Utility Computing on Global Grids

Chee Shin Yeo, Rajkumar Buyya[1], Marcos Dias de Assunção, Jia Yu, Anthony Sulistio, Srikumar Venugopal, and Martin Placek

Grid Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Melbourne, VIC 3010, Australia
Email: {csyeo, raj, marcosd, jiayu, anthony, srikumar, mplac}@csse.unimelb.edu.au

Keywords: Utility Computing, Grid Computing, Service Level Agreement, Market-based Resource Allocation, On Demand Computing, Middleware, Service-Oriented Architecture, Virtual Organization, Adaptive Enterprise.

1. Introduction

The growing popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we do computing. These technological developments have led to the possibility of using networks of computers as a single, unified computing resource, known as *cluster computing* [1][2]. Clusters appear in various forms: high-performance clusters, high-availability clusters, dedicated clusters, non-dedicated clusters, and so on. In addition, computer scientists in the mid-1990s, inspired by the electrical power grid's pervasiveness and reliability, began exploring the design and development of a new infrastructure, *computational power grids* for sharing computational resources such as clusters distributed across different organisations [3].

In the business world, cluster architecture-based large-scale computing systems, called *data centers*, offering high-performance and high-available hosting services are widely used. The reliable and low-cost availability of data center services has encouraged many businesses to outsource their computing needs; thus heralding a new utility computing model.

Utility computing is envisioned to be the next generation of Information Technology (IT) evolution that depicts how computing needs of users can be fulfilled in the future IT industry [4]. Its analogy is derived

[1] Corresponding author, email – raj@csse.unimelb.edu.au

from the real world where service providers maintain and supply utility services, such as electrical power, gas, and water to consumers. Consumers in turn pay service providers based on their usage. Therefore, the underlying design of utility computing is based on a service provisioning model, where users (consumers) pay providers for using computing power only when they need to.

These developments appears like realization of the vision of Leonard Kleinrock, one of the chief scientists of the original Advanced Research Projects Agency Network (ARPANET) project which seeded the Internet, who said in 1969 [5]: "As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of '*computer utilities*' which, like present electric and telephone utilities, will service individual homes and offices across the country."

Benefits of Utility Computing

The utility computing model offers a number of benefits to both service providers and users. From the provider's perspective, actual hardware and software components are not set up or configured to satisfy a single solution or user, as in the case of traditional computing. Instead, virtualized resources are created and assigned dynamically to various users when needed. Providers can thus reallocate resources easily and quickly to users that have the highest demands. In turn, this efficient usage of resources minimizes operational costs for providers since they are now able to serve a larger community of users without letting unused resources go unutilized. Utility computing also enables providers to achieve a better Return On Investment (ROI) such as Total Cost of Ownership (TCO) since shorter time periods are now required to derive positive returns and incremental profits can be earned with the gradual expansion of infrastructure that grows with user demands.

For users, the most prominent advantage of utility computing is the reduction of IT-related operational costs and complexities. Users no longer need to invest heavily or encounter difficulties in building and maintaining IT infrastructures. Computing expenditures can now be modeled as a variable cost depending on the usage patterns of users, instead of as a static cost of purchasing technologies and employing staff to manage operations. Users neither need to be concerned about possible over- or under-utilization of their own self-managed IT infrastructures during peak or non-peak usage periods, nor worry about being confined to any single vendor's proprietary technologies. With utility computing, users can obtain appropriate amounts of computing power from providers dynamically, based on their specific service needs and requirements. This is particularly useful for users who experience rapidly increasing or unpredictable computing needs. Such an outsourcing model thus provides increased flexibility and ease for users to adapt to their changing business needs and environments [6].

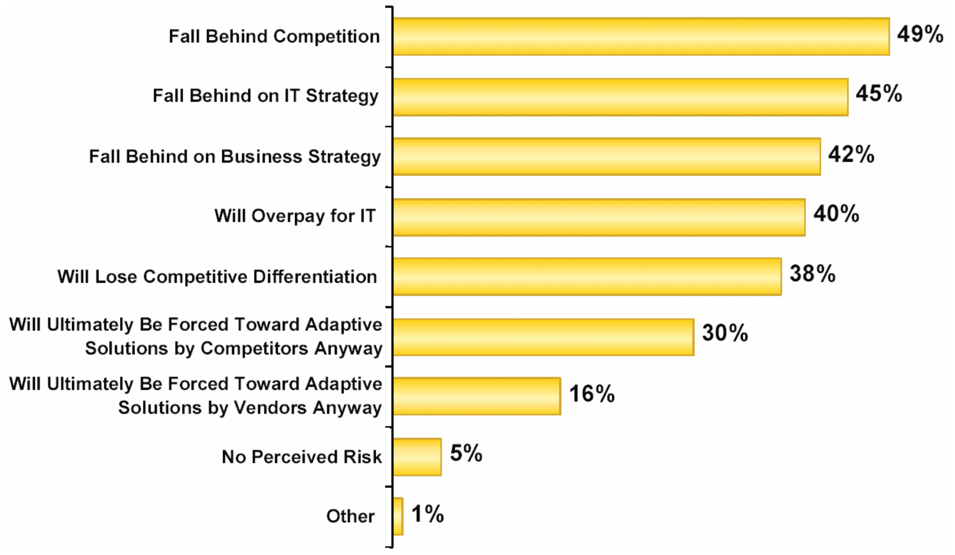

Figure 1. Risks of not becoming an adaptive organization (Source: META Group [7]).

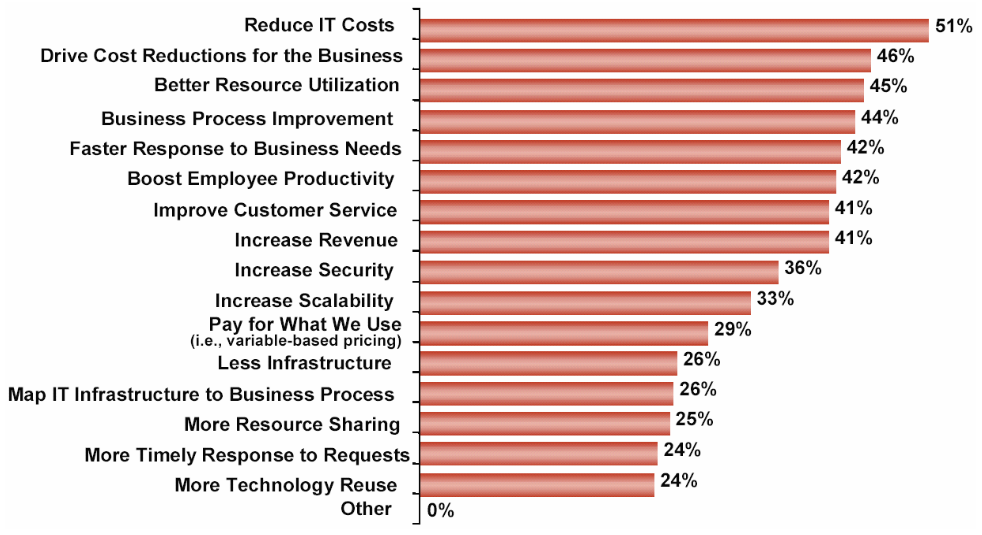

Figure 2. Goals of using adaptive solutions (Source: META Group [7]).

In today's highly competitive and rapidly changing market environment, business organizations aim to be more adaptive in their business and IT strategies, in order to stay ahead of other competitors (see Figure 1). An adaptive organization requires enhanced IT processes and better resource utilization in order to deliver faster response, higher productivity, and lower costs (see Figure 2). Therefore, there seems to be a potential to employ utility computing models for business organizations to be more adaptive and competitive.

1.1. Potential of Grids as Utility Computing Environments

The aim of *Grid computing* is to enable coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [8]. An infinite number of computing devices ranging from high performance systems such as supercomputers and clusters, to specialized systems such as visualization devices, storage systems, and scientific instruments, are logically coupled together in a *Grid* and presented as a single unified resource [9] to the user. Figure 3 shows that a Grid user can easily use these globally distributed Grid resources by interacting with a Grid resource broker. Basically, a Grid user perceives the Grid as a single huge virtual computer that provides immense computing capabilities, identical to an Internet user who views the World Wide Web as a unified source of content.

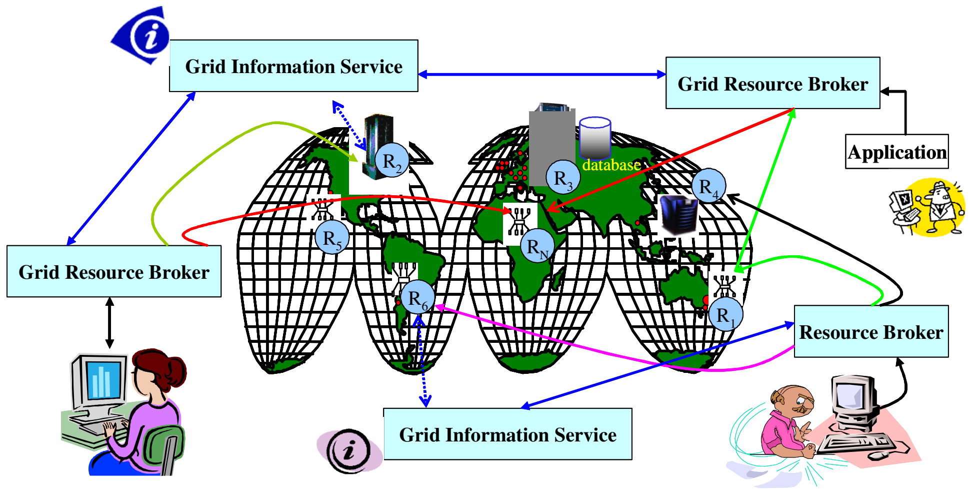

Figure 3. A generic view of a global Grid.

A diverse range of applications are currently or soon to be employed on Grids, some of which include: aircraft engine diagnostics, earthquake engineering, virtual observatory, bioinformatics, drug discovery, digital image analysis, high energy physics, astrophysics, and multi-player gaming [3]. Grids can be primarily classified into the following types, depending on the nature of their emphasis [10] as depicted in Figure 4:

- *Computational Grid*: Aggregates the computational power of globally distributed computers (e.g. TeraGrid [11], ChinaGrid [12], and APACGrid [13]).
- *Data Grid*: Emphasizes on a global-scale management of data to provide data access, integration, and processing through distributed data repositories (e.g. LHCGrid [14] and GriPhyN [15]).
- *Application Service Provisioning (ASP) Grid*: Focuses on providing access to remote applications, modules, and libraries hosted on data centers or Computational Grids (e.g. NetSolve/GridSolve [16]).
- *Interaction Grid*: Focuses on interaction and collaborative visualization between participants (e.g. AccessGrid [17]).
- *Knowledge Grid*: Aims towards knowledge acquisition, processing, management, and provide business analytics services driven by integrated data mining services (e.g., Italian KnowledgeGrid [18] and EU DataMiningGrid [19]).
- *Utility Grid*: Focuses on providing all the Grid services including compute power, data, and services to end-users as IT utilities on a subscription basis and the infrastructure necessary for negotiation of required Quality of Service (QoS), establishment and management of contracts, and allocation of resources to meet competing demands from multiple users and applications (e.g. Gridbus [20] and Utility Data Center [21]).

