Strategies for managing business disruption due to Grid Computing

by

Vidyadhar Phalke

Ph.D. Computer Science, Rutgers University, New Jersey, 1995
M.S. Computer Science, Rutgers University, New Jersey, 1992
B.Tech. Computer Science, Indian Institute of Technology, Delhi, 1989

Submitted to the MIT Sloan School of Management
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in the Management of Technology

at the

Massachusetts Institute of Technology

June 2003

Signature of Author:____
MIT Sloan School of Management
9 May 2003

Certified By:____
Starling D. Hunter III
Theodore T. Miller Career Development Assistant Professor
Thesis Supervisor

Accepted By:____
David A. Weber
Director, Management of Technology Program

Strategies for managing business disruption due to Grid Computing

by

Vidyadhar Phalke

Submitted to the MIT Sloan School of Management on May 9 2003 in partial fulfillment of the requirements for the degree of

Master of Science in the Management of Technology

ABSTRACT

In the technology centric businesses disruptive technologies displace incumbents time and again, sometimes to the extent that incumbents go bankrupt. In this thesis we would address the issue of what strategies are essential to prepare for and to manage disruptions for the affected businesses and industries. Specifically we will look at **grid computing** that is poised to disrupt (1) certain Enterprise IT departments, and (2) the software industry in the high-performance and web services space**.** In this thesis, we will present an analysis for addressing this disruption on a segment by segment basis across the distributed computing and application software value chain.

Thesis Supervisor:	Starling D. Hunter III
Title:	Theodore T. Miller Career Development Assistant Professor

Table of Contents

List of Figures

ACKNOWLEDGEMENTS

I have benefited from talking to several leading industry and academic experts for my thesis. I thank Peter Richards, Tracy Trent, Prof. Andrew Grimshaw, Dr. Wolfgang Gentzsch, Ahmar Abbas, and Dr. Rao Mikkelinini Their insight and feedback has played a crucial role in identifying and addressing major issues for my thesis. I also thank several industry experts that have influenced this thesis but have chosen to stay anonymous.

I am grateful to my advisor Prof. Starling Hunter whose guidance and support has been vital in completing this thesis.

I thank my wonderful wife Debjani for her constant encouragement and support throughout this past year. Finally, I thank my lovely daughter Meghna whose joyful presence gave me a constant reminder of why I am doing this program at MIT Sloan.

1. INTRODUCTION

Grid computing is poised to disrupt enterprise IT architectures in the coming years. It is not only a major technological change from the traditional client-server model, but is also supposed to create a major organizational impact on the IT departments and the enterprise end-users. Traditionally, IT departments are used to managing clusters of servers and applications on hardware and equipment they own. This allows them to make changes, do maintenance, and other operational activities in a controlled and predictable manner. On the other hand grid architecture changes this model completely – different sub-systems and processes of an application will move from one computer to another without any administrator intervention. To make the matters more complicated, the applications need not run on servers managed by the IT groups at all. This raises a host of issues related to planning, management, security, and help-desk for which the IT groups are unprepared for today.

The second major disruption grid computing will have is on the software application providers – specifically high-performance application software vendors. The ultimate promise of the grid architecture is in the cross-enterprise deployment of software and solutions. This will allow enterprises to collaborate over the net using each other's computing resources, e.g. Boeing will be able to run their simulation software on Pratt and Whitney's computers thereby avoiding massive CAD file transfers and trade-secret issues. This development has a huge business impact on the current software pricing and licensing models. Today, software is typically sold to an enterprise using a per-seat license model. With the advent of grid, this model breaks down since the actual software will be able to move across enterprise boundaries and across license limitations.

Moreover there are issues regarding technical support and liabilities, e.g. if software purchased by Boeing crashes on Pratt and Whitney's computer, then who owns the problem, and how does it get fixed?

We also envision disruption in the current software development practices that typically treat resource constraints, high-availability, security, and manageability as an after-thought. With the advent of grid computing these features will have to be thought through very carefully during the software design, development, and the QA process.

After addressing the scope of grid computing disruptions, we address the issue of how an incumbent can manage this impending architectural disruption. We categorize our solution based on which segment this affected company falls in. If it is an IT department of an enterprise that is getting affected, then the approach should be to weigh the benefits of cost-reduction due to grid architecture against a possible increase in management, security and monitoring costs. Moreover, as the prices of computing resources continue to fall, an obvious alternative to a pure grid architecture is a server cluster architecture with ultra thin clients which is much simpler, easier to manage, and possibly cheaper than a grid architecture.

If it is a software development group that needs to make its applications grid enabled, they possibly need to rewrite parts of their code that deals with memory allocation, file systems, and other system resources. Also, additional work needs to be done for remote checkpointing, recovery, and result aggregation. Obviously before embarking on such activity the benefits should be well understood via analysis and by doing simulations.

Finally, for the software licensing and pricing issues in the grid architecture, the pay-as-you-go business model seems to hold some promise. This model will pave the way for embedding highly granular monitoring mechanisms in the software, which in turn, can also be used for solving support and liability issues.

The rest of thesis is organized as follows. In Chapter 2 we present the background of grid computing and the various segments that make up this business. In Chapter 3 we discuss various changes that grid computing will bring about and the impacts they will have on the current marketplace. In Chapter 4 we discuss strategies for managing these impacts, and finally in chapter 5 we present our conclusions and discuss potential follow-on work.

2. BACKGROUND

Grid computing is a model that envisions that high-performance computing will be delivered via a network of systems in which all entities are treated as resources. Thus processing, storage, sensors, networks – all get treated as resources, and applications are able to leverage them in a flexible, coordinated, secure, and robust manner in order to achieve whatever they are supposed to achieve.

A classic example that illustrates this notion is the SETI@home (Search for Extraterrestrial Intelligence) project. SETI@home is a scientific experiment that marshals the processing power of Internet-connected computers in the search for extraterrestrial intelligence (SETI). Participants on the Internet install a free software program packaged as a screen saver. While the screen saver runs, the software downloads, analyzes, and uploads radio telescope data from a server at the University of California, Berkeley, host of the SETI@home project. In this way, the SETI@home application is harnessing idle CPU cycles on computers that are connected to the internet virtually for free. Over the past 13 years, this project has been able to use over 4 million computers across more than 200 countries. The amount of CPU resource that it has been able to leverage is equivalent to having more than 100,000 dedicated computers – an obvious saving of hundreds of millions of dollars for the SETI@home project.

SETI@home inspired a range of other grid deployments: Folding@home is a protein-folding simulation for sifting potential drugs from the wealth of data revealed by the recent decoding of the human genome; Xpulsar@home sifts astronomical data for pulsars; Evolutionary@home is tackling problems in population dynamics.

Although the SETI@home examples and others clearly illustrates the power of harnessing idle CPU power available on computers connected to the net, it leaves a lingering doubt about the general application of this idea in the business world due to issues related to security, performance, fault management, etc. Before we address these issues let us look at the computer and networking industry trends leading up to the point where we are today.

