



# **Grid Computing Looking Forward**

By Enrique Castro-Leon and Joel Munter  
Intel® Solution Services

# Contents

Introduction .....	2	Implications of the Grid-Related Technology Transitions .....	14
Section 1: Technology Overview .....	2	Parallel Distributed Computation .....	14
Hardware Configurations: Nodes, Clusters, and Grids .....	3	The Grid and Multi-Core Processors.....	16
Business Advantages that Drive Grid Adoption .....	3	The Role of Services in Grid Adoption .....	16
Shared Heterogeneity—A Transportation Analogy .....	4	Workload Characterization.....	16
Fungibility and Virtualization in Grids .....	4	Technology Transitions Conclusion.....	16
Technology Overview Conclusion .....	5	Section 4: Industry Viewpoints.....	16
Section 2: Usage Models .....	5	Grid-Computing Business Models in Various Industries.....	17
Cycle Scavenging Versus Dedicated Hardware .....	6	Industry Leadership and Grid-Computing Early Adoption .....	18
Application and Data Grids: Economic Advantages.....	6	Grid-System Hardware and Software Design.....	19
Parallel-Computing Grids: Productivity Advantages .....	7	Short-Term Deployment Strategies: Hardware Investment .....	19
Overcoming Cost Hurdles in Grid Business Models .....	7	Medium-Term Deployment Strategies: Application Focus .....	20
Business-Process Innovation Drives Grid Ecosystems.....	8	Long-Term Deployment Strategies: Harnessing Future Technology Transitions.....	21
Grid-Computing Standards Enable Future Innovation ...	11	Industry Viewpoints Conclusion.....	22
Usage Models Conclusion .....	11	References and Related Links .....	23
Section 3: Technology Transitions.....	11	About the Authors.....	24
Hardware-Architecture Advances .....	12		
Transitions in Business Practices and Infrastructure..	12		

## Introduction

*Grid computing is expected to become a mainstream business-enterprise topology during the rest of the decade.*

This paper includes the following four sections:

- **Section 1: Technology Overview** gives an overview of current and emerging technologies in this area.
- **Section 2: Usage Models** presents the roles of the grid ecosystem and international standards in the development of grid-computing business models.
- **Section 3: Technology Transitions** provides insights to decision makers and engineers about the way grid computing is impacted by the general development of contemporary technology.
- **Section 4: Industry Viewpoints** illustrates the challenges, benefits and strategies associated with grid-computing deployment generally and in specific industries.

## Section 1: Technology Overview

A number of technology transitions are taking place or will take place within the next five years that will lower the barriers that exist today to deploy, maintain, and run applications on computer grids. Most of the literature dwells on performance gains and application capabilities enabled by the new technologies. Perhaps a more interesting exercise is to take these transitions to their logical conclusions and speculate as to what new business models will become feasible. A second exercise is to determine the optimal strategies for organizations contemplating grid deployment.

The grid is not only of interest to scientists and engineers running applications—that is the traditional user community for grids. Grid deployments in the next decade will encompass a broad swath of industry verticals that will take the grid well beyond its High Performance Computing (HPC) roots. Beyond capabilities delivered to end users, every

participant in the ecosystem has a vested interest in the acceleration in grid uptake: users enjoying new and powerful capabilities, vendors seeking new channels and additional revenues, and organizations discovering that grid deployment can bring associated cost reductions and a welcome competitive edge.

While attempts at predicting discontinuous events are not usually very accurate at determining actual outcome, the authors believe that the process of building a thought experiment is intrinsically useful. Moreover, the readers, far from being mere witnesses, will find that these ideas will bring other powerful ideas by association that will lead to a positive influence when it comes to grid evolution.

## Hardware Configurations: Nodes, Clusters, and Grids

For the first part of this discussion, we will use a simple three-level abstraction to describe the following grid hardware:

- **Nodes**—A computer in the traditional sense: a desktop or laptop personal computer (PC), or a server in any incarnation, including a self-standing pedestal, a rack module, or a blade, containing one or more central processing units (CPUs) in a Symmetric Multiprocessor (SMP), NUMA, or Cache Coherent Non-uniform Memory Access (ccNUMA) configuration<sup>1</sup>.
- **Cluster**—A collection of related nodes.
- **Grid**—A collection of clusters.

The nodes in a cluster are connected via some fast interconnect technology. Before the introduction of InfiniBand\* and PCI Express\* technologies, there was a tradeoff between a relatively high-performance, single-sourced, expensive technology and an economical, standards-based, but lower-performance technology. Ethernet, a technology designed for networking, is commonly used in cost-constrained clusters. This setup introduces bottlenecks in parallel applications that require tight node-to-node coordination. The adoption of InfiniBand-based interconnects promises to remove this tradeoff.

The clusters in a grid can be connected via local area network (LAN) technology, constituting an *intra-grid*—that is a grid deployed within departmental boundaries—or connected by a wide area network (WAN) technology, constituting an *inter-grid* that can span the whole globe.

This model includes boundary cases as particular instances: a grid consisting of exactly one cluster is exemplified by a cluster accessible to a large community, front ended with grid middleware. Through Web services technology, users in a HPC shop can submit jobs for execution through a single,

local interface, not even realizing that the job may end up being executed thousands of miles away. In this way, it is possible for the supporting information technology (IT) department to optimize costs across a number of facilities around the world, including outsourced service providers.

Conversely, a large cluster—even one that contains thousands of nodes—may not be a grid if it does not have the infrastructure and processes that characterize a grid. Remote access may need to be accomplished through relatively limited operating system (OS) utilities such as *rlogin* or *telnet* or through customized Web interfaces<sup>2</sup>.

A grid made up of single nodes defaults to the setup used in cycle scavenging. Cycle scavenging is discussed in “Section 2: Usage Models” of this paper.

This three-tier node-cluster-grid model encompasses grids of greater complexity through recursion: grids of grids are possible, including grids with functional specialization. This functional specialization can happen at the lower levels for technical reasons (for example, a grid might consist of nodes of a certain memory size) or for economic reasons (for example, a grid might be deployed at a certain geographical location because of cost considerations).

## Business Advantages that Drive Grid Adoption

As described, a grid is essentially a set of computing resources shared over a network. Grids differ from more traditional distributed systems, such as the classic n-tier systems, in the way its resources are utilized. In a conventional environment, resources are dedicated: a PC or laptop has an owner, and a server supports a specific application. A grid becomes useful and meaningful when it both encompasses a large set of resources and serves a sizable community.

The large set of resources associated with a grid makes it attractive to users in spite of the overhead (and the complexity) of sharing the resource, and the grid infrastructure allows the investment to be shared over a large community. If the grid were an exclusive resource, it would have to be a lot smaller for the same level of investment.

In a grid environment, the binding between an application and the host on which it runs begins to blur: the execution of a long-running program can be allocated to multiple machines to reduce the time (also known as *wall clock* time or *actual* time) that it takes to run the application. Generally, a program designed to run in parallel will take less time to run as more nodes are added, until algorithmic or physical bottlenecks develop or until the account limits are reached.

Two assumptions must hold for an application to take advantage of a grid:

- Applications need to be re-engineered to scale up and down in this environment.
- The system must support the dynamic resource allocation as called by applications.

As technology advances, it will become easier to attain both these conditions, although most commercial applications today cannot satisfy either of them without extensive retrofitting.

## Shared Heterogeneity— A Transportation Analogy

Transportation systems follow a similar philosophy as grids, in terms of making large-scale resources available to users on a shared basis. Jet aircraft may cost anywhere between \$50 and \$200 million. A private aircraft might provide excellent service to its owner on a coast-to-coast flight. The obvious shortcoming of this solution, however, is that the cost of the plane and the fuel it takes to fly it across the continent are out of reach for most people, and in any case, it probably does not represent the best use of capital for general-purpose transportation. The reason why millions of passengers can travel like this every year is because aircraft resources are shared—and any single user pays only for the seats used—not for a complete jet and the infrastructure behind it.

Shared-resource models come with overheads: users need to make reservations and manage their time to predetermined schedules, and they must wait in line to get a seat. The actual route may not be optimized for point-to-point performance: passengers may have to transfer through a hub, and the departure and destination airports may not be convenient relative to the passenger's travel plans, requiring additional hops or some travel in a car.

Note that aircraft used for shared transportation are architected for this purpose. Aircraft designed for personal transportation are significantly smaller and would not be very efficient as a shared resource.

Transportation systems are also heterogeneous, where sharing exists on a continuum. In an air-transportation system, users choose among a variety of dedicated resources, including general aviation, executive aircraft, time-shared aircraft, commuter aircraft, and the very large aircraft used in long-haul flights. Likewise, grids tend to gravitate toward heterogeneity in equipment availability during their lifetime, with nodes going through incremental upgrades. Grids tend to be deployed under diverse business models.

While the air-transportation system is an instructive instantiation of a *grid*, it is so embedded in the fabric of society that we scarcely consider it as such<sup>3</sup>. Computer systems will likely evolve in a similar way as aviation did sixty years ago—gradually gravitating toward an environment of networked, shared resources as technology and processes improve.

## Fungibility and Virtualization in Grids

Ideally, the resources in a computing grid should be *fungible* and *virtualized*. Two resources in a system are fungible if one can be used instead of the other with no loss of functionality. Two single dollar bills are fungible, in the sense that they will each purchase the same amount of goods, even if one is destroyed. In contrast, in most computer systems today, if one of two physically identical servers breaks, the second is not likely to be able to take over smoothly. The second server may not be in the right place, or the broken server may contain critical data on one of its hard drives without which the computation cannot continue.

A system can be architected to attain fungibility, for instance, by keeping data separate from the servers that process it. A long-running computation can checkpoint its data every so often, so that if a host breaks, the new host can, when it comes online, pick up the computation at the last checkpoint when it comes online. If the server was running an enterprise application, it could unwind any uncommitted transactions and proceed from there. An online user may notice a hiccup, but the computations are correct.

