Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility

Rajkumar Buyya[1,2], Chee Shin Yeo[1], Srikumar Venugopal[1], James Broberg[1], and Ivona Brandic[3]

[1] **Gri**d Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
Email: {raj, csyeo, srikumar, brobergj}@csse.unimelb.edu.au

[2] Manjrasoft Pty Ltd, Melbourne, Australia

[3] Institute of Information Systems
Vienna University of Technology
Argentinierstraße 8, 1040 Vienna, Austria
Email: ivona@infosys.tuwien.ac.at

Abstract

With the significant advances in Information and Communications Technology (ICT) over the last half century, there is an increasingly perceived vision that computing will one day be the 5th utility (after water, electricity, gas, and telephony). This computing utility, like all other four existing utilities, will provide the basic level of computing service that is considered essential to meet the everyday needs of the general community. To deliver this vision, a number of computing paradigms have been proposed, of which the latest one is known as Cloud computing. Hence, in this paper, we define Cloud computing and provide the architecture for creating Clouds with market-oriented resource allocation by leveraging technologies such as Virtual Machines (VMs). We also provide insights on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain Service Level Agreement (SLA)-oriented resource allocation. In addition, we reveal our early thoughts on interconnecting Clouds for dynamically creating global Cloud exchanges and markets. Then, we present some representative Cloud platforms, especially those developed in industries along with our current work towards realizing market-oriented resource allocation of Clouds as realized in Aneka enterprise Cloud technology. Furthermore, we highlight the difference between High Performance Computing (HPC) workload and Internet-based services workload. We also describe a meta-negotiation infrastructure to establish global Cloud exchanges and markets, and illustrate a case study of harnessing 'Storage Clouds' for high performance content delivery. Finally, we conclude with the need for convergence of competing IT paradigms to deliver our 21st century vision.

Keywords: Cloud Computing, Data Centers, Utility Computing, Virtualization, Market-Oriented Resource Allocation

1. Introduction

Computing is being transformed to a model consisting of services that are commoditized and delivered in a manner similar to traditional utilities such as water, electricity, gas, and telephony. In such a model, users access services based on their requirements without regard to where the services are hosted or how they are delivered. Several computing paradigms have promised to deliver this *utility computing* vision and these include cluster computing, Grid computing, and more recently *Cloud computing*. The latter term denotes the infrastructure as a "*Cloud*" from which businesses and users are able to access applications from anywhere in the world on demand. Thus, the computing world is rapidly transforming towards developing software for millions to consume as a service, rather than to run on their individual computers.

At present, it is common to access content across the Internet independently without reference to the underlying hosting infrastructure. This infrastructure consists of data centers that are monitored and maintained around the

clock by content providers. Cloud computing is an extension of this paradigm wherein the capabilities of business applications are exposed as sophisticated services that can be accessed over a network. Cloud service providers are incentivized by the profits to be made by charging consumers for accessing these services. Consumers, such as enterprises, are attracted by the opportunity for reducing or eliminating costs associated with "in-house" provision of these services. However, since cloud applications may be crucial to the core business operations of the consumers, it is essential that the consumers have guarantees from providers on service delivery. Typically, these are provided through Service Level Agreements (SLAs) brokered between the providers and consumers.

Providers such as Amazon, Google, Salesforce, IBM, Microsoft, and Sun Microsystems have begun to establish new data centers for hosting Cloud computing applications in various locations around the world to provide redundancy and ensure reliability in case of site failures. Since user requirements for cloud services are varied, service providers have to ensure that they can be flexible in their service delivery while keeping the users isolated from the underlying infrastructure. Recent advances in microprocessor technology and software have led to the increasing ability of commodity hardware to run applications within *Virtual Machines* (VMs) efficiently. VMs allow both the isolation of applications from the underlying hardware and other VMs, and the customization of the platform to suit the needs of the end-user. Providers can expose applications running within VMs, or provide access to VMs themselves as a service (e.g. Amazon Elastic Compute Cloud) thereby allowing consumers to install their own applications. While convenient, the use of VMs gives rise to further challenges such as the intelligent allocation of physical resources for managing competing resource demands of the users.

In addition, enterprise service consumers with global operations require faster response time, and thus save time by distributing workload requests to multiple Clouds in various locations at the same time. This creates the need for establishing a computing atmosphere for dynamically interconnecting and provisioning Clouds from multiple domains within and across enterprises. There are many challenges involved in creating such Clouds and Cloud interconnections.

Therefore, this paper discusses the current trends in the space of Cloud computing and presents candidates for future enhancements of this technology. This paper is primarily divided into two parts. The first part examines current research issues and developments by:

- presenting the 21st century vision of computing and describing various computing paradigms that have promised or are promising to deliver this grand vision (Section 2),
- differentiating Cloud computing from two other widely explored computing paradigms: Cluster computing and Grid computing (Section 3),
- focusing on VM-centric Cloud services and presenting an architecture for creating market-oriented Clouds using VMs (Section 4),
- providing insights on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain SLA-oriented resource allocation (Section 5),
- revealing our early thoughts on interconnecting Clouds for dynamically creating global Cloud exchanges and markets (Section 6), and
- comparing some representative Cloud platforms, especially those developed in industries along with our Aneka enterprise Cloud technology (Section 7).

The second part introduces our current work on Cloud computing which include:

- realizing market-oriented resource allocation of Clouds as realized in Aneka enterprise Cloud technology and highlighting the difference between High Performance Computing (HPC) workload and Internet-based services workload (Section 8),
- incorporating a meta-negotiation infrastructure for QoS management to establish global Cloud exchanges and markets (Section 9), and
- creating 3rd party cloud services based on high performance content delivery over commercial cloud storage services (Section 10).

2. The 21st Century Vision of Computing

With the advancement of modern society, basic essential services (*utilities*) are commonly provided such that everyone can easily obtain access to them. Today, utility services, such as water, electricity, gas, and telephony are deemed necessary for fulfilling daily life routines. These utility services are accessed so frequently that they need to be available whenever the consumer requires them at any time. Consumers are then able to pay service providers based on their usage of these utility services.

In 1969, Leonard Kleinrock [1], one of the chief scientists of the original Advanced Research Projects Agency Network (ARPANET) project which seeded the Internet, said: "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." This vision of the computing utility based on the service provisioning model anticipates the massive transformation of the entire computing industry in the 21st century whereby computing services will be readily available on demand, like other utility services available in today's society. Similarly, computing service users (consumers) need to pay providers only when they access computing services. In addition, consumers no longer need to invest heavily or encounter difficulties in building and maintaining complex IT infrastructure. Hence, software practitioners are facing numerous new challenges toward creating software for millions of consumers to use as a service, rather than to run on their individual computers.

The creation of the Internet has marked the foremost milestone towards achieving this grand 21st century vision of '*computer utilities*' by forming a worldwide system of computer networks that enables individual computers to communicate with any other computers located elsewhere in the world. This internetworking of standalone computers reveals the promising potential of utilizing seemingly endless amount of distributed computing resources owned by various owners. As such, over the recent years, new computing paradigms (shown in Figure 1) have been proposed and adopted to edge closer toward achieving this grand vision. Applications making use of these utility-oriented computing systems emerge simply as catalysts or market makers, which brings buyers and sellers together. This creates several trillion dollars worth of the utility/pervasive computing industry as noted by Sun Microsystems co-founder Bill Joy [2]. He also indicated "It would take time until these markets to mature to generate this kind of value. Predicting now which companies will capture the value is impossible. Many of them have not even been created yet."