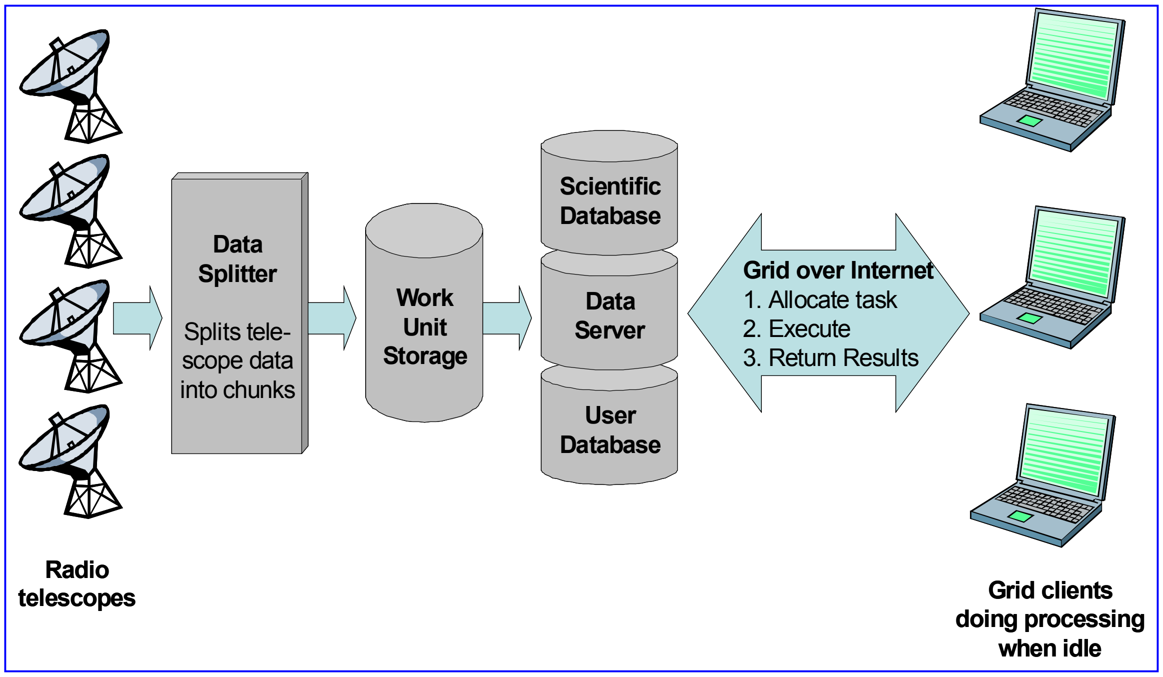

Figure 1: The architecture of SETI@home

In the mid to late 80s, computers and software were treated as standalone tools in the enterprises – manufacturing industry would run control systems; financial and service industry would keep customer accounts and generate periodic reports; sectors such as telecom used computers for controlling and monitoring switching equipment, and the scientific industry would run batch simulations. In the early 90s, as networked PCs became the norm, businesses shifted to client-server model for enterprise applications. The use of computers started to become critical for the day to day operations. Finally, with the advent of the Internet in the mid 90s, IT became a strategic component in most enterprises across most industries. Applications became mission-critical, and started to span not only departments and divisions within a company but also partners and suppliers across extranets. Technology wise, applications became distributed in nature – typically architected in an *n*-tier fashion.

Moving forward in the late 90s, with the hi-tech boom, four major trends emerged. First, enterprises invested heavily in software and hardware to gear themselves up for an incoming high growth scenario. Typically they would buy hardware based on peak performance requirements of the software they were buying. Second, network speeds went up exponentially while the prices got slashed. This trend has been continuing with Gigabit Ethernet (network that can carry over 1 billion bits per second) becoming a commodity today. Third, security standards such as IPSec and SSL got established making privacy and security extremely robust and foolproof. Finally, software became modular with platform independent technologies such as Java taking off. On the system side virtualization has created a separation between the hardware and the operating system, allowing applications to move from one system to another in a seamless manner. All these major trends have brought computing and networking to a level where users, communities, applications, and data seem to move anywhere in a fairly seamless manner – leveraging the resources in the best possible manner. In addition, as businesses and enterprises have become more collaborative and teamwork oriented the need for seamless grid computing has become stronger than ever.

On the business side, the grid computing value chain is made up of five major layers. These layers reflect the layering along the traditional OSI stack – moving from network layer all the way to application layer. Following diagram describes these layers with a brief description of the functionality and some of the key industry players in each layer.

Layer	Function	Companies
Application	Distributed or perf-ormance intensive application software	LifeSciences, Bio-Informatics, Supply Chain, Simulation
Serviceware	Tools for usage metrics, billing, accounting, etc.	Sun, IBM, HP, PlatformComputing Avaki, Entropia, United Devices, DataSynapse, Globus
Middleware	Grid resource map-ping, scheduling, queuing, security	
Resource	Collection of distri-buted storage and compute resources	Sun, IBM, HP, Dell, IBM, storage and server startups
Fabric	High Performance Networking (Ethernet centric)	Cisco, Juniper, Foundry, etc.

Figure 2: Grid computing value chain

Application Layer: Performance intensive distributed applications make up this top layer. Typically, these applications address problems in the compute intensive industries such as Life Sciences, Engineering Modeling and Simulation, Supply Chain Management, and Scientific Computing. These applications are characterized by a SIMD (Single Instruction Multiple Data) style of problem decomposition, i.e. processing needs to be done on a large data set that can be easily partitioned – allowing for the processing to be done in parallel across different CPU resources. To get an idea of the kinds of applications that would benefit from the grid, here is an illustrative list:

1. Large-scale, multi-institutional engineering design and multi-disciplinary science that require locating and co-scheduling many resources – e.g., design of next generation diesel engines, next generation space shuttle, etc.
2. Real-time data analysis for on-line instruments, especially those that are unique national resources – e.g. Department of Energy's LBNL and ANL synchrotron light sources, PNNL gigahertz NMR machines, etc.
3. Generation, management and use of very large, complex data archives that are shared across an entire community – e.g. DOE's human genome data and NASA's EOS data.
4. On-line medical imaging systems that provide real-time generation and cataloguing of large data-objects such as CTs, MRIs, for patient record keeping as well as research.
5. Bank One needs to run massive risk analytics software in its derivatives trading business. The software more or less remains the same but the data changes in a continuous fashion due to market events.
6. Charles Schwab was able to speed up a portfolio-rebalancing application from four minutes to 15 seconds by exploiting unused processing cycles on non-dedicated servers.

Serviceware Layer: Since grid computing allows for resources to be shared across administrative and ownership domains a mechanism is needed to track who is using what and how much. The serviceware layer encapsulates that function – it defines usage metrics and implements tools for running the grid as a managed service.

Middleware Layer: This is the core infrastructure of a grid. It provides mechanisms for managing and mapping resources, scheduling compute jobs, and security. Another way to think of this layer is as a Distributed Network Operating System.

Large companies that are providing solutions in the Serviceware and the Middleware layers are IBM, HP, and Sun. A longer list is shown in Figure 2.

Resource Layer: All the resources – supercomputers, servers, desktops, storage nodes, sensors, and other sharable elements make up this layer. The minimum requirement is that they should allow for sharing and should be networked in a reliable fashion. All the vendors that have traditionally provided computing, storage and networking elements fall in this category – the major ones are IBM, HP, Sun, Dell, EMC, and Hitachi.

Fabric Layer: Finally, the layer that forms the nuts and bolts of the grid architecture is the networking fabric. The dominant technology in this layer is the Ethernet with all the major vendors standardizing on IP as the networking layer. In any case, if some other protocol is needed, the vendor will create an abstraction to hide the lower layer implementation.

Along the time dimension, grid computing market for enterprises will follow a pattern depicted in

Figure 3. Currently it is dominated by *Cluster Computing*, which over time will evolve to *Intra-grids* and *Extra-grids*. The ultimate goal of grid computing is to reach the *Inter-grid* stage where resources across enterprises and across a global scale are leveraged.

