Post-Cluster Computing and the Next Generation of Scientific Applications

Gerd Heber and David Lifka
Cornell Theory Center, Cornell University
Ithaca, NY 14853, USA

Paul Stodghill
Department of Computer Science, Cornell University
Ithaca, NY 14853, USA

ABSTRACT

In this paper, we argue that the deployment of high performance wide area networks coupled with the availability of commodity middleware will produce a new paradigm of high performance computing that we call *Post-Cluster* computing. Further we argue that the computational science community is on the cusp of this change, and already we are seeing systems deployed that utilize this new style of computing. We discuss two systems that we are developing along these lines, a framework for deploying scientific applications as web services and a finite-element method (FEM) analysis system that uses commercial relational database systems (RDBMS's) instead of flat files. We argue that both of these systems leverage commodity middleware in order to allow us to write applications that are more distributed, flexible, and easier to implement than, and yet perform competitively with, our previous systems.

Keywords: Post-Cluster computing, middleware, distributed computing, finite-element analysis, relational database management systems, enterprise computing, grid computing, componentware.

1. INTRODUCTION/MOTIVATION

Cluster computing has matured to the point where computational scientists can directly purchase turnkey, high-performance clusters. While having clear price/performance, user interface, and maintenance advantages over traditional supercomputers, clusters remain primarily centralized machines, and the applications that are being written for them are also centralized and monolithic. In this paper, we argue that the computational science community is on the cusp of a new era in high-performance computing. In order to understand the trends that are bringing this about, we will review how cluster computing has already changed computational science and examine several emerging technologies that will effect how scientific applications are developed over the next five to ten years.

Traditional High Performance Computing (HPC) - Up until only a few years ago, high performance computing was done primarily on a class of machines called "supercomputers." Supercomputers were distinct from other, more general-purpose computers in that they attempted to deliver the highest level of performance for scientific applications. Because these machines were so expensive and difficult to maintain, they were usually found in dedicated supercomputing centers at a small number of locations around the country and the world.

There were a number of different classes of supercomputer, including vector, SIMD, and MIMD, and there were a number of different manufacturers, all of whom employed different instruction sets and operating systems. The dissimilarity between machines made it extremely difficult to port a piece of software from one machine to machine another, or to get different machines to work together on a single problem. It also meant that applications often had to be developed directly on these machines.

Cluster-based HPC - In the nineties, the gap between super- and general-purpose computers grew increasingly smaller. Several groups started building clusters from commodity desktop computers and networks and found that these machines were able to achieve a level of performance that was competitive with supercomputers for a number of scientific applications. In the past few years, as PC's and their interconnect networks became faster, the performance gap between commodity clusters and traditional supercomputers has continued to close. Today, the price/performance ratio is clearly in the favor of machines built around commodity hardware.

Today, it is possible for scientists to order commodity cluster computers from the same manufacturer that they ordered their desktop computers. This means that they can develop and debug applications on their desktop machines using their favorite programming environment and then simply copy their applications onto a cluster for execution. Furthermore, in many cases, these computers share a common file system and authentication mechanism, so jobs can be submitted to a cluster directly from desktop machines.

While having better price/performance ratios than their predecessors, cluster computers are still primary centralized computers. That is, applications that are designed for the current generation of high-performance computers execute, primarily, on a single machine in a single location.

Post-Cluster HPC - We see several emerging trends that we believe will radically change scientific computing. They are as follows.

- *High performance WAN's:* Trans- and inter-continental networks are being deployed that will provide very high bandwidth between geographically distributed institutions [1],[2]. This will enable collaborating scientists at different institutions to move relatively large amounts of data over the Internet. In addition, devices for 3D "virtual reality" visualization are now

This research is partially supported by NSF grants EIA-9726388, EIA-9972853, and ACIR-0085969.

commercially available [23], and their price will continue to drop as they become increasing based on commodity hardware and software [22]. We believe that *immersive visualization* will be consumer of bandwidth on the new networks.