A virtualized resource has been abstracted out of certain physical limitations. For instance, any 32-bit program can access a 4-GB memory virtual space, even if the amount of actual physical memory is substantially less. Virtualization can also apply to whole machines: multiple logical servers can be created out of a single physical server. These logical servers run their own copies of the operating system and applications. This setup makes sense in a consolidation setting, where the cost of maintaining the consolidated server is less than it would cost if the machines were hosted in separate, smaller machines. A hosting service provider can provide a client with what looks like an isolated machine but which is actually a virtualized portion of a larger machine.

The nodes in a cluster may be “heavy” in the sense of being built as two, four, or more CPUs sharing memory in an SMP configuration. Programs that take more than one node to run can operate in a hybrid Message Passing Interface (MPI)/OpenMP\* configuration. These programs expose large-grain parallelism, with major portions running in different nodes

using the MPI message-passing library. Within one node, each portion is split into a number of threads that are allocated to the CPUs within a node. Building software to a hybrid configuration can increase development costs enormously.

Fungibility helps improve operational behaviors. A node operating in a fungible fashion can be taken out of operation and replaced by another one on the fly. In a lights-out environment, malfunctioning nodes can be left in the rack until the next scheduled maintenance.

In a highly virtualized, fungible, and modularized environment, deploying computing resources in small increments to respond to correspondingly small variations in demand is possible. Contrast this to the mainframe environment two decades ago: because of the expense involved, a shop would wait until the resources of an existing mainframe were maxed out before purchasing and bringing in a new one in what was literally a forklift upgrade.

The main innovation brought up by IBM's System/360\* was the ability to run the same software base over a range of machine sizes. An organization could purchase a bigger machine as business grew. This change was expected to happen over months or years. This capability represented enormous progress over having to re-implement the application base for every new model, as the case was before.

The bar for business agility today is much higher. The expectation for the grid is that resources dedicated to applications can be scaled up and down almost in real-time. Outsourcing to service providers represents an alternative over long procurement cycles. Because commodity servers are less expensive than mainframes, the budgetary impact of adding a new server is much smaller than adding or upgrading a mainframe. Despite this affordability, however, not all applications can take advantage of extra servers smoothly.

The capability for incremental deployment simplifies business processes and reduces the cost of doing business. It enables new business models, such as *utility* computing, where service provisioning is metered to match demand.

A pure utility model is not yet practical today, because the concept can be taken only so far. Even traditional utilities have different granularities and costs. Consider, for example, a traditional electric utility company, where electrons have different costs depending on the time of day and the energy source with which they were generated. Most utilities hide this fact, presenting most residential customers with a single, integrated bill. *On-demand* computing is a more attainable

degree of utility computing, where relatively non-fungible resources are allocated dynamically, within certain restrictions. One example is *capacity-on-demand*, where a large server is sold with extra CPUs that are turned on at customer request. A restriction is that the new CPUs cannot be turned off and, hence, the rates cannot be rolled back.

## Technology Overview Conclusion

With the enormous flexibility and reliability afforded by computing grids, it may seem surprising that they not more pervasive today. The primary explanation for that fact is that grids exist in the context of a large ecosystem. It is not possible to go to a store and purchase a grid. Roadblocks to wider adoption are both technical and business-oriented in nature. From a technical perspective, it is safe to assume that applications not designed in multiprocessor environments are by default uni-processor applications. They can be executed on a multiprocessor node, but they will not use more than one processor, even if more are available, and hence the total run time won't be shorter.

From a cost perspective, it might be attractive to share resources across organizations, including different companies, even in different countries. Doing so implies additional overhead to ensure data integrity, security, and resource billing. The technology to support these functions is still evolving. The lack of precedents makes potential users squeamish about trusting their code and data to be executed by someone else in a shared resource environment represented by a grid. Therefore, few grids today cross company boundaries. The largest user communities for grids today belong to government and academic research.

This challenge translates directly into opportunity for those solution providers and system integrators that can overcome them. As the ecosystem of solutions for grid computing continues to evolve, adoption is likely to increase by private companies that seek to harness the power and cost advantages of grid computing. This paper provides background both for those who seek to create those solutions and for those who wish to implement them.

## Section 2: Usage Models

While economic advantages are likely to be a prime motivator for the implementation of grid computing in enterprise environments, those advantages resist simple analysis from a traditional return-on-investment (ROI) point of view. This complexity is directly related to the fact that building a dedicated grid to be used by a single business entity is generally prohibitively expensive. The real value of grid computing,

however, is largely generated by the notion of a shared-resource model in which many users take advantage of the collective resources of a grid.

As the usage models associated with grid computing continue to advance, it seems likely that the worldwide grid infrastructure may develop along the lines of a utility. This model suggests the development of service providers at multiple levels, including those that build and maintain grids, as well as those that broker the services of grid resources to users. If grid computing does develop along these lines, business entities will be able to take advantage of vast computing resources that they pay for on an as-needed basis, much in the way that individual users pay only for indirect and incremental shares, for instance, of a power-generation facility but are able to use electricity from it as needed.

## **Cycle Scavenging Versus Dedicated Hardware**

As discussed in “Section 1: Technology Overview,” an essential motivation behind the adoption of grid computing is to increase resource utilization through sharing. An incremental way of improving utilization in an existing environment is through the model of *cycle scavenging*.

In a dedicated environment, resource utilization is quite low: utilization factors in desktops might range between 1 and 10 percent. Cycle scavenging essentially overlays a grid-usage model on top of a traditional, interactive usage model. The application of cycle scavenging, however, is subject to a number of limitations. For instance, putting a workstation in a grid may end up inconveniencing both the owner of the workstation and the grid users without showing much gain in terms of ROI or the amount of work accomplished. In a cycle-scavenging environment, users have little control over the configuration of pre-existing hosts. This configuration may not be optimal for running grid jobs.

If a grid model brings some inconvenience to users, consider how it would affect the owner of the host machine. By definition, the normal use of a workstation in a cycle-scavenging grid is interactive. Grid jobs tend to be large and they are, therefore, likely to impact the responsiveness of the host. Even when a grid job is designed to be pre-empted, it may take several seconds for the workstation to flush the gigabyte or so of data in the job. Furthermore, the shared use of the host may pose security problems, especially when resource sharing is conducted among mutually suspicious organizations or user communities.

The compromises associated with sharing a resource under the cycle-scavenging model suggest that building grids out of resources that are exclusively dedicated to grid computing can improve hardware configuration and the user experience. The overhead of a shared-resource model implies some inconvenience. Hosting the service on resources originally deployed for a different purpose may lead to an undeserved negative perception that will slow adoption.

Hosting a grid on dedicated hardware allows the deployment of a system that has been optimized for the intended purpose. Hence, it opens the possibility of specialization, where a grid is handled by an organization whose main goal is to provide grid services. Such an organization may viably provide this service to an entire company or even several companies or state entities and, in fact, separate companies could thrive whose sole charter is to provide grid services to the marketplace.

## **Application and Data Grids: Economic Advantages**

The licensing costs of specialized engineering and productivity applications can be quite high, particularly as development costs cannot necessarily be recovered through selling a large number of copies. A grid infrastructure can potentially increase the utilization of a few expensive software licenses by sharing them over a relatively large user base. This arbitrage may be only temporary, as software vendors adjust licensing models to prevent revenue loss from this type of shared usage.

In a similar fashion to CPU processing power, storage can be distributed and shared in a grid. As in computing grids, where physically distributed computing resources pose a challenge, using unused storage resources on unused hard drives across thousands of clients can be quite difficult. The economic gain from increasing the utilization of otherwise unused storage space must be offset by the cost of moving it across the network. The performance behaviors of a data grid are quite different from a highly concentrated FibreChannel-based Storage Area Network (SAN).

The need to adjust to the differences in those behaviors between data grids and SANs may well lead to new usage models and business opportunities. For instance, a grid with 10,000 data nodes is effectively a device with 10,000-fold redundancy. The aggregation of so many nodes can be an advantage to reduce the probability of data loss. An application could implement a file system probably designed to meet

virtually any level of reliability. It can spread out the data in the files so widely that the system behaves like a hologram: even if many of the nodes are lost, it would still be possible to recover all of the original data from the remaining ones.

The geographical spread is not always disadvantageous. Consider this analogous example. The traditional method for a movie studio to release a motion picture is by physically shipping film cans to movie theatres<sup>4</sup>—this is the film-industry version of the *sneaker net*. It is only matter of time until the entire distribution process becomes digital, where a movie is digitally transmitted and projected with a digital projector. The storage required for a theater-quality motion picture can span several terabytes. Using a central server for sending copies to every movie house in the world is obviously an inefficient way of using long-haul Internet connectivity. Instead, the servers in each theater can be conceived of as nodes in a data grid. Using a tree topology, the studio could send copies of the file to a few designated distribution points in each country or state. Copies would then be sent from the distribution points to a local distribution point in each city, and then locally to all theaters within a city.

## **Parallel-Computing Grids: Productivity Advantages**

Another usage model associated with grids is *parallel computation*<sup>5</sup>. For instance, if it takes one server-node 10 minutes to update 100,000 records, 10 nodes working together (that is, in “parallel”) could theoretically do the same job in one minute. In practice, of course, the time required would be somewhat more than a minute due to overhead: the input/output (I/O) subsystem may experience interference with 10 nodes doing simultaneous updates, there might be data dependencies, and one processor might have to wait until another is finished. Nevertheless, parallel processing would reduce the time required to perform the work.

In some cases, wall-clock time is of primary importance; for instance, a weather simulation done for forecasting purposes needs to be completed on a deadline. If these calculations can be accelerated by applying more CPUs, even if the CPUs interfere with each other, the reduction in execution time can make the difference between success and failure at meeting the deadline.

A similar dynamic applies to simulation and analysis jobs in engineering shops, albeit less dramatically. Because of the potential savings in worker time involved, it is enormously valuable to be able to run jobs that take several CPU hours in a few minutes of clock time. Because design is an iterative

process, detecting a flaw more quickly can equate to significant savings in terms of workers' time, increasing productivity. In the late phases of a design cycle, parametric runs (that is, similar runs with slightly different data) may be necessary. With a job that takes eight hours to run on one CPU, a one-CPU workstation running for an entire month will yield about 100 data points. If an unexpected flaw is discovered in the data at the end of that month, and the run needs to be repeated, the project essentially slips by a month.