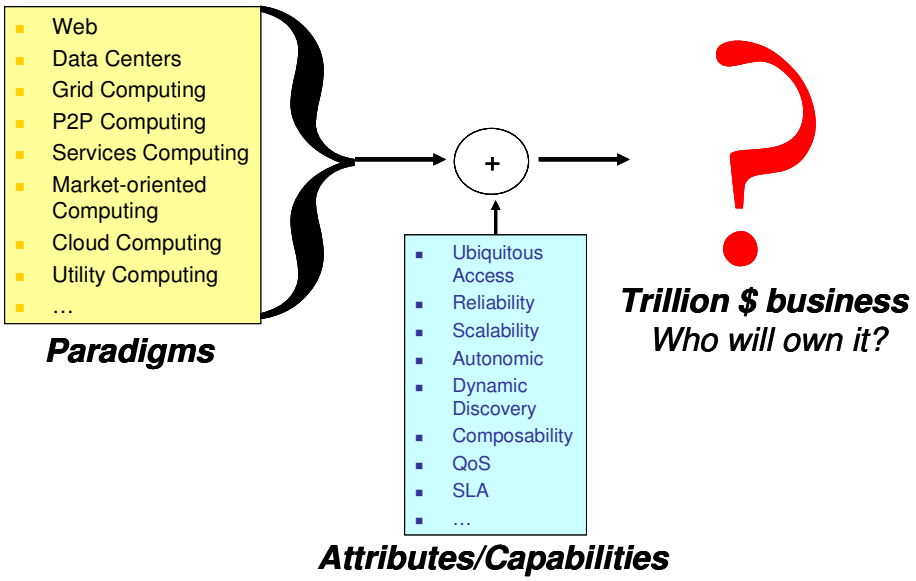


Figure 1: Various paradigms promising to deliver IT as services.

Grid computing [3] enables the sharing, selection, and aggregation of a wide variety of geographically distributed

resources including supercomputers, storage systems, data sources, and specialized devices owned by different organizations for solving large-scale resource-intensive problems in science, engineering, and commerce. Inspired by the electrical power Grid's pervasiveness, ease of use, and reliability [4], the motivation of Grid computing was initially driven by large-scale, resource (computational and data)-intensive scientific applications that required more resources than a single computer (PC, workstation, supercomputer) could have provided in a single administrative domain. Due to its potential to make impact on the 21st century as much as the electric power Grid did on the 20th century, Grid computing has been hailed as the next revolution after the Internet and the World Wide Web.

Peer-to-Peer (P2P) computing allows peer nodes (computers) to share content directly with one another in a decentralized manner. In pure P2P computing, there is no notion of clients or servers since all peer nodes are equal and concurrently be both clients and servers. The goals of P2P computing include cost sharing or reduction, resource aggregation and interoperability, improved scalability and reliability, increased autonomy, anonymity or privacy, dynamism, and ad-hoc communication and collaboration [5].

Services computing focuses on the linkage between business processes and IT services so that business processes can be seamlessly automated using IT services. Examples of services computing technologies include Service-Oriented Architecture (SOA) and Web Services. The SOA facilitates interoperable services between distributed systems to communicate and exchange data with one another, thus providing a uniform means for service users and providers to discover and offer services respectively. The Web Services provides the capability for self-contained business functions to operate over the Internet.

Market-oriented computing views computing resources in economic terms such that resource users will need to pay resource providers for utilizing the computing resources [6]. Therefore, it is able to provide benefits, such as offering incentive for resource providers to contribute their resources for others to use and profit from it, regulating the supply and demand of computing resources at market equilibrium, offering incentive for resource users to back off when necessary, removing the need for a central coordinator (during the negotiation between the user and provider for establishing quality of service expectations and service pricing), and enabling both users and providers to make independent decisions to maximize their utility and profit respectively.

Today, the latest paradigm to emerge is that of Cloud computing [7] which promises reliable services delivered through next-generation data centers that are built on virtualized compute and storage technologies. Consumers will be able to access applications and data from a "Cloud" anywhere in the world on demand. The consumers are assured that the Cloud infrastructure is very robust and will always be available at any time. Computing services need to be highly reliable, scalable, and autonomic to support ubiquitous access, dynamic discovery and composability. In particular, consumers indicate the required service level through Quality of Service (QoS) parameters, which are noted in SLAs established with providers. Of all these paradigms, the recently emerged Cloud computing paradigm appears to be the most promising one to leverage and build on the developments from other paradigms.

3. Definitions, Characteristics, and Trends

In order to facilitate a clear understanding of what exactly is Cloud computing, we compare Cloud computing with two other recent, widely-adopted or explored computing paradigms: Cluster Computing and Grid Computing. We first examine the respective definitions of these three paradigms, then differentiate their specific characteristics, and finally highlight their recent web search trends.

3.1 Definitions

A number of computing researchers and practitioners have attempted to define clusters, Grids, and Clouds [8] in various ways. Here are some definitions that we think are generic enough to stand the test of time.

The essence of Pfister's [9] and Buyya's [10] work defines clusters as follows:

- "A cluster is a type of parallel and distributed system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource."

Buyya defined one of the popular definitions for Grids at the 2002 Grid Planet conference, San Jose, USA as follows:

- "A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed 'autonomous' resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements."

Based on our observation of the essence of what Clouds are promising to be, we propose the following definition:

- "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers."

At a cursory glance, Clouds appear to be a combination of clusters and Grids. However, this is not the case. Clouds are clearly next-generation data centers with nodes "*virtualized*" through hypervisor technologies such as VMs, dynamically "*provisioned*" on demand as a personalized resource collection to meet a specific service-level agreement, which is established through a "*negotiation*" and accessible as a composable service via Web Service technologies such as SOAP and REST.

3.2 Characteristics

A set of characteristics that helps distinguish cluster, Grid and Cloud computing systems is listed in Table 1. The resources in clusters are located in a single administrative domain and managed by a single entity whereas, in Grid systems, resources are geographically distributed across multiple administrative domains with their own management policies and goals. Another key difference between cluster and Grid systems arises from the way application scheduling is performed. The *schedulers* in cluster systems focus on enhancing the overall system performance and utility as they are responsible for the whole system. On the other hand, the schedulers in Grid systems called *resource brokers*, focusing on enhancing the performance of a specific application in such a way that its end-users' QoS requirements are met.

