SAND2013-6833C

Investigating the Integration of Supercomputers and Data-Warehouse Appliances

Ron A. Oldfield, George Davidson, Craig Ulmer, and Andrew Wilson

Sandia National Laboratories**
{raoldfi,gsdavid,cdulmer,atwilso}@sandia.gov

Abstract. Two decades of experience with massively parallel supercomputing has given insight into the problem domains where these architectures are cost effective. Likewise experience with database machines and more recently massively parallel database appliances has shown where these architectures are valuable. Combining both architectures to simultaneously solve problems has received much less attention. In this paper, we describe a motivating application for economic modeling that requires both HPC and database capabilities. Then we discuss hardware and software integration issues related to a direct integration of a Cray XT supercomputer and a Netezza database appliance.

1 Introduction

High-performance computing (HPC) systems, particularly large-scale parallel supercomputers, have traditionally been exclusively used to solve complex scientific problems; however, recent interest in data-analytic problems in the areas of cyber security, social networking, economic modeling, and others, along with the exponential growth of data required for these applications, is motivating the use of HPC systems to solve these problems as well.

Some analytics applications are particularly well suited for HPC computing because of their numerical requirements. For example, applications that use latent semantic analysis (LSA) [7] and eigen-value decomposition could make use of highly-tuned numerical algorithms such as those in Sandia's Trilinos numerical solvers [14]. Other applications could exploit the tremendous in-core memory of HPC systems. While cluster solutions require substantial culling of the data before analysis, a supercomputer could analyze much larger data sets, perhaps identifying global relationships among the data that could not be found at a smaller scale.

Although HPC platforms provide effective capability to address numerical-computing requirements of analytics codes, they are missing other key features. For example, a common objective is to find relationships between elements in a large data-warehouse appliances (DWA) such as a Netezza, XtremeData, LexisNexis, and others. Social networking and document clustering both fit into this category. Other applications, like

** Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia's N-ABLE code [10], generate data that has to be "ingested" into a DWA for post analysis. While HPC systems provide access to large parallel file systems, there is often no support for database access. In addition, HPC systems are designed for fast access through proprietary networks to internal resources, but are not designed for efficient access to remote resources. Thus, providing network access from an HPC system to a remote DWA creates severe bottlenecks.

The increasing desire to support a dual-use for HPC platforms that includes traditional physics-driven science codes and data-driven analytics is creating serious concerns for system architects. One approach is to design a system capable of efficiently handling both workloads, but the characteristics of science-based and analytics-based applications differ enough that this approach may not be practical or cost-effective. An alternative is to find ways to tightly couple HPC systems and data-warehouse appliances to exploit each systems special hardware and software features. In this paper, we explore the second option.

This paper describes an effort at Sandia to couple a Cray XT4 and a small Netezza DWA. We describe a motivating use case based on economic modeling, describe the necessary software and hardware modifications to support such an integration, and perform three different experiments to evaluate our approach.

2 Motivating Use Case: Analysis of Complex Economic Systems

A number of interesting use cases exist to motivate the integration of HPC and DWA systems, but there is one particular Sandia application we targeted for this work, economic modeling. The National Infrastructure Simulation and Analysis Center (NISAC) is a center at Sandia National Laboratories that analyzes economic impacts and interdependencies of disruptions in our national infrastructure. Disruptions include, for example, changes in U.S. border security technologies, terrorist acts on commodity futures markets, disruptions in food supply chains, and power and rail disruptions.

The tool used for economic modeling is the NISAC Agent-Based Laboratory for Economics (N-ABLE) [10]. N-ABLE is an agent-based microsimulation tool with the goal of modeling complex economic systems through the interactions of massive numbers of individual entities. N-ABLE agents employ simple market-based rules for production and consumption that, when considered economy-wide, lead to "emergent" behavior that would be hard to predict in a purely mathematical model.

N-ABLE agents are simple pieces of C++ code that model economic activity of entities such as banking firms, power companies, telecommunication firms, and others. On HPC systems, a simulation consists of tens of thousands of agents that communicate events using the Message Passing Interface (MPI).

Agent simulations produce "snapshots" at periodic intervals that represent the set of transactions executed by representative agents since the last snapshot. Data from the simulation needs to be ingested into a database for interactive exploratory analysis; however, constraints in I/O performance and functionality make this process cumbersome and slow. The standard approach is to stage this data to files. An out-of-band, sequential, process then merges and ingests that data into the database.