If, instead of one CPU, 100 CPUs can be applied to the same problem in a grid environment, it is very likely that the computation will not be done 100 times faster—perhaps just 25 times faster. Thus, the grid system might yield one data point every 20 minutes or so (at 25 percent efficiency). Furthermore, let us assume that a grid with 4,000 nodes is available. In this case, 40 jobs can be launched in parallel and, hence, the team might be able to deliver the 100 data points in one hour.

The productivity implications of being able to do a month's work in one hour are epochal. It might mean saving the production time of a \$100M movie by a few weeks and a few million dollars through the use of parallel rendering engines, or the ability to base real-time quotes on complex derivative securities calculations.

## **Overcoming Cost Hurdles in Grid Business Models**

In the example above, it might be argued that few shops can afford to purchase a 4,000-node grid. The main issue here is that few organizations will keep a 4,000-node grid busy all the time; so they could probably not justify owning a 4,000-node grid. Instead, because of its shared nature, a grid could become a resource shared by an entire economy, much in the same way as other collective resources, such as transportation networks. If there is enough demand across an entire sector, grid services will become a viable model, providing opportunities to entrepreneurs worldwide.

A film-production company makes the news when it spends \$10M to purchase a server farm for image rendering. In the near future, such a purchase may make as much sense as an organization purchasing a jumbo jet for shipping packages across the continent. Any of the commercial parcel-freight companies will do this work for a lot less, and they *do* own jumbo jets for this purpose. Those providers can achieve success from this model because of their process expertise in the transportation business and the ability to amortize the cost of their jumbo jets over millions of accounts and billions of packages.

A similar phenomenon could happen with grids, with entrepreneurs rising to the occasion to provide grid services for specific verticals, or perhaps even an entire economy. They will be able to amortize their capital costs over multiple clients and optimize their business on a global basis.

The over-building of optical fiber that occurred during the dot-com boom led to radical reduction in the cost of moving data across the world. For better or worse, lower communication costs facilitated the emergence of outsourcing in countries like India, Costa Rica and the Philippines. Grids can be placed in emerging economies as well. Further cost efficiencies are expected as people figure out ways of utilizing fiber-optic cable that is already in the ground but not in use.

A few technical hurdles still need to be overcome for this to happen. Security is a vital concern in this area, particularly when international borders are involved. Security may need to evolve to the point that service providers are unable to tell what the host machines are running. The service provider might not even know who the end user is, because jobs may be passed around as commodities in a complex grid-services supply network. This dynamic has a precedent, for instance, in the way mortgage loans are issued and later passed around among institutions, or the way insurance companies co-insure each other to manage their risks.

Another hurdle is being able to package code and data in a way that can be handled by any grid in the world. Today, it is easier to package data than the code that uses it. In some cases, the code needs to be already installed in the host machines. This problem is solvable, for instance, if code is written to a standardized virtual machine such as a Java

Virtual Machine\* (JVM). There is some efficiency loss for the sake of portability and interoperability, but that performance loss could be compensated through the use of performance run-time libraries. The virtual machine could also be architected to take advantage of multi-core CPUs.

Data interoperability will also be facilitated further as representatives of specific industry verticals get together and agree on specific Extensible Markup Language (XML) interoperability standards. Ultimately, what matters in a grid job is a committed service-level agreement. The service provider can turn around and delegate the execution to another service provider, perhaps a consolidator with expertise in a specific vertical industry. The success of this industry will be measured precisely in terms of the richness of the ecosystem that develops around it.

## **Business-Process Innovation Drives Grid Ecosystems**

We just discussed how grid deployments involve much more than the purchase of some hardware; these deployments are intimately associated with an ecosystem around it that ultimately may span an entire economy.

The node/cluster/grid three-layer model for grids (which is discussed at some length under “Nodes, Clusters and Grids” in “Section 1: Technology Overview” of this paper) is actually embedded in a much larger environment, with hardware at the bottom and business models at the top. The characteristics of this ecosystem model are captured in Table 1, The Grid Ecosystem Model. Each layer in the table represents an abstraction that includes all of the layers below it.

**Table 1.** The Grid Ecosystem Model

<b>Level of Abstraction</b>	<b>Enabling Factors</b>		
<b>Virtual Organizations</b>	Legal frameworks, Service-Level Agreements, international treaties, intellectual property, privacy		
<b>Business Vertical</b>	Organization mission, capital sources, investment, business strategy	Corporate Finance, Human Resources	
<b>Business Model</b>	Insource/outsource, depreciation schedules, capital/expense, Application Service Provider, asset service provider, resource utilization rates, cycle scavenging		
<b>Business Function</b>	Research & Development (R&D), Business Operations	Department of Finance	
<b>Applications</b>	Domain-specific codes, application middleware (checkpoint/restart, transaction-processing monitors, application servers)	<i>Web services, System Management (Intel® Active Management Technology (Intel® AMT) Management Frameworks: Tivoli*, UniCenter*, OpenView*)</i>	
<b>Grid Platforms</b>	<i>Internet, LAN, WAN, Grid middleware: Globus Toolkit*</i>		
<b>Clusters</b>	<i>CPU interconnect architectures: InfiniBand*, PCI-Express*, Myrinet*, Qsnet*, proprietary, cluster tools, middleware, Message-Passing Interface (MPI), LAN, WAN, Hypertransport* Technology</i>		
<b>Nodes: Blades, Rack Units, Pedestals</b>	<i>I/O architecture, InfiniBand, compilers, debuggers, LAN, performance libraries, operating systems</i>	<i>PCI-Express, Extended Firmware Interface (EFI)</i>	<i>Box Management Intel® AMT and Intel® Cross-Platform Manageability Program</i>
<b>Baseboards</b>	<i>Chipsets, Customer Subscriber Identification (CSI), Peripheral Component Interconnect (PCI), Hypertransport Technology</i>		
<b>CPU Technologies</b>	<i>Intel NetBurst® microarchitecture, Hyper-Threading (HT) Technology, multi-core processors, New Product Introduction (NPI), Intel® Extended Memory 64 Technology</i>		

This emphasis on the greater context is relevant primarily because the investment organizations will make in grid hardware and software will represent only a fraction of the total economic impact. Research done by Erik Brynjolfsson, Director for the Center for e-Business at MIT's Sloan School of Management<sup>6</sup> indicates that for every dollar of IT hardware capital investment, there are up to \$9 of IT intangible assets, such as business processes, training and human skills involved. It is the linkage between the initial grid investment and the effectiveness of the resulting processes that will ultimately determine the payoff of the investment on a grid

and, hence, the success of grid adoption for that organization. Conversely, grid adoption will not reach a tipping point until the linkage to the other 90 percent is firmly established in the industry's psyche. Because of the nature of the grid as a level playing field, this linkage cannot be established through product features; it must be done at the business-process level.

Table 1 summarizes the layers of the grid ecosystem along with enabling factors. The enabling factors tend to be technology-oriented in the lower layers and business-oriented in the upper layers. The items where there is

significant Intel presence or activity as provider of technology building blocks have been emphasized in *italics*. Some factors can span multiple layers. These have been placed in the rightmost column.

At the bottom layer, the “atom” of the grid today is the microprocessor or CPU. Cost being a strong driver, there is an advantage to using mass-produced microprocessors as the basis for a grid infrastructure. CPU chips, chipsets and other devices are attached to a baseboard, the main module of a computer. The baseboard can come packaged in a laptop, desktop, pedestal server, racked server or a server blade to constitute a node.

The next levels of integration comprise nodes, clusters and grid abstractions. Nodes are usually packaged into cabinets in dedicated grid installations, although it is not unusual to see rows upon rows of PCs in low-cost grid installations.

The *application* layer encompasses all the elements that comprise a delivered application. For instance, the SETI@home application under the SETI@home project<sup>7</sup> encompasses the application software, the servers distributing the software and the millions of PCs running it. In addition to domain-specific codes being run, an application also includes the middleware to make it run. This middleware is the main differentiator between a grid and a traditional cluster. It carries out functions like secure access to a grid facility and ensuring that the authenticated user is entitled to the resources being requested. It allows linking together multiple services into a logical single service. The middleware also provides ancillary services, such a checkpoint/restart, and in enterprise-oriented grids, services to support transactions and application servers.

Grids exist in the context of a *business function*, whether it is an R&D department or a datacenter operations department. If parts of a grid are outsourced, the user community may be within a department and jobs may be submitted locally. The grid may be largely hidden, however, with only a small portion visible. Jobs may run in virtual nodes whose physical counterparts are deployed somewhere else in the world; hence, users may end up using thousands of nodes collectively, even though an individual user might not see more than just a handful at any time.

The next layer up encompasses the business models within which grids are deployed: whether the grid in use is wholly owned by the organization, considerations for sourcing and outsourcing of the various functions in the infrastructure, and whether the grid uses dedicated hardware versus cycle scavenging.

The business vertical segment being served by a grid influences decisions for all layers below. Examples of vertical segments include government research, oil exploration, electric energy systems, automotive research and aerospace structural analysis, computational electro-magnetics and computational fluid dynamics.

The ability of grids to link resources across organizations in a very fluid fashion led to the notion of *virtual organizations*, first described by Ian Foster, Carl Kesselman and Steven Tuecke in their seminal paper “The Anatomy of the Grid.”

Each layer in this system is subject to specific considerations. These considerations are predominantly technical in the bottom layers, becoming gradually more business-oriented as we move up in levels of abstraction. For instance, the main consideration at the bottom layer is processor selection: CPU architecture, associated features and specific technologies (for example, 32-bit or 64-bit architecture, HT Technology, Intel NetBurst<sup>®</sup> microarchitecture and multi-core design). Some considerations, such as manageability, span multiple layers of abstraction.

The deployment environment is of concern at the business-function level; it matters, for instance, whether a grid is deployed in a R&D or a business-operations setting. Not surprisingly, Intel, as a semiconductor manufacturer, has a significant presence at the CPU layer. A very significant number of grid installations run on Intel<sup>®</sup> architecture-based machines. CPU technology innovations introduced by Intel have helped the grid become feasible. While it would be technically possible, it is hard to conceive of a grid comprised of mainframes or discrete-logic computers.