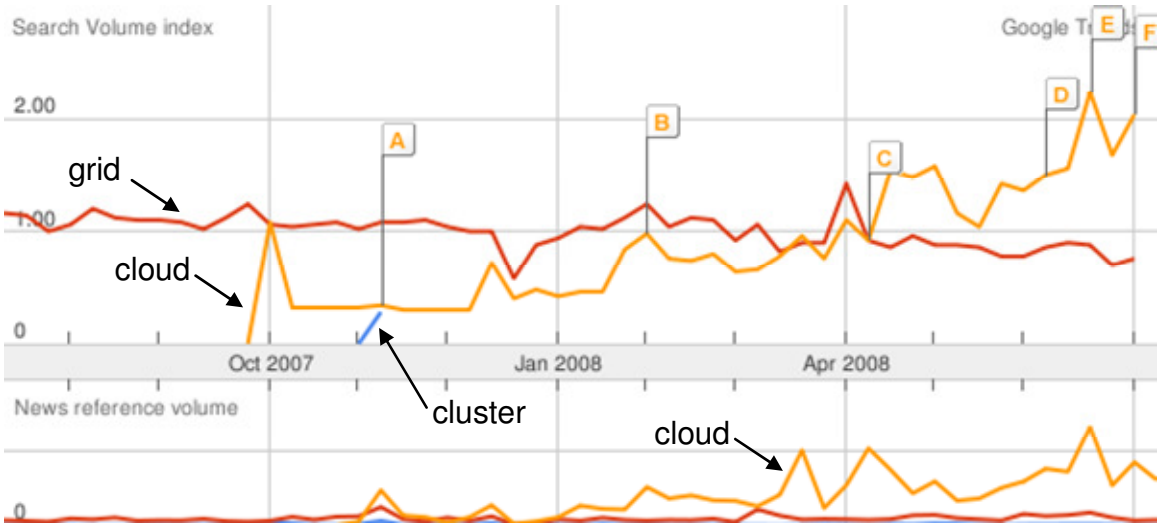
Cloud computing platforms possess characteristics of both clusters and Grids, with its own special attributes and capabilities such strong support for virtualization, dynamically composable services with Web Service interfaces, and strong support for creating 3rd party, value added services by building on Cloud compute, storage, and application services. Thus, Clouds are promising to provide services to users without reference to the infrastructure on which these are hosted.

Table 1: Key Characteristics of Clusters, Grids, and Cloud Systems.

Systems / Characteristics	Clusters	Grids	Clouds
Population	Commodity computers	High-end computers (servers, clusters)	Commodity computers and high-end servers and network attached storage
Size / Scalability	100s	1000s	100s to 1000s
Node Operating System (OS)	One of the standard OSs (Linux, Windows)	Any standard OS (dominated by Unix)	A hypervisor (VM) on which multiple OSs run
Ownership	Single	Multiple	Single
Interconnection Network | Speed	Dedicated, high-end with low latency and high bandwidth	Mostly Internet with high latency and low bandwidth	Dedicated, high-end with low latency and high bandwidth
Security/Privacy	Traditional login/password-based. Medium level of privacy – depends on user privileges.	Public/private key pair based authentication and mapping a user to an account. Limited support for privacy.	Each user/application is provided with a virtual machine. High security/privacy is guaranteed. Support for setting per-file access control list (ACL).
Discovery	Membership services	Centralised indexing and decentralised info services	Membership services
Service Negotiation	Limited	Yes, SLA based	Yes, SLA based
User Management	Centralised	Decentralised and also Virtual Organization (VO)-based	Centralised or can be delegated to third party
Resource Management	Centralized	Distributed	Centralized/Distributed
Allocation / Scheduling	Centralised	Decentralised	Both centralised/decentralised
Standards / Inter-Operability	Virtual Interface Architecture (VIA)-based	Some Open Grid Forum standards	Web Services (SOAP and REST)
Single System Image	Yes	No	Yes, but optional
Capacity	Stable and guaranteed	Varies, but high	Provisioned on demand
Failure Management (Self-healing)	Limited (often failed tasks/applications are restarted).	Limited (often failed tasks/applications are restarted).	Strong support for failover and content replication. VMs can be easily migrated from one node to other.
Pricing of Services	Limited, not open market	Dominated by public good or privately assigned	Utility pricing, discounted for larger customers
Internetworking	Multi-clustering within an Organization	Limited adoption, but being explored through research efforts such as Gridbus InterGrid	High potential, third party solution providers can loosely tie together services of different Clouds
Application Drivers	Science, business, enterprise computing, data centers	Collaborative scientific and high throughput computing applications	Dynamically provisioned legacy and web applications, Content delivery
Potential for Building 3rd Party or Value-added Solutions	Limited due to rigid architecture	Limited due to strong orientation for scientific computing	High potential – can create new services by dynamically provisioning of compute, storage, and application services and offer as their own isolated or composite Cloud services to users

3.3 Web Search Trends

The popularity of different paradigms varies with time. The web search popularity, as measured by the Google search trends during the last 12 months, for terms “cluster computing”, “Grid computing”, and “Cloud computing” is shown in Figure 2. From the Google trends, it can be observed that cluster computing was a popular term during 1990s, from early 2000 Grid computing become popular, and recently Cloud computing started gaining popularity.



Legend: **Cluster Computing**, **Grid Computing**, **Cloud Computing**

Figure 2: Google search trends for the last 12 months.

Spot points in Figure 2 indicate the release of news related to Cloud computing as follows:

A IBM Introduces 'Blue Cloud' Computing, CIO Today - Nov 15 2007.

B IBM, EU Launch RESERVOIR Research Initiative for Cloud Computing, IT News Online - Feb 7 2008.

C Google and Salesforce.com in Cloud computing deal, Siliconrepublic.com - Apr 14 2008.

D Demystifying Cloud Computing, Intelligent Enterprise - Jun 11 2008.

E Yahoo realigns to support Cloud computing, 'core strategies', San Antonio Business Journal - Jun 27 2008.

F Merrill Lynch Estimates "Cloud Computing" To Be $100 Billion Market, SYS-CON Media - Jul 8 2008.

Other more recent news includes the following:

- Yahoo, Intel and HP form Cloud computing labs, Reseller News - Jul 29 2008.
- How Cloud Computing Is Changing The World, Pittsburgh Channel.com - Aug 4 2008.
- SIMtone Corporation Takes Cloud Computing to the Next Level with Launch of First Wireless, "Zero-Touch" Universal Cloud Computing Terminal, TMCnet - Sep 8 2008.

4. Market-Oriented Cloud Architecture

As consumers rely on Cloud providers to supply more of their computing needs, they will require specific QoS to be maintained by their providers in order to meet their objectives and sustain their operations. Cloud providers will need to consider and meet different QoS parameters of each individual consumer as negotiated in specific SLAs. To achieve this, Cloud providers can no longer continue to deploy traditional system-centric resource management architecture that do not provide incentives for them to share their resources and still regard all service requests to be

of equal importance. Instead, market-oriented resource management [11] is necessary to regulate the supply and demand of Cloud resources to achieve market equilibrium (where supply = demand), providing feedback in terms of economic incentives for both Cloud consumers and providers, and promoting QoS-based resource allocation mechanisms that differentiate service requests based on their utility. In addition, clients can benefit from the "potential" cost reduction of providers, which could lead to a more competitive market and thus lower prices.