The primary goal of the N-ABLE simulation is to rapidly identify key economic impacts of changes to our national infrastructure. To address those concerns in a timely manner it is critical to minimize the total time to solution for the entire workflow, which includes the HPC simulation, ingest of results into a database, and analysis of the simulation results by analysts. In the current implementation, the most significant time constraint is the time to ingest simulation data into the database.

N-ABLE is a great candidate for an integrated HPC/DWA system. The snapshot operation could avoid the off-line staging process by ingesting data directly into the database, and a tight coupling of a DWA and HPC could reduce ingest time, addressing the network bottleneck between the HPC system and the database host.

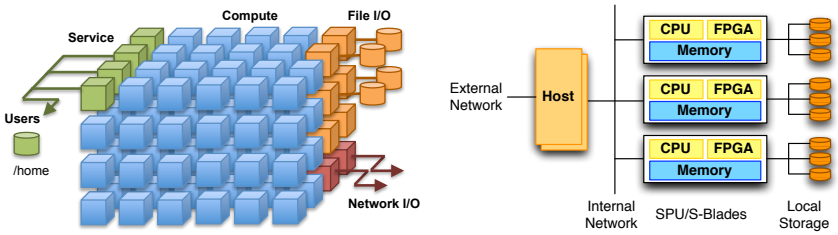
3 Architectural Differences Between HPC and DWA

Many years of research have gone into hardware and software design for HPC and DWA systems. In both cases, systems are designed to support a relatively small set of applications. While HPC systems are generally designed for problems in *computational science* [21], DWAs are primarily designed for analytics applications in business intelligence [17] or cyber intelligence [25]. As we discuss in the next few paragraphs, the hardware and software needed to support these applications do not always overlap.

3.1 HPC Systems

Computational science applications typically model some physical phenomena such as predicting climate changes, simulating nuclear fusion, or simulating the interactions between molecules. To model this behavior, scientists rely on approximations that use parallel numerical methods to solve partial differential equations. The finite-element and finite-difference methods used to solve these equations have distinct phases of computation, communication, and I/O; and require effective load balancing to maximize resource utilization [4]. These requirements motivated supercomputing vendors to build large-scale distributed-memory systems with high-speed, low latency networks and powerful parallel file systems. Software and allocation strategies are designed to be predictable and simple, using a space-sharing as opposed to time sharing for node allocation, explicit message passing, and user control of memory management.

Figure 1a illustrates the partitioned architecture typically used for parallel supercomputers. This is the model used by the Cray XT, XE, and XMT series and the IBM BlueGene series of supercomputers. A partitioned architecture system [13] is comprised of compute nodes that use a lightweight kernel [16, 24] operating system and service nodes running a full-fledged operating system, typically some variant of Linux. By design, lightweight kernels provide a very limited set of functionality to the application. In the case of Catamount [15] and Compute Node Kernel [24], the operating systems used by Cray XT and IBM, there is no support for demand paged virtual memory, no support for local disk, and little or no support for multi-threading. Cray Linux Environment [27], the operating system used by compute nodes of the more modern multi-core systems such as the Cray XT and XE sysetms, provides some of these services, but is still somewhat restricted when compared to full Linux distributions.



(a) HPC System Architecture. (b) Netezza System Architecture.

Fig. 1: Architecture of a traditional supercomputer and Netezza DWA [11].

In addition to the specialized operating systems, high-end parallel supercomputers typically use proprietary networks with specialized protocols. For example, the Cray XT3 and XT4 use the SeaStar network interface [2] that provides extremely high bandwidth, but is only accessible through the Portals programming interface [3]. Cray XE systems use the Gemini interconnect capable of millions of MPI messages per second [1], but it also uses the proprietary uGNI API for access. These systems have limited support for TCP on the compute nodes, and no support for ODBC, the standard API used to access remote SQL databases.

3.2 Data Warehouse Appliances

Data warehouse appliances (DWA) are systems with special hardware to process relational queries (e.g., SQL) on massive datasets. The history of special database architectures for SQL processing goes back many decades to the use of data base computers (DBC) [8]. Commercial uses for database processing became the driver for more advanced capabilities later used for scientific computing. For example, the Teradata DBC/1012 [23] developed in the early 1980s was one the first to demonstrate commercially viable use case for massively parallel processing (MPPs).