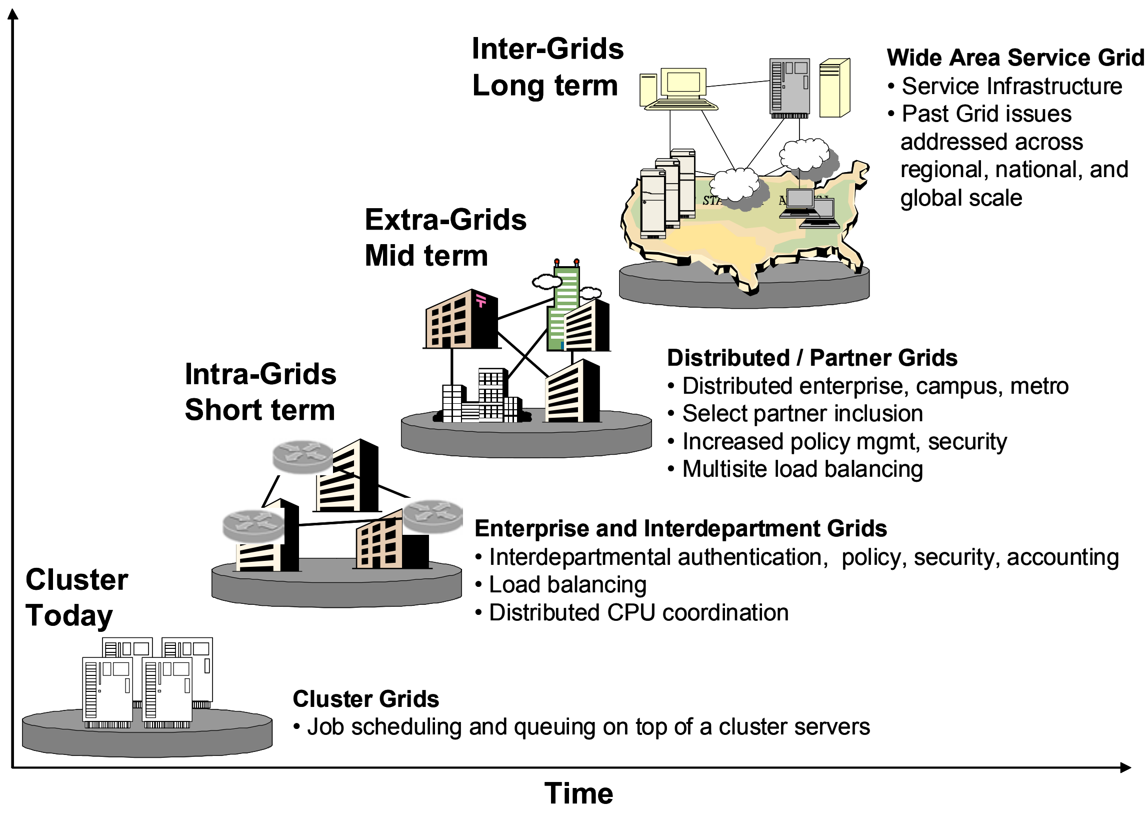
Inter-Grids
Long term
Wide Area Service Grid
• Service Infrastructure
• Past Grid issues addressed across regional, national, and global scale
Extra-Grids
Mid term
Distributed / Partner Grids
• Distributed enterprise, campus, metro
• Select partner inclusion
• Increased policy mgmt, security
• Multisite load balancing
Intra-Grids
Short term
Enterprise and Interdepartment Grids
• Interdepartmental authentication, policy, security, accounting
• Load balancing
• Distributed CPU coordination
Cluster
Today
Cluster Grids
• Job scheduling and queuing on top of a cluster servers
Time

Figure 3: The evolution of grid computing

Finally, to nail down the scope of grid computing, a note on what grid computing can not do to. The grid is not some magic that can take any arbitrary application and run it 1000 times faster without the need for buying any more hardware or software. Not every application is suitable or enabled for running on a grid. Some kinds of applications simply cannot be parallelized and decomposed. For others, it can take a large amount of work to modify them to achieve faster throughput. The configuration of a grid can greatly affect the performance, reliability, and security of an organization’s computing infrastructure. For all of these reasons, it is important for the users to understand how far the grid has evolved today and which features are coming tomorrow or in the distant future.

3. WHERE IS THE DISRUPTION (OR WHAT IS NEW)

In this chapter I would address the key business shifts due to grid computing. These could be architectural – as defined by Henderson and Clark [Henderson and Clark 90] or disruptive – as defined by Christensen [Christensen 97]. The chapter is based on my research and interviews that I conducted with various people involved in this line of business – from grid middleware, to application developers, to grid users.

As discussed in chapter 1 the value chain in the grid computing industry is made us of hardware vendors, operating system software vendors, middleware software vendors, tools vendors, application software vendors, and finally the application users. The grid computing model affects each one of these segments – in some cases severely and in some marginally. These affects are different in each one of the segments.

3.1 Hardware Vendors

Traditionally hardware vendors such as IBM, HP, Sun, Dell have relied on rapid obsolescence of existing hardware and the need for efficiency and digitization to sell more and more of their hardware products. With the dot-com boom a large number of companies grossly increased their computational capacities and as such there is a major computational glut in enterprises. At one of the major financial services company in New York that I visited, the average number of computers per employee was over 2.5! This is a phenomenon that is quite rampant across Corporate America. One of the key promises of grid computing called *scavenging* addresses exactly this situation. Companies that have large amount of idle hardware can use grid computing to do an efficient job of utilizing their existing hardware. What it implies is that enterprises will use grid computing, a software solution, to cut down on their hardware spending, directly affecting the top-line of all the hardware vendors.

3.2 System Software Vendors

System Software vendors such as those providing Operating Systems, Databases, Networking, provide the necessary layer on top of specific hardware so that applications see a uniform view of the computer hardware. Most hardware companies either provide their own System Software or partner with other software vendors. IBM, HP, and Sun provide their own Operating Systems with UNIX being the dominant design. Dell typically uses Operating Systems and other System Software from Microsoft. Over the past few years a range of standardization has happened in the System Software space. These include – run time environments based on shared libraries, database connectivity, networking and network security, and standardized interfaces to storage and devices. Another major development in the Operating System space has been a need for running multiple Operating Systems on the same hardware platform. This need has been driven mainly by data center requirements for isolating customers that are hosted on the same hardware platform. This need created a concept of *virtualization* that is now being extended to other parts of the computational and networking software value chain.

The evolution of standards in the System Software alongwith Virtualization has created the right set of enabling conditions for grid computing. In other words, grid computing does not affect the system software space in a negative manner but will only lead to an acceleration in this space if grid computing really takes off.

3.3 Middleware Software

Middleware typically refers to the software that bridges applications with operating systems. Traditionally, before applications became highly distributed, middleware software did not exist because applications were typically coded for a specific operating system and they would run on a single system. For CPU intensive applications, high-end parallel processing systems would be used. With the advent of internet and high speed connectivity, applications became more distributed, and an *n*-tier model of software architecture emerged that has more and less become a dominant design today.

Middleware software technologies emerged during these times to create a solution that would provide maximum flexibility in distributing an application while not forcing the developer to rewrite his or her code every time a change was introduced in the distribution architecture. Middleware software became significant in the early to mid 90s with OMG's CORBA open standard, soon Microsoft introduced its own middleware model called DCOM, and today we see SOAP and web services touting the same functionality.

The major advantage of middleware software is in the creation of a transparency that allows components of a distributed application to talk to each other without knowing where the other component is. It is very likely that grid computing will erode this value-add of middleware software. The grid model itself forces resources to conform to a uniform standard that allows transparent access from anywhere in the network thus providing the same core feature that OMG style middleware does.