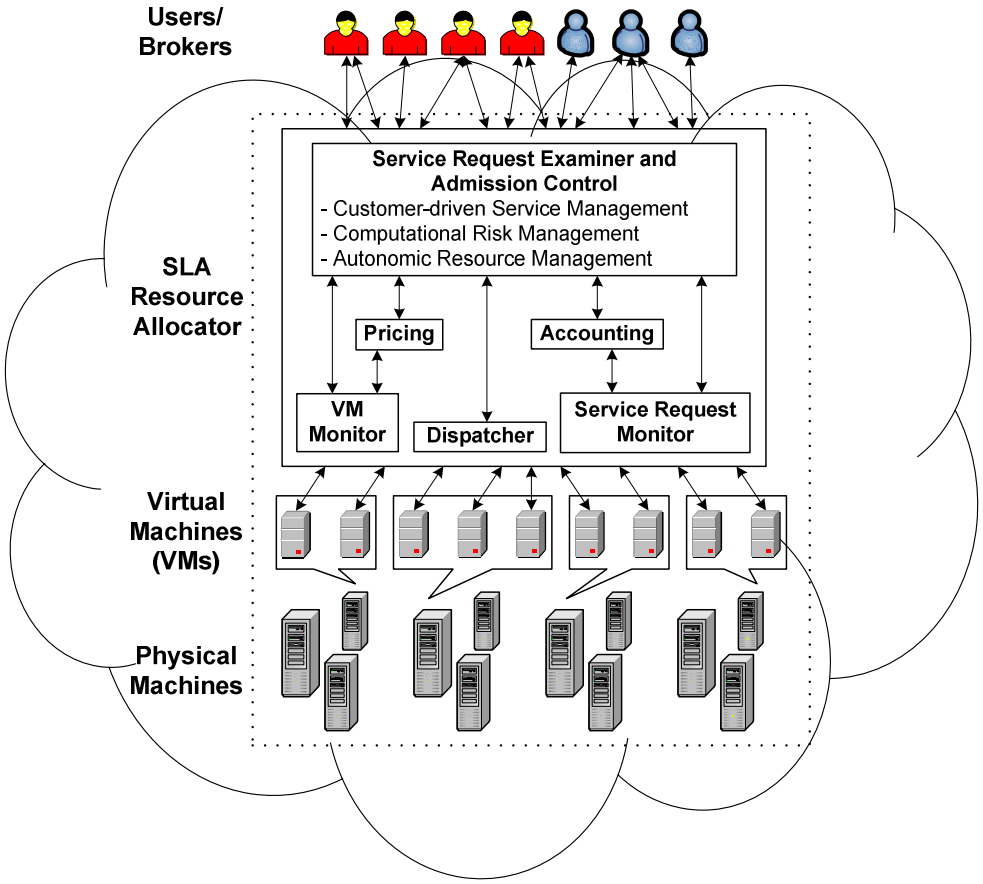


Figure 3: High-level market-oriented Cloud architecture.

Figure 3 shows the high-level architecture for supporting market-oriented resource allocation in Data Centers and Clouds. There are basically four main entities involved:

- ***Users/Brokers***: Users or brokers acting on their behalf submit service requests from anywhere in the world to the Data Center and Cloud to be processed.
- ***SLA Resource Allocator***: The SLA Resource Allocator acts as the interface between the Data Center/Cloud service provider and external users/brokers. It requires the interaction of the following mechanisms to support SLA-oriented resource management:
    - *Service Request Examiner and Admission Control*: When a service request is first submitted, the Service Request Examiner and Admission Control mechanism interprets the submitted request for QoS requirements before determining whether to accept or reject the request. Thus, it ensures that there is no overloading of resources whereby many service requests cannot be fulfilled successfully due to limited resources available. It also needs the latest status information regarding resource availability (from the VM Monitor mechanism) and workload processing (from the Service Request Monitor mechanism) in order to make resource allocation decisions effectively.

Then, it assigns requests to VMs and determines resource entitlements for allocated VMs.

- *Pricing*: The Pricing mechanism decides how service requests are charged. For instance, requests can be charged based on submission time (peak/off-peak), pricing rates (fixed/changing) or availability of resources (supply/demand). Pricing serves as a basis for managing the supply and demand of computing resources within the Data Center and facilitates in prioritizing resource allocations effectively.
- *Accounting*: The Accounting mechanism maintains the actual usage of resources by requests so that the final cost can be computed and charged to the users. In addition, the maintained historical usage information can be utilized by the Service Request Examiner and Admission Control mechanism to improve resource allocation decisions.
- *VM Monitor*: The VM Monitor mechanism keeps track of the availability of VMs and their resource entitlements.
- *Dispatcher*: The Dispatcher mechanism starts the execution of accepted service requests on allocated VMs.
- *Service Request Monitor*: The Service Request Monitor mechanism keeps track of the execution progress of service requests.

- ***VMs***: Multiple VMs can be started and stopped on-demand on a single physical machine to meet accepted service requests, hence providing maximum flexibility to configure various partitions of resources on the same physical machine to different specific requirements of service requests. In addition, multiple VMs can concurrently run applications based on different operating system environments on a single physical machine since every VM is completely isolated from one another on the same physical machine.
- ***Physical Machines***: The Data Center comprises multiple computing servers that provide resources to meet service demands.

In the case of a Cloud as a commercial offering to enable crucial business operations of companies, there are critical QoS parameters to consider in a service request, such as time, cost, reliability and trust/security. In particular, QoS requirements cannot be static and may change over time due to continuing changes in business operations and operating environments. In short, there should be greater importance on customers since they pay for accessing services in Clouds. In addition, the state-of-the-art in Cloud computing has no or limited support for dynamic negotiation of SLAs between participants and mechanisms for automatic allocation of resources to multiple competing requests. Recently, we have developed negotiation mechanisms based on alternate offers protocol for establishing SLAs [12]. These have high potential for their adoption in Cloud computing systems built using VMs.

Commercial offerings of market-oriented Clouds must be able to:

- Support customer-driven service management based on customer profiles and requested service requirements,
- Define computational risk management tactics to identify, assess, and manage risks involved in the execution of applications with regards to service requirements and customer needs,
- Derive appropriate market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain SLA-oriented resource allocation,
- Incorporate autonomic resource management models that effectively self-manage changes in service requirements to satisfy both new service demands and existing service obligations, and
- Leverage VM technology to dynamically assign resource shares according to service requirements.

5. Resource Management Strategies for Market-Oriented Clouds

Since customer satisfaction is the crucial success factor to excel in the service industry [13], computing service providers have to be aware of user-centric objectives and meet them in order to achieve customer satisfaction. But, many service quality factors can influence customer satisfaction [13][14]. Hence, we need to design SLA-oriented resource management strategies for Data Centers and Clouds that provide personalized attention to customers, such as enabling communication to keep customers informed and obtain feedback from them, increasing access and

approachability to customers, and understanding specific needs of customers. These strategies can also encourage trust and confidence in customers by emphasizing on the security measures undertaken against risks and doubts, the credibility of the provider, and the courtesy towards customers.

Our initial work [15] has also presented examples of how various elements of utility-based resource management can be perceived as risks and hence identified risk analysis from the field of economics as a probable solution to evaluate them. However, the entire risk management process [16][17] comprises of many steps and needs to be studied thoroughly so as to fully realize its effectiveness in managing risks. Hence, we need to first establish the context of risk management in Data Centers and Clouds, and then identify the risks involved. Each of the identified risks will be thoroughly assessed, before deriving appropriate strategies to manage these risks.