The Netezza [11], formed in 2000 and acquired by IBM in 2010, was one of the first companies to treat the database machine as a true appliance [6], exposing the machine through a query interface, but providing powerful storage and processing capabilities internally. While there are now a number of companies offering hardware and software support for SQL-based data-warehouse appliances such as XtremeData, Oracle, Teradata, and Greenplum, we focus our discussion on the Netezza architecture.

In the same way Teradata demonstrated commercial uses for MPP, the Netezza system was the first to demonstrate a commercially-viable use case for active storage [22]. The Netezza system, illustrated in Figure 1b, leverages key concepts of the active-storage research to construct an MPP SQL-processing system that performs most of the processing at or near the stored data. They even employ field-programmable gate arrays (FPGAs) to filter data before it even hits the processor memory. A host presents a standard interface to external applications, compiles SQL queries into distributable code segments called “snippets” that then get passed to the MPP nodes–originally called snippet processing units (SPUs), now called S-Blades–for processing. The S-Blades contain multi-core CPUs, multi-engine FPGAs, and multiple gigabytes of memory.

Since snippet programs are generally "embarrassingly parallel", there is no need for a proprietary interconnect, such as those found on HPC supercomputers. 1Gib Ethernet, along with proprietary network protocols designed specifically for query-based traffic provides sufficient bandwidth to handle typical workloads.

3.3 Summary of Differences Between Architectures

The primary differences in HPC architectures and DWA architectures is reflected in the diverging requirements of the supported applications. Here we contrast application requirements for three important components: computation, communication, and I/O.

Historically, computation of floating point operations (flops) is considered the most important quality of an HPC system. Since 1993, the LINPACK benchmark [9], used to calculate the top 500 supercomputers, has been the most common metric used to rate HPC systems. Thus, HPC architects are continually delivering systems with leading-edge CPU performance and parallelism. DWA designers, at least in the case of Netezza, seem less interested in aggregate CPU performance and are more interested in providing the minimum amount processing required to keep up with the disk I/O rates.

In the same way computation is a critical requirement for HPC systems, communication is equally important, and essentially the distinguishing feature separating HPC systems and commodity clusters. The major supercomputing vendors like IBM and Cray use proprietary interconnects to provide this performance. In contrast, DWA architects put much less emphasis on a high-speed interconnect. Computing distributed SQL queries is essentially a streaming computation problem [20] where most of the data is filtered very close to the disk device. DWA systems do not need an expensive proprietary network to get good parallel performance.

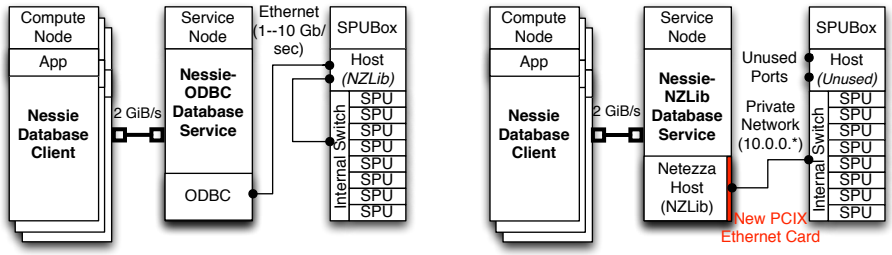
The I/O and storage systems are also very different in supercomputers and DWA systems. Supercomputing applications are often write intensive, writing large volumes of simulation results and checkpoint files [5]. DWA applications are typically read-intensive, performing most of their I/O for queries as part of an analysis operation. The I/O system of an HPC is often separate from the compute nodes, as illustrated in Figure 1a, requiring an efficient parallel file system. In contrast, although there is a tremendous amount of aggregate I/O for DWA applications, Netezza is an active-storage [22] architecture that filters and processes most of this data on nodes connected to the storage devices. The devices are independent and do not require a parallel file system for access.

4 An Integration Environment for HPC and DWA

The goal of this work is to explore application workflows that need capabilities of both supercomputers and DWAs. However, a single general-purpose system, designed for both types of applications, does not exist. Instead of trying to force one system to manage both workloads, we explore two ways to connect a supercomputer with a DWA. The first integration is a "loose" coupling that uses a standard Ethernet to connect to a remote Netezza. The second approach is a "tight" coupling that places a Netezza front-end host on the fast interconnect of the HPC system. In this section, we describe the software and hardware modifications needed to support these approaches.