The good news for the middleware software providers is that their target market does not intersect with that of grid head-on. Even then they need to extend their functionality to be able to inter-operate with grid system. These include extensions for administration across multiple administrative domains, dynamic interfaces, security management, and workflow management.

3.4 Tools Software

Software tools from the viewpoint of grid computing can be broadly subdivided into two broad categories (1) Software Development, (2) Software Deployment and Administration. We will discuss the impact of grid computing on these two categories separately although there are some situations where the grid paradigm makes these two categories intertwined which we will explain later.

3.4.1 Software Development Tools

This category contains technologies such as modeling tools, compilers, libraries, development frameworks, revision control, and testing tools. A significant portion of these technologies will transparently carry over to the grid computing paradigm. One of the key issues that needs to be resolved though is that of dynamic shared libraries and shared objects. Today most applications contain only the object code relevant to their functionality. For everything else they depend on dynamic or run-time libraries that already exist on a computer. Thus, when an application is executed, it searches and finds these libraries as and when needed. The positive advantage of this model is that application code size tends to be much smaller and computer memory gets used more efficiently. The downside is that when the dynamic libraries can not be found or if they are incompatible, that typically leads to the main application crashing. This happens many a times when an application is copied from one computer to another and the user attempts to execute it on the second computer. Since grid computing enables scavenging for executing an application anywhere on the grid this immediately raises the concern of dynamic library mismatch. In order to prevent such problems the grid administrators and the middleware will have to ensure that dynamic libraries are compatible throughout the nodes that make up the grid.

The second major issue in development tools is that of a multiplicative effect on Quality Assurance (QA). Most likely a grid deployment will be heterogeneous, i.e. it will have nodes (computers) that will run different Operating Systems with different versions and patches. Thus in order to certify that an application can run on any node of a grid, it has to be tested in-house on each of the possible variation of an Operating System. This implies increased QA capital costs since multiple variations of target Operating Systems will have to be maintained in-house. In addition, the QA cycles will be longer even if parallel testing can be achieved via automation.

3.4.2 Software Deployment and Operations

Software deployment consists of system sizing, user-acceptance cycle, and creating operations procedures. Operations involves administrative work including installation, upgrades, help-desk, security, health monitoring, and recovery. The grid computing model places additional burden on the deployment phase in that various nodes have to be preconfigured with administrative and security policies, the application itself has to be annotated with its minimal system requirements, and the soak-in period will be large since various dynamic scenarios will have to be tested.

From IT operations point of view, grid computing is a radical departure from the assumptions and practices IT departments have been following traditionally. Within IT operations, grid computing can be viewed either as a next step from *cluster computing* or a paradigm shift from a *client-server model* – depending on what they are used to.

Grid computing versus cluster computing: With respect to cluster computing, grid computing is seen as an evolutionary step. Cluster computing is defined as a system where a collection of general purpose servers, usually collocated, are networked together using high speed connections. They are not dedicated to run a particular application or service but instead serve as a pool of nodes. Typically a user requests to run an application to a master node in the cluster, and that master decides which node within the cluster that application should run on. After the application finishes executing, the results are returned to the user. This style of computational usage on clusters is the classic *job model* for mainframes.

The model of grid computing is a generalization of cluster computing in that it takes the concept of remote execution of a user's application to the next level in which any node on a grid with available processing resource can do the remote execution. Businesses that are already using cluster computing need to consider the following key differences that my impact their IT business in making a change to grid computing:

1. Unlike a cluster, the nodes in grid are not dedicated. This implies issues with performance guarantees and system availability. Moreover you may not know what can be expected from the nodes in a grid.
2. The way a grid gets constructed is dynamic – various resource nodes can come in and go as they please. On the other hand since a cluster is typically owned by an IT department, an organization has full control over it. This dynamic nature makes the grid less reliable from the view point of resource availability. It could also lead

performance degradation if a job keeps getting moved around due to nodes entering and leaving the grid in an abnormal fashion.

3. The resources in a grid are typically managed by different organizations and as such they have different administration, security, and management policies. This obviously implies that a grid administrator, unlike a cluster administrator, has to work across organizational and sometimes geographic boundaries.
4. Finally, due to the sharing model, the costs in a grid are usage based and are recurring. In the cluster model the cost is usually upfront as a capital expense. This has implications on the end-user and he/she has to be careful about submitting jobs since the usage will have to be paid for.

Grid computing versus client-server computing: With respect to client-server computing, grid computing is seen as a major disruption. In a client-server model, applications such as enterprise web servers, ERP, CRM, research databases are hosted on dedicated computers called servers. Access to these applications is provided to desktop computers called clients. Majority of CPU intensive work is done at the server and the client does minimal presentation and formatting work. Management of this type of system is done by the IT department where they own the servers and the client software portion on the desktops. The IT department has full control and full picture on the server side in this system.

If a client-server model has to be migrated to the grid computing model, such as we have witnessed at some of the major financial institutions, the disruption to the IT mindset is very significant. Following list highlights the major changes that need to be done at the IT level to adopt grid computing:

1. **Planning and Installation:** The grid middleware must be installed on an appropriately configured set of machines. These machines should be connected using networks with sufficient bandwidth to other machines on the grid. Another issue of prime importance is fail-over. Unlike in a client-server model where failure is handled by a simple mirroring technique per server, in a grid it is the management machines and critical database machines that need to have redundant backups. Security is more important in grid since root passwords of all machines in the grid has to be known to the grid middleware – this puts additional responsibility on IT.
2. **Licensing issues:** Some software vendors might enforce strict licensing. This implies the IT group has to add additional automated processing to allow for software to migrate to any node on a grid seamlessly. The application should not fail when it moves to a new node for lack of a license privilege.
3. **Managing enrollment of donors and users:** Grid administrators have an ongoing task to manage the members of the grid, both the machines donating resources and the users. The administrator is responsible for controlling the rights of the users in the grid. Donor machines may have access rights that require management as well. Grid jobs running on donor machines may be executed under a special grid user ID on behalf of the users submitting the jobs. The rights of these grid user IDs must be properly set so that grid jobs do not allow access to parts of the donor machine to which the users are not entitled. Similar enrollment activity is usually required to enroll donor machines – which involves more complex steps with Certification Authority. Corresponding procedures for removing users and machines must be executed by the administrator.
4. **Certificate authority:** It is critical to ensure the highest levels of security in a grid because the grid is designed to execute code and not just share data. Thus, it can be prime target for viruses, Trojan horses, and other attacks. The Certificate Authority is

one of the most important aspects of maintaining strong grid security – this is very unlike the client-server model that may or may not need Certificates depending on the applications being provided.

5. **Resource management:** Another responsibility of the administrator is to manage the resources of the grid. This includes setting permissions for grid users to use the resources as well as tracking resource usage and implementing a corresponding accounting or billing system. Usage statistics are useful in identifying trends in an organization that may require the acquisition of additional hardware, reduction in excess hardware to reduce costs, and adjustments in priorities and policies to achieve utilization that is fairer or better achieves the overall goals of an organization. Some grid components, usually job schedulers, have provisions for enforcing priorities and policies of various kinds. It is the responsibility of the administrator to configure these to best meet the goals of the overall organization. Software license managers can be used in a grid setting to control the proper utilization. These may be configured to work with job schedulers to prioritize the use of the limited licenses.
6. **Data sharing:** For small grids, the sharing of data can be fairly easy, using existing networked file systems, databases, or standard data transfer protocols. As a grid grows and the users become dependent on any of the data storage repositories, the administrator should consider procedures to maintain backup copies and replicas to improve performance. All of the resource management concerns apply to data on the grid.