In addition, service requirements of users can change over time and thus may require amendments of original service requests. As such, our proposed resource management strategies will be able to self-manage the reservation process continuously by monitoring current service requests, amending future service requests, and adjusting schedules and prices for new and amended service requests accordingly. Hence, we need to investigate self-configuring components to satisfy new service requirements, so that more autonomic and intelligent Data Centers and Clouds can better manage the limited supply of resources with dynamically changing service demand. For users, there can be brokering systems acting on their behalf to select suitable providers and negotiate with them to achieve ideal service contracts. Thus, providers also require autonomic resource management to selectively choose appropriate requests to accept and execute depending on a number of operating factors, such as the expected availability and demand of services (both current and future), and existing service obligations.

Recently, virtualization [18] has enabled the abstraction of computing resources such that a single physical machine is able to function as a set of multiple logical VMs. A key benefit of VMs is the ability to host multiple operating system environments which are completely isolated from one another on the same physical machine. Another benefit is the capability to configure VMs to utilize different partitions of resources on the same physical machine. For example, on a physical machine, one VM can be allocated 10% of the processing power, while another VM can be allocated 20% of the processing power. Hence, we need to leverage existing VM technologies so that VMs can be started and stopped dynamically to meet the changing demand of resources by users as opposed to limited resources on a physical machine. In particular, we need to investigate how VMs can be assigned various resource management policies catering to different user needs and demands to better support the implementation of SLA-oriented resource allocation for Data Centers and Clouds.

6. Global Cloud Exchanges and Markets

Enterprises currently employ Cloud services in order to improve the scalability of their services and to deal with bursts in resource demands. However, at present, service providers have inflexible pricing, generally limited to flat rates or tariffs based on usage thresholds, and consumers are restricted to offerings from a single provider at a time. Also, many providers have proprietary interfaces to their services thus restricting the ability of consumers to swap one provider for another.

For Cloud computing to mature, it is required that the services follow standard interfaces. This would enable services to be commoditized and thus, would pave the way for the creation of a market infrastructure for trading in services. An example of such a market system, modeled on real-world exchanges, is shown in Figure 4. The market directory allows participants to locate providers or consumers with the right offers. Auctioneers periodically clear bids and asks received from market participants. The banking system ensures that financial transactions pertaining to agreements between participants are carried out.

Brokers perform the same function in such a market as they do in real-world markets: they mediate between consumers and providers by buying capacity from the provider and sub-leasing these to the consumers. A broker can accept requests from many users who have a choice of submitting their requirements to different brokers. Consumers, brokers and providers are bound to their requirements and related compensations through SLAs. An SLA specifies the details of the service to be provided in terms of metrics agreed upon by all parties, and penalties for meeting and violating the expectations, respectively.

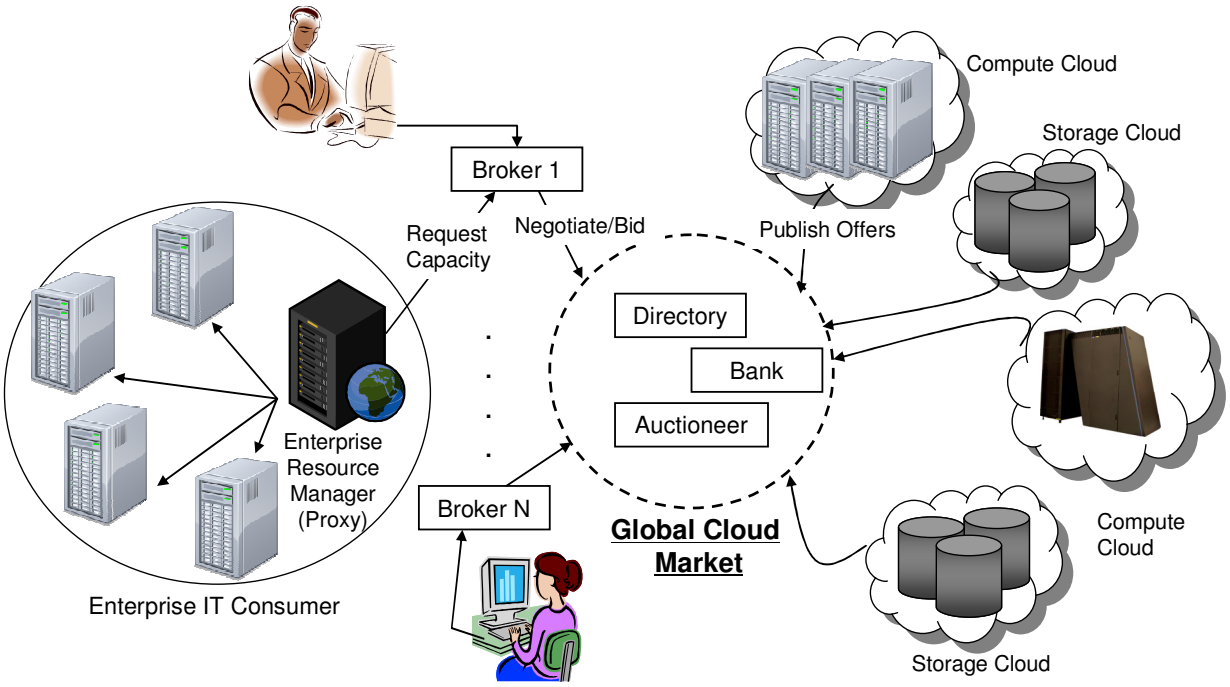


Figure 4: Global Cloud exchange and market infrastructure for trading services.

Such markets can bridge disparate Clouds allowing consumers to choose a provider that suits their requirements by either executing SLAs in advance or by buying capacity on the spot. Providers can use the markets in order to perform effective capacity planning. A provider is equipped with a price-setting mechanism which sets the current price for the resource based on market conditions, user demand, and current level of utilization of the resource. Pricing can be either fixed or variable depending on the market conditions. An admission-control mechanism at a provider's end selects the auctions to participate in or the brokers to negotiate with, based on an initial estimate of the utility. The negotiation process proceeds until an SLA is formed or the participants decide to break off. These mechanisms interface with the resource management systems of the provider in order to guarantee the allocation being offered or negotiated can be reclaimed, so that SLA violations do not occur. The resource management system also provides functionalities such as advance reservations that enable guaranteed provisioning of resource capacity.

Brokers gain their utility through the difference between the price paid by the consumers for gaining resource shares and that paid to the providers for leasing their resources. Therefore, a broker has to choose those users whose applications can provide it maximum utility. A broker interacts with resource providers and other brokers to gain or to trade resource shares. A broker is equipped with a negotiation module that is informed by the current conditions of the resources and the current demand to make its decisions.

Consumers have their own utility functions that cover factors such as deadlines, fidelity of results, and turnaround time of applications. They are also constrained by the amount of resources that they can request at any time, usually by a limited budget. Consumers also have their own limited IT infrastructure that is generally not completely exposed to the Internet. Therefore, a consumer participates in the utility market through a resource management proxy that selects a set of brokers based on their offerings. He then forms SLAs with the brokers that bind the latter to provide the guaranteed resources. The enterprise consumer then deploys his own environment on the leased resources or uses the provider's interfaces in order to scale his applications.

The idea of utility markets for computing resources has been around for a long time. Recently, many research projects such as SHARP [19], Tycoon [20], Bellagio [21], and Shirako [22] have come up with market structures for trading in resource allocations. These have particularly focused on trading in VM-based resource slices on networked infrastructures such as PlanetLab. The Gridbus project has created a resource broker that is able to negotiate with resource providers. Thus, the technology for enabling utility markets is already present and ready to be deployed.