For our experiments, we integrated a Cray XT4 with a small SPUBox Netezza system. Our Cray XT4 consists of a single cabinet containing 94 2.1 GHz quad-core AMD Opteron processors, with 2 GB of memory per core. The nodes are connected by a SeaStar router [2] using a 3-dimensional torus topology capable of 2 GiB/s transfers per link. The service nodes have an additional external 1 Gib Ethernet card that allows connections to/from the machine. The compute nodes of the Cray XT4 use a light-weight kernel operating system called the Cray Linux Environment.

The Netezza SPUBox [1] uses the older generation "Mustang" architecture consisting of a single front-end host and 8 back-end snippet-processing units (SPUs). Each SPU contains a single PowerPC for computation, an FPGA, and a SATA controller for the disk. The host is connected to the SPUs through a 1 Gib/s Ethernet router. An additional 1 Gib/s Ethernet card provides external access to the host node. The host uses a standard Linux operating system and proprietary software, called *NZLib*, that manages processing of code snippets on the back-end SPUs.



(a) Loosely-coupled integration. (b) Tightly-coupled integration.

Fig. 2: Organization of hardware for HPC/DWA integration experiments.

4.1 Loosely-Coupled Integration of Cray and Netezza

Figure 2a illustrates the configuration used to integrate the Cray XT4 and a remote Netezza SPUBox. Since the compute nodes of the XT4 do not have access to an external network, we developed a "proxy" database service [19] using the Network Scalable Service Interface (Nessie) [18]. Nessie provides a remote procedure call abstraction, along with RDMA functions to perform direct memory-to-memory transfers between compute and services nodes on the high-speed interconnect.

Our database proxy aggregates and forwards database requests to servers that have connectivity to the external network. The servers, running a full linux OS, use standard interfaces to access the remote Netezza database. Our servers use the implementation of ODBC provided by Netezza.

[1] A commercial version of the SPUBox is now available as the IBM Netezza 100.

4.2 Tightly-Coupled Integration of Cray and Netezza

Although the loosely-coupled approach provides access to a remote database, we anticipate a significant bottleneck between the ODBC service and the remote database. The SeaStar network provides 2 GiB/sec bandwidth per link to the service, but our Ethernet link provides 1 Gib/sec (a factor of 16 slower) from the ODBC service to the Netezza. In addition the Portals interface can achieve nearly full bandwidth of the SeaStar network, but congestion-control algorithms of TCP/IP limit network bandwidth, and overheads in the ODBC protocols further limit performance.

To improve data-transfer rates between the Cray XT system and Netezza, we developed a hardware-integrated solution that allows the host functionality of the Netezza to be resident on a service node of an XT system. Figure 2b shows the system architecture. This configuration allows efficient access from compute nodes to the host through Portals, and we could avoid overheads of software abstraction layers such as ODBC by leveraging the host libraries (*NZLib*) provided by Netezza to interact directly with the back-end SPUs. In particular, we want to leverage the bulk ingest capability called *NZLoad* that is available in NZLib but not in the ODBC interface.

5 Evaluation

To evaluate performance of the different integration schemes, we developed a simple parallel benchmark to generate and ingest 4 MiB of data per client process into a database. The intent is to emulate the I/O behavior of the N-ABLE economic-modeling code. Our experiments measure three different use cases:

ODBC: The loosely-coupled integration described in Figure 2a. We measure the time to generate and ingest data into a remote Netezza database through the ODBC database proxy.

NZLoad: The tightly-coupled integration described in Figure 2b. We measure the time to generate and ingest data through our hardware-integrated platform that uses the bulk ingest NZLoad operation from an HPC service node.

SQLite: A *control* experiment that measures the time to ingest data directly to a shared SQLite database on the HPC parallel file system.

Figure 3 shows combined results of the three different experiments as we increase the number of bytes ingested into the database. As expected, the loosely-couple case, although it demonstrates useful functionality, performs very poorly, taking over 40 minutes to ingest 16 KB of data. This poor performance is due to known issues with ingest using SQL [12]. The issue is that the typical enterprise use of databases assumes that databases are built up gradually over time. Ingesting data using the INSERT command only allows one row at a time to be ingested. Considering each INSERT command is a remote network transfer, it is no surprise this benchmark performs so poorly.

Writing to SQLite files performs much better than remote ODBC, but still starts to degrade rapidly when approaching 1 MB of data. This case suffers the same SQL-interface limitations as the ODBC experiment. In both of these cases, it is the interface, not the hardware that is limiting bulk-ingest performance.

The final experiment, NZLoad, demonstrates a hardware-integration and use of the bulk-ingest capability of the Netezza libraries performs orders of magnitude faster than the other two experiments, achieving an ingest performance of 256 MB in 150 seconds.