3.5 Application Development

Most grid computing middleware and tools providers strongly believe in seamless migration of existing applications to the grid framework. This is logical since otherwise the cost of adapting to grid will be too high for the end-users. Even then application developers need to bring in certain discipline within their code.

The key mind shift in this segment is regarding assumptions that developers make about available resources on a system. Typically application or component developers assume that certain amount of memory, temporary file storage, and privilege levels will be available for their code when it gets executed on the target machine. A lot of times these pre-conditions are neither well documented nor it is known how the code will behave at run-time if these pre-conditions are not satisfied. Grid computing model's ability to run an application anywhere requires a disciplined approach for handling this developmental mind-set. The developers will have to provide a list of assumptions for their software's run-time environment to the grid. In addition, mechanisms will have to be put in for allowing graceful degradation when some of the assumptions get overlooked and the application can not continue on a computer. Alternately, tools need to be developed that can automatically find the scope of an application's resource usage by doing an analysis of the source code or by doing simulations of potential grid scenarios.

Another important fact to remember is that grid computing starts losing its advantages when the application becomes I/O bound, i.e. when the application spends more time reading and writing to storage than doing intensive processing. From the application developer's point of view it may not be obvious whether an application is I/O bound or CPU bound. Worse, the same application could be both I/O bound as well as CPU bound depending on the input or how it is configured or the section of code it is currently executing. This raises a tough challenge for the proponents of grid computing namely how to 'grid enable' applications that can not be clearly marked CPU intensive.

Finally, for the application developer, debugging is a major issue. In the grid computing paradigm, due to issues related to security, production time debugging will be extremely limited and at times impossible. The developer needs to build in remote logging and remote core dump mechanisms to allow basic triage and for collecting enough information to recreate the scenario and debug the problem.

4. MANAGING THE BUSINESS IMPACTS

Among the impacts and changes discussed in the previous chapter, the ones that standout from the viewpoint of business could be classified either as short term or long term. The short term impacts are characterized by the need for *scavenging* CPU cycles within an enterprise, while the long term impacts are characterized by the notion of *virtual organizations* – crossing enterprises boundaries to achieve a new level of efficiencies.

The major business impacts in the short term are:

1. Reduced IT costs to enterprises since they will be able to use their internal resources much more effectively and efficiently.
2. Increased cost and margins for software companies that have applications appropriate for grid architectures.
3. Reduced margins for mid-level servers and desktop providers since the demand will go down due to improved usage of existing hardware.

In the long term, the biggest impact for businesses will be in two stages:

1. Enterprises will treat computing and applications as a true pay-per-use service. This will finally lead to the revival of the Application Service Provider (ASP) business model.
2. For large businesses the notion of “virtual organization” will finally become a reality. In this scenario, suppliers, partners and at times customers will be able to access each other’s resources in a seamless manner. This will produce new kinds of efficiencies across corporate boundaries without compromising any confidential information.

In this chapter we analyze various issues that need to be managed in order to brace the changes being brought about by grid computing. Although most companies we surveyed seemed to be aware of the benefits and pitfalls of grid, the longer term impact of how grid computing enables the creation of virtual organization is not well understood or is seen as less disruptive by the industry.

4.1 Enterprise IT Departments

The enterprise IT department that need to adapt grid computing are typically characterized by three key features (1) Operational dependence on CPU intensive applications, (2) Surplus hardware in terms of desktops per employee and internal bandwidth, and (3) Have traditionally tried cluster based servers for computing needs.

An illustrative example in this category is that of Bristol-Meyers Squibb – a major pharmaceuticals and health care products company. Their business analytics software was suffering from poor performance due to lack of processing power. By investing around $500,000, the company was able to scavenge CPU cycles from desktop PCs giving them a 100 fold increase in available CPU cycles. In contrast a cluster based solution with similar investment would have given them only a single digit increase in CPU cycles. Figure 4 shows the schematic of their grid computing implementation that was done by Platform Computing Inc. There are similar examples in the pharmaceutical, financial, and automotive industries where IT departments have adapted grid computing for leveraging unused CPU resources within the enterprise.

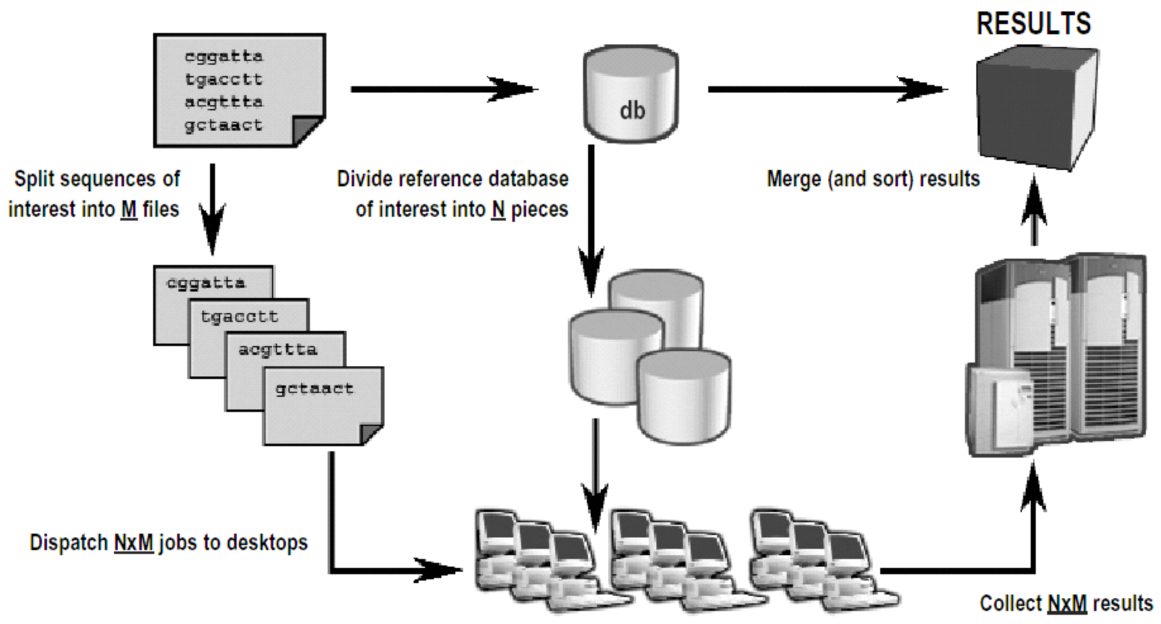
RESULTS
cggatta
tgacctt
acgttta
gctaact
db
Split sequences of interest into M files
Divide reference database of interest into N pieces
Merge (and sort) results
cggatta
tgacctt
acgttta
gctaact
Dispatch NxM jobs to desktops
Collect NxM results

Figure 4: Bristol-Meyers grid computing Implementation

Within enterprises, IT departments are usually cost-centers. For them the cost savings are obvious. But then there are increased overheads from the viewpoint of total cost of ownership. Following table lists a cost scorecard that an IT department could use in choosing between a grid solution versus a cluster or server based solution.

Decision Criteria	Grid computing	Server or Cluster Computing
Application type (whether embarrassingly parallel)	Need to assess if current and future applications are parallelizable.	No such assessment needed.
Application type (issues of security)	Need a security audit on the application to protect it from a disgruntled employee and vice versa	No such audit needed.
Assessment of IT needs	Needs to be more frequent since resources are being shared.	Less frequent – depends on the sizing done when an application is purchased or productized.