These various types of Grids follow a layered design, with the Computational Grid as the bottom-most layer and the Utility Grid as the top-most layer. A Grid on a higher layer utilizes the services of Grids that operate at lower layers in the design. For example, a Data Grid utilizes the services of Computational Grid for data processing and hence builds on it. In addition, lower-layer Grids focus heavily on infrastructural aspects, whereas higher-layer ones focus on users and QoS delivery. Accordingly, Grids are proposed as the emerging cyber infrastructure to power utility computing applications.

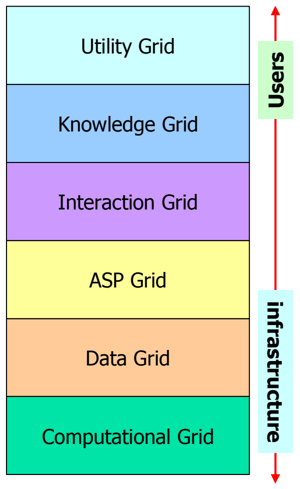

Figure 4: Types of Grids and their focus.

Grids offer a number of benefits such as:

- Transparent and instantaneous access to geographically distributed and heterogeneous resources.
- Improved productivity with reduced processing time.
- Provisioning of extra resources to solve problems that were previously unsolvable due to the lack of resources.
- A more resilient infrastructure with on-demand aggregation of resources at multiple sites to meet unforeseen resource demand.

- Seamless computing power achieved by exploiting under-utilized or unused resources that are otherwise wasted.
- Maximum utilization of computing facilities to justify IT capital investments.
- Coordinated resource sharing and problem solving through virtual organizations [8] that facilitates collaboration across physically dispersed departments and organizations.
- Service Level Agreement (SLA) based resource allocation to meet QoS requirements.
- Reduced administration effort with integration of resources as compared to managing multiple standalone systems.

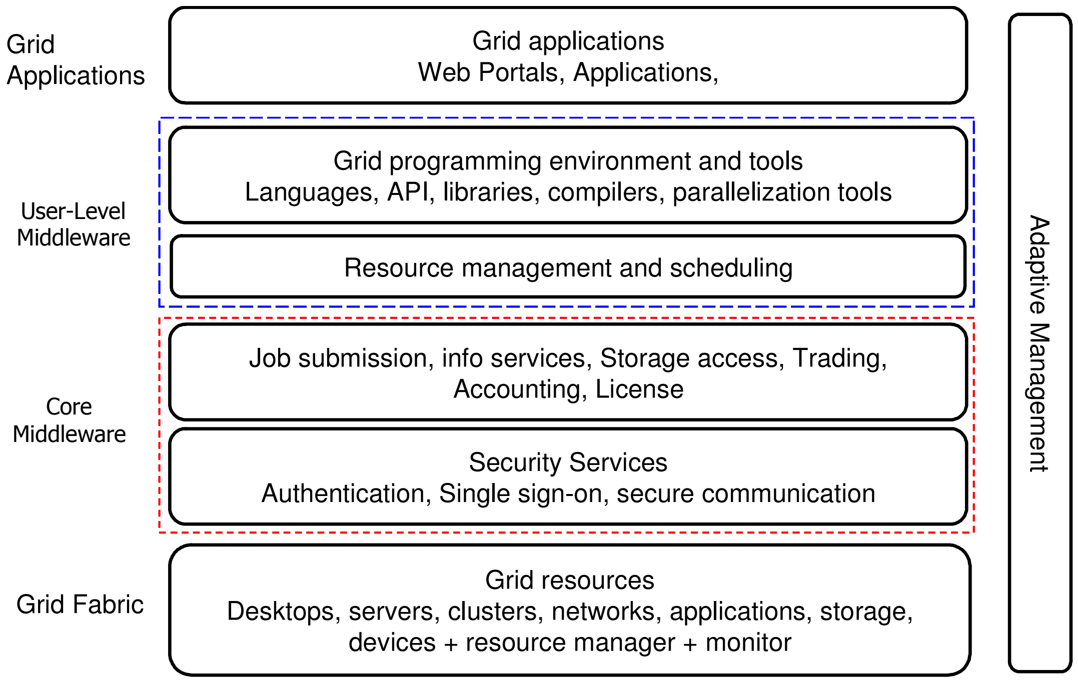
Grid Applications
Grid applications
Web Portals, Applications,
User-Level Middleware
Grid programming environment and tools
Languages, API, libraries, compilers, parallelization tools
Resource management and scheduling
Core Middleware
Job submission, info services, Storage access, Trading, Accounting, License
Security Services
Authentication, Single sign-on, secure communication
Grid Fabric
Grid resources
Desktops, servers, clusters, networks, applications, storage, devices + resource manager + monitor
Adaptive Management

Figure 5. A layered Grid architecture.

Layered Grid Architecture

The components that are necessary to form a Grid are shown in Figure 5. The layered Grid architecture organizes various grid capabilities and components such that high level services are built using lower-level services. Grid economy [22] is essential for achieving adaptive management and utility-based resource allocation and thus influences various layers of the architecture:

- *Grid fabric software layer*: Provides resource management and execution environment at local Grid resources. These local Grid resources can be computers (e.g. desktops, servers, or clusters) running a variety of operating systems (e.g. UNIX or Windows), storage devices, and special devices such as a radio telescope or heat sensor. As these resources are administered by different local resource managers and monitoring mechanisms, there needs to be Grid middleware that can interact with them.
- *Core Grid middleware layer*: Provides Grid infrastructure and essential services which consists of information services, storage access, trading, accounting, payment, and security. As a Grid environment is highly dynamic where the location and availability of services are constantly changing, information services provide the means for registering and obtaining information about Grid resources, services, and status. Resource trading based on the computational economy approach is suitable given the complex and decentralized manner of Grids. This approach provides incentives for both resource providers and users to be part of the Grid community, and allows them to develop strategies to maximize their objectives. Security services are also critical to address the confidentiality, integrity, authentication, and accountability issues for accessing resources across diverse systems that are autonomously administered.

- *User-level middleware layer*: Provides programming frameworks and policies for various types of applications, and resource brokers to select appropriate and specific resources for different applications. The Grid programming environment and tools should support common programming languages (e.g. C, C++, Fortran, and Java), a variety of programming paradigms (e.g. message passing [23] and Distributed Shared Memory (DSM) [24]), and a suite of numerical and commonly used libraries. Resource management and scheduling should be transparent to the users such that processor time, memory, network, storage, and other resources in Grids can be utilized and managed effectively and efficiently using middleware such as resource brokers.
- *Grid applications Layer*: Enables end-users to utilize Grid services. Grid applications thus need to focus on usability issues so that end-users can find them intuitive and easy to use. They should also be able to function on a variety of platforms and operating systems so that users can easily access them. Therefore, an increasingly number of web portals are being built since they allow users to ubiquitously access any resource from anywhere over any platform at any time.

The design aims and benefits of Grids are analogous to those of utility computing, thus highlighting the potential and suitability of Grids to be used as utility computing environments. The current trend of implementing Grids based on open standard service-based architectures to improve interoperability is a step towards supporting utility computing [25]. Even though most existing Grid applications are scientific research and collaboration projects, the number of Grid applications in business and industry-related projects is also gradually increasing. It is thus envisioned that the realization of utility computing through Grids will follow a similar course as the World Wide Web, which was first initiated as a scientific project but was later widely adopted by businesses and industries.

1.2. Challenges of Realizing Utility Computing Models

There are several challenges that need to be addressed in order to realize utility computing. One challenge is that both providers and users need to redraft and reorganize their current IT-related procedures and operations to include utility computing [9]. New IT policies need to be negotiated and agreed upon between providers and users, compared to the previous situation where providers and users owned and controlled their standalone policies. Providers must also understand specific service needs and requirements of users in order to design suitable policies for them. Open standards need to be established to facilitate successful adoption of utility computing so that users and producers experience fewer difficulties and complexities in integrating technologies and working together, thus reducing associated costs. Table 1 lists some of the major computing standards organizations and the activities they are engaged in.

With the changing demand of service needs from users, providers must be able to fulfill the dynamic fluctuation of peak and non-peak service demands. Service contracts known as SLAs are used by providers to assure users of their level of service quality. If the expected level of service quality is not met, providers will then be liable for compensation and may incur heavy losses. Therefore, providers seek to maximize customer satisfaction by meeting service needs and minimize the risk of SLA violations [27]. Improved service-oriented policies and autonomic controls [28][29] are essential for achieving this.

Other than managing the technological aspects of delivering computing services, providers also need to consider the financial aspects of service delivery. Financial risk management for utility computing [30] is comprised of two factors: delivery risk and pricing risk. Delivery risk factors examine the risks concerned with each possible scenario in which a service can be delivered. Pricing risk factors study the risks involved with pricing the service with respect to the availability of resources. Given shorter contract durations, lower switching costs, and uncertain customer demands in utility computing environments, it is important to have dynamic and flexible pricing schemes to potentially maximize profits and minimize losses for providers [31].

There are also potential non-technical obstacles to successful adoption of utility computing such as cultural and people-related issues that will require organizations to change their current stance and perceptions [32]. The most worrying issues being perceived are loss of control or access to resources, risks associated with enterprise-wide deployment, loss or reduction of budget dollars, and reduced priority of projects. Thus, overcoming these non-technical obstacles is extremely critical and requires the dissemination of correct information to all levels of management within organizations to prevent the formation of misperceptions.

Table 1. Some major standards organizations (Source: J. Joseph, et al. [26]).

Organization	Website	Standards Activities
Open Grid Forum (OGF)	http://www.ogf.org	Grid computing, distributed computing, and peer-to-peer networking.
World Wide Web Consortium (W3C)	http://www.w3c.org	World Wide Web (WWW), Extensible Markup Language (XML), web services, semantic web, mobile web, and voice browser.
Organization for the Advancement of Structured Information Standards (OASIS)	http://www.oasis-open.org	Electronic commerce, systems management and web services extensions, Business Process Execution Language for Web Services (BPEL4WS), and portals.
Web Services Interoperability Organization (WS-I)	http://www.ws-i.org	Interoperable solutions, profiles, best practices, and verification tools.
Distributed Management Task Force (DMTF)	http://www.dmtf.org	Systems management.
Internet Engineering Task Force (IETF)	http://www.ietf.org	Network standards.
European Computer Manufacturers Organization (ECMA)	http://www.ecma-international.org	Language standards (C++, C#).
International Organization for Standardization (ISO)	http://www.iso.org	Language standards (C++, C#).
Object Management Group (OMG)	http://www.omg.org	Model-Driven Architecture (MDA), Unified Modeling Language (UML), Common Object Resource Broker Architecture (CORBA), and real-time system modeling.
Java Community Process (JCP)	http://www.jcp.org	Java standards.