- *Standardized middleware:* We will discuss a number of these technologies: CORBA [3], COM/DCOM [4], ODBC [5], XML [6], HTTP [7], and SOAP [8]. As these API's converge, middleware implementations are becoming cheaper and more widely available. As these technologies become more mature, vendors have started to deploy entire application frameworks that are built around them. Microsoft's .NET framework [9] is, perhaps, the most well known instance of this. Globus [31] is framework targeted towards high performance computing.

These trends will give rise to what we call "post-cluster computing". In the near future, we expect to see scientists leverage these commodity technologies to write applications that are constructed in a decentralized and distributed manner from a set of loosely coupled components that allow for a greater degree of collaboration between scientists and that provide much simpler and intuitive interfaces. In fact, we already see hints of this. For instance, there are a number of applications in computational genomics [10] and astronomy [11],[12] that use commodity databases to store and access tera- and petabytes of scientific data. Peer-to-peer computations, such as SETI@Home [12] and Distributed.Net [14], use simple standard Internet protocols in order to communicate between servers and clients. Our thesis is that the emerging commodity technologies will make it easier to develop such systems, and therefore they will become more prevalent. Furthermore, we believe that scientists will use these emerging commodity technologies in ways that we are only beginning to imagine.

In the rest of this paper, we will discuss ongoing work in the Adaptive Software Project that leverages commodity in ways that are novel for scientific computing. Section 2 discusses the Adaptive Software Project's Distributed Simulation Environment, a framework for deploying components for simulation in a distributed, and yet, easy to use manner. Section 3 discusses a component for performing FEM analysis that will be deployed in this framework, and makes novel use of commercial RDBMS's. In Section 4 we summarize our work.

2. THE ASP DISTRIBUTED SIMULATION ENVIRONMENT

What is ASP

The Adaptive Software Project (ASP) is a research program to develop adaptive software and systems for scientific simulations. We are studying adaptivity at three different levels, the application level (e.g., switching between different physical models as the simulation evolves), the algorithmic level (e.g., switching between different linear solvers based upon the conditioning of a system of equations or changing the discretization method from Finite Elements to wavelets), and the system-level (e.g., providing fault-tolerance and dynamic resource management).

Researchers from a number of institutions, including Cornell University, Mississippi State University (MSU), and the College of William and Mary (CWM), are participating in the project. The participants also come from a number of disciplines, including Computer Science, Civil Engineering, Physics and Applied Math. The physical phenomena that we are interested in simulating are multiple physics (e.g., chemically reacting flows, thermal stresses, structural mechanics and fracture mechanics) at multiple time and length scales (e.g., from angstroms to centimeters and meters).

In order to pursue our research agenda, we need a "nimble" simulation environment in which novel adaptive components can be developed, tested, and used by members of this project. Consider this to be enabling technology for the sorts of adaptive software that we want to deploy. In the past, we might have chosen to develop a software testbed in which each of our codes would compile and run. However, in addition to being geographically distributed, our computing resources are based upon different computer architectures and operating systems (Cornell – Windows NT on x86 and Itanium, MSU – Solaris on SPARC, CWM – Linux on x86). Based upon our experience with past projects, we concluded that single framework that would run on any single machine and that would support all of our codes was simply too expensive and time consuming to develop.

Requirements

Instead of building a centralized framework for building monolithic simulation applications, we have chosen to develop a distributed web-based framework for building loosely coupled simulation applications that will leverage the bandwidth that will become available as ever better WAN's are deployed. Our vision is to have each project member supply their individual components/software to the other project members in the form of Web services [27]. Then, a member can perform a simulation by choosing a set of Web services that are required to perform the computation and by specifying how the data must flow between them.

There are several advantages to the Web services based approach over the traditional centralized approach.