Decision Criteria	Grid computing	Server or Cluster Computing
IT Human resources	Need specialized training. Possibly needs additional help-desk resources.	Traditional IT sufficient.
Capital Investment	All software. Much cheaper.	Hardware and software.
Impact on users that are being ‘scavenged’	Major issue if grid leads to performance degradation on donor machines.	No such issue.
Impact on users that are running their applications on grid or cluster	Issues of guarantees and reliability.	No such issue since end-user know the precise behavior.

4.2 Application Developers

Software businesses that plan to ‘grid enable’ their applications need to look at a cost-benefit analysis that takes into account not only the current usage of their software but also how the patterns will evolve as new models such as *web services* come into play. Specifically, following cost-benefit issues need to be looked at:

Potential Business Impacts	Type of impact	Size of impact
Resurrection of compiler technologies that automated the process of parallelizing an application. This is key since grid computing is not much applicable to code that needs to execute serially, e.g. transactional code.	Organizational	High R&D cost
Testing and debugging on various OS versions on which the software could run.	Capital expense	High QA costs
Software support will be technically deeper.	Organizational	High support costs
Software licensing and certification	Legal	Increased sales time
Profit margins. The product will have better quality and more features.	Financial	Based on licensing model
Additional services revenue	Organizational	More Professional Services staff

4.3 Server and Desktop Vendors

One of the major issues facing server and desktop vendors such as Sun Microsystems and Dell is a perceived threat to their hardware business. If grid computing really takes off and enterprises are able to scavenge and leverage their existing idle CPU cycles then they may not need to buy additional hardware – at least not at the rate they used to. Although this threat is real in the near term, one needs to remember that ultimately grid computing is good for these server and desktop vendors. In the long term grid computing will spur growth in new kind of software business models and automation – a growth that will outstrip the current CPU cycle glut very quickly and create new opportunities for hardware vendors.

To manage this disruption, the mid-range server and desktop vendors need to focus on the following key issues:

1. Creation of a new sales channel to *`Grid Service Operators.'* Going beyond an enterprise boundary a class of service providers will emerge that will facilitate and operate nationwide and international grids that link various resources such as supercomputers and distributed large databases. We already see such efforts underway with iVDGL and the TerraGrid. Recently IBM and others have installed a National Digital Mammographic Archive on a grid that will need to be operated as a service. Establishing formal ties and building relationships with these new sales channels will be key for server vendors.
2. Additional functionality to the servers and desktops that makes them 'seamlessly grid enabled.' If some of the core grid functionalities can be embedded within the server or desktop it will create additional value for the customer – virtualization of Operating

System, protection mechanisms such as sandboxes and elements of guarantees such as those offered by Real-Time Operating Systems (RTOS) are some of the features that these vendors need to focus on.

3. Finally, these vendors need to understand that current erosion of growth in this sector is not only due to CPU glut but also due to the Internet boom and bust. As the bubble has burst we are seeing more and more used equipment showing up on secondary markets such as eBay and refurbished outlets. For example, a search on eBay for Sun E450 (a mid range server) listed over 30 used servers for sale – and this was a search for a very specific server. As such, server and desktop vendors need to work on other market factors, besides grid, that are impeding their growth.

4.4 Virtual Organizations – Disruption for Software Vendors

To illustrate this disruption consider the web services offering from Amazon.com that was announced in July 2002. With its web services program, Amazon.com offers visitors to third-party web sites the ability the search or browse Amazon's product database and conduct shopping cart operations remotely on the third-party web site itself. Visitors can add to wish-lists and wedding registries, as well as shopping carts using Web services. Amazon.com offers incentives to sites and companies in its referral program, Amazon.Com Associates, to use its web services, where referral sales can earn Associates up to 15 percent of the volume. The end-user interacts only with the third-party web site while beneath the covers the third-party web site is leveraging and using Amazon.com’s software. A potential example could be (say) Toys-r-us’ baby registry web-site coupled to Amazon.com via web services. When a user searches on Toys-r-us’ site for items with keyword “Winnie the pooh,” he/she is returned “Winnie the pooh” toy items from the Toys-r-us catalog while the books and CDs on “Winnie the pooh” come from Amazon.com. The benefit to the end-user will be a one-stop registry, while the two vendors will be able to sell their wares in a much more convenient and seamless manner.

The most important issue to understand here is that the third-party software, in this case Toys-r-us web-site, is actually using Amazon.com's software – crossing enterprise boundaries. This arrangement need not be across just two enterprises but could go even further. For example, if Mastercard wants to provide a special incentive for using Mastercard over the Web, it could link its own web services to Amazon.com's billing and checkout portal. Thus when a Toys-r-us customer buys an item, the software pieces that would interact together to make this transaction happen could span Toys-r-us, Amazon.com, and Mastercard – in a completely transparent manner to the end-user.

Although the example here is that of web services, it falls under the grid computing paradigm in that software components on one site (Toys-r-us) are using software components on another site (Amazon.com) by merely treating them as resources. A grid computing 'scavenging purist' may argue that Amazon.com is not actually executing 'code' supplied by Toys-r-us and hence it is not grid computing. To refute this argument one only needs to look under the covers to see how the above interaction happens in practice. The Toys-r-us software will not only pass through the end-user's search criteria, in this case the keyword "Winnie the pooh" to Amazon.com, but it will also pass a piece of software written in XSL (Extensible Stylesheet Language) that Amazon.com needs to execute on behalf of Toys-r-us to generate the final formatted output. In this way Amazon.com is executing an arbitrary piece of code supplied to it by a third-party. In Figure 5 we depict this interaction pictorially.