2. Utility Grids

This chapter focuses on the use of Grid technologies to achieve utility computing. An overview of how Grids can support utility computing is first presented through the architecture of Utility Grids. Then, utility-based resource allocation is described in detail at each level of the architecture. Finally, some industrial solutions for utility computing are discussed.

A reference service-oriented architecture for Utility Grids is shown in Figure 6. The key players in a Utility Grid are the Grid user, Grid resource broker, Grid middleware services, and Grid Service Providers (GSPs). The Grid user wants to make use of Utility Grids to complete their applications. Refactoring existing applications is thus essential to ensure that these applications are Grid-enabled to run on Utility Grids [33]. The Grid user also needs to express the service requirements to be fulfilled by GSPs. Varying QoS parameters, such as deadline for the application to be completed and budget to be paid upon completion, are defined by different Grid users, thus resulting in dynamic fluctuation of peak and non-peak service demands. The Grid resource broker then discovers appropriate Grid middleware services based on these service demand patterns and QoS requirements, and dynamically schedule applications on them at runtime, depending on their availability, capability, and costs. A GSP needs tools and mechanisms that support pricing specifications and schemes so they can attract users and improve resource utilization. They also require protocols that support service publication and negotiation, accounting, and payment.

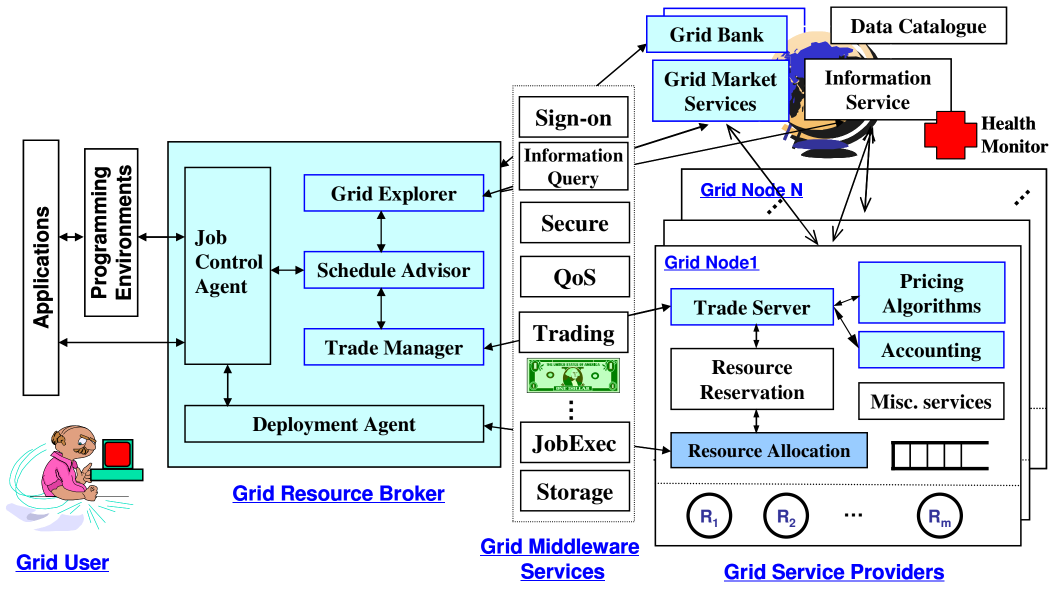

Figure 6. A reference service-oriented architecture for Utility Grids.

The Grid resource broker comprises the following components:

- *Job control agent*: Ensures persistency of jobs by coordinating with schedule advisor for schedule generation, handling actual creation of jobs, maintaining job status, and interacting with users, schedule advisor, and deployment agent.
- *Grid explorer*: Interacts with Grid information service to discover and identify resources and their current status.
- *Schedule advisor*: Discovers Grid resources using the Grid explorer, and select suitable Grid resources and assign jobs to them (schedule generation) to meet users' requirements.
- *Trade manager*: Accesses market directory services for service negotiation and trading with GSPs based on resource selection algorithm of schedule advisor.
- *Deployment agent*: Activates task execution on the selected resource according to schedule advisor's instruction and periodically updates the status of task execution to job control agent.

Traditional core Grid middleware focuses on providing infrastructure services for secure and uniform access to distributed resources. Supported features include security, single sign-on, remote process management, storage access, data management, and information services. An example of such middleware is the Globus toolkit [34] which is a widely adopted Grid technology in the Grid community. Utility Grids require additional service-driven Grid middleware infrastructure that includes:

- *Grid market directory*: Allows GSPs to publish their services so as to inform and attract users.
- *Trade server*: Negotiates with Grid resource broker based on pricing algorithms set by the GSP and sells access to resources by recording resource usage details and billing the users based on the agreed pricing policy.
- *Pricing algorithms*: Specifies prices to be charged to users based on the GSP's objectives, such as maximizing profit or resource utilization at varying time and for different users.
- *Accounting and charging*: Records resource usage and bills the users based on the agreed terms negotiated between Grid resource broker and trade server.

Figure 7 shows how services are assembled on demand in a Utility Grid. The application code is the legacy application to be run on the Utility Grid. Users first compose their application as a distributed application such as parameter sweep using visual application composer tools (Step 1). The parameter sweep model creates multiple independent jobs, each with a different parameter. This model is well suited for Grid computing environments wherein challenges such as load volatility, high network latencies, and high probability of individual node failures make it difficult to adopt a programming approach which favors tightly coupled systems. Accordingly, a parameter sweep application has been termed as a "killer application" for the Grid [35].

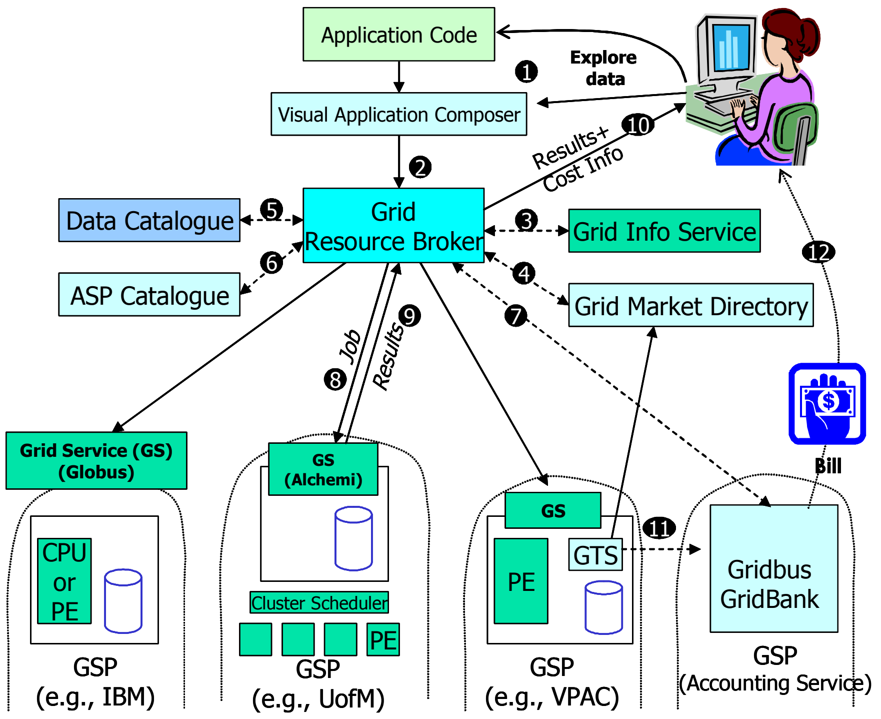

Figure 7. On demand assembly of services in a Utility Grid.

Visual tools allow rapid composition of applications for Grids by hiding the associated complexity from the user. The user's analysis and QoS requirements are submitted to the Grid resource broker (Step 2). The Grid resource broker first discovers suitable Grid services based on user-defined characteristics, including price, through the Grid information service and the Grid market directory (Steps 3 and 4). The broker then identifies the list of data sources or replicas through a data catalogue and selects the optimal ones (Step 5). The broker also identifies the list of GSPs that provides the required application services using the Application Service Provider (ASP) catalogue (Step 6). The broker checks that the user has the necessary credit or authorized share to utilize the requested Grid services (Step 7). The broker scheduler assigns and deploys jobs to Grid services that meet user QoS requirements (Step 8). The broker agent on the Grid resource at the GSP then executes the job and returns the results (Step 9). The broker consolidates the results before passing them back to the user (Step 10). The metering system charges the user by passing the resource usage information to the accounting service (Step 11). The accounting service reports remaining resource share allocation and credit available to the user (Step 12).

Layered Grid Architecture Realization

To enable Utility Grids, the Gridbus project has offered open-source Grid middleware [36] for various layers (as highlighted in Figure 8) that include:

- *Grid fabric software layer*: Libra, a utility-driven cluster scheduler that considers and enforces SLAs for jobs submitted into the cluster.
- *Core Grid middleware layer*: Alchemi which is a .NET-based desktop Grid framework, Grid Market Directory which is a directory publishing available Grid services, Grid Bank which provides accounting, authentication and payment facilities, and GridSim which is a event-driven simulator that models Grid environments.
- *User-level middleware layer*: Gridbus broker that selects suitable Grid services and schedules applications to run on them, Grid workflow engine that provides workflow execution and

monitoring on Grids, and Visual Parametric Modeler that provides a graphical environment to parameterize applications.

- *Grid application layer*: Web portals such as Gridscape that provide interactive and dynamic web-based Grid monitoring portals and G-Monitor that manages execution of applications on Grids using brokers.

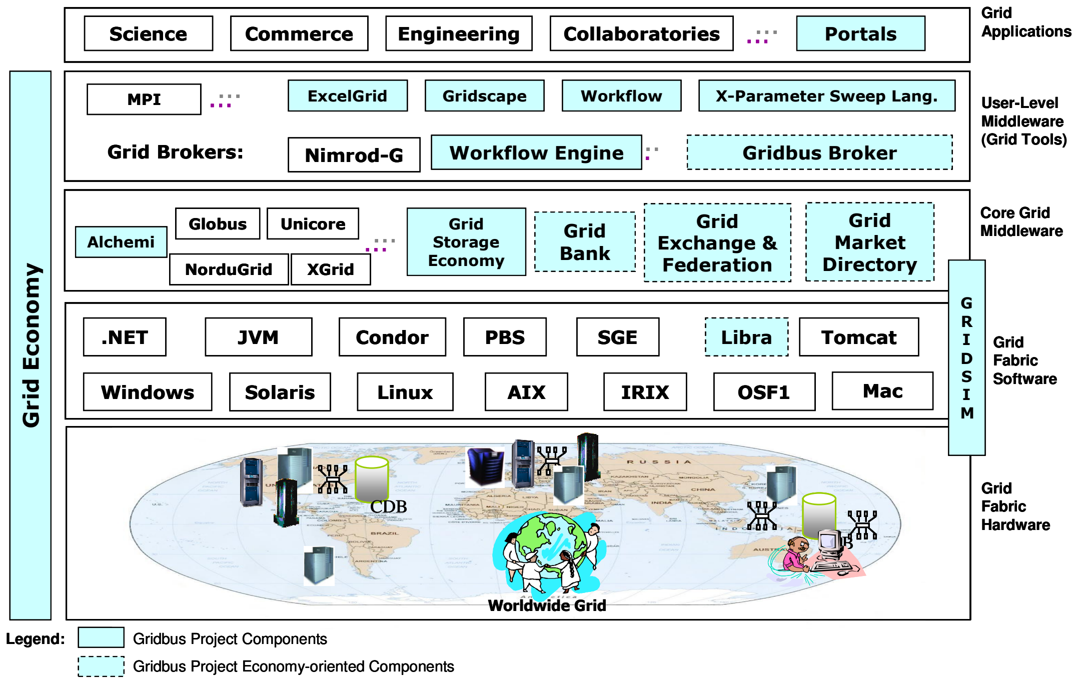

Figure 8. Realizing Utility Grids: Gridbus and complementary technologies.