Chipsets, baseboards and computer building blocks built by Intel ensure that ecosystem participants, whether system integrators, value added resellers or original-equipment manufacturers, can bring the capabilities of the newest processors to market very quickly.

Intel also had a pioneering role in the development of InfiniBand I/O technology, a derivative of the Virtual Interconnect Architecture of the late 90s, which in turn had roots in the Paragon<sup>®</sup> supercomputer mesh interconnect built by Intel starting in the mid-90s. The Paragon interconnect was architected based on the early experience of the Intel<sup>®</sup> iPSC interconnect of the late 80s. This topic is discussed with more detail under “Parallel Distributed Computation” in “Section 3: Technology Transitions” of this paper.

Beyond I/O interconnects, Intel has a strong presence in the manufacture of networking components.

## **Grid-Computing Standards Enable Future Innovation**

Analogous to the technology transitions that have been described in this paper, an ecosystem is not possible without standards supporting it. As with many maturing technologies, there are de facto standards and standards in development at various Standard Development Organizations, also known as SDOs. One relevant example of an SDO is the Organization for the Advancement of Structured Information Standards (OASIS). The Global Grid Forum (GGF\*) is another significant group working in this arena.

Examples of related work going on at SDOs include the four newly created OASIS Data Center Meta Language (DCML) Technical Committees. They have been established in the areas of Framework, Network, Server and Applications & Services. The DCML is one language that describes schemas for how servers, networks, applications and services can utilize data that had been previously isolated in an automated, on-demand fashion.

This is not the only area where standardization is occurring. Additional efforts are underway at the Distributed Management Task Force (DMTF) in a Utility Computing Working Group. This effort is designed to utilize DMTF's Common Information Model (CIM). The DMTF Utility Computing Working Group will define how to assemble complete service definitions. This will include work on the composition of the models in CIM, as well as business- and domain-specific functional interfaces.

The GGF will continue to be a key driver during the next five years. Their Web site describes the GGF as a non-profit "community-initiated forum of thousands of individuals from industry and research [to] promote and support the development, deployment, standardization, and implementation of Grid technologies and applications." They carry out this mission through the development of Best Practice guides (technical specifications), user experiences and implementation guidelines. Intel is a Platinum Sponsor of this effort.

Recent draft submissions to the GGF include topics such as "Operations for Access, Management, and Transport at Remote Sites," "Open Grid Services Architecture: Glossary of Terms," and "Guidelines for IP Version Independence in GGF Specifications." There are more than 150 final documents posted on their Web site. Within that set of final documents, you can find information covering myriad grid issues, including "Managements of Grid Services" and "Networking Issues Within Grid Infrastructures."

As standards continue to mature, they will provide the basis for the software development that will bring these technologies to maturity. Improving datacenter performance is a central theme for next-generation computing architectures, as customers search for ways to simplify their infrastructure and cut costs. The initial drafts being standardized at OASIS and other organizations will largely come to maturity in the next five years.

## **Usage Models Conclusion**

Grids can only be understood in a larger context that includes usage and business models and processes—and even issues of national and economic development policy—given that a grid can span multiple organizations and international boundaries.

Large-scale grids are inherently federated and heterogeneous. It is only through commonly agreed-upon standards that enable interoperability that grids can exist. It is unrealistic to build a grid that depends solely upon components or products of the same type, because even products from a single manufacturer evolve over time. This fact does not, however, preclude smaller-scale homogeneous grids deployed early on to facilitate institutional learning.

The grid playing field is extremely level and wants to stay level. Manufacturers can introduce improved products, such as a better-performing InfiniBand switch chip, as long as it is interoperable. An exclusionary implementation of a standard will not even yield a tactical advantage to the manufacturer. It is more likely that the product will end up selected out.

## **Section 3: Technology Transitions**

Grid computing has strong HPC roots, perhaps because the early demands of HPC dictated that some of the solutions, technologies and usage models associated with grid computing be investigated in an HPC context first. This is the case with cycle scavenging and distributed parallel computation. Technology developments like multi-core CPUs may become forced functions for the pervasive use of multithreaded and parallel programming techniques that have been in use in the HPC space for more than 20 years, both for grid computing and in other areas of the computing industry. A quantum jump in the beneficial impact of grid computing will take place when the grid gets adopted in a broader context, including the enterprise and consumer spaces.

The architectural advances that are taking place today, and which will continue to develop over the coming years, will set the stage for widespread adoption of grid computing among relatively small users. This sphere of technology, which is currently widely associated with government and university research environments and the largest corporations, may become well within the reach of all businesses by the end of the decade. The parallel distributed computation enabled by this model will enable businesses to undertake computationally intensive operations such as sophisticated rendering and analysis that would otherwise be impossible for them to do directly.

## Hardware-Architecture Advances

A number of technology transitions will take place in the next few years that will accelerate the adoption of grid computing. The following list provides a sample that hints at developments to come.

**Multi-core CPUs:** For the past twenty years, single-chip CPUs have been the de-facto building blocks for computers. It is hard to believe that these twenty years are but a snapshot of a larger trend toward integration in the 60 years or so that computers have been built using electronic components. The initial use of tubes in the late 1940s led to the use of discrete transistors in the 1950s and to the use of integrated circuits in the 1960s.

The advent of integrated circuits accelerated the rate of integration—beginning with simple gates (ANDs, ORs, flip-flops and such others), to Run-time Library (RTL) modules, to functional units—until Intel squeezed a whole microprocessor in the 4004 chip of the early 1980s. Most of the advances at this stage were enabled by increasingly smaller trace features, with improvements in fabrication that allowed building larger processor dies reliably. While a modern Itanium® 2 processor or Intel® Xeon™ processor runs many orders of magnitude faster than the original 4004, this improvement has been scalar in nature, essentially allowing a single program to run faster. A state-of-the-art microprocessor today can contain billions of transistors.

Unfortunately, technology is reaching a point of diminishing returns, where the transistor budget is growing much faster than performance gains. This fact, coupled with ongoing fabrication advances, have led to another milestone: it is now possible to place two or more CPU cores on a chip. And the aggregate performance of these cores is faster than if the transistors were placed in a single, more complex core. The pervasive presence of multi-core CPUs could become an incentive for building parallel applications.

### **High-bandwidth, low-latency memory architectures:**

Improvements in memory technology have taken place at a slower pace than CPU technology. Still, reductions in cost per byte have made it possible to build mainstream systems with more than 4 GB of physical memory.

**PCI-Express-enabled chipsets:** The introduction of the PCI bus for attaching peripherals to a CPU in the early 1990s brought considerable improvement over the older Industry Standard Architecture (ISA) standard. The PCI protocol is arbitrated, and in most implementations, data from the CPU to a peripheral needs to cross at least two chips. The performance of this setup is increasingly out of balance, relative to the bandwidth and latency needs of present-day CPUs. The new PCI-Express standard is point-to-point and can be aggregated to fit a target bandwidth. Implementations where data crosses only one chip are possible.

**InfiniBand:** InfiniBand is a point-to-point protocol that allows moving I/O streams to be moved out of a baseboard to a peripheral device or another baseboard a few feet away. It extends the reach of the predecessor PCI I/O bus that is limited to no more than a few inches. InfiniBand will increase the flexibility with which distributed systems are architected and operated. For instance, having computation physically separate from storage facilitates provisioning. Nodes without spinning storage can be installed or removed almost at will. In fact, even if a node has a local boot drive, if it carries no other data than temporary buffers, this node can be pulled out and replaced by another one that gets re-imaged out of the common store in very short order.

**Backplane interconnects:** Reductions in component size now make it practical to build large-scale bladed systems. Computers or servers are arranged like books in a bookcase, instead of the pancake paradigm used in rack units. Blades are inserted in a metal enclosure or cage. The blades carry no connecting wires. Instead, they plug into the back of the cage, the backplane. The backplane has built-in conductive traces that carry power and I/O signals to feed the blades. I/O can be done with a number of technologies, including Ethernet, FibreChannel and InfiniBand. Most backplanes are passive, which is to say that they carry just wires, with no chips.

## Transitions in Business Practices and Infrastructure

As businesses around the world become more nimble with regard to adopting new technologies, the infrastructure and support associated with those technologies continues to grow. IT departments become more sophisticated and

integral to business processes within the companies they serve. And each internal technology transition better positions a company for the next one.

One important aspect of recent infrastructure advances has been the advent of worldwide high-bandwidth communication. The dot-com boom led to a fiber build-out that occurred much faster than bandwidth consumption. Many of the companies that built this infrastructure went into bankruptcy when the hoped-for revenue stream never materialized. Much of the fiber in the ground today is literally “dark” because equipment has never been connected at its ends. The economics of fiber led to an initial overcapacity: the incremental cost of adding strands is very small and, hence, cables with 1000 or more pairs were laid when only two or three were needed. Companies that had rights of way, such as railroad, gas, electric and gas utilities, laid out fiber every time they had to dig—even though there might not be a business model for the utilization of this capability—because the cost of the cable was a small fraction of the cost to dig in the first place.

Another key technology transition has been the rise of technologies that enable virtualization, automation and modularity, reducing the cost to provision, manage and maintain large systems. This is a second-order effect stemming from the introduction of InfiniBand, PCI-Express, and Internet Small Computer System Interface (iSCSI), along with Moore’s Law.

These technologies allow great freedom in how the architectural components in a system are laid out. Traditional systems, for instance, have hard drives connected to the baseboard through a fairly short cable. This requires hard drives close to the CPU in a Direct-Attached Storage Devices (DASD) layout. With InfiniBand, this is no longer a requirement; hence, storage can be consolidated on the side, in another room or somewhere else in the world, enabling much denser blade form factors that encourage the application of parallelism.

Security enhancements are another key technology transition that enables next-generation grid computing. While the implementation of security enhancements carry processing and organizational overhead, improvements to security functionality are vital to the development of new computing capabilities. The new capabilities associated with grid infrastructures include single sign-on, which allows access to multiple resources in a single authentication operation. The presence of a security infrastructure reduces risk for users when data crosses organizational boundaries. It also simplifies administrative processes such as billing. Jobs safely run across organizational boundaries, while preserving data and code integrity and privacy.