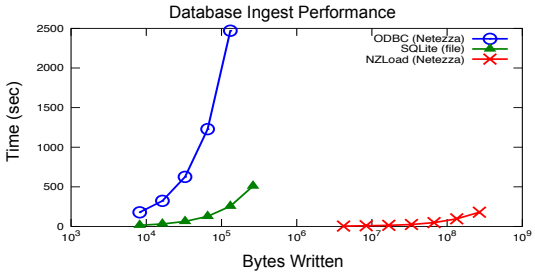
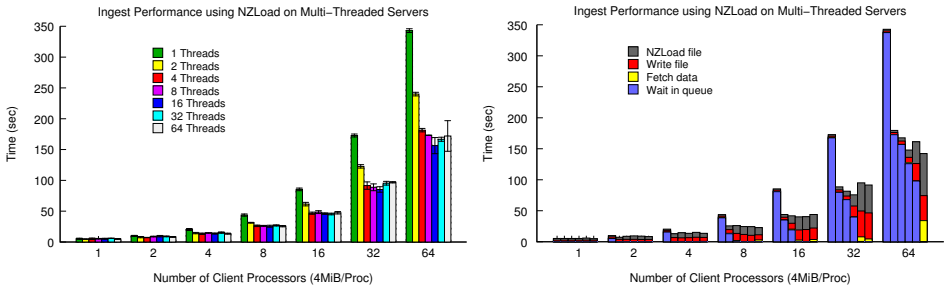


Fig. 3: Ingest performance of the three different experiments.

The next two figures look in detail at the hardware-integration experiment. To improve performance, our Netezza collaborators suggested multi-threading on the host to increase the number of outstanding requests. Figure 4a shows scaling results of the NZLoad experiment for 1 to 64 client processors, each writing 4MiB to a single server that uses up to 64 threads to send data to the back-end database. The x-axis shows the number of client nodes sending data to the database, and also represents the number of outstanding requests. The plots show a large boost in performance between 1 and 2 threads, but no significant advantage to using more than 2 threads per server.



(a) Thread Timings. (b) Operation Timings.

Fig. 4: Multithreaded ingest performance for hardware-integrated solution. Columns show performance using different number of threads on the service node of the Cray.

To better understand results from Figure 4a, we created a stack plot (Figure 4b) showing the time to fetch data from the client, write data to a file, ingest/load data into the database, and finally the time to wait if the system is busy. The plots show that four threads provides a sufficient level of concurrency to overlap data-transfers and writes with the ingest (NZLoad) to the database. Any more than that has no benefit. This is

not surprising given that we are piping all our data through a single network interface. Additional parallelism to the back-end would likely provide better results.

6 Conclusions and Future Work

Recent interest in using supercomputers for data-analytic operations are causing HPC system architects to explore interesting new designs. In this paper, we propose an integration of two special-purpose architectures: a supercomputer designed primarily for scientific simulations, and a data-warehouse appliance, tuned for analytics jobs.

Our economic modeling application requires rapid ingest into a relational database for analysis. This bulk-ingest problem created serious issues when using SQL-based tools, but showed a clear benefit from a tight coupling that exploited bulk-ingest capabilities provided in the vendor libraries. Another alternative is to explore data-parallel approaches such as LexisNexis DAS, Sector and Sphere, IBM's InfoSphere, and Hadoop. There has already been a good study on using these types of data-warehouse appliances for scientific data analysis [26]. Perhaps similar benefits exist for economic modeling.

Although we propose an integration of two special-purpose machines, it is not clear that an HPC/DWA integration would be difficult with existing supercomputer hardware. For example, an extra partition of "special" DWA nodes, along with Netezza software, could provide capabilities useful for analytics on the fast network.

Although we expect exascale architectures to motivate the integration of simulation and analytics, we have not fully explored the space of possible applications that would benefit from such an architecture. In addition, these new problem domains introduce a number of issues in the operating system, runtime system, and programming models that lead to a possible explosion of research opportunities in the next decade.

7 Acknowledgements

Special thanks to Eric Eidsen and John Masciatoni for providing a great motivating application, and to Kevin Pedretti for helping with the hardware modifications for the Cray XT4 and Netezza experiments.