1. From the end user's point of view, a set of Web services can be more easily combined and recombined to form different simulation applications.
2. From the developer's point of view, providing a component in the form of a Web service means that the component does not have to be ported and maintained on more than one architecture and operating system. In the long run, we believe that this approach will dramatically reduce our implementation costs, while providing us with a much more flexible simulation toolkit.
3. From a philosophical point of view, it remains doubtful that a system capable to adapt can exist in form of a monolithic framework. [31]

In order to realize our vision, we need an infrastructure for deploying and using these Web services. We call this infrastructure the ASP Distributed Simulation Environment (DSE). This infrastructure must provide a simple interface to both the end-user and the component developer. It must allow clients and servers running on different architecture and within different security domains to communicate and interact.

Commodity Middleware

We have discovered that much of the functionality that we need in order to implement the ASP DSE is already present in systems that support commerce on the web today. Here are a number of examples,

- SSL [29] for communication authentication and privacy,
- XML [28] for application-neutral data exchange,
- SOAP [30],[27] for remote procedure and service invocation,
- UDDI [26] for description, discovery and integration of web-services and
- ODBC [5] for accessing databases.

Not only are these technologies standardized (or at least documented), they are widely available for a number of different platforms and from a number of different vendors. In effect, what we are starting to see with the wide deployment of these technologies is the commodization of middleware systems [33],[32]. Computational scientists, like us, will greatly benefit from this development.

Status and Related work

At present, we are in the midst of laying the foundations for our ASP DSE. In the process of studying what has been done in this area, we have found two bodies of work that are of immediate relevance.

Gridware – Systems like Globus [31] attempt to provide middleware for distributed scientific computing. [37] describes how a system like Globus can be used to deploy scientific applications as Web services. Globus is composed of a number of different services, including the Grid Security Infrastructure (GSI) [38]. The GSI is built on top of SSL and provides end-to-end authentication and encryption. Unfortunately, we know of no gridware system that runs on all of the platforms that we need to support (e.g., Windows).

Componentware – Systems like COM/DCOM [4] and CORBA [3] are designed to wrap traditional standalone applications into interoperable components and to combine them to form new applications running on a single machine or within the limits of an intranet. These systems were not designed for loosely coupled distributed components exchanging messages in a form independent of the middleware over the Internet. They also do not support the notion of inheritance for building new components on top of existing, possibly distributed, components.

What we have found is that, while many useful pieces are readily available, a complete solution for the ASP DSE does not exist at the present time. However, over the next 5 years, or so, we expect to seem dramatic improvements in this area. In the meantime, we plan to use the existing technologies like XML, SOAP, WSDL, and UDDI to build a "light-weight" infrastructure to meet our projects needs. We plan to incorporate additional gridware and componentware technologies as they mature and are able to meet our project requirements of ease of use, interoperability and performance. The technologies mentioned above, on the other hand, give us the necessary flexibility for rapid development and deployment of the components dealing specifically with adaptivity.

3. A CASE STUDY: FEM ANALYSIS AND SQL SERVER

In this section, we examine one component that will execute within the ASP DSE in more detail. In order to illustrate further how software commodization adds mature, standardized and easy-to-use tools to the scientific toolset and how they enable us to re-think traditional approaches and write better software, we outline in this section a new design for a Finite Element Method (FEM) analysis system.

FEM analysis is something that we thought we knew how to do "right" all along. Of course, the definition of "right" depends on how the software is used and this mode of use itself evolves. One of the lessons the increasing involvement in interdisciplinary research collaborations across different schools and research institutions has taught us is that the "you port our code to your platform and we do the same with yours" approach is no longer an option and that therefore interoperability, not portability, is the most important criterion.

Relational database management systems (RDBMS) are a mature enterprise technology: Entire industries depend on their robustness, interoperability and performance. Apart from huge datasets, e.g. in genomics, particle physics or astronomy, they are easily overlooked by the scientific community. RDBMS are something to store customer information and product orders, and subsequently mine those data to increase sales, - right? – That is precisely the impression one gets from reading standard texts on RDBMS and SQL and it is no surprise that many people finally believe that that's the ONLY thing RDBMS are useful for.[1] How can I and why should I flatten my unstructured grid data into simple-minded tables?

In this section, we will highlight some of the gains in interoperability, productivity, robustness and performance obtained from making an RDBMS an integral part of a simulation environment for computational fracture mechanics.