The rise of Web services also represents a locus of opportunity in the grid-computing sphere. Web services, as an integration technology, constitute a natural match for building heterogeneous computing grids. This property of Web services is so useful that some prior standards and platforms, such as the Globus Toolkit, were re-engineered to incorporate the use of Web services. The article “Web Services Extend High-Performance Computing Grid Capabilities” discusses this aspect of the technology in more detail.

As the prevalence of embedded computing continues to develop, the notion of a grid node can be extended downward toward simpler devices, such as the following:

- Home appliances
- Portable Digital Assistants (PDAs)
- Cell phones
- Electronic “motes”
- Active Radio Frequency Identification (RFID) devices
- Passive RFID devices

Grids can conceivably be used to implement “sensor networks,” where data exists as a continuum between the physical space and cyberspace. Data entry, or perhaps more important, data re-entry, becomes unnecessary in principle. A package-delivery company could use embedded RFID tags in packages that are registered with the system at pickup time with the tag remaining ‘in sight’ of the computing system until it is actually delivered.

As soon as the sender hands the package to the carrier, it is detected by a wireless device they wear, which relays the data to the truck. The truck, in turn, relays the data further inside the grid, triggering several database updates, including retrieving the sender’s account and making a credit-card charge. As the package moves to the local warehouse, the regional hub, the local warehouse at the delivery end, and finally at the destination drop-off, different grid nodes would be involved along the way. The carrier might be wearing a specialized combined PDA/Voice over IP (VoIP) mobile phone with an RFID detector; while the truck might be fitted with a router equipped with Wireless Fidelity (WiFi), WiMAX and satellite links.

The truck might communicate with the company’s data-center, and again it might not. The database itself could be distributed, with different pieces of business logic provided by a number of service providers, and the whole infrastructure integrated with Web services. The prevalence of standards allows the carrier to outsource nearly any

component in the system—including trucks, aircraft and services—but because of the system interoperability, the package goes through a series of smooth handoffs as it moves throughout, no matter who “owns” it.

## Implications of the Grid-Related Technology Transitions

Grid-computing systems are essentially federated systems with enormous variety and autonomy across constituent subsystems. Looking at a grid as a “product” will yield an incomplete picture. For instance, no one can walk into a store and “purchase” a grid. No vendors offer “grid-ready” or “grid-compatible” components, nor are such offerings likely to be available in the foreseeable future.

For more insight into this dynamic, it is useful to look at how grids work in other contexts. Consider again the case of electric utility companies: electric utilities don’t acquire grids as a unit; they *build grids* and these grids evolve over time. Alternatively, they may acquire other companies that may already own grids. These companies are more than the sum of the energy sources, generators, substations and transmission lines they own. The character of an electric grid is shaped by the business processes used to run the grid, the available sources of capital, the ownership models, the regulatory environment (including federal, state and public utility commissions), and their relationships with subscribers.

Computing grids can (and will) be every bit as rich and complex as any utility. In this environment, compatibility, interoperability and flexibility are fundamental traits. Some business models will become obsolete; for instance, the per-processor license basis on which most software is sold today, which assumes that the software is bound to a CPU, might become untenable. This scheme becomes impractical in an environment where a program run can be shipped anywhere in the world, where one run takes 10 processors, and where the next one requires 10,000. Using a grid to share the license of software running in a few nodes represents adapting the grid to the current licensing model. This is untenable in a dynamic grid environment and, eventually, licensing models will need to be changed and adapted correspondingly.

The grid supports proprietary solutions only insofar as they provide interoperable interfaces at some level that allow customers to do useful work. Some vendors will attempt to “throttle” interoperability in an attempt to maintain or gain market share, with implementations that are difficult to work with. These attempts will ultimately fail in a Darwinian

fashion. The transparent, community approach of Open Source software is well aligned with the dynamics of grid deployment; this is one of the reasons behind the prominent role of Linux\* in grid computing.

The technology building-block approach associated with Intel architecture is also well-aligned with the grid environment, where Intel supplies the “atoms” and “molecules” for the grid, while the “compounds” or solutions to be built from these elements are built by many players in the ecosystem, as driven by customer needs.

The grid is an extremely level playing field, where none of the grid constituents is single-sourced and where excellence is measured in technological and business capability, not in proprietary advantage or exclusivity. Locking out the competition is ultimately counterproductive because it leads to reduced synergy.

## Parallel Distributed Computation

The core technological capability of the grid is parallel distributed computing. The first decade of the third millennium brings to fruition a 25-year chapter in the evolution of the technology that underlies parallel distributed computing: namely, the trend toward commoditization. In the early 1980s, advancing the state of the art in high performance, parallel computation required building everything from scratch, including the CPUs, memory, component packaging and operating system.

By the late 1980s, commercial, off-the-shelf (COTS) commodity CPUs were becoming powerful enough for high-performance applications. The enormous research expenditure required to build a state-of-the-art CPU to be amortized over a few hundred or at most thousands of deployments was no longer necessary. The Intel® Supercomputer Systems Division was founded to take advantage of the rapidly evolving commodity CPU technology.

The use of commodity processors in this second generation led to a 25-fold improvement in cost/performance. A representative of the first generation was the Cray-1\* supercomputer that yielded about 1 GigaFLOPS and cost about \$5 million. An Intel architecture-based machine of equivalent power could be built for around \$200K.

Nevertheless, the CPU of a second-generation machine was less powerful than a custom first-generation processor. Making a virtue out of necessity, second-generation machines were built as a collection of *nodes*, each consisting of one or more CPUs with memory attached. These machines could

be scaled by replication, or by scaling out, in today's terms. Second-generation machines were faster in terms of price/performance and even in absolute performance for some applications. One of these machines, the ASCI Red\* supercomputer deployed at Sandia National Laboratories in 1996, was the first to reach the watershed performance of 1 TeraFLOPS, or one trillion floating-point operations per second. First-generation machines had a high SMP configuration connected to a single bus, which ultimately limited the number of processors it could serve. The concept of a parallel distributed application became fundamental to reaching the desired target performance goals, as it is today with grid computing.

Similarly to first-generation machines, because there was no precedent technology, the commodity processors and memory of second-generation machines were placed in specially built baseboards. And a fast node-to-node communication interconnect system was built with proprietary or single-sourced components. Existing networking technologies, such as Ethernet, could not provide the required performance in terms of throughput and latency. The OS was also customized to handle thousands of nodes as a logical entity.

The third generation in this evolution came around the year 2000 with the increasing adoption of commodity clusters. The first clusters were built on small budgets using Ethernet technology as the interconnect. These clusters are severely limited in scalability to no more than a few tens of nodes. At that time, a GigaFLOPS machine could be built for about \$4,000. Fortunately, the work from the second generation was not lost; the interconnect developed by the Intel Supercomputer Division underwent an evolution of its own, eventually becoming the basis for the InfiniBand I/O technology, which is an industry standard.

Today, the capability of the second-generation machines can be re-created entirely from off-the-shelf components and GigaFLOPS capability can be achieved for less than \$500. In fact, a single Itanium processor is several times as powerful as the Cray-1 of yore.

The completion of the commoditization stage may signal the arrival of a turning point in the computer industry, with the opening of significant business and economic opportunities. Those opportunities would be available by taking advantage of the application of commodity components in parallel distributed computation, in general, and grid computing, in particular. Capabilities that used to be the domain of university and government research labs are now within the reach of Value Added Resellers and Systems Integrators.

The opportunities derived from commoditization are also open

to emerging economies. Because of the organic nature of the grid, it is almost certain that this model will take different evolutionary paths in emerging economies, although it will be built out of the same components everywhere in the world.

An interesting topic for speculation is whether the grid will mirror the patterns for outsourcing seen elsewhere in the information industry. Because a grid architecture tends to blur the effect of geographical distance and because labor is a significant component of the operation of a data center, it would not be surprising to see grid datacenters migrate to countries with lower labor costs. This effect may be tempered by security and privacy concerns. Security technology will continue improving, although privacy is a non-technical issue that is not likely to go away anytime soon. These concerns may limit initial grid deployments to multinationals where presence in multiple countries keeps a grid within organizational boundaries.

A third barrier can't be improved: the speed of light. It takes up to 0.3 seconds for a signal to travel half-way around the world over a satellite link, or a little less over a fiber-optic link. This timing does not consider additional equipment delays. This delay, or latency, determines the minimum unit of work, or working set, that can be processed efficiently by the system.

For instance, let's assume a hypothetical case where a computer in the United States requests a transaction to be processed at a Chinese datacenter and it takes one millisecond to execute with a 1-second round-trip latency. Furthermore, assume that in order to send a second transaction, the results of the first one are needed. In this setup, the grid utilization is 1 millisecond per second, or a mere 0.1 percent, which is probably unacceptable. Circumventing this problem often requires clever programming. In this case, if it were possible to have 1000 transactions in transit simultaneously, the problem might be solved with some tinkering and re-engineering.

The same inference can be made about data: if a grid can process 6.4 GB of data per second, data sets need to be at least 6.4 GB in size. If the data sets are smaller than that, the system starves and utilization goes down. The product of processing speed in terms of bytes per second times the latency time in seconds yields the characteristic working set in bytes for a given problem. This is the smallest problem set that will fully utilize the grid. The inefficiency associated with a grid working on undersized data sets is analogous to that of a jetliner flying with too many empty seats. Thus, small problems are still better processed locally with a single computer.

## The Grid and Multi-Core Processors

While Moore's Law continues unabated in terms of gates per chip, another turning point has been reached in this decade: until very recently, extra performance came from an ever-faster-running processor and from the use of functional units to uncover parallelism within the instruction stream. Continuing on this path has led to increasing heat-dissipation problems.

At this point, it becomes more power-efficient to run two or more processor cores on the same CPU chip. One core of a dual-core CPU may be slightly less powerful than the prior-generation single-core version. However, when the two cores are used together, they are significantly faster than the single-core version.