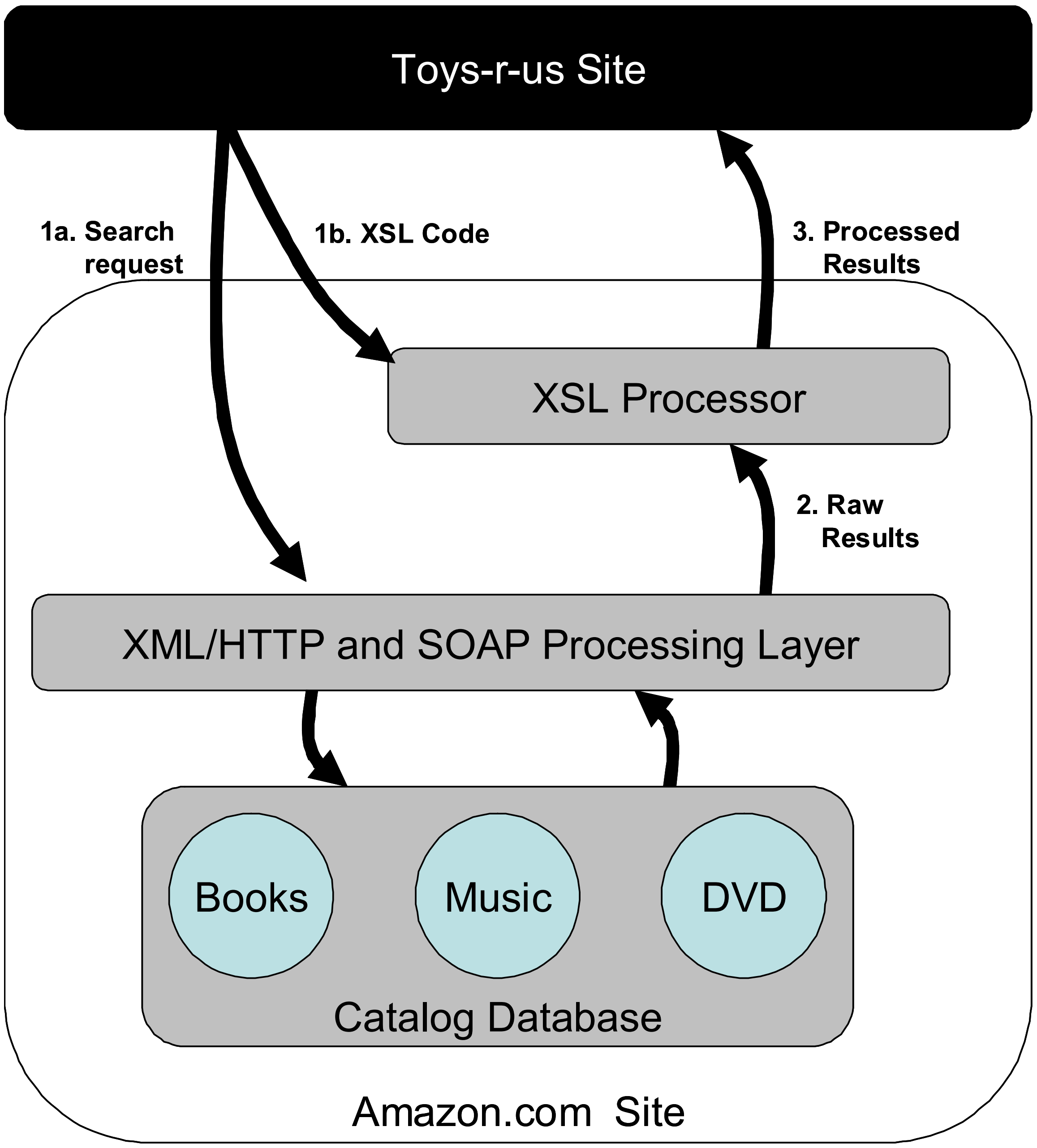

Figure 5: Web services example at Amazon.com

To see the first business problem with the above arrangement we need to look into the details of the above transaction. Let us assume that Amazon.com is using Oracle Corp.'s database to store information about the books and merchandise it sells. Assuming Oracle sold its database license to Amazon.com only, it brings up an interesting license violation issue when Toys-r-us' website indirectly starts using Amazon.com's Oracle database via the web services interface. From Oracle's point of view it potentially lost a customer – Toys-r-us since Toys-r-us' website does not need to license Oracle's database.

Another licensing issue exists in terms of the software that is being remotely executed on behalf of another party. In the example discussed above the XSL code represents such software. Supposing the XSL code was licensed exclusively to Toys-r-us from another company, then we get into yet another license violation issues if Amazon.com's website executes that code. Another issue is that if that XSL software causes a glitch at Amazon.com's website, how does one go about solving the issue? Who owns the problem – Amazon.com or Toys-r-us? It gets worse when the interaction is across more than two companies.

Today, these issues do not appear to be a major cause for concern due to three key reasons:

1. Most of the software that is being made available on a grid is exclusively owned by the party that is making it available. This is especially true of some of the Scientific and Experimental grids.
2. Companies that have purchased the software are paying extra for allowing third parties to access that software, as well as for running that software some place else. This overhead is currently tiny since most of it is being done at a narrow experimental scale.
3. Finally, some software vendors are hoping their licensing models based on number of client connections or strong license validations will scale automatically to the grid computing.

While one might be tempted to assume that current licensing models based on number of seats and connections may be sufficient for the grid, an understanding that having a highly dynamic nature is critical for grid's success is sufficient to demonstrate that these licensing models are not scalable. In a highly dynamic grid, computational components will move around freely leveraging the best resources possible, services will bundle, unbundled, and combine themselves across enterprise boundaries to create new services.

To address this issue, it seems only fair that licensing and pricing of software be addressed from the view point of the *value* it provides. Typically in the software business value is associated with the amount of usage. For example, Oracle does pricing and licensing based on the number of connections its database server will have at a customer site. The assumption here is that more the number of connections needed, more the database is being used, and hence more value is being derived from it. This *number of seats* based model works fine as long as all the connections to the database are coming from the same cost accounting pool. It starts becoming an issue when it is not easy to predetermine how many connections are needed, or when the database is shared – such as in the case of an Application Service Provider, or as we saw in the Amazon.com and Toys-r-us example above when new services are being created and are being accessed on the fly.

From the above discussion, we can conclude that in the software industry, since *usage* is a well defined metric for measuring the value delivered, it is only appropriate that in the grid infrastructure provisions be made to measure who is using what resource and how much. Once this common framework is embedded throughout the fabric of grid computing it will be fairly easy to build all sorts of sophisticated pricing, licensing, and billing machinery on top.

We next discuss various pieces of a solution for managing this disruption for the ISVs (Independent Software Vendors).

4.4.1 License Management by a Controller Scheme

For the grid model of running software on any available CPU licenses may be set up to where they can be shared among multiple CPUs and geographies. In a computing grid, where several applications or users may need the same licensed product, one may not want to purchase a license for every user and application. Instead, it may be more desirable to buy fewer licenses and have a system that manages the use of these licenses. This system may implement a fair share scheme where all users and applications have equal access to these licenses, or a priority system may be established where higher priority applications or users are able to use the licenses when needed. A scheme along these lines has been implemented in IBM's Platform Global License Broker.

4.4.2 Usage based Pricing

While actively tracking who is requesting a license maybe sufficient in some cases for tracking usage, it may cause a bottleneck in the system if large amount of time is spent requesting for and getting a license from the license management system. Worse, in some cases it may not be possible to find out what licenses are needed a-priori. For example, in the above scenario of Amazon.com and Toys-r-us web services interaction, the request from Toys-r-us does not even know that it needs an Oracle license to execute its query on the Amazon.com server. A more scalable and transparent approach is to track usage and then build *usage based billing and licensing models.* Although this is a radical shift from the licensing model that software vendors are used to, it is not something that hasn't been tried – in the late 1990s the Application Service Providers (ASP) model was exactly based on this concept.

4.4.3 Pricing Implementation

Telecommunications business models give an insight into what will be needed for building such schemes. Carriers across the country - and around the world - seamlessly transfer callers between various networks and, through peering agreements, share profits equitably. Given the volume of calls, this can be an exceedingly complex undertaking, but the carriers have it down pat – a true service grid in which consumers and service providers all benefit.

However, there is no comparable usage tracking or billing system that exists on the Internet. Towards that end what is needed is a universal identity system that doesn't play favorites. Such a system could be borrowed from telecommunications (peering, roaming and settlement), credit card systems (trusted third-party intermediaries), and governments (passports for international travel, issued and honored by many different countries, not by a single entity).

Once the identity scheme is established, the web services meter and accounting model, serves as a base platform for enabling this critical functionality in a grid. The model operates on a base assumption that web services and grid resources with a high degree of value are contracted via *Service Level Agreements* (SLA's) or their equivalent, which implies that both parties agree to the contract. The contract lays the foundation for metering the services to be used, by covering all the attributes of the service customs and how they shall be employed by provider and requestor. The contract can also include environmental prerequisites for the use of the Web service or resource. Once the requestor (or client) has contracted (or registered) with the provider, that requestor is said to be known to the provider and to the service meter. This is achieved via electronically signed contracts and certificates for the use of the Web service or resource.