Traditional FEM Analysis

A simulation typically consists of several phases with different data-access patterns like *pre-processing, equation solving* and *post-processing*. Geometric and material modeling, and mesh generation are examples of pre-processing operations. The formulation and setup of the (non-)linear equations, error estimation and finally running an appropriate equation solver make up the solution phase. Error analysis, visualization or model adaptation, are typical post-processing tasks.

The individual phases usually "communicate" via files or data structures in memory. This approach has some serious disadvantages.

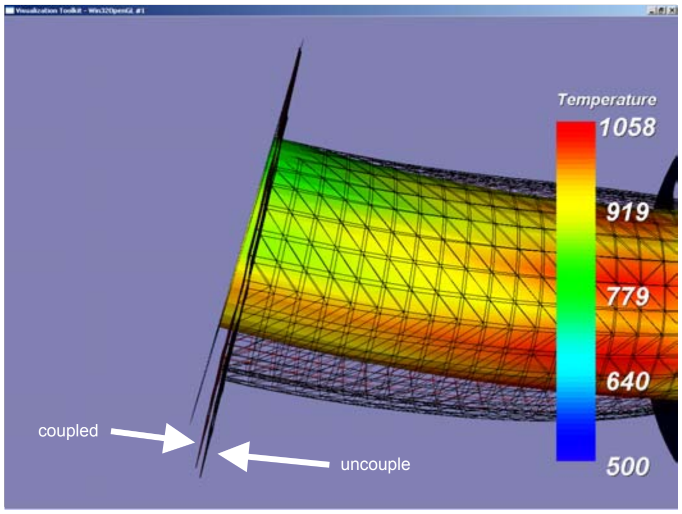
1. Every package supports a limited number of more or less standardized data structures/file formats. Besides the conversion effort, it requires an API (and more) to extract useful information. Usually, such APIs are provided in the form of libraries, which too often limit potential uses and users because of ties to a particular language.

[1] There are notable exceptions as for example [15] where the implementation of complex arithmetic using stored procedures is shown. Reference [16] has some examples of matrix arithmetic in SQL, which to traditional users of RDBMS may appear as useless academic exercises. From an HPC point of view, they in fact are, but to HPC users, on the other hand, a matrix is a more familiar object than a customer database.

2. The costs of providing and maintaining an API can be overwhelming in the light of evolving demands. In addition, users must write everything that is not (yet) part of the API and replication of superfluous, non-problem-oriented code is a consequence.
3. Files as the primary persistent storage are of poor interoperability in a distributed computing environment where different views and subset of the same underlying data frequently are created and/or adapted.
4. Files may contain invalid data in terms of data and/or referential integrity and it is difficult to detect this once large datasets are in place. Transactions and fault-tolerance mechanisms can be layered on top of file or in-core data structures, but our goal is to quickly deploy tools for doing material not computer science.
5. Parallel and distributed computing adds another layer of complexity. Usually, it destroys the natural global view of the model and replaces it by local patches that need a synchronization mechanism for consistency. Parallelization of full-fledged FEM codes is non-trivial. (Otherwise, it would be difficult to understand why commercial codes like ANSYS [20] or ABAQUS [21] exist as sequential codes only.)

An Alternative Approach Using RDBMS

We have implemented a coupled thermo-mechanical simulation that allows us to study crack growth taking the thermal state of the material into account. On the figure below, the results of a coupled and an uncoupled thermo- mechanical simulation for a combustion pipe are shown. (The picture shows only parts of the 3D model's surface with the color contour indicating the temperature distribution and the wireframes indicating the deformed shape.)