This situation will create a powerful motivation for both hardware suppliers and applications vendors to incorporate parallelism into their solutions. Vendors may experience significant user resistance to migrating to a multi-core environment if the application can use only one of the cores, where the performance running with one core is lower than in a prior-generation single-core CPU.

Over the long term, application vendors and consumers will become increasingly comfortable with building parallelism into their applications. This familiarity with parallelism will also make it easier, eventually, to port and run these applications in a grid environment.

## The Role of Services in Grid Adoption

Grids and expert services are strongly correlated because of the role that integration at multiple levels of abstraction plays in grid build-outs. A product's architecture and feature set are not sufficient to determine its suitability as a grid component, any more than the behavior of a gate design can determine the behavior of a finished PC. Additional context is needed and architectural layers need to be added to encompass the complete ecosystem described previously.

For an emerging technology such as grid computing, the existence of service organizations in the ecosystem offers the opportunity to accelerate the discovery and diffusion of the collective knowledge and experience necessary to build grids. This knowledge benefits both grid technology providers and grid consumers.

## Workload Characterization

HPC and enterprise transactional workloads exhibit fundamentally different behaviors. The unit of execution for

most enterprise applications is the *transaction*—a transaction runs a small piece of code that carries out a single or a small number of functions, which trigger database updates. An example of a transaction is an account-to-account transfer where one account is debited and another is credited in a single operation. A high-end server can easily execute hundreds of thousands of transactions in a second.

In contrast, data sets associated with HPC applications may be enormous: anywhere from hundreds of megabytes to terabytes. Because of the significant number-crunching involved, a single run may take decades to finish if run in one CPU.

Multiple CPUs are applied to transactional loads to increase throughput, whereas multiple CPUs can be applied to an HPC load to reduce the total run time as measured by a wall clock. Grids can be designed to run any of these loads.

## Technology Transitions Conclusion

While the adoption of the grid is gaining momentum due to recent technology developments, such as Open Source software, virtualization, progress in manageability technology and the emergence of InfiniBand and PCI-Express, technology alone is insufficient to explain the dynamics behind grid computing. And as much as product companies wish it were true, correlating grid benefits with product features makes even less sense, because a successful grid deployment is not conditioned to any single feature or even a combination of features. Grids can be deployed with 32-bit CPUs or 64-bit CPUs, with desktops, laptops, racked servers, blades or pedestals—and no single feature will make a deployment “better” than any other.

Ironically, it is very likely that the grid will disappear into the woodwork just as it reaches its tipping point, not because it is going away, but because it will be so commonplace that it will become implicit. An example of this dynamic is virtual memory, which is a given in any modern OS.

## Section 4: Industry Viewpoints

The advance of grid computing into industries that are currently largely outside the sphere of this technology will allow new capabilities for those industries by cutting costs dramatically to undertake increasingly ambitious projects and to offer more advanced capabilities to their customers. While the implementation details are different for each industry, certain strategies are common to all of them.

As a first stage toward the implementation of grid computing in their business models, most businesses should explore the viability of deploying dedicated hardware resources, rather than attempting to scavenge spare cycles from existing equipment. By creating a homogeneous environment initially, the company greatly simplifies the effort and minimizes the amount of application optimization required to run efficiently on the grid. Once the first-generation grid environment is in place, the company can move toward refining their applications to take better advantage of the environment and toward incorporating future technologies as they become available.

## **Grid-Computing Business Models in Various Industries**

**Healthcare:** Some of the thorniest challenges in this industry concern addressing supply-chain and enterprise resource-planning issues. Private-sector solution providers have thrived providing services to this segment. Some of these applications could be implemented as grid-sensor networks. For example, where patients in a hospital are issued active RFID tags to keep track of vitals, treatments and prescription schedules to reduce the errors that would otherwise jeopardize the quality of patient care. These tags will also make it easier to implement regulatory mandates and to manage insurance claims, while minimizing the opportunities for fraud.

Tagging the most expensive drugs, which might cost tens of dollars a pill, will help manage inventory, batch management and expiration. It will be easier to track a batch from production to consumption, and to manage recalls and safety advisories. RFID tagging, combined with a grid infrastructure, could also reduce counterfeiting, tampering and shrinkage. In a grid system, the processing of the data will be distributed. Data to be sent to the manufacturer can be aggregated and processed locally to protect patient privacy in a provable way.

Where health systems are being consolidated, whether by mergers and acquisitions or by process reengineering, a grid-inspired distributed database for storing patient records might make more sense than using a more traditional consolidated, massive database.

**Financial-services industries:** The financial-services industry has been a pioneer in the application of high-performance computing technology and is today a leader in the application of grid technology. An application ten years ago deployed on a Paragon supercomputer produced real-time quotes on certain mortgage-based securities.

Computations that took several hours to run on a mainframe could be run in seconds on a parallel supercomputer, allowing customer representatives to deliver quotes immediately.

Today, grids are promising for securities trading, for tasks such as performing risk and derivative calculations, trading decision support, performing “what if” analyses to assist in building optimization strategies, and in data mining. They can be equally useful in banking, asset management and insurance, speeding up tasks such as risk analysis, fraud detection and actuarial analysis.

Benefits conferred by grid computing will include fault tolerance through virtualization and geographical distribution through multiple service providers. Grids allow throttling resources up and down to meet a service-level agreement. For instance, when a computation needs to finish within a pre-determined time, the application can be designed to take advantage of parallelism. Once this capability is architected within the application, it is matter of scheduling the appropriate number of processors to ensure that the run time does not exceed a pre-determined interval. Furthermore, it is not necessary to wait until the next procurement period; the extra processors can be summoned from a service provider just for the duration of a run.

The interoperability among grid components is also applicable to legacy integration. Where it makes sense, pre-existing applications could be integrated into the new grid infrastructure, perhaps through the use of a Web services Application Programming Interface (API). The grid middleware should be able to keep track of resource usage, allowing highly deterministic cost accounting.

**Government and academic research:** For government labs and universities, grids will make it easier for large communities to access clusters hosted in national laboratories or in regional computing centers. Standardized front ends and access APIs will facilitate resource marshalling as needed, including combining the resources of several clusters.

The agility possible from this capability can support time-constrained calculations that can have immediate public benefits. For instance, the results of a predictive weather-modeling simulation can help the planning of emergency preparedness during a severe storm. As another example, running a real-time electric power system contingency analysis could help system operators take defensive measures to minimize transients that can bring the system down, preventing a blackout.

**The film industry:** The increasing use of computer effects in feature films requires massive amounts of computation. Tasks include physics modeling and particle simulations, ray simulations, running animation and character tools, and compositing (combining scenes shot against a blue-screen background with actual backgrounds).

Until very recently, the largest rendering jobs were done on server farms based on Reduced Instruction-Set Computers (RISC) processors. The compelling cost advantage of Intel architecture-based platforms has triggered a migration to these platforms, but a new server-farm deployment can still cost several million dollars.

If widespread deployment of the grid-computing model successfully decouples data and programs from execution vehicles, providing the ability to command thousands of nodes on a per-job basis, such massive investment would become unnecessary in many cases. It will be possible to securely ship and run a job that takes years of processor time and run it on massively parallel systems in a dramatically shorter period. The job could be done by a grid services provider that does not even own the grid, but acts instead as an aggregator for lower-level grid services in a rich ecosystem.

Small studios with big needs could pay only for the time they need, thereby converting an otherwise-untenably large capital expenditure into a manageable operating cost. This ability could enable independent studios to undertake projects that would otherwise be impossible. Grid infrastructure, as opposed to individual investment, makes the sharing of a cluster across multiple organizations possible.

**The electronic-gaming industry:** Game development bears some resemblance to film production in that it often involves very large numbers of pre-rendered images. Grids can be equally useful in game operations, especially in the support of massively multiplayer games that may involve tens of thousands of simultaneous players<sup>8</sup>. The traditional architecture of using a centralized server infrastructure is not scalable with the demands of the game or with the number of players who sign on. Game response times can therefore deteriorate during high usage periods.

Under a traditional architecture, relief does not come until additional servers are purchased. Under a grid infrastructure, the gaming application can be designed to dynamically allocate additional servers, tracking the usage demand and ensuring that performance does not degrade.

The game data can be distributed throughout the grid to optimize locality behavior. Likewise, the game can be designed

to optimize the balance between local computations done at the customer's client versus computations performed at a service provider's server. A customer with a powerful PC might have more computations done locally, yielding better game responsiveness. With a customer using a PDA, the application may be designed to rely more on the servers.

The application designers will likely use the authentication, authorization and billing services built into the grid middleware, with a corresponding reduction in development cost.

## **Industry Leadership and Grid-Computing Early Adoption**

In this section, we will document two early adopters and industry grid pioneers: eBay\* and Google\*. It is said that Google operates the largest grid in the world, although the company has consistently downplayed the number of servers it operates. Estimates run between 50,000 and 100,000 in 2004. This infrastructure is tended by fewer than 100 people. Unlike the experiences of other companies, server sprawl is not an issue for Google.

Their competitive advantage today lies in the software implemented to automate administration and the processes and continuous improvement established for system management. Google has implemented the grid principle of stateless servers: a failed server is left in place until the next scheduled service, since others can take on their work. One of the reasons behind the popularity of Google is the responsiveness of the search application. This responsiveness has been achieved by the use of parallelism in query execution and by replication of the Web-crawler database.

eBay experienced a major outage in 1999 that resulted in a total loss of service. The architecture of the system made it vulnerable: bid information was maintained in a single massive database that was both a point of contention and a single point of failure. The business logic for all except one application was also centralized.

The lessons learned from the 1999 outages led to a system redesign with applications written in portable Java 2 Platform, Enterprise Edition (J2EE\*). The monolithic system was disaggregated and re-engineered as an array of service components and to facilitate fault isolation. The single back-end was split to maintain four or five large search databases that took into account geographical locality. The servers in the original design had tens of CPUs. These were scaled out with servers in the six-to-twelve CPU range. Some of the servers are used as spinning reserve; they do not come online unless a primary server fails.

The extra redundancy built into the system allows it to be run 24x7; it is never brought down for scheduled maintenance as was required with the old implementation. The operating results are exceptional: although the system usage went up by about an order of magnitude, downtime went down from about 15 days per year to just a few hours.