3. Utility-based Resource Allocation at Various Levels

Utility-based resource allocation is essential at various levels of the Utility Grid in order to realize the utility computing model. This section examines the challenges involved in clusters, distributed storage, Computational Grid brokering, Data Grids, workflow scheduling, advanced reservation, and cooperative virtual organizations.

3.1. Clusters

A *cluster* is a type of parallel or distributed computer system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource [1][2]. Clustering these stand-alone computers together has resulted in high-performance, high-availability, and high-throughput processing on a network of computers at a much lower cost than traditional supercomputing systems, thus resulting in *cluster computing* being a more viable choice as a supercomputing solution. As clusters are extensively utilized in Data Centers that promise to provide managed computing, storage, and application services at low cost, the cost of ownership and maintenance need to be reduced. But, clusters are heavily focused on increasing peak performance using tens of thousands of power hungry components, which is leading to intolerable operating cost and failure rates [37]. Thus recently, power-aware clusters have been built to reduce power consumption by leveraging DVS (Dynamic Voltage Scaling) techniques and employing distributed performance-directed DVS scheduling strategies. Such clusters are able to gain similar performance, yet reduce the amount of power consumption.

Currently, service-oriented Grid technologies are employed to enable utility computing environments where the majority of Grid resources are clusters. Grid schedulers such as brokers and workflow engines can then discover suitable Grid resources and submit jobs to them on the behalf of the users. If the chosen Grid resource is a cluster, these Grid schedulers then interact with the cluster Resource Management System (RMS) to monitor the completion of submitted jobs. The cluster RMS is a middleware which provides a uniform interface to manage resources, queue jobs, schedule jobs and execute jobs on multiple compute nodes within the cluster. With a utility model, users can specify different levels of service needed to complete their jobs. Thus, providers and users have to negotiate and agree on SLAs that serve as contracts outlining the expected level of service performance. Providers can then be liable to compensate users for any service under-performance. So, the cluster RMS must be able to enable utility-driven cluster computing by supporting SLA based resource allocation that can meet competing user demands and enforce their service needs.

Existing cluster RMSs need to be enhanced or extended to adopt utility-driven resource allocation rather than the current system-centric resource allocation that maximizes resource throughput and utilization of the cluster. System-centric approaches assume that all job requests are equally important and thus neglect the actual levels of service required by different users. The cluster RMS should have the following components:

- *Request examiner*: Interprets the QoS parameters defined in the SLA.
- *Admission control*: Determines whether a new job request should be accepted or not. This ensures that the cluster is not overloaded with too many requests such that service performance deteriorates.
- *SLA based scheduler*: New service-oriented policies need to be incorporated to allocate resources efficiently based on service requirements.
- *Job monitor*: New measurement metrics need to be used to keep track of whether the execution progress of jobs meets the service criteria.
- *Accounting*: Maintains the actual usage of resources so that usage charges can be computed. Usage information can also be used to make better resource allocation decisions.
- *Pricing*: Formulate charges for meeting service requests. For example, requests can be charged based on submission time (peak/off-peak), pricing rates (fixed/variant), or availability of resources (supply/demand).
- *Job control*: Enforces resource assignment for executing requests to fulfill specified service needs.

The key design factors and issues for a utility-driven cluster RMS can be addressed from five perspectives [38]:

- *Market model*: Considers market concepts present in real-world human economies are can be applied for service-oriented resource allocation in clusters to deliver utility.
- *Resource model*: Addresses architectural framework and operating environments of clusters that need to be conformed.
- *Job model*: Examines attributes of jobs to ensure that various job types with distinct requirements can be fulfilled successfully.
- *Resource allocation model*: Analyzes factors that can influence the resource assignment outcome.
- *Evaluation model*: Assesses the effectiveness and efficiency of the cluster RMS in satisfying the utility model.

There is therefore growing research interest in formulating effective service-oriented resource allocation policies to satisfy the utility model. Computation-at-Risk (CaR) [39] determines the risk of completing jobs later than expected based on either the makespan (response time) or the expansion factor (slowdown) of all jobs in the cluster. Cluster-On-Demand [40] reveals the significance of balancing the reward against the risk of accepting and executing jobs, particularly in the case of unbounded penalty. QoPS [41] incorporates an admission control to guarantee the deadline of every accepted job by accepting a new job only if its deadline can be guaranteed without violating the deadlines of already accepted jobs. LibraSLA [42] accepts jobs with hard deadlines only if they can be met, and accepts jobs with soft deadlines depending on their penalties incurred for delays. Another work [43] addresses the difficulties encountered by service providers when they rent resources from resource providers to run jobs accepted from clients. These include difficulties such as which jobs to accept, when to run them, and which resources to rent to run them. It analyzes the likely impact on these difficulties in several scenarios, including changing workload, user impatience, resource availability, resource pricing, and resource uncertainty.

3.2. Distributed Storage

Storage plays a fundamental role in computing as it is present in many key components, from registers and Random Access Memory (RAM) to hard disk drives and optical disk drives. Combining storage with networking has created a platform for Distributed Storage System (DSS). The wide proliferation of the Internet has created a global network – a platform with innovative possibilities such as offering access to storage as a utility. DSSs functioning in a global environment support sharing of storage across geographic, institutional, and administrative boundaries. The network speed (bandwidth and latency) and usage load impact the speed of remote storage access, however, they can be overcome through smart access management techniques such as demand prediction, preloading, and caching. The key benefits of such DSS are, it (1) enables aggregation of storage resources from different sources to create mass storage systems at lower cost, (2) enhances reliability of storage through replication, (3) supports disaster management and recovery due to replication of content across multiple sites, (4) provides ability to offer access to storage as utility services, which in turn reduces the cost of ownership and management of storage systems for end users, and (5) allows organisations to barter or harness each other's storage systems for transparent sharing and preservation of digital assets such as e-books, e-music, e-journals, and scientific experimental data.

Some prominent examples of DSS are Parallel Virtual File System (PVFS) [44], General Parallel File System (GPFS) [45], and Google File System (GFS) [46]. A key challenge in DSS is ensuring that storage services are shared fairly among users and providers are offered incentive for making storage services available. One way to achieve this is applying economic principles. Examples of DSS which apply economic principles to manage various aspects operational behavior include: Mungi which manages storage quota, Mojo Nation which instills cooperative behavior, Stanford Archival Repository (SAR) which encourages the sharing of storage services and exchanges, and OceanStore which provides utility storage. SAR [47] discusses the Stanford Archival Repository, a bartering storage system for preserving information. Institutions that have common requirements and storage infrastructure can use the framework to barter with each other for storage services. For example, libraries may use this framework to replicate their archives among each other for the purpose of preservation. OceanStore [48] is a globally scalable storage utility, providing paying users with a durable, highly available storage service by utilizing non-trusted infrastructure. Mungi [49] is a Single-Address-Space Operating System (SASOS), which employs economic principles to manage storage quota. Mojo Nation [50] uses digital currency, *Mojo*, to encourage users to share resources on its network. Users that contribute services are rewarded with *Mojo*, which can then be traded for services. Storage Exchange [51] applies a Double Auction (DA) market model allowing storage services to be traded in a global environment.

Treating storage as a tradable commodity provides incentives for users to participate in the federating and sharing of their distributed storage services. However, there are still some challenges that need to be overcome before storage utility can be fully realized:

- *Federate*: A plethora of heterogeneous storage systems exist, creating a homogenous interface is a key step in federating storage.
- *Share*: Ensuring that distributed storage resource is shared fairly among users and that no single user can deny access to others, accidentally or otherwise.
- *Security*: Operating on non-trusted infrastructure requires the use of cryptographic mechanisms in order to enforce authentication and prevent malicious behavior.
- *Reliability*: Storage medium and network failures are common in a global storage infrastructure, and therefore mechanisms of remote replicas and erasure codes need to be employed to ensure that persistent reliable access to stored data is achieved.

Whilst the challenges are many, the weight of incentives and future possibilities from realizing a globally distributed storage utility ensure the continuing research and development in this area:

- *Monetary gain*: Institutions providing storage services (providers) are able to better utilize existing storage infrastructure in exchange for monetary gain. Institutions consuming these storage services (consumers) have the ability to negotiate for storage services as they require them, without needing to incur the costs associated with purchasing and maintaining storage hardware.
- *Common objectives*: There may be institutions that wish to exchange storage services between themselves due to the presence of a mutual goal such as preservation of information [47].

- *Spikes in storage requirements*: Research institutions may require temporary access to mass storage [52] such as needing access to additional storage to temporarily store data generated from experiments that are carried out infrequently. In exchange, institutions may provide access to their storage services for use by others when they are unused.
- *Donate*: Institutions may wish to donate storage services, particularly if these services are going to a noble cause.
- *Autonomic storage*: Development of a framework to support future autonomic storage systems that will allow agents to broker storage on an as needed basis.

3.3. Computational Grid Brokering

A *Computational Grid broker* acts as an agent for the user in Computational Grids, by performing various tasks such as resource discovery, job scheduling, and job monitoring on the behalf of the user. To determine which resources to select, the broker needs to take into account various attributes from both the user perspective such as resource requirements of the application and the resource perspective such as resource architecture and configuration, resource status (available memory, disk storage, and processing power), resource availability, network bandwidth, resource workload, and historical performance.

However, the Computational Grid broker needs to consider additional service-driven attributes such as QoS requirements specified by users in order to support the utility model. For example, users may specify a deadline for the completion of their application. The user may also state the maximum price to be paid for the completion. With the user's request of deadline and price for an application, the broker tries to locate the most suitable resources. Thus, during resource discovery, the broker also needs to know the costs of resources that are set by GSPs which can be obtained from a Grid market directory service. The broker must be able to negotiate with GSPs to establish an agreed price for the user, before selecting the most suitable resources with the best price based on the provided QoS. For instance, a more relaxed deadline should be able to obtain cheaper access to resources and vice-versa. So, different users often have varying prices based on their specific needs. To achieve this, we need new scheduling algorithms that take into consideration the application processing requirements, Grid resource dynamics, users' QoS requirements such as the deadline and budget, and their optimization preferences.