In the following we describe the main algorithmic steps and how the simulation interacts with the RDBMS. The parallel code (MPI) is written in C++ and uses the ODBC [5] API to communicate with the SQL Server. One of the code's characteristics is its brevity (compared to the traditional version): the library functions consist mostly of SQL statements (strings!) and result set parsing. *SQL is a universal API!*

Algorithmic Steps

1. **Partitioning:** To run the simulation on a cluster of distributed memory machines, data must be partitioned. A simple and cheap partitioner is sufficient to get started. We implemented recursive coordinate bisection (RCB) [19] and a space-filling curve (Peano curve) [34],[35] based partitioner as stored procedures, which partition the sets of vertices, edges and tetrahedra. This partitioning information is also used to create certain clustered indices. For complex geometries, the quality of the partitions provided by RCB may not be satisfactory in terms of load balancing. In this case, a Kernighan-Lin method [19] or a high-quality partitioner like ParMeTis [18] can be used to post-process the partitions in memory.
2. **Read:** Given the initial partitioning, vertices and tetrahedra are read into core where each MPI process `SELECT`s only objects tagged with its process id.
3. **Formulation:** The read is followed by the formulation of the FEM elemental matrices. (The underlying finite elements are 10-noded tetrahedral elements. The elemental matrices are 10x10 and 30x30 matrices for the thermal and the mechanical analysis, respectively.) Contributions to the right hand sides of the equations from internal heat sources or sinks and volume forces are computed during the formulation as well. The formulation is embarrassingly parallel and floating point intensive (numerical integration). There is no interaction with the database during formulation.
4. **Equation numbering:** The equation numbering is simple enough that it can be handled by a stored procedure. It is independent of the formulation and can be performed on the SQL Server while the elemental matrices are formulated on the cluster. As in the case of the partitioning, a post-processing of the equation numbering in core is optional and depends on the problem. Having SQL Server perform the equation numbering is particularly simple, because no synchronization (communication) between partitions is necessary.
5. **Assembly:** Once equation numbers are assigned, all elemental matrices are assembled into a global matrix. There is no interaction with the database during assembly.
6. **Boundary Conditions:** The assembly can be overlapped with the incorporation of boundary conditions like surface temperatures, heat fluxes, fixed displacements or tractions on the surface. For a database's global view of topology, geometry, boundary conditions and mesh there is no need for synchronization among partitions about boundary conditions. Below, we show an example of a query that returns the ids of all tetrahedra in a given partition, which have a facet on a surface where certain Neumann-type boundary conditions are imposed.

```
SELECT A.m_tet_id FROM
 (SELECT m_tet_id FROM MVerticesOfTetrahedron
    WHERE m_vertex_id IN
     (SELECT C.m_vertex_id FROM
        MVerticesOfMTrianglesOnTSurface AS C
       JOIN
         Wall_conditions AS D
       ON C.m_triangle_id = D.m_triangle_id)
    GROUP BY m_tet_id
    HAVING COUNT(m_vertex_id) = '3') AS A
  JOIN
    TPartitioning AS B
  ON A.m_tet_id = B.m_tet_id
  WHERE partition = 'my_MPI_rank'
```

7. **Solution:** The equations for temperature and displacement are solved in this step using one of the iterative solvers from PETSc's [17] suite. There is no interaction with the database during solution.
8. **Write:** The results are stored in the database for post-processing. (Distributed) Pre- and post-processing tools can query data from SQL Server via HTTP, XML, ODBC, ADO or OLE DB.

Evaluation

We ran a series of tests on the Cornell Theory Center's AC3 clusters and the RDBMS of our choice was Microsoft's SQL Server 2000 Enterprise Edition, which was installed on a dual processor DELL PowerEdge 2450 Dual Pentium III processor running Windows 2000 Advanced Server (SP2). 100 MBit Ethernet connects the cluster and the database server. For a fixed problem size, we were interested in how many clients (MPI processes) a single server would be able to serve without significant performance degradation. The underlying problem was mesh of about 60,000 tetrahedra resulting in about 300,000 degrees of freedom for the Finite Element problem. In the table below, we show the time for those phases of the simulation when all clients access the database. The solution time is shown for comparison.