Although grid principles were applied by both Google and eBay, these systems are not “true” grids, in the sense that some of the components were implemented in-house, and today they represent a competitive advantage for the companies. This is true for most early adopters where the missing solution pieces are implemented in house. Eventually, one might expect that these pieces will be implemented using industry standards.

Table 2 shows the dramatic growth of resource usage and site availability by eBay users.

**Table 2.** Historic eBay\* Usage<sup>9</sup>

	June 1999	December 2003	June 2004
<b>Page Views</b>	54M	644M	509M
<b>Searches</b>	5M	139M	205M
<b>Listings</b>	532,000	3.4M	3.5M
<b>Bids</b>	900,000	7.7M	6.5M
<b>Outbound e-Mail</b>	1M	22.1M	25M
<b>Peak Net Usage</b>	268 Mbps	7.4 Gbps	7.1 Gbps
<b>Site Availability</b>	95.2 percent	99.93 percent	99.94 percent

**Note:** December is peak season for eBay, which explains why the December 2003 traffic is higher than the June 2004 traffic. This data suggests that eBay has enough reserve capacity to handle anticipated peaks while maintaining quality of service.

## Grid-System Hardware and Software Design

The number of CPUs packaged in a node can range between one and 64, or more. The number of nodes in a cluster can range from a handful to 20,000 and beyond. A grid can span the whole world, because most any PC connected to the Internet can join it. Grid technology is the only means known today to build systems spanning upwards of thousands of nodes.

Software designers apply more than one CPU to a problem to reduce the wall clock time that it takes to solve it. This is the same dynamic that takes place when more workers are brought in to speed up the completion time of a project. In real life, bringing more workers will help up to a certain point. This is because workers need to communicate and coordinate the tasks to be performed. As the number of

workers increases, duplication and interference leads to diminishing returns. Additional resources, such as project management, need to be brought in that do not directly contribute to the project, only to coordinating resources. Extra up-front planning is required as well, even before the project proper starts, lengthening completion times.

A similar dynamic occurs in a computing environment. A program originally designed to execute on a single CPU will still execute on a single CPU, even when it is run on a two-way or four-way machine, and the other CPUs will just sit idle. A program needs to be re-designed, or at the very least re-compiled, to make it run in a *multithreaded* manner in order to take advantage of more than one CPU in a node. If more performance is needed, further enhancements are needed to make it also run in a *distributed* fashion. These modifications are labor intensive and take a high degree of expertise to implement correctly.

In the computer equivalent of the need for worker communication, every once in a while, the different parts of a distributed program will require the results of computations performed by other nodes. These intermediate results need to be moved very rapidly amongst nodes. A fast communication network, usually faster than a WAN or even a LAN, is used for this purpose: the node-to-node *interconnect*. This interconnect needs to operate at near-memory speeds. Otherwise, the CPUs will sit idle waiting for data to arrive or for the information transmission to be completed. Inexpensive clusters are sometimes built using LAN technology. These clusters are severely limited in scalability, except when running a narrow class of problems, called embarrassingly parallel, that require very little communication.

A cluster can be optimized to run either HPC- or enterprise-type transactional loads. Both types of loads require an efficient interconnect. HPC-type loads use the interconnect for the instances of a program running in different nodes to communicate with each other. In a cluster with mesh topology running transactional loads, the interconnect is used to load the code and data necessary to process a transaction and to push the results back into the database as fast as possible.

## Short-Term Deployment Strategies: Hardware Investment

For shops contemplating a first deployment of grid technology, in spite of the purported cost savings of cycle scavenging using pre-existing computing resources, cycle scavenging is probably *not* the deployment mode to try first.

First consideration should be given to the deployment of dedicated grid resources where the constituent nodes are homogeneous. The drive to mop up every idle cycle in a shop may be rooted more in history than in today's reality.

As an example, some design houses have been using grid-like infrastructures for the past few years for semiconductor design. The goal early on was to increase the utilization of the expensive RISC workstation cycles of that time. Now, the cost of hardware has come down by two orders of magnitude or more. A \$1,000 desktop today is more powerful than a \$150,000 workstation was then. The hardware-acquisition cost today is a small fraction of the total cost of ownership (TCO). Larger components are the cost of the software stack, including applications, both in terms of acquisition cost and the cost of maintenance over the life cycle of the system.

Because it is hard to quantify, another factor seldom incorporated into TCO considerations is the quality of the user experience. Far from being a secondary consideration, however, user experience correlates to worker productivity, which has an impact on the organization's bottom line. In the worst case, a lowest-cost system is still "expensive" if the targeted audience refuses to use it. A dedicated, homogeneous environment makes it easier to run parallel applications. Some of these applications will only run in homogeneous environments; others will run at the speed of the lowest performing node. Faster nodes are left waiting until the stragglers catch up. If the owner of one of the workstations on which the application is running decides to take it off the grid, the entire run may hang.

Applications can be optimized to run in a heterogeneous environment, but optimization takes time and money, thereby increasing the labor component of the TCO, or introducing project delays until the Independent Software Vendor (ISV) incorporates the optimizations. The user community may see this optimization as a hurdle and opt out of grid computing.

Even if a shop starts with a homogeneous environment, the installation will, over time, gravitate toward becoming a heterogeneous system. This is because as the system is upgraded, more advanced nodes will be incorporated. Furthermore, at some point, especially for large companies, additional grids or clusters will be added to the original grid. These additional grids may come from consolidation, mergers and acquisitions, and deployments in different division and geographical regions within the company. These nodes are, of course, different from the originals and, by definition, they make the system heterogeneous.

## **Medium-Term Deployment Strategies: Application Focus**

A medium-term consideration is the harnessing of application parallelism. Parallel applications run over networked nodes may experience performance bottlenecks at the network level. One approach to overcoming these bottlenecks is to re-host the applications in a cluster. A cluster has an interconnect that is faster in terms of bandwidth and latency than Ethernet-based networks.

Also medium-term, applications will need to be optimized to take advantage of multi-level data hierarchies within a node: the CPUs in an SMP node, the cores in a multi-core CPU and multiple levels of cache. AMD Hypertransport\* technology-based nodes have one extra layer of complexity because of their ccNUMA configuration and the difference between near—and far—memory accesses.

For some classes of problems, multiple cores and large caches are beneficial. An example in the HPC space is represented by dense linear-algebra problems where data size grows proportionally to the square of problem size and where the number of operations grows proportionally to the cube of problem size. Dense linear-algebra algorithms are designed to load chunks of data into the CPU's caches and to flush results to memory in a pipelined fashion. Large-cache cores allow a large number of operations between reloads, whereas multiple cores can ensure that these operations are done fast. These capabilities will come for free, in the sense that the CPU fraction of the TCO will likely remain constant or shrink a bit. However, realizing these gains will require hard work and a significant investment from all players in the ecosystem. Ensuring computational balance between the cores in a CPU, the CPUs in a node, the nodes in a multi-level cluster and the clusters in a grid, while maintaining logical consistency across the entire system, is the architectural equivalent of juggling five balls.

For very large data sets, the same dynamic between memory and cache storage also applies between disk storage and memory. In this case, instead of megabyte-sized buffers between cache and memory, memory can be used as a gigabyte-sized cache for terabyte-size data sets.

64-bit addressing can be useful in two ways. First, the larger addressability over 32-bit addressing makes it possible to fit data sets of tens of gigabytes entirely in the physical memory of a cluster. Being able to do so has a significant impact in application design. For some HPC applications, a data set that does not fit in physical memory can be run with an application that has "out of core" capability, which is essentially

an application-optimized virtual memory system. An out-of-core version of an application can be 10 times more expensive to develop than the plain vanilla version. The developers need to be versed in OS architecture, in addition to specific application-domain skills.

There is a significant gain in efficiency in computations done against a very large database when the entire database fits in memory. One such database is the one associated with the human genome project; the human genome consists of 30,000 genes and 3.2 billion base pairs. Roughly assuming the use of one byte to encode a base pair suggests a 3.2-GB dataset, which pushes the limits of 32-bit addresses because the entire 4-GB space is not necessarily available for data addressing.

The second advantage afforded by 64-bit addressing is that, for applications whose number of operations grows faster than data sizes—such as the linear algebra example mentioned previously—the ratio of computation to I/O increases, making the system run more efficiently overall. Storing a database in memory is an example of caching and data replication traded off against the latency and limited bandwidth of accessing a data repository across the globe. Computational genomics problems are especially amenable to this kind of treatment.

One behavioral trait of applications that has not changed in the past 50 years is their locality of reference: given a large address space, a program is likely to reference a minute portion of that address space. This principle applies to both code and data; it is why caches work. For instance, if all the code associated with a loop fits in the cache, potentially the entire code segment can be loaded into cache and, for running this code segment, memory is referenced exactly once when loaded into the cache for the first time—whether the loop executes 1,000 times or a million times. Most applications exhibit this desirable locality behavior.

The portion of the address space referenced by a program over a certain interval is defined as the working set for that interval. It is also interesting to note that this “lumpy” behavior happens at different time scales concurrently, whether the interval is seconds, minutes or even hours. This behavior allows mapping working sets for different timescales to specific elements in the grid architecture. For instance, sub-second working sets are better handled at the cache level, while an application can spend a few minutes between flushing and reloading memory buffers from disk. Transactional workloads typical of enterprise applications also exhibit a well-defined locality of behavior, running a relatively small portion of code updating a few records in a database.

One way of achieving efficiency in a grid environment is to make an application self-adjusting with respect to the application’s working set at each level of abstraction in the system<sup>9</sup> and at some interesting time constant. The reason this is possible is that optimizing for one level of abstraction can be done without undue interaction or interference with layers up and down. For instance, the optimization of cache utilization is embedded in a library routine, or perhaps in the code generated by compiler. These optimizations rarely affect the way I/O buffering is managed. Applications could be written to allow for metadata exchange, where the host can pass information such as the available physical and virtual memory, the number of CPUs per node, cache size and memory performance parameters such as latency and bus bandwidth. The application could then adjust the operating parameters for a particular run, including how arrays are allocated, their sizes and the buffering and I/O strategies for a particular run.