However, significant challenges persist in the universal application of such markets. Enterprises currently employ conservative IT strategies and are unwilling to shift from the traditional controlled environments. Cloud computing uptake has only recently begun and many systems are in the proof-of-concept stage. Regulatory pressures also mean that enterprises have to be careful about where their data gets processed, and therefore, are not able to employ Cloud services from an open market. This could be mitigated through SLAs that specify strict constraints on the location of the resources. However, another open issue is how the participants in such a market can obtain restitution in case an SLA is violated. This motivates the need for a legal framework for agreements in such markets, a research issue that is out of scope of themes pursued in this paper.

7. Emerging Cloud Platforms

Industry analysts have made bullish projections on how Cloud computing will transform the entire computing industry. According to a recent Merrill Lynch research note [23], Cloud computing is expected to be a "$160-billion addressable market opportunity, including $95-billion in business and productivity applications, and another $65-billion in online advertising". Another research study by Morgan Stanley [24] has also identified Cloud computing as one of the prominent technology trends. As the computing industry shifts toward providing Platform as a Service (PaaS) and Software as a Service (SaaS) for consumers and enterprises to access on demand regardless of time and location, there will be an increase in the number of Cloud platforms available. Recently, several academic and industrial organizations have started investigating and developing technologies and infrastructure for Cloud Computing. Academic efforts include Virtual Workspaces [25], OpenNebula [26], and Reservoir [27]. In this section, we compare six representative Cloud platforms with industrial linkages in Table 2.

Amazon Elastic Compute Cloud (EC2) [28] provides a virtual computing environment that enables a user to run Linux-based applications. The user can either create a new Amazon Machine Image (AMI) containing the applications, libraries, data and associated configuration settings, or select from a library of globally available AMIs. The user then needs to upload the created or selected AMIs to Amazon Simple Storage Service (S3), before he can start, stop, and monitor instances of the uploaded AMIs. Amazon EC2 charges the user for the time when the instance is alive, while Amazon S3 [29] charges for any data transfer (both upload and download).

Google App Engine [30] allows a user to run web applications written using the Python programming language. Other than supporting the Python standard library, Google App Engine also supports Application Programming Interfaces (APIs) for the datastore, Google Accounts, URL fetch, image manipulation, and email services. Google App Engine also provides a web-based Administration Console for the user to easily manage his running web applications. Currently, Google App Engine is free to use with up to 500MB of storage and about 5 million page views per month.

Microsoft Azure [31] aims to provide an integrated development, hosting, and control Cloud computing environment so that software developers can easily create, host, manage, and scale both Web and non-web applications through Microsoft data centers. To achieve this aim, Microsoft Azure supports a comprehensive collection of proprietary development tools and protocols which consists of Live Services, Microsoft .NET Services, Microsoft SQL Services, Microsoft SharePoint Services, and Microsoft Dynamics CRM Services. Microsoft Azure also supports Web APIs such as SOAP and REST to allow software developers to interface between Microsoft or non-Microsoft tools and technologies.

Sun network.com (Sun Grid) [32] enables the user to run Solaris OS, Java, C, C++, and FORTRAN based applications. First, the user has to build and debug his applications and runtime scripts in a local development environment that is configured to be similar to that on the Sun Grid. Then, he needs to create a bundled zip archive (containing all the related scripts, libraries, executable binaries and input data) and upload it to Sun Grid. Finally, he can execute and monitor the application using the Sun Grid web portal or API. After the completion of the application, the user will need to download the execution results to his local development environment for viewing.

Table 2: Comparison of Some Representative Cloud Platforms.

System / Property	Amazon Elastic Compute Cloud (EC2)	Google App Engine	Microsoft Azure	Sun Network.com (Sun Grid)	GRIDS Lab Aneka
Focus	Infrastructure	Platform	Platform	Infrastructure	Software platform for enterprise Clouds
Service Type	Compute, Storage (Amazon S3)	Web application	Web and non-web application	Compute	Compute
Virtualization	OS level running on a Xen hypervisor	Application container	OS level through Fabric Controller	Job management system (Sun Grid Engine)	Resource manager and scheduler
Dynamic Negotiation of QoS Parameters	None	None	None	None	SLA-based resource reservation on Aneka side.
User Access Interface	Amazon EC2 Command-line Tools	Web-based Administration Console	Microsoft Windows Azure portal	Job submission scripts, Sun Grid web portal	Workbench, web-based portal
Web APIs	Yes	Yes	Yes	Yes	Yes
Value-added Service Providers	Yes	No	Yes	Yes	No
Programming Framework	Customizable Linux-based Amazon Machine Image (AMI)	Python	Microsoft .NET	Solaris OS, Java, C, C++, FORTRAN	APIs supporting different programming models in C# and other .Net supported languages

Aneka [33], which is being commercialized through Manjrasoft, is a .NET-based service-oriented resource management platform. It is designed to support multiple application models, persistence and security solutions, and communication protocols such that the preferred selection can be changed at anytime without affecting an existing Aneka ecosystem. To create an Aneka Cloud, the service provider only needs to start an instance of the configurable Aneka container hosting required services on each selected desktop computer. The purpose of the Aneka container is to initialize services and acts as a single point for interaction with the rest of the Aneka Cloud. Aneka provides SLA support such that the user can specify QoS requirements such as deadline (maximum time period which the application needs to be completed in) and budget (maximum cost that the user is willing to pay for meeting the deadline). The user can access the Aneka Cloud remotely through the Gridbus broker. The Gridbus broker [34] also enables the user to negotiate and agree upon the QoS requirements to be provided by the service provider.

8. Aneka: From Enterprise Grids to Market-Oriented Cloud Computing

We are working towards implementing a market-oriented Cloud using a .NET-based service-oriented resource management platform called Aneka [33]. Aneka is initially developed as a 3rd generation enterprise Grid technology. Recently, various new capabilities have been added to exhibit properties and potentials of the Cloud computing paradigm. An enterprise Grid [35] harnesses computing resources of desktop computers (connected over an internal network or the Internet) within an enterprise without affecting the productivity of their users. Hence, it increases the amount of computing resources available within an enterprise to accelerate application performance. This capability can be combined with other dedicated resources in the enterprise to enhance the overall system capability and reliability.

To support scalability, the Aneka container is designed to be lightweight by providing the bare minimum functionality needed for an Aneka Cloud node. It provides the base infrastructure that consists of services for persistence, security (authorization, authentication and auditing), and communication (message handling and dispatching). The Aneka container can host any number of optional services that can be added to augment the capabilities of an Aneka Cloud node. Examples of optional services are indexing, scheduling, execution, and storage services. This provides a single, flexible and extensible framework for orchestrating various application models. This section describes how pricing can be implemented in an Aneka Cloud with advanced reservations.

8.1 Market-Oriented Resource Pricing and Allocation in Aneka Cloud

To create an Aneka Cloud, we implement a bi-hierarchical advance reservation mechanism with a Reservation Service at a master node that coordinates multiple execution nodes and an Allocation Service at each execution node that keeps track of the reservations at that node. This architecture was previously introduced in [12]. To use the Aneka Cloud, the resource user (or a broker acting on its behalf) first makes advanced reservations during the reservation phase for resources required at a designated time in the future. If the reservation phase is successful, the user/broker can then submit applications later during the execution phase when the designated time in the future arrives.

