <http://www.genome.gov/sequencingcosts/>.

34. Uduak Grace Thomas. *NCBI to End Support for Sequence Read Archive as Federal Purse Strings Tighten*. 2011 February 18 [accessed 2011 August 27]; Available from: <http://www.genomeweb.com/informatics/ncbi-end-support-sequence-read-archive-federal-purse-strings-tighten>.