	4 procs	8 procs	16 procs	32 procs
Fetch Vertices and Tets	1.04	1.07	1.22	2.48
Number equations	2.63	4.36	7.1	14.58
Essential BCs elasticity	5.01	8.61	14.68	30.31
Natural BCs elasticity	1.65	4.73	9.67	19.66
Solve	278.89	153.67	84.17	56.84
Write displacements	26.37	26.47	26.73	26.39

There is a noticeable performance degradation going from 16 to 32 MPI processes, where the total of the database access times actually exceeds the solve time. Except for the writing of the results, all other accesses are read-only. The writing of the results is done by MPI process 0 only, which gathers the result vector and updates the result table. The performance is competitive with traditional file-based simulations, although we did not perform a direct comparison. To make a "fair comparison" is not as simple as it might appear. For example, large clusters, such as the AC3 clusters, are not served by a single file server and the ability of the database-assisted simulation to overlap certain steps of the simulation may be viewed an unfair advantage.

The fetching of vertices and tetrahedra scales so well because the two underlying tables have clustered indices [24]. The same can be done for the tables of bounding entities to make the formulation of the boundary conditions scalable, which we have not done for this experiment. However, the scalability problem at large cannot be solved by clustered indices alone. A single server is a potential bottleneck and it will serve well only a limited number of clients, whose number also depends on the size of the database. State-of-the-art RDBMS can be run on clusters of servers where the underlying databases are either replicated or distributed across several linked servers. The former is applicable for small and medium sized models. The latter is definitely the more scalable approach for large solid models and Finite Element or Finite Volume meshes, which we currently explore using distributed partitioned views [15],[24] in SQL Server 2000.

With respect to databases, there are at least two major challenges on the software side:

1. Although, thanks to the use of databases, we have been able to dramatically reduce the fraction of non problem-oriented code in our simulation, the newly introduced code that targets the ODBC interface hardly lives up to contemporary software standards. Greater type safety and a more "organic" integration of dynamic SQL queries with libraries like the ANSI/ISO C++ Standard Library would be helpful. Embedded SQL is a nice, though outdated, example for the integration of static SQL queries and the C programming language. Microsoft's ADO.NET [36] offers a much cleaner interface.
2. Given that clustered RDBMS will serve simulations running on large HPC clusters, an intermediate software layer is necessary to facilitate the mapping between the two and to isolate the application from organizational details of the database cluster. Parallel libraries that offer explicit support for (distributed) domains, relations, and associations, like the JANUS library [25], can be used to build such a layer.

In this section, we focused on how RDBMS can be used to accelerate and simplify the solution phase of a typical FEM simulation. Let us point out that standard pre- and post-processing can be greatly simplified and made more powerful by the use standard RDBMS as well. The conversion of this environment into a Web service [8] and a visualization tool that interacts directly with the RDBMS are in progress.

4. SUMMARY

In this paper, we have argued that the computational science community is on the cusp of a new era in high-performance computing. We have identified two emerging trends that we believe will radically change the way that scientific applications are written: the availability of very high bandwidth wide area networks and the commodization of application middleware. These trends will give rise to what we call "post-cluster computing". In the near future, we expect to see scientists leverage these commodity technologies to write applications that are constructed in a decentralized and distributed manner from a set of loosely coupled components that allow for a greater degree of collaboration between scientists and that provide much simpler and intuitive interfaces.

To illustrate our point, we have presented two pieces of work being done within the Adaptive Software Project. The first was the Distributed Simulation Environment, a framework that we are developing in order to deploy the adaptive scientific components that we are developing as web services. We are planning on making extensive use of the commodity middleware systems that are becoming widely available.

The second piece of work is a novel system for coupled thermal-mechanical simulations based on the Finite-Element Method (FEM). What differentiates our system from previous systems is that it uses a commercial relational database system (RDBMS), in form of Microsoft SQL Server 2000, instead of the usual file-based approach for storing and retrieving data. Our new system is implemented using fewer lines of code but is able to interface with other systems in more general ways. For instance, because it interacts with data through standard database mechanisms, our new FEM solver can easily be set up as a service on the WWW or combined with other packages (e.g., CAD modeling tools, visualization systems) to produce extremely rich and flexible engineering tools. On the technical side, our system exhibits a higher degree of concurrency because considerable workload is pushed to the RDBMS. Furthermore, because our system manipulates data using transactions, it never leaves data in an inconsistent state if a failure occurs.