The Nimrod-G resource broker [53] and Gridbus Grid service broker [54] are examples of a service-oriented Computational Grid brokers for parameter sweep applications. Both brokers schedule jobs based on economic principles (through the budget that the user is willing to pay) and a user-defined QoS requirement (the deadline within which the user requires the application to be completed). Nimrod-G implements four adaptive algorithms for scheduling compute-intensive parameter sweep applications:

- *Cost Optimization*: Execution time is within the specified deadline and execution cost is the cheapest.
- *Time Optimization*: Execution time is the shortest and execution cost is within the specified budget.
- *Cost-Time Optimization*: Similar to cost optimization, but if there are multiple resources with the same cost, it applies time optimization so that execution time is the shortest given the same cost.
- *Conservative Time Optimization*: Similar to cost-time optimization, but ensures that each unprocessed job in the parameter sweep application has a minimum budget-per-job.

The Gridbus broker extends cost and time optimization to schedule distributed data-intensive applications that require access to and processing of large datasets stored in distributed repositories.

3.4. Data Grids

Data is one of the most important entities within any IT infrastructure. Therefore, any utility computing platform must be able to provide secure, reliable and efficient management of enterprise data, and must be able to abstract the mechanisms involved. One of the key factors in the adoption of Grids as a utility computing platform is the creation of an infrastructure for storing, processing, cataloguing and sharing the ever-expanding volumes of data that are being produced by large enterprises such as scientific and commercial collaborations. This infrastructure, commonly known as *Data Grids*, provides services that allow users to discover, transfer and maintain large repositories of data. At the very minimum, a Data Grid provides a high-performance and reliable data transfer mechanism and a data replica management infrastructure. Data manipulation operations in a Data Grid are mediated through a security layer that, in addition to the facilities provided by the general Grid security services, also provides specific operations such as managing access permissions and encrypted data transfers.

In recent years, the Grid community has adopted the Open Grid Service Architecture (OGSA) [55] which leverages web service technologies such as XML and SOAP to create Grid services that follow standard platform-independent mechanisms for representation, invocation and data exchange. A subset of OGSA deals with providing basic interfaces called data services that describe data and the mechanisms to access it. Data services virtualize the same data by providing multiple views that are differentiated by attributes and operations. They also enable different data sources such as legacy databases, data repositories and even spreadsheets to be treated in the same manner via standard mechanisms [56].

Virtualization of data creates many possibilities for its consumption, enabled by the coupling of Grid services to enable applications such as data-intensive workflows. Already, projects such as Virtual Data Grid (VDG) [57] represent data not in terms of physical storage but as results of computational procedures. This has at least two possibilities for users: the ability to determine the provenance of data, i.e. determines how it was produced and if it were valid, and the ability to reuse the data products in future experiments where the same procedures with the same inputs are involved. Pegasus [58] is a workflow management system from the GriPhyN project which uses the VDG to reduce workflows by substituting previously generated data wherever possible. A similar procedure is followed by Storage Resource Broker (SRB) [59] which uses stored procedures in its SRB Matrix to reduce dataflow graphs. Therefore, data virtualization isolates the users from the physical Data Grid environment.

Data virtualization is an important technology for creating a information-rich utility computing environment that is able to provide its users with the following abilities:

- Seamlessly discover and use available data services
- Plan ahead for future requirements and take preemptive action
- Create dynamic applications by combining services.

In such an environment, QoS parameters associated with a data service play an important role in data service discovery. Such parameters include the size of the data, permissions associated with access and modification, available bandwidth to storage locations, and relevance. Relevance of data can be determined from the provenance data that describes the procedures used for producing the data. Planning, scheduling, and reserving resources in advance is conducted by resource brokers that take QoS parameters into account while selecting data sources and storage locations.

3.5. Workflow Scheduling

With the advent of Grid and application technologies, scientists and engineers are building more and more complex applications to manage and process large data sets, and execute scientific experiments on distributed resources. Such application scenarios require means for composing and executing complex workflows. *Workflows* are concerned with the automation of procedures whereby files and data are passed between participants according to a defined set of rules to achieve an overall goal [60]. A workflow management system defines, manages and executes workflows automatically on computing resources. Imposing the workflow paradigm for application composition on Grids offers several advantages [61] such as:

- Ability to build dynamic applications which orchestrate distributed resources.
- Utilization of resources that are located in a particular domain to increase throughput or reduce execution costs.
- Execution spanning multiple administrative domains to obtain specific processing capabilities.
- Integration of multiple teams involved in managing different parts of the experiment workflow, thus promoting inter-organizational collaborations.

QoS support in workflow management is required by many workflow applications. For example, a workflow application for maxillo-facial surgery planning [62] needs results to be delivered before a certain time. However, QoS requirements cannot be guaranteed in a conventional Grid where resources provide only best effort services. Therefore, there is a need to have Utility Grids that allow users to negotiate with service providers on a certain service agreement with the requested QoS.

In general, scheduling workflows on Utility Grids is guided by users' QoS expectations. Workflow management systems are required to allow the user to specify their requirements, along with the descriptions of tasks and their dependencies using the workflow specification. In general, QoS constraints express the preferences of users and are essential for efficient resource allocation. QoS constraints can be classified into five dimensions [63]: *time*, *cost*, *fidelity*, *reliability* and *security*. Time is a basic measure of performance. For workflow systems, it refers to the total time required for completing the execution of a workflow. Cost represents the cost associated with the execution of workflows, including the cost of managing workflow systems and usage charge of Grid resources for processing workflow tasks. Fidelity refers to the measurement related to the quality of the output of workflow execution. Reliability is related to the number of failures for execution of workflows. Security refers to confidentiality of the execution of workflow tasks and trustworthiness of resources.

Several new issues arise from scheduling QoS constrained workflows on Utility Grids:

- In general, users would like to specify a QoS constraint for the entire workflow. It is required that the scheduler determines a QoS constraint for each task in the workflow, such that the global QoS is satisfied.
- The scheduler is required to be adaptive to evaluate a proposed SLA and negotiate with a service provider for one task with respect to its current accepted set of SLAs and expected return of unscheduled tasks.
- The description language and monitoring mechanism for QoS based workflows will be more complex as compared to traditional workflow scheduling.

To date, several efforts have been made towards QoS aware workflow management. Web Services Agreement (WS-Agreement) [64] allows a resource provider and a consumer to create an agreement on the expected service qualities between them. Grid Quality of Service Management (G-QoSm) [65] provides Grid services which workflow schedulers can negotiate and reserve services based on certain quality levels. The Vienna Grid Environment (VGE) [66] develops a dynamic negotiation model that facilitates workflow schedulers to negotiate various QoS constraints with multiple service providers. It also extends the Business Process Execution Language (BPEL) [67] to support QoS constraint expression. A QoS based heuristic for scheduling workflow applications can be found in [68]. The heuristic attempts to assign the task into least expensive computing resources based on assigned time constraint for the local task. A cost based scheduling algorithm [69] minimizes the cost, while meeting the deadline of the workflow by distributing the deadline for the entire workflow into sub-deadlines for each task. More recently, a budget constrained workflow scheduling has been developed in [70]. It uses genetic algorithms to optimize workflow execution time while meeting the user's budget.

However, supporting QoS in scheduling of workflow applications is still at a very preliminary stage. There is a need for many advanced capabilities in workflow systems and resource allocation such as support of cyclic and conditional checking of workflow structure, adaptive scheduling based on dynamic negotiation models, and advanced SLA monitoring and renegotiation.

3.6. Advanced Reservation

In existing Grid systems, incoming jobs to resources are either scheduled via Space-shared or Time-shared mode. The Space-shared mode runs jobs based on their submission times; similar to First Come First Serve (FCFS), whereas the Time-shared mode allows multiple executions of jobs; hence it behaves like a Round Robin approach. With a Utility Grid, jobs can be prioritized based on users' QoS by a resource scheduler. However, a Utility Grid is not necessarily able to handle high priority jobs or guarantee reliable service. Therefore, *advance reservation* needs to be introduced in a Utility Grid system to secure resources prior to their execution.

Advanced Reservation (AR) is a process of requesting resources for use at a specific time in the future [71]. Common resources that can be reserved or requested are processors, memory, disk space and network bandwidth or a combination of any of those. The main advantage of AR is that it guarantees the availability of resources to users and applications at specific times in the future. Hence, from the user's perspective:

- Jobs can be executed straight away at a specified time rather than being held up in a queue by other jobs. This is a highly-desirable approach for executing workflow applications that have one or more dependencies.
- Avoiding any dropouts in a network transfer which is not an option in multimedia streaming applications such as video-conferencing.

Combining utility computing with AR allows resource providers to specify criteria and requirements of usage. In addition, resource providers can match and satisfy user's QoS. Utility computing applies to different stages of AR [72] as follows:

- *Requesting a new reservation slot*: A user asks a resource about the availability of a reservation slot by giving details such as start time, duration time, and number of processors. A resource can then either accept or reject the request (if the slot is already booked). Both the user and resource can continue negotiating until they reach an agreement, i.e. to accept or not to accept.
- *Modifying an existing reservation slot*: A user or a resource can request to modify an existing reservation slot. Modification is used as a way to shorten or extend the duration of a reservation. If a user's jobs are running longer than expected, extending the duration time is needed to prevent them from being preempted by a resource. A user can request to shorten a reservation if jobs have finished, in order to save some costs.
- *Canceling an existing reservation slot*: A user or a resource can request to cancel an existing reservation slot. However, this can be done only when both parties have agreed beforehand. Canceling a reservation before it starts can easily be agreed to by a resource. However, some resource providers may not allow the cancellation of a reservation once it has started.

3.7. Cooperative Virtual Organizations

The concept of *virtual organization* (VO) is crucial to the Grid. In current Grid collaborations, physical organizations engage in projects and alliances such as joint ventures that require shared access to compute and data resources provided by their members. The model adopted for such endeavors is that of a VO, in which resource providers and users are organized in a structure that may comprise several physical organizations. VOs may vary in several ways, such as scope, dynamism, and purpose, even though they impose similar challenges regarding their formation, operation and dissolution [73].

Over the years, applications and compute and data resources have been virtualized to enable the utility model for business processes. With regard to the creation of a VO, it may be assumed that a VO is formed because of the need from a business process or some project. Despite the virtualization provided by Grid technologies, organizations may have difficulties in expressing their needs and requirements to their potential partners. Additional challenges are the selection of partners and the establishment of trust at a level that allows the automated creation of VOs [74]. However, to enable the utility model in VOs, issues regarding the responsive or even the automated creation of VOs need to be tackled.