The contract provides details concerning:

1. The type of contract: long term, ad hoc, restricted time, unlimited time, etc.
2. Start dates and expiration dates of the contract
3. The time model to be used: everyday, Monday to Friday, etc.
4. The amount model, which defines limits to the amount of service to be provided
5. Security: Signatures or certificates for encryption and authentication

The meter and accounting service stores the relationship between the requestor and provider as detailed in the contract by the internal use of certificates. This is especially important for billing purposes, in order to prevent inaccurate charges to the service requestor. Note that the use of contracts does not preclude relationships between Web services and grid resources published dynamically differently; it merely requires that an appropriate usage model be employed.

The meter and accounting Web service acts as a resource-counter. It provides input for:

1. Billing according to the service provider's rating models (e.g. Common Billing Interface)
2. Payment and tax processing
3. Defining functions and accounting figures like the following:
    a. Log user begin and end times for a particular service

b. Report total resource usage for a specific user
c. Report used service statistics per request
d. Create Service Requestor (SR) accounts (ad hoc account & contract)
e. Create Service Provider (SP) accounts
f. Create IPDR.org conforming XML usage records

The meter service makes it possible to retrieve usage-data in standard form as XML files. This data can be used in a batch-like model, that is, by a billing product which supports processing standards conform input.

To conclude this section, for the business of grid computing and web services to take off, lot of lessons can be learned from the telecommunication industry regarding shared services, peering arrangements, and profit sharing. Combining these lessons with a fine-grained usage tracking and brokering model will provide the necessary fundamentals to scale this technology to its fullest potential.

5. CONCLUSIONS

Over the past decade compute and data-intensive applications have become entrenched in most industries and across all functions. This trend coupled with Moore’s law for server, storage, and networking technologies has created a networked business world that is ripe for sharing compute and data resources to achieve the next level of global growth at much reduced costs. We are already witnessing this phenomenon in scientific grids as well as in web services – both exemplify the current state of grid computing.

This trend will affect the current way of doing business in multitude of ways. First, in the short term, businesses that use CPU intensive applications containing significant parallelism will be able to leverage their internal IT resources much more efficiently. This phenomenon, called *scavenging*, will lead to an immediate productivity gain. Of course this gain has to be carefully weighed against the need for a more sophisticated IT and security management, and various socio-political resistances. The downside of this gain, in the short term, is on the desktop and mid-range server vendors in that they will see a reduction in new orders since their customers will be able to gain more from their existing hardware. It is in the best interest of these vendors not to worry about cannibalization but to actively promote grid technologies so that they can incorporate important features such as virtualization in their offerings. This will help them position themselves better when scavenging reaches its limitations and customers start looking for new servers to fulfill their processing needs.

Second, for the vendors of grid applications, grid computing implies increased developmental and support costs. This is because now their applications will have to run on a range of platforms and Operating Systems variations. In addition they will have to abstract out I/O and data interfaces in order to move freely about in a network. On the business side they will have to adopt sophisticated run-time licensing and monitoring schemes for deployment.

Finally, for most software vendors, grid computing implies a major shift away from the traditional licensing model. If we include *web services* in the definition of grid computing, it becomes very obvious that a per-seat or a per-connection licensing model will not be enough since enterprises will start leveraging software across corporate boundaries in a manner that has never been done before. Collaboration across supply-chains, business alliances, and technology eco-systems is already on the rise as a spate of outsourcing as well as a need for providing vertical solutions across horizontal industries is on the rise in the IT and some other areas of the Hi-Tech sectors.

The best bet for managing this business disruption for the software vendors is to first change their mind-set from the traditional one-time licensing and pricing model to a more usage based – possibly a subscription model. Although this shift creates serious impact on their sales and financial projections they will have to eventually embrace it to facilitate the creation of new business models and for ultimately delivering value to final end-user. The advantage of this model is that it allows the software vendor to grow with the business of its customer. Thus, the vendor is forced to build software that adds true value to its customer's business – a win-win situation for both parties. It also reduces the gorilla heavy handedness that some large software vendors employ while dealing with small customers, and vice-versa.

Another important driving factor behind a subscription or a usage based model is that grid computing forms the basis of the utility computing or on-demand computing model.

While grid computing is the technology - the service that gets deployed is on-demand computing – either offered internally by a company or purchased from external vendors such as Gateway or IBM. Such services are inherently subscription based and as such it makes sense for the software vendors to align themselves along that business model.

5.1 Future Work

In this thesis we have identified key business changes along the distributed software value chain due to grid computing. It is obvious that most long term effects will be on the application vendors in this business. Though we have laid out most of the major issues around these impending changes a thorough analysis of technologies such as Sun's Java, Microsoft's .Net, and IBM's Globus is necessary to find out how close are these technologies in delivering the promise of grid and web services to the non-technical enterprise and consumer mass-market.

In addition to addressing licensing and pricing models across enterprise boundaries the issue of managing authentication and authorization is also of prime importance. We did not address it in this thesis since we believe it is a discussion that is beyond the scope of this work – it is applicable to the entire networking framework. What needs to be looked at, specifically for remote execution of code, is a prescreening process to eliminate viruses and Trojan horses that can cause havoc in a grid. In addition to preventive schemes, other throttling mechanisms need to be put in place to detect abnormal activities – such as an attempt by a grid job to suddenly start hundreds of jobs on other nodes.

BIBLIOGRAPHY

[Christensen 97] Clayton M. Christensen. The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail. Harvard Business School Press, June 1997.

[Christensen et al 02] Clayton M. Christensen, Mark W. Johnson and Darrell K. Rigby, 2002, *Foundations for Growth: How to Identify and Build Disruptive New Businesses,* MIT Sloan Management Review, Vol. 43, Number 3, pp. 22-31.

[Econ 01] “Computer Networks: Computing power on tap,” The Economist, 6/21/2001.

[Ferreira et al 02] Introduction to grid computing with Globus, Luis Ferreira et al, IBM Redbooks, ISBN 0738427969, (2002).

[Foster and Kesselman 99] Ian Foster, Carl Kesselman (Eds), 1999, *The Grid: Blueprint for a New Computing Infrastructure,* Morgan Kaufmann, 1999.

[Gerstner 02] Gerstner, Louis, 2002, *Gerstner predicts information technology will become a utility*, MIT Tech Talk, Oct 30 2002.

[Henderson & Clark 90] Henderson, Rebecca M., and Kim B. Clark. "Architectural Innovation: The Reconfiguration Of Existing Product Technologies And The Failure Of Established Firms." *Administrative Science Quarterly* 35.1 (1990): 9-30.

[IVDGL 02] “International Virtual Data Grid Laboratory,” http://www.ivdgl.org, since 2002.

[Scannell 02] Ed Scannell, 2002, Pushing ‘smart’ computing, InfoWorld, November 15, 2002.

[Schwartz 02] Jeffrey Schwartz, Pushing ‘Grid computing’ To New Heights, VARBusiness, Sept. 20, 2002.

[Tech Review 03] “Ten emerging technologies that will change the world “ showcases Ian Foster and Carl Kesselman for Grid computing. MIT Technology Review, February 2003.

[Tushman et al 86] Tushman, Michael L. and Philip Anderson, 1986, *Technological Discontinuities and Organizational Environments*, Administrative Science Quarterly, Vol. 31. pp. 439-465.

[Trent 97] Trent, Tracy 1997, *Changing the rules on market leaders: strategies for survival in the high-performance workstation industry*, MIT-Sloan Management of Technology M.S Thesis.

[Utterback94] Utterback, James M., 1994, *Mastering the Dynamics of Innovation*, Harvard Business School Press.

[Wladawsky-Berger 01] Irving Wladawsky-Berger, 2001, Grid computing: Advancing e-business into the future, http://www-1.ibm.com/servers/events/gridcomputing.pdf, Kennedy Consulting Summit.