References

[1] "Internet2 Website." http://www.internet2.edu/

[2] "GEANT - The pan-European Gigabit Research and Education Network." http://www.dante.net/geant/

[3] "Welcome to the OMG's CORBA Website." http://www.corba.org/

[4] "Microsoft COM Technologies." http://www.microsoft.com/com/

[5] "Microsoft ODBC." http://www.microsoft.com/data/odbc/default.htm

[6] "Extensible Markup Language (XML)." http://www.w3.org/XML/

[7] "HTTP – Hypertext Transfer Protocol Overview." http://www.w3.org/protocols

[8] Kenn Scribner and Mark C. Stiver, Applied SOAP: Implementing .NET XML Web services, Sams Publishing 2002.

[9] "Microsoft .NET." http://www.microsoft.com/net/

[10] "AC3 Case Study: Genomics Data Warehouse." http://www.tc.cornell.edu/ac3/News/CaseStudies/2001/casestudy.cartinhour.html

[11] Jim Gray, "The World Wide Telescope: Mining the Sky." Supercomputing '01. Denver, Colorodo, November 10-16, 2001.

[12] "Sloan Digital Sky Survey SkyServer". http://skyserver.sdss.org/

[13] "SETI@Home: Search for Extraterrestrial Intelligence at Home." http://setiathome.berkeley.edu/

[14] "distributed.net: Node Zero." http://distributed.net/

[15] Itzik Ben-Gan and Tom Moreau, Advanced Transact-SQL for SQL Server 2000, Apress 2000.

[16] Joe Celko, SQL for Smarties, 2nd Edition, Morgan Kaufmann Publishers 2000.

[17] "PETSc: The Portable, Extensible Toolkit for Scientific Computation." http://www.mcs.anl.gov/petsc/

[18] "METIS: Family of Multilevel Partitioning Algorithms." http://www-users.cs.umn.edu/~karypis/metis/

[19] Ullrich Elsner, Graph partitioning – a survey, Technische Universität Chemnitz, SFB 393. SFB393-Preprint 97-27, 1997

[20] "Welcome to ANSYS.COM." http://www.ansys.com/

[21] "ABAQUS Home Page." http://www.abaqus.com/

[22] "Cornell Theory Center funded by NASA and New York State to provide 'Educluster' and Windows CAVE-like environment for education". Cornell Theory Center Press Release. April 23, 2001. http://www.tc.cornell.edu/news/releases/2001/educluster.asp

[23] Fakespace Systems, Inc. http://www.fakespacesystems.com/

[24] Kalen Delaney, Inside Microsoft SQL Server 2000, Microsoft Press 2001.

[25] "JANUS Home Page." http://www.first.gmd.de/janus/

[26] "The Universal Description, Discovery and Integration project." http://www.uddi.org/

[27] "LernXmlWS XML and Web services Resources". http://www.learnxmlws.com/

[28] "Extensible Markup Language (XML)". http://www.w3.org/XML/

[29] "SSL 3.0 Specification". http://www.netscape.com/eng/ssl3/

[30] "Web Services". http://www.w3.org/2002/ws/

[31] John H. Holland, Hidden Order: How Adaptation Builds Complexity, Perseus Print, 1996.

[32] "WS-I Web Services Interoperability Organization". http://www.ws-i.org/

[33] "The World Wide Web Consortium". http://www.w3.org/

[34] Jim Gray, private communication.

[35] Hans Sagan, Space-Filling Curves. Springer Verlag, 1994.

[36] "MSDN Home". http://msdn.microsoft.com/"The Globus Project". http://www.globus.org/

[37] "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration." I. Foster, C. Kesselman, J. Nick, S. Tuecke; January, 2002.

[38] A National-Scale Authentication Infrastructure**.** R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch. *IEEE Computer*, 33(12):60-66, 2000.