The operational phase of a VO is also a complex task. Resource sharing in VOs is conditional and rules-driven. Also, the relationship in some VOs is peer-to-peer. To complicate matters further, the collection of participating entities is dynamic [26]. This scenario complicates tasks such as the negotiation of SLAs among the participants of the VO or between the participants and the VO itself. Furthermore, the reconciliation, management, and enforcement of resource usage control policies in the VO poses several challenges as presented in [75]. For example, a simple model for providing resources as utilities in VOs requires the presence of a trusted VO manager. Resource providers are committed to deliver services to the VO according to contracts established with the VO manager. The manager is therefore responsible for assigning quotas of these resources to VO groups and users based on some VO policy. Users are allowed to use services according to these quotas and the VO policy. However, in this context, the delivery of compute and data resources in a utility-model to VO users and groups makes tasks such as enforcement of policies in a VO level and accounting difficult.

A simple model in a VO that follows a peer-to-peer sharing approach is of best effort, in which “you give what you can and get what others can offer”. A more elaborate model in which the presence of a VO manager does not exist allows the delivery of services following a "you get what give" approach [76]. Other approaches require multilateral agreements among the members of the VO and give rise to challenges in the enforcement of resource usage policies, as described before.

Current works have not focused on aspects related to the dissolution of VOs, since this problem involves more legal and social issues rather than technical issues. Hence, the delivery of compute resources as a utility in VOs requires the investigation and solving of these problems related to various aspects of a VO. Automation and responsiveness are also required in every stage of the lifecycle of a VO.

4. Industrial Solutions for Utility Computing

Various commercial vendors have launched industrial solutions to support utility computing. Competing marketing terms are used by different vendors even though they share the same vision of providing utility computing. This section discusses four major industrial solutions, as listed in Table 2: HP's Adaptive Enterprise, IBM's E-Business On Demand, Oracle's On Demand, and Sun Microsystems's Sun Grid. All four solutions use Grids as the core enabling technology.

Table 2. Some major industrial solutions for utility computing.

Vendor	Solution and Website	Brief Description	Core Enabling Technology and Website
HP	Adaptive Enterprise http://www.hp.com/go/adaptive	Simplifies, standardizes, modularizes, and integrates business processes and applications with IT infrastructures to adapt effectively in a changing business environment.	Grids http://www.hp.com/go/grid
IBM	E-Business On Demand http://www.ibm.com/ondemand	Performs on demand business processes that include research and development, engineering and product design, business analytics, and enterprise optimization.	Grids http://www.ibm.com/grid
Oracle	On Demand http://www.oracle.com/ondemand	Standardizes and consolidates servers and storage resources to automate IT process management.	Grids http://www.oracle.com/grid
Sun Microsystems	Sun Grid http://www.sun.com/service/sungrid	Offers computing power pay-as-you-go service utility by charging users $1 for every hour of processing.	Grids http://www.sun.com/software/grid

4.1. HP Adaptive Enterprise

The vision of HP's *Adaptive Enterprise* [77] is to synchronize the business and IT processes in an enterprise in order to allow it to benefit from changes in market demands. IT processes are coordinated through the Adaptive Enterprise architecture which comprises two dimensions: IT management and IT service capabilities.

The IT management dimension involves:

- *IT business management*: Long-term IT strategies such as asset management, customer and supplier relationship management, and project portfolio management need to be developed.
- *Service delivery management*: Various operational aspects of service delivery such as availability, cost, capacity, performance, security, and quality need to be considered.
- *Service delivery*: IT services need to be provided by highly automated systems to users.

The IT service capabilities dimension that is addressed under both service delivery management and service delivery in the IT management dimension consists of:

- *Business services*: Represent top-level services related to business processes such as the composition of workflows.
- *Information services*: Consolidate and manipulate information for business services that are independent from application services.
- *Application services*: Automate and handle the processing of applications.
- *Infrastructure services*: Create a common infrastructure platform to host the application, information, and business services.

The Adaptive Enterprise architecture also defines four design principles that are to be realized for an enterprise to become more adaptive:

- *Simplification*: Complex IT environments can be streamlined through application integration, process automation, and resource virtualization to facilitate easier management and faster response.
- *Standardization*: Standardized architectures and processes enable easier incorporation of new technologies, improves collaboration and saves cost.
- *Modularity*: Smaller reusable components can be deployed faster and more easily, increasing resource sharing and reducing cost.
- *Integration*: Dynamic linking of business processes, applications, and infrastructure components enhance agility and cost efficiency.

Grid technologies enable the successful implementation of the Adaptive Enterprise architecture to link IT infrastructure dynamically to business processes by fulfilling the following design rules:

- *Service-oriented architecture (SOA)*: Grid services are defined based on OGSA [55], an open standard that leverages web services to allow large-scale collaboration across the Internet.
- *Virtualization*: Grids harness a large pool of resources that can be shared across applications and processes to meet business demands.
- *Model-based automation*: Grid technologies integrate standalone resources and automate services, thus simplifying the process of deployment, management, and maintenance.

HP customizes the Globus Toolkit [34] as the Grid infrastructure for its platforms. The Grid solutions offered by HP and its partners are listed in Table 3.

Table 3. HP and its partners' Grid solutions.

HP Solutions and Website	HP Partner Solutions and Website
▪ Management Solutions: *OpenView* http://www.hp.com/go/openview	Infrastructure:
▪ Server Solutions: *BladeSystem* http://www.hp.com/go/bladesystem	▪ Application Infrastructure: *DataSynapse* http://www.datasynapse.com
▪ Storage Solutions: *StorageWorks Grid* http://www.hp.com/go/storageworksgrid	▪ Grid Infrastructure: *United Devices* http://www.ud.com
	▪ Resource Management: *Axceleon* http://www.axceleon.com
	▪ Workflow Management: *TurboWorx* http://www.turboworx.com
	▪ Workload Management: *PBS Pro* http://www.altair.com
	▪ Workload Management: *Platform* http://www.platform.com

4.2. IBM E-Business On Demand

IBM's *E-business On Demand* [78] aims to improve the competitiveness and responsiveness of businesses through continuous innovation in products and services. To achieve this aim, E-business On Demand optimizes the following business processes:

- *Research and development*: New innovative products and services need to be researched and developed quickly to make an impact in the highly competitive market.
- *Engineering and product design*: Products and services need to be well-engineered and designed to meet customers' requirements.
- *Business analytics*: Swift and accurate business decisions need to be made based on market performance data in order to remain a market leader.
- *Enterprise optimization*: Standalone resources at various global branches need to be integrated so that workload can be distributed evenly and resources utilized fully to satisfy demand.

E-business On Demand targets numerous industries that include:

- *Automotive and aerospace*: Collaborative design and data-intensive testing.
- *Financial services*: Complex scenario simulation and decision-making.
- *Government*: Coordinated operation across civil and military divisions and agencies.
- *Higher education*: Advanced compute and data-intensive research.
- *Life sciences*: Biological and chemical information analysis and decoding.

Table 4. IBM and its partners' Grid solutions.

IBM Solutions and Website	IBM Partner Solutions and Website
▪ Application Server: *Websphere* http://www.ibm.com/websphere	Infrastructure:
▪ Resource Provisioning: *Tivoli* http://www.ibm.com/tivoli	▪ Application Infrastructure: *DataSynapse* http://www.datasynapse.com
▪ System Server: *eServer* http://www.ibm.com/eserver	▪ Grid Infrastructure: *United Devices* http://www.ud.com
	▪ Grid Infrastructure: *Univa* http://www.univa.com
	▪ Workload Management: *PBS Pro* http://www.altair.com
	▪ Workload Management: *Platform* http://www.platform.com
	Application:
	▪ Document Production: *Sefas* http://www.sefas.com
	▪ Grid Deployment: *SAS* http://www.sas.com/grid
	▪ Risk Management: *Searchspace* http://www.searchspace.com

IBM applies four core enabling technologies for E-Business On Demand: Grid computing, autonomic computing, open standards, and integration technologies. Grid technologies acts as the key component to provide the flexibility and efficiency required for various E-Business On Demand environments:

- *Research and development*: Highly compute- and data-intensive research problems can be solved with lower cost and shorter time by harnessing extra computational and data resources in a Grid.
- *Engineering and product design*: Industry partners are able to collaborate by sharing resources and coordinating engineering and design processes through VOs and open standards-based Grid architecture.

- *Business analytics*: Heavy data analysis and processing can be sped up with extra computational and data resources so that results are derived in time for decision-making.
- *Enterprise optimization*: Virtualization and replication using Grid technologies ensures that under-utilized resources are not wasted, but are instead utilized for backup and recovery purposes.

The core component that IBM deploys for its Grid infrastructure is called the IBM Grid Toolbox which is an enhanced version of the Globus Toolkit [34]. Table 4 lists Grid solutions that are available by IBM and its partners. IBM provides a general integrated Grid solution offering called Grid and Grow for interested customers to easily deploy and sample Grid technologies. In addition, it has created customized Grid offerings for specific industries and applications to drive E-business On Demand.

4.3. Oracle On Demand

Oracle's On Demand aims to enable customers to focus on more strategic business objectives by improving their IT performance and maximize the return on investment in four areas:

- *Quality*: A comprehensive and configurable set of services specifically designed to improve IT performance will be continuously delivered.
- *Cost*: IT expenses are more easily predicted and lower as there is no need to spend on additional unexpected repairs and upgrades.
- *Agility*: The offered service is flexible and can be tailored to satisfy changing complexities, environments, and business needs.
- *Risk*: Service-level commitments guarantees problem resolution, enhancements, and expansions to maximize accountability.

Oracle's On Demand is implemented using Grid solutions through three basic steps:

- *Consolidation*: Hardware, applications, and information can be shared across multiple data centers.
- *Standardization*: Using common infrastructure, application, and information services bridges the gap between various servers, storages, and operating systems.
- *Automation*: Less system administration work is required as multiple resources can be managed concurrently and more easily.

Table 5. Oracle and its partners' Grid solutions.

Oracle Solutions and Website	Oracle Partner Solutions and Website
▪ Data Provisioning: *Oracle Database 10g* http://www.oracle.com/database	Infrastructure:
▪ Management Solutions: *Oracle Fusion Middleware* http://www.oracle.com/middleware	▪ Grid Infrastructure: *Apple* http://www.apple.com
▪ Resource Provisioning: *Oracle Enterprise Manager 10g* http://www.oracle.com/technology/products/oem	▪ Grid Infrastructure: *Egenera* http://www.egenera.com
	▪ Grid Infrastructure: *HP* http://www.hp.com
	▪ Grid Infrastructure: *Network Appliance* http://www.netapp.com
	Application:
	▪ Database Management: *GridApp* http://www.gridapp.com
	▪ Database Management: *Grid-Tools* http://www.grid-tools.com
	▪ Enterprise Automation: ORSYP http://www.orsyp.com

Table 5 shows the Grid solutions from Oracle and its partners. Oracle Database 10g is the first database designed for Grid computing and offers data provisioning capabilities, such as detaching part of a database and attaching it to another database without unloading and reloading.

4.4. Sun Microsystems Sun Grid

Sun Microsystems's *Sun Grid* aims to provide affordable commodity-based computing power pay-as-you-go service. Sun Grid currently supports three types of compute utility service (see Table 6 for comparison):

- *Compute utility*: A standard offering that provides instant deployment for anyone with Internet access at $1 per hour of processing.
- *Commercial utility*: A single tenant standard offering in a multi tenant hosting center that targets medium and large enterprises.
- *Variable cost infrastructure*: A customized modular offering and single tenant utility model for large enterprises and system integrators.