References

1. R. Alverson, D. Roweth, and L. Kaplan. The Gemini system interconnect. In *Proceedings of the 18th Annual Symposium on High Performance Interconnects (HOTI)*, pages 83–87, Mountain View, CA, August 2010. IEEE Computer Society Press.
2. R. Brightwell, K. Pedretti, K. Underwood, and T. Hudson. SeaStar interconnect: Balanced bandwidth for scalable performance. *IEEE Micro*, 26(3):41–57, 2006.
3. R. Brightwell, R. Riesen, B. Lawry, and A. B. Maccabe. Portals 3.0: protocol building blocks for low overhead communication. In *Proceedings of the International Parallel and Distributed Processing Symposium*, page 268, Fort Lauderdale, FL, April 2002.
4. A. M. Bruaset and A. Tveito, editors. *Numerical Solution of Partial Differential Equations on Parallel Computers*, volume 51 of *Lecture Notes in Computational Science and Engineering*. Springer, 2006.

5. P. Carns et al. Understanding and improving computational science storage access through continuous characterization. In *IEEE Conference on Mass Storage Systems and Technologies*, pages 1–14, 2011.
6. G. S. Davidson et al. Data-centric computing with the Netezza architecture. Technical Report SAND2006-3640, Sandia National Laboratories, 2006.
7. S. C. Deerwester et al. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
8. D. J. DeWitt and P. B. Hawthorn. A performance evaluation of data base machine architectures. In *Proceedings of the seventh international conference on Very Large Data Bases*, VLDB '81, pages 199–214, Cannes, France, 1981. Invited paper.
9. J. Dongarra, P. Luszczek, and A. Petitet. The LINPACK benchmark: past, present, and future. *Concurrency - Practice and Experience*, 15(9):803–820, 2003.
10. E. D. Eidson and M. A. Ehlen. NISAC agent-based laboratory for economics (N-ABLE^{TM}): Overview of agent and simulation architectures. Technical Report SAND2005-0263, Sandia National Laboratories, 2005.
11. P. Francisco. The Netezza data appliance architecture: A platform for high performance data warehousing and analytics. IBM Redguide, http://www.redbooks.ibm.com/redpapers/pdfs/redp4725.pdf, 2011.
12. S. Graves. HPC databases: The data ingest challenge. HPCWire, June 2007.
13. D. S. Greenberg et al. A system software architecture for high-end computing. In *Proceedings of SC97: High Performance Networking and Computing*, pages 1–15, San Jose, California, November 1997. ACM Press.
14. M. Heroux et al. An overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.
15. S. M. Kelly and R. Brightwell. Software architecture of the Light Weight Kernel, Catamount. In *Proceedings of the Cray User Group Meeting*, Albuquerque, NM, May 2005.
16. A. B. Maccabe and S. R. Wheat. Message passing in PUMA. Technical Report SAND-93-0935C, Sandia National Labs, 1993.
17. S. Negash. Business intelligence. *Communications of the Association for Information Systems*, 13(15), 2004.
18. R. A. Oldfield et al. Trilinos I/O Support (Trios). *Scientific Programming*, August 2012.
19. R. A. Oldfield, T. Kordenbrock, and J. Lofstead. Developing integrated data services for Cray systems with a Gemini interconnect. In *Cray User Group Meeting*, April 2012.
20. B. Plale and K. Schwan. Dynamic querying of streaming data with the dQUOB system. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):422–432, April 2003.
21. A. Ralston, E. D. Reilly, and D. Hemmendinger, editors. *Encyclopedia of Computer Science*. Wiley, 4th edition, 2003.
22. E. Riedel, C. Faloutsos, G. A. Gibson, and D. Nagle. Active disks for large-scale data processing. *IEEE Computer*, 34(6):68–74, June 2001.
23. R. D. Sloan. A practical implementation of the data base machine – Teradata DBC/1012. In *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, pages 320–327, January 1992.
24. T. B. Team. An overview of the BlueGene/L supercomputer. In *Proceedings of SC2002: High Performance Networking and Computing*, Baltimore, MD, November 2002.
25. B. M. Thuraisingham. *Web data mining and applications in business intelligence and counter-terrorism*. CRC Press, 2005.
26. C. Ulmer, G. Bayer, Y. R. Choe, and D. Roe. Exploring data warehouse appliances for mesh analysis applications. In *Proceedings of the 43rd International Conference on System Sciences*, pages 1–10, Koloa, Kauai, Hawaii, January 2010. IEEE Press.
27. D. Wallace. Compute Node Linux: Overview, progress to date & roadmap. In *Proceedings of the Cray User Group Meeting*, Helsinki Finland, May 2007.