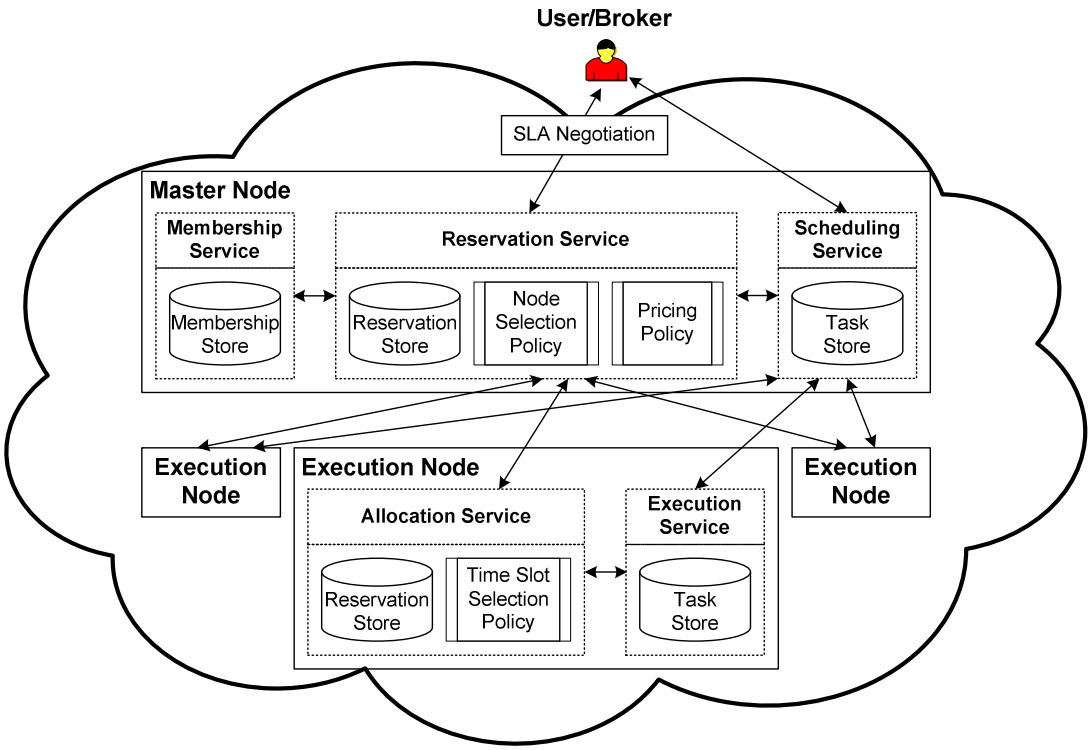


Figure 5: Interaction of services in Aneka Cloud environment.

Figure 5 shows that the process of allocating advanced reservations happens in two levels: the Allocation Service at each execution node and the Reservation Service at the master node. Both services are designed to support pluggable policies so that the provider has the flexibility to easily customize and replace existing policies for different levels and/or nodes without interfering with the overall resource management architecture.

During the reservation phase, the user/broker submits reservation requests through the Reservation Service at the master node. The Reservation Service discovers available execution nodes in the Aneka Cloud by interacting with the Allocation Service on them. The Allocation Service at each execution node keeps track of all reservations that have been confirmed for the node and can thus check whether a new request can be satisfied or not.

The Allocation Service determines how to schedule a new reservation at the execution node. For simplicity, we implement the same time slot selection policy for the Allocation Service at every execution node. The Allocation Service allocates the requested time slot if the slot is available. Otherwise, it assigns the next available time slot after the requested start time that can meet the required duration. By allocating reservations at each execution node instead of at the master node, computation overheads arising from making allocation decisions are distributed across multiple nodes and thus minimized, as compared to overhead accumulation at a single master node.

The Reservation Service performs node selection by choosing the required number of available time slots from execution nodes and administers admission control by accepting or rejecting a reservation request. It also calculates the price for a confirmed reservation based on the implemented pricing policy. Available time slots are selected taking into account the application requirement of the user.

The application requirement considered here is the task parallelism to execute an application. A sequential application has a single task and thus needs a single processor to run, while a parallel application needs a required number of processors to concurrently run at the same time.

For a sequential application, the selected time slots need not have the same start and end times. Hence, available time slots with the lowest prices are selected first. If there are multiple available time slots with the same price, then those with the earliest start time available are selected first. This ensures that the cheapest requested time slot is allocated first if it is available. Selecting available time slots with the lowest prices first is fair and realistic. In reality, reservations that are confirmed earlier enjoy the privilege of cheaper prices, as compared to reservation requests that arrive later.

But, for a parallel application, all the selected time slots must have the same start and end times. Again, the earliest time slots (with the same start and end times) are allocated first to ensure the requested time slot is allocated first if available. If there are more available time slots (with the same start and end times) than the required number of time slots, then those with the lowest prices are selected first.

The admission control operates according to the service requirement of the user. The service requirements examined are the deadline and budget to complete an application. We assume both deadline and budget are hard constraints. Hence, a confirmed reservation must not end after the deadline and cost more than the budget. Therefore, a reservation request is not accepted if there is insufficient number of available time slots on execution nodes that ends within the deadline and the total price of the reservation costs more than the budget.

During the execution phase, the user/broker submits applications to be executed to the Scheduling Service at the master node. The Scheduling Service first checks whether the submission satisfies the starting time, ending time, and duration of the reservation that have been specified by the user/broker during the reservation phase. If the reservation is still valid, the Scheduling Service then determines whether any of the reserved execution nodes are available before dispatching applications to them for execution, otherwise applications are queued to wait for the next available execution nodes that are part of the reservation. The Execution Service at each execution node starts executing an application after receiving it from the Scheduling Service and updates the Scheduling Service of changes in execution status. Hence, the Scheduling Service can monitor executions for an application and notify the user/broker upon completion.

8.2 Performance Evaluation

High-end computing systems such as Clouds are used for hosting applications containing short-lived or long-lived processing tasks. Applications providing Internet services often have short-lived tasks and are designed to serve millions of users concurrently. Examples include search engines (e.g. Google), e-commerce sites (e.g. Amazon.com online shopping store), and social networking sites (e.g. Facebook). Many business and scientific applications such as investment risk analysis, supply chain management, flight simulation, and molecular docking often contain tasks that are resource-intensive and long-lived. We will first present the performance of HPC workload (with long-lived tasks) in our Aneka enterprise Cloud environment. We will then discuss another performance study on Internet-based services workload (with short-lived tasks).

8.2.1 High Performance Computing (HPC) Workload

Figure 6 shows the Aneka enterprise Cloud environment setup used for performance evaluation. The Aneka Cloud contains 33 personal computers (PCs) with 1 master node and 32 execution nodes located across 3 student computer

laboratories in the Department of Computer Science and Software Engineering, The University of Melbourne. This setup demonstrates that the Aneka Cloud is able to present a unified resource to the users/brokers by harnessing computing resources located physically apart in the 3 laboratories.