Table 6. Comparison of Sun Grid compute utility services.

Comparison	Sun Grid Compute Utility Services		
	Compute Utility	Commercial Utility	Variable Cost Infrastructure
Access	Portal	Dedicated	Customer or system integrator hosted
Availability	Instantaneous	Short notice	Longer term contract
Scheduling	No reservation	Time-based reservation	Dedicated or time-based reservation
Pricing	All-inclusive $1/CPU-hour	Negotiated $/CPU-hour	Negotiated
Business Terms	Standard	Service level	Capacity provisioning
Technology	Solaris 10 x64	Solaris 10 x64 or Redhat Linux	Menu of options
Storage	10 GB standard	Customer defined	Menu of options

The Sun Grid compute utility [79] provides a golden opportunity for non-IT users to make use of utility computing by providing a simple and easy to use web interface that hides the complex Grid technologies involved. Given this assumption of a simple utility computing environment, the Sun Grid compute utility has several limitations:

- Submitted applications must be able to execute on Solaris 10 operating system.
- Submitted applications must be self-contained and scripted to work with Sun N1 Grid Engine [80] software without requiring interactive access.
- Submitted applications has to be implemented using standard object libraries included with Solaris 10 Operating system or user libraries packaged with the executable.
- The application can obtain finer control over compute resources through the interfaces provided by the Sun N1 Grid Engine software.
- The total maximum size of applications and data must be less than 10 GBytes.
- Applications and data can only be uploaded through the web interface and may be packaged into compressed ZIP files of less than 100 MBytes each.

Grid technologies are employed for the Sun Grid compute utility with the following compute node configuration:

- 8 GBytes of memory.
- Solaris 10 operating system.
- Sun N1 Grid Engine 6 software for resource management of compute nodes.
- Grid network infrastructure built on Gigabit Ethernet.

- Web-based portal for users to submit jobs and upload data.
- 10 GBytes of storage space for each user.

Industries that are targeted by the Sun Grid compute utility consists of:

- *Energy*: Reservoir simulations and seismic processing.
- *Entertainment/Media*: Digital content creation, animation, rendering, and digital asset management.
- *Financial services*: Risk analysis and Monte Carlo simulations.
- *Government education*: Weather analysis and image processing.
- *Health sciences*: Medical imaging, bioinformatics, and drug development simulations.
- *Manufacturing*: Electronic design automation, mechanical computer-aided design, computational fluid dynamics, crash-test simulations, and aerodynamic modeling.

Sun Microsystems, Gridwise Tech, and the Globus project have been collaborating in the joint development of interfaces between Sun Grid solutions and the Globus Toolkit [81]. Table 7 shows Grid solutions that are developed by Sun Microsystems and its partners.

Table 7. Sun Microsystems and its partners' Grid solutions.

Sun Microsystems Solutions and Website	Sun Microsystems Partner Solutions and Website
▪ Resource Management: *N1* http://www.sun.com/software/n1gridsystem	Infrastructure:
	▪ Application Server: *GigaSpaces* http://www.gigaspaces.com
	▪ Autonomic Processing: *Paremus* http://www.paremus.com
	▪ Workload Management: *Platform* http://www.platform.com
	Service Management:
	▪ Collaborative Solutions: *SAP* http://www.sap.com
	▪ Database Solutions: *Oracle* http://www.oracle.com
	▪ Service-Oriented Solutions: *BEA* http://www.bea.com
	Software As a Service:
	▪ Pricing and Risk Solutions: *CDO2* http://www.cdo2.com

5. Summary

In this chapter, the utility computing model and its vision of being the next generation of IT evolution is introduced. The utility computing model is significantly different from traditional IT models, and thus requires organizations to amend their existing IT procedures and operations towards this outsourcing model so as to save costs and improve quality. There is also increasing emphasis on adopting Grid computing technologies to enable utility computing environments.

This chapter has focused on the potential of Grids as utility computing environments. A reference service-oriented architecture of Utility Grids has been discussed, along with how services are assembled on demand in the Utility Grid. The challenges involved in utility-based resource allocation at various levels of the Utility Grid are then examined in detail. With commercial vendors rapidly launching utility computing solutions, industrial solutions by three pioneer vendors (HP, IBM, and Sun Microsystems) and their realization through Grid technologies are also presented.

For recent advances in Grid computing technologies and applications, readers are recommended to browse the proceedings of CCGrid [82], Grid [83], and e-Science [84] conference series organized by the IEEE Technical Committee on Scalable Computing (TCSC) [85].

6. Acknowledgements

The authors thank Hussein Gibbins and the anonymous reviewers for their comments. This work is partially supported through the Australian Research Council (ARC) Discovery Project grant.

7. Glossary

Adaptive Enterprise – An organization that is able to adjust and benefit according to the changes in its operating environment.

Grid Computing – A model allowing organizations to access a large quantity of remotely distributed computing resources on demand.

Market-based Resource Allocation – Assignment of resources based on market supply and demand from providers and users.

Middleware – Software designed to interface and link separate software and/or hardware.

On Demand Computing – Computing services that can be accessed when required by the user.

Service Level Agreement (SLA) – A contract agreed upon between a service provider and a user which formally specifies service quality that the provider is required to provide.

Service-Oriented Architecture (SOA) – An architectural framework for the definition of services that are able to fulfill the requirements of users.

Utility Computing – A model whereby service providers offer computing resources to users only when the users need them and charges the users based on usage.

Virtual Organization (VO) – A temporary arrangement formed across physically dispersed departments and organizations with a common objective to facilitate collaboration and coordination.

8. References

[1] Rajkumar Buyya (editor), *High Performance Cluster Computing: Architectures and Systems*, Volume 1, Prentice Hall, 1999.

[2] Gregory F. Pfister, *In Search of Clusters*, Second Edition, Prentice Hall, 1998.

[3] Ian Foster and Carl Kesselman (editors), *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 2003.

[4] Michael A. Rappa, “The utility business model and the future of computing services,” *IBM Systems Journal*, vol. 43, no. 1, 2004, pp. 32-42.

[5] Leonard Kleinrock, “A vision for the Internet,” ST Journal of Research, vol. 2, no. 1, November 2005, pp. 4-5.

[6] Jeanne W. Ross and George Westerman, “Preparing for utility computing: The role of IT architecture and relationship management,” *IBM Systems Journal*, vol. 43, no. 1, 2004, pp. 5-19.

[7] META Group, “The Adaptive Organization: An Examination of On Demand Computing,” META Group Multiclient Study, May 2004.

[8] Ian Foster, Carl Kesselman, and Steven Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations,” *The International Journal of High Performance Computing Applications*, vol. 15, no. 3, Fall 2001, pp. 200-222.

[9] Rajkumar Buyya, Toni Cortes, and Hai Jin, “Single System Image,” *The International Journal of High Performance Computing Applications*, vol. 15, no. 2, Summer 2001, pp. 124-135.

[10] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Software: Practice and Experience*, vol. 32, no. 2, February 2002, pp. 135-164.
[11] TeraGrid, http://www.teragrid.org (accessed November 2006).
[12] ChinaGrid, http://www.chinagrid.edu.cn (accessed November 2006).
[13] APACGrid, http://www.apac.edu.au (accessed November 2006).
[14] LHCGrid, http://www.cern.ch/lcg (accessed November 2006).
[15] GriPhyN, http://www.griphyn.org (accessed November 2006).
[16] NetSolve/GridSolve, http://icl.cs.utk.edu/netsolve (accessed November 2006).
[17] AccessGrid, http://www.accessgrid.org (accessed November 2006).
[18] Mario Cannataro and Domenico Talia, The Knowledge Grid, *Communications of the ACM*, vol. 46, no. 1, 2003, pp. 89-93.
[19] EU Data Mining Grid, http://www.dataminininggrid.org (accessed November 2006).
[20] Rajkumar Buyya and Srikumar Venugopal, "The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report", *Proceedings of the First IEEE International Workshop on Grid Economics and Business Models (GECON 2004, April 23, 2004, Seoul, Korea)*, 19-36pp, ISBN 0-7803-8525-X, IEEE Press, New Jersey, USA.
[21] Sven Graupner, Jim Pruyne, and Sharad Singhal, "Making the Utility Data Center a Power Station for the Enterprise Grid", *HP Labs Technical Report*, Palo Alto, USA, 2003. http://www.hpl.hp.com/techreports/2003/
[22] Rajkumar Buyya, David Abramson, and Srikumar Venugopal, "The Grid Economy," *Proceedings of the IEEE*, vol. 93, no. 3, March 2005, pp. 698-714.
[23] Message Passing Interface (MPI) Forum, http://www.mpi-forum.org (accessed November 2006).
[24] Bill Nitzberg and Virginia Lo, "Distributed Shared Memory: A Survey of Issues and Algorithms", *IEEE Computer*, vol. 24, no. 8, 1991, pp.52-60.
[25] Joshy Joseph and Craig Fellenstein, *Grid Computing*, Prentice Hall, 2004.
[26] Joshy Joseph, Mark Ernest, and Craig Fellenstein, "Evolution of grid computing architecture and grid adoption models," *IBM Systems Journal*, vol. 43, no. 4, 2004, pp. 624-645.
[27] Melissa J. Buco, Rong N. Chang, Laura Z. Luan, Christopher Ward, Joel L. Wolf, and Philip S. Yu, "Utility computing SLA management based upon business objectives," *IBM Systems Journal*, vol. 43, no. 1, 2004, pp. 159-178.
[28] Jeffrey O. Kephart and David M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, vol. 36, no. 1, January 2003, pp. 41-50.
[29] Richard Murch, *Autonomic Computing*, Prentice Hall, 2004.
[30] Chris Kenyon and Giorgos Cheliotis, "Elements of Financial Risk Management for Grid and Utility Computing," Abderrahim Labbi (editor), *Handbook of Integrated Risk Management for E-Business: Measuring, Modeling, and Managing Risk*, Chapter 8, pp. 169-191, J. Ross Publishing, 2005.
[31] Giuseppe A. Paleologo, "Price-at-Risk: A methodology for pricing utility computing services," *IBM Systems Journal*, vol. 43, no. 1, 2004, pp. 20-31.
[32] Platform Computing, *The Politics of Grid*, http://www2.platform.com/adoption/politics (accessed November 2006).
[33] IBM developerWorks, *Six Strategies for Grid Application Enablement*, http://www.ibm.com/developerworks/grid/library/gr-enable (accessed November 2006).
[34] The Globus Alliance, *The Globus Toolkit*, http://www.globus.org/toolkit (accessed November 2006).
[35] David Abramson, Jon Giddy, and Lew Kotler, "High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?" *Proceedings of the 14th International Parallel and Distributed Processing Symposium (IPDPS 2000)*, Cancun, Mexico, May 2000, pp. 520-528.
[36] The Gridbus Project, *The Gridbus Toolkit*, http://www.gridbus.org/middleware (accessed November 2006).
[37] Rong Ge, Xizhou Feng, and Kirk W. Cameron, "Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters," *Proceedings of the 2005 ACM/IEEE Supercomputing Conference (SC 2005)*, Seattle, WA, November 2005.
[38] Chee Shin Yeo and Rajkumar Buyya, "A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing," *Software: Practice and Experience*, vol. 36, no. 13, 10 November 2006, pp. 1381-1419.
[39] Stephen D. Kleban and Scott H. Clearwater, "Computation-at-Risk: Assessing Job Portfolio Management Risk on Clusters," *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, Santa Fe, NM, April 2004.
[40] David E. Irwin, Laura E. Grit and Jeffrey S. Chase, "Balancing Risk and Reward in a Market-based Task Service," *Proceedings of the 13th International Symposium on High Performance Distributed Computing (HPDC13)*, Honolulu, HI, June 2004, pp. 160-169.
[41] Mohammad Islam, Pavan Balaji, Ponnuswamy Sadayappan, and Dhabaleswar K. Panda, "Towards Provision of Quality of Service Guarantees in Job Scheduling," *Proceedings of the 6th International Conference on Cluster Computing (Cluster 2004)*, San Diego, CA, September 2004, pp. 245-254.