A useful way to look at grids is as a composition of services and business processes to attain specific business goals. Hardware expenditures may not map very well to ROI, because doing such analysis would be difficult without considering the intervening logical layers. For instance, without the intermediate analysis, it would be very difficult to explain why using four-way servers is more desirable than using two-way servers.

## **Long-Term Deployment Strategies: Harnessing Future Technology Transitions**

Contemplating grid deployments from a purely technological perspective, perhaps as a means of speeding up current tasks and processes, is likely to result in missed opportunities. It will be difficult for chief information officers (CIOs) to justify grid deployments purely on the basis of the technological advantages that it confers, although these benefits might be substantial to some stakeholders. The driver for success will be tangible, quantifiable business benefits stemming from grid deployment to critical organization stakeholders. Technical arguments alone do not paint the complete picture.

Because of the emerging nature of grid technology, service organizations with prior experience in grid deployments play a valuable role in the ecosystem, accelerating grid adoption by sharing their experience. These service organizations can come in many forms, including in-house or external expertise. Outsourced expertise can come from pure-play consulting houses or from product-based companies. Each option has pros and cons. A detailed discussion of the subject is outside the scope of this paper.

Success in a deployment breeds additional success. Sharing of prior experience can be a critical success factor. Conversely, organizations venturing out on their own can easily step into blind alleys with their first attempts. A negative initial experience can deter further attempts for months or years. Failures might be unrelated to inherent limitations of grids, but without the proper expertise, it may be difficult to tell. In such a case, potential benefits to the organization are not realized.

### **Industry Viewpoints Conclusion**

It is safe to say that, as an emerging technology, most grid applications or even “killer” applications have not been invented yet. It would be interesting to explore, for instance, whether the now-ubiquitous wireless access point could be enhanced to become part of a mesh-oriented sensor

network, a particular case of an embedded grid. Only a few of these devices would be wired, functioning as gateways into the wired Internet. The rest of the access points would be truly wireless, talking to neighboring access points. The devices could be fitted with environmental sensors that, for example, could act as fire alarms or function as relay stations for VOIP calls. Users with multi-modal communication devices could use VOIP for intra-company calls and use the regular cellular network when no other medium is possible. The system would take care of managing multiple phone numbers, international access codes, credit card access codes, or IP addresses to reach a certain person. Such creative implementations are likely to become more prevalent in the next several years. Grid computing will generate tremendous benefits to the companies that deploy such solutions, as well as to the service providers that support them.

## References and Related Links

### References

<sup>1</sup>SMP: symmetric multiprocessing; NUMA: non-uniform memory access; ccNUMA: cache-coherent NUMA.

<sup>2</sup>An example of a Web interface is entering a genetic sequence to be matched against a database using the BLAST (Basic Local Alignment Search Tool) application\*.  
([www.ncbi.nlm.nih.gov/Education/blasttutorial.html](http://www.ncbi.nlm.nih.gov/Education/blasttutorial.html))

<sup>3</sup>The 451 Group, "Grids 2004: From Rocket Science to Business Service\*," page 17  
([www.the451group.com/special\\_reports/special\\_report\\_detail.php?icid=8](http://www.the451group.com/special_reports/special_report_detail.php?icid=8))

<sup>4</sup>Arik Hesseldahl, "Attack Of The Digital Movie\*," Forbes.com, March 18, 2002. Parallel computation and cycle scavenging are independent of each other; you can have one without the other, or both.  
([www.forbes.com/2002/03/18/0318digitaldistribution.html](http://www.forbes.com/2002/03/18/0318digitaldistribution.html))

<sup>5</sup>Erik Brynjolfsson, "The IT Productivity GAP\*." *Optimize* magazine, Issue 21, July 2003.  
([http://ebusiness.mit.edu/erik/Optimize/pr\\_roi.html](http://ebusiness.mit.edu/erik/Optimize/pr_roi.html))

<sup>6</sup>John Dix, "Focus on processes, not the technology\*." *Network World*, May 24, 2004.  
([www.nwfusion.com/columnists/2004/0524edit.html](http://www.nwfusion.com/columnists/2004/0524edit.html))

<sup>7</sup>SETI@home\* (<http://setiathome.ssl.berkeley.edu/>)

<sup>8</sup>White paper: "Intel® Solution Services maximizes return-on-investment in online gaming arena."

<sup>9</sup>This data appeared in *eWeek\**, August 30, 2004, p. 23. Data from eBay. ([www.eweek.com/](http://www.eweek.com/))

### Related Links

- **Intel® Developer Services High Performance Computing Developer Center** ([www.intel.com/software/dc/hpc/](http://www.intel.com/software/dc/hpc/)) provides technical background and resources for implementing grids and clusters for large-scale computing tasks.
- **Grid Computing Harnesses the Power of Multitudes** ([www.intel.com/cd/ids/developer/asmo-na/eng/segments/enterprise/61106.htm](http://www.intel.com/cd/ids/developer/asmo-na/eng/segments/enterprise/61106.htm)) discusses how specialized hardware creates huge, aggregate virtual computers from dispersed machines.
- **HPC and Intel® Cluster Tools Intel® Developer Forum** (<http://softwareforums.intel.com/ids/board?board.id=HPC>) is a discussion board for discussing technical issues related to High Performance Computing with industry peers and Intel experts.
- **Intel® Developer Services High Performance Computing Developer Center** ([www.intel.com/cd/ids/developer/asmo-na/eng/segments/hpc/index.htm](http://www.intel.com/cd/ids/developer/asmo-na/eng/segments/hpc/index.htm)) provides technical background and resources for implementing grids and clusters for large-scale computing tasks.
- **Multiprocessors, Clusters, Grids and Parallel Computing: What's the Difference?** ([www.intel.com/cd/ids/developer/asmo-na/eng/95581.htm](http://www.intel.com/cd/ids/developer/asmo-na/eng/95581.htm)) Understanding how clusters and grids work—and which processors support them best—is the first step in identifying the many ways they can add raw processing muscle to your infrastructure.
- **Highly Reliable Linux HPC Clusters: Self-Awareness Approach** ([www.intel.com/cd/ids/developer/asmo-na/eng/183307.htm](http://www.intel.com/cd/ids/developer/asmo-na/eng/183307.htm)) discusses detailed solutions for the high-availability and serviceability enhancement of clusters by means of the HA-OSCAR software stack to handle runtime system configuration changes caused by transient failures.
- **HPC and Intel® Cluster Tools Intel® Developer Forum** (<http://softwareforums.intel.com/ids/board?board.id=HPC>) is a discussion board for discussing technical issues related to High Performance Computing with industry peers and Intel experts.
- **Trends in Distributed Computing** ([www.intel.com/cd/ids/developer/asmo-na/eng/95223.htm](http://www.intel.com/cd/ids/developer/asmo-na/eng/95223.htm)) is a white paper that explores the latest trends in distributed computing and provides examples of its uses.
- **Professional Services in High Performance Computing** ([www.intel.com/cd/ids/developer/asmo-na/eng/61399.htm](http://www.intel.com/cd/ids/developer/asmo-na/eng/61399.htm)) shows by example how the High Performance Computing industry is often a proving ground where advanced research and technologies are funded and tried out first before being adopted later in a wider setting.

## About the Authors

**Enrique Castro-Leon** – *Principal Enterprise Architect, Intel® Solution Services, Software and Solutions Group, Intel Corporation*

As Enterprise Architect for Intel Solution Services, Enrique Castro-Leon assists Intel Solution Services' corporate clients in matters of technology assessment, management, diffusion, and adoption, helping clients build technology transition road maps that incorporate business considerations.

Enrique's 25-year career includes 21 years with Intel Corporation spanning OS design and architecture, software engineering, platform definition, and business development, with occasional teaching activities at the Oregon Graduate Institute, Portland State University, and the University of Costa Rica. Enrique also served as a lead architect during the formation of Intel Solution Services.

Enrique has authored more than 30 papers, white papers, and articles on subjects ranging from high-performance computing to Web services.

He holds PhD and MS degrees in Electrical Engineering and Computer Science from Purdue University.

**Joel Munter** – *Program Manager, Technology Office, Intel® Solution Services, Software and Solutions Group, Intel Corporation*

As Program Manager for the Intel Solution Services Technology Office, Joel Munter is working to establish lateral linkages with key representatives from stakeholder organizations critical to the achievement of the division's objectives. Joel is using his extensive Intel-wide network to facilitate an information exchange that will develop and ratify strategic roadmaps for the key technologies, standards, and usage models essential to developing and delivering services and value for product groups across Intel.

Joel has worked in the information technology industry for 22 years including experience in the software product development, software consulting, hotel reservation, and aerospace industries. His 12 years of Intel experience include software development and program management in materials, manufacturing, and corporate services. Supporting Intel's investments in Web services standards, Joel led the Intel effort that led to a successful UDDI specification. Most recently, he led several successful power and energy-related research efforts for Intel® XScale™ microarchitecture within the Corporate Technology Group. Joel's interests include the timely facilitation of information sharing. Getting the right information to all of the necessary people in time to be useful is one of his key goals.

Joel has three patents pending and several more in process. He holds a BS degree in Mechanical and Aerospace Engineering.

---

Experience 64-bit computing on Intel® Architecture.  
Visit [www.intel.com/software/enterprise](http://www.intel.com/software/enterprise).

\*Other names and brands may be claimed as the property of others.

This document and the information described in it are furnished for informational use only and subject to change without notice. No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

THIS DOCUMENT, RELATED MATERIALS AND INFORMATION DESCRIBED HEREIN ARE PROVIDED "AS IS" WITH NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE. INTEL ASSUMES NO RESPONSIBILITY FOR ANY ERRORS CONTAINED IN THIS DOCUMENT AND HAS NO LIABILITIES OR OBLIGATIONS FOR ANY DAMAGES ARISING FROM OR IN CONNECTION WITH THE USE OF THIS DOCUMENT OR THE INFORMATION PROVIDED HEREIN.

Intel may make changes to specifications, product descriptions and features, and plans at any time, without notice.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