[42] Chee Shin Yeo and Rajkumar Buyya, "Service Level Agreement based Allocation of Cluster Resources: Handling Penalty to Enhance Utility," *Proceedings of the 7th International Conference on Cluster Computing (Cluster 2005)*, Boston, MA, September 2005.

[43] Florentina I. Popovici and John Wilkes, "Profitable services in an uncertain world," *Proceedings of the 18th Conference on Supercomputing (SC 2005)*, Seattle, WA, November 2005.

[44] Walter B. Ligon III and Rob Ross, "PVFS: A Parallel File System for Linux Clusters," Thomas Sterling (editor), *Beowulf Cluster Computing with Linux*, pp. 391-430, MIT Press, 2001.

[45] Jason Barkes, Marcelo R. Barrios, Francis Cougard, Paul G. Crumley, Didac Marin, Hari Reddy, and Theeraphong Thitayanun, "GPFS: A Parallel File System," IBM Redbook SG24-5165-00, April 1998.

[46] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "The Google File System," *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003)*, Bolton Landing, NY, October 2003, pp. 29-43.

[47] Brian F. Cooper and Hector Garcia-Molina, "Peer-to-Peer Data Trading to Preserve Information," *ACM Transactions on Information Systems*, vol. 20, no. 2, April 2002, pp. 133-170.

[48] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage," *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, Cambridge, MA, November 2000, pp. 190-201.

[49] Gernot Heiser, Kevin Elphinstone, Jerry Vochteloo, Stephen Russell, and Jochen Liedtke, "The Mungi Single-Address-Space Operating System," *Software Practice and Experience*, vol. 28, no. 9, 25 July 1998, pp. 901-928.

[50] Bryce Wilcox-O'Hearn, "Experiences Deploying a Large-Scale Emergent Network," *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, Lecture Notes in Computer Science (LNCS) 2429/2002, Cambridge, MA, March 2002, pp. 104-110.

[51] Martin Placek and Rajkumar Buyya, "Storage Exchange: A Global Trading Platform for Storage Services", Proceedings of the 12th International European Conference on Parallel Computing (Euro-Par 2006), Lecture Notes in Computer Science (LNCS), Dresden, Germany, August 2006.

[52] Sudharshan S. Vazhkudai, Xiaosong Ma, Vincent W. Freeh, Jonathan W. Strickland, Nandan Tammineedi, and Stephen L. Scott, "FreeLoader: Scavenging Desktop Storage Resources for Scientific Data," *Proceedings of the 18th Supercomputing Conference (SC 2005)*, Seattle, WA, November 2005.

[53] David Abramson, Rajkumar Buyya, and Jonathan Giddy, "A computational economy for Grid computing and its implementation in the Nimrod-G resource broker," *Future Generation Computer Systems*, vol. 18, no. 8, October 2002, pp. 1061-1074.

[54] Srikumar Venugopal, Rajkumar Buyya, and Lyle Winton, "A Grid service broker for scheduling e-Science applications on global data Grids," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 6, May 2006, pp. 685-699.

[55] Open Grid Forum, *The Open Grid Services Architecture*, http://forge.gridforum.org/projects/ogsa-wg (accessed November 2006).

[56] Mario Antonioletti, Shannon Hastings, Amy Krause, Stephen Langella, Simon Laws, Susan Malaika, and Norman W. Paton, "Web Services Data Access and Integration – The XML Realization (WS-DAIX)," GGF DAIS Working Group Informational Draft, September 2004.

[57] Ian Foster, Jens Vöckler, Michael Wilde, and Yong Zhao, "The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration," *Proceedings of the 1st Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, January 2003.

[58] Ewa Deelman, James Blythe, Yolanda Gil, and Carl Kesselman, "Workflow Management in GriPhyN," Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz (editors), *Grid Resource Management: State of the Art and Future Trends*, Chapter 7, pp. 99-117, Kluwer Academic Publishers, 2003.

[59] Reagan W. Moore, Arcot Rajasekar, and Michael Wan, "Data Grids, Digital Libraries and Persistent Archives: An Integrated Approach to Sharing, Publishing, and Archiving Data," *Proceedings of the IEEE*, vol. 93, no. 3, March 2005, pp. 578-588.

[60] David Hollinsworth, "The Workflow Reference Model," The Workflow Management Coalition Specification, Document Number TC00-1003, January 1995.

[61] Daniel P. Spooner, Junwei Cao, Stephen A. Jarvis, Ligang He, and Graham R. Nudd, "Performance-Aware Workflow Management for Grid Computing," *The Computer Journal*, vol. 48, no. 3, 2005, pp. 347-357.

[62] Guntram Berti, Siegfried Benkner, John W. Fenner, Jochen Fingberg, Guy Lonsdale, Stuart E. Middleton, and Mike Surridge, "Medical Simulation Services via the Grid", *Proceedings of the 1st HealthGRID Conference (HealthGRID 2003)*, Lyon, France, January 2003.

[63] Jia Yu and Rajkumar Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," *Journal of Grid Computing*, vol. 3, no. 3-4, September 2005, pp. 171-200.

[64] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu, “Web Services Agreement Specification (WS-Agreement) Version 1.1,” Draft 18, GGF GRAAP Working Group, May 2004.

[65] Rashid J. Al-Ali, Kaizar Amin, Gregor von Laszewski, Omer F. Rana, David W. Walker, Mihael Hategan, and Nestor Zaluzec, “Analysis and Provision of QoS for Distributed Grid Applications”, *Journal of Grid Computing*, vol. 2, no. 2, June 2004, pp. 163-182.

[66] Siegfried Benkner, Ivona Brandic, Gerhard Engelbrecht, and Rainer Schmidt, “VGE - A Service-Oriented Grid Environment for On-Demand Supercomputing”, *Proceedings of the 5th International Workshop on Grid Computing (Grid 2004)*, Pittsburgh, PA, USA, November 2004, pp. 11-18.

[67] Ivona Brandic, Siegfried Benkner, Gerhard Engelbrecht, and Rainer Schmidt, “QoS Support for Time-Critical Grid Workflow Applications”, *Proceedings of the 1st International Conference on e-Science and Grid Computing (e-Science 2005),* Melbourne, Australia, December 2005, pp. 108-115.

[68] Daniel A. Menascé and Emiliano Casalicchio, “A Framework for Resource Allocation in Grid Computing,” *Proceedings of the 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2004)*, Volendam, Netherlands, October 2004, pp. 259-267.

[69] Jia Yu, Rajkumar Buyya, and Chen Khong Tham, “Cost-based Scheduling of Scientific Workflow Applications on Utility Grids”, *Proceedings of the 1st International Conference on e-Science and Grid Computing (e-Science 2005),* Melbourne, Australia, December 2005, pp. 140-147.

[70] Jia Yu and Rajkumar Buyya, “A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms”, *Workshop on Workflows in Support of Large-Scale Science (WORKS), Proceedings of the 15th International Symposium on High Performance Distributed Computing (HPDC 2006)*, Paris, France, June 2006.

[71] Warren Smith, Ian Foster, and Valerie Taylor, “Scheduling with Advanced Reservations,” *Proceedings of the 14th International Parallel and Distributed Processing Symposium (IPDPS 2000)*, Cancun, Mexico, May 2000, pp. 127-132.

[72] Anthony Sulistio and Rajkumar Buyya, “A Grid Simulation Infrastructure Supporting Advance Reservation,” *Proceedings of the 16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, Cambridge, MA, November 2004, pp. 1-7.

[73] Jigar Patel, W. T. Luke Teacy, Nicholas R. Jennings, Michael Luck, Stuart Chalmers, Nir Oren, Timothy J. Norman, Alun Preece, Peter M. D. Gray, Gareth Shercliff, Patrick J. Stockreisser, Jianhua Shao, W. Alex Gray, Nick J. Fiddian, and Simon Thompson, “Agent-based Virtual Organisations for the Grid,” *International Journal of Multi-Agent and Grid Systems*, vol. 1, no. 4, 2005, pp. 237-249.

[74] Theo Dimitrakos, David Golby, and Paul Kearney, “Towards a Trust and Contract Management Framework for Dynamic Virtual Organisations,” *Proceedings of the eChallenges Conference (eChallenges 2004)*, Vienna, Austria, October 2004.

[75] Catalin L. Dumitrescu, Michael Wilde, and Ian Foster, “A Model for Usage Policy-based Resource Allocation in Grids,” *Proceedings of the 6th International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*, Stockholm, Sweden, June 2005, pp. 191-200.

[76] Glenn Wasson and Marty Humphrey, “Policy and Enforcement in Virtual Organizations,” *Proceedings of the 4th International Workshop on Grid Computing (Grid 2003)*, Phoenix, AZ, November 2003, pp. 125-132.

[77] HP, “Adaptive Enterprise: Business and IT synchronized to capitalize on change,” HP White Paper 4AA0-0760ENW, June 2005.

[78] IBM, “Unleash the power of e-business on demand,” IBM White Paper G522-2580-00, October 2003.

[79] Sun Microsystems, “Sun Grid Compute Utility Reference Guide,” Part No. 819-5131-10, January 2006.

[80] Sun Microsystems, *Sun N1 Grid Engine 6*, http://www.sun.com/software/gridware (accessed November 2006).

[81] GridwiseTech, *Grid Engine-Globus Toolkit adapter*, http://www.gridwisetech.com/ge-gt (accessed November 2006).

[82] International Symposium on Cluster Computing and the Grid (CCGrid), http://www.buyya.com/ccgrid (accessed November 2006).

[83] International Conference on Grid Computing (Grid), http://www.gridcomputing.org (accessed November 2006).

[84] International Conference on e-Science and Grid Computing (e-Science), http://www.escience-meeting.org (accessed November 2006).

[85] IEEE Technical Committee on Scalable Computing (TCSC), http://www.ieeetcsc.org (accessed November 2006).