Synthetic workloads are created by utilizing trace data of HPC applications. The experiments utilize 238 reservation requests in the last 7 days of the SDSC SP2 trace (April 1998 to April 2000) version 2.2 from Feitelson's Parallel Workloads Archive [36]. The SDSC SP2 trace from the San Diego Supercomputer Center (SDSC) in USA is chosen due to the highest resource utilization of 83.2% among available traces to ideally model a heavy workload scenario. However, the trace only provides the inter-arrival times of reservation requests, the number of processors to be reserved (downscaled from a maximum of 128 nodes in the trace to a maximum of 32 nodes), and the duration to be reserved. Hence, we adopt a methodology similar to that adopted by Irwin et al. [37] to synthetically assign service requirements (deadline and budget) through two request classes: (i) low urgency and (ii) high urgency.

A reservation request *i* in the *low urgency* class has a deadline of high $deadline_i$ / $duration_i$ value and budget of low $budget_i$ / $f(duration_i)$ value. $f(duration_i)$ is a function representing the minimum budget required based on $duration_i$. Conversely, each request *i* in the *high urgency* class has a deadline of low $deadline_i$ / $duration_i$ value and budget of high $budget_i$ / $f(duration_i)$ value. This is realistic since a user who submits a more urgent request to be met within a shorter deadline offers a higher budget for the short notice. Values are normally distributed within each of the deadline and budget parameters.

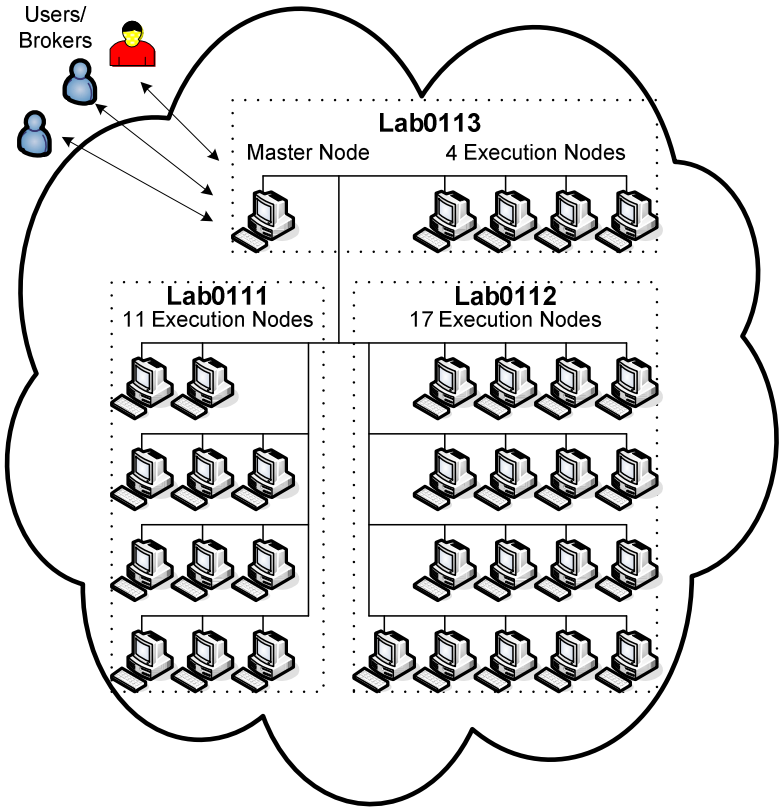


Figure 6: Configuration of Aneka Cloud.

For simplicity, the Aneka Cloud only charges users for utilizing the computing resource type based on per processor (CPU) per hour (Hr). Thus, users are not charged for using other resource types such as memory, storage, and bandwidth. In addition, we assume that every user/broker can definitely accept another reservation time slot proposed by the Aneka Cloud if the requested one is not possible, provided that the proposed time slot still satisfies both application and service requirements of the user.

As listed in Table 3, we evaluate the performance of seven pricing mechanisms representing three basic types: (i) Fixed, (ii) FixedTime, and (iii) Libra+$. Each of these three pricing mechanisms has a maximum and minimum configuration to highlight their performance range. The Fixed mechanism charges a fixed price at all times. The FixedTime mechanism charges a fixed price for different time periods of resource usage where a lower price is charged for off-peak (12AM—12PM) and a higher price for peak (12PM—12AM).

Table 3: Pricing Mechanisms.

Name	Configured Pricing Parameters
FixedMax	$3/CPU/Hr
FixedMin	$1/CPU/Hr
FixedTimeMax	$1/CPU/Hr (12AM–12PM) $3/CPU/Hr (12PM–12AM)
FixedTimeMin	$1/CPU/Hr (12AM–12PM) $2/CPU/Hr (12PM–12AM)
Libra+$Max	$1/CPU/Hr ($PBase_j$), $\alpha = 1$, $\beta = 3$
Libra+$Min	$1/CPU/Hr ($PBase_j$), $\alpha = 1$, $\beta = 1$
Libra+$Auto	same as Libra+$Min

Libra+$ [38] uses a more fine-grained pricing function that satisfies four essential requirements for pricing of resources to prevent workload overload: (i) flexible, (ii) fair, (iii) dynamic, and (iv) adaptive. The price P_{ij} for per unit of resource utilized by reservation request i at compute node j is computed as: $P_{ij} = (\alpha * PBase_j) + (\beta * PUtil_{ij})$. The base price $PBase_j$ is a static pricing component for utilizing a resource at node j which can be used by the service provider to charge the minimum price so as to recover the operational cost. The utilization price $PUtil_{ij}$ is a dynamic pricing component which is computed as a factor of $PBase_j$ based on the utilization of the resource at node j for the required deadline of request i: $PUtil_{ij} = RESMax_j / RESFree_{ij} * PBase_j$. $RESMax_j$ and $RESFree_{ij}$ are the maximum units and remaining free units of the resource at node j for the deadline duration of request i respectively. Thus, $RESFree_{ij}$ has been deducted units of resource committed for other confirmed reservations and request i for its deadline duration.

The factors α and β for the static and dynamic components of Libra+$ respectively provides the flexibility for the service provider to easily configure and modify the weight of the static and dynamic components on the overall price P_{ij}. Libra+$ is fair since requests are priced based on the amount of different resources utilized. It is also dynamic because the overall price of a request varies depending on the availability of resources for the required deadline. Finally, it is adaptive as the overall price is adjusted depending on the current supply and demand of resources to either encourage or discourage request submission.

However, these three mechanisms rely on static pricing parameters that are difficult to be accurately derived by the service provider to produce the best performance where necessary. Hence, we propose Libra+$Auto [39], an autonomic Libra+$ that automatically adjusts β based on the availability of compute nodes. Libra+$Auto thus considers the pricing of resources across nodes, unlike Libra+$ which only considers pricing of resources at each node j via P_{ij}.

Figure 7 shows the normalized pricing and revenue performance of seven pricing mechanisms in the Aneka Cloud for high urgency requests (with short deadline and high budget) from sequential applications (requiring one processor to execute) over a 7-days time period. In Figure 7, the two performance metrics are: (i) the price for a confirmed reservation (in $/CPU/Hr) and (ii) the accumulated revenue for confirmed reservations (in $). Both metrics have been normalized to produce standardized values within the range of 0 to 1 for easier comparison. The revenue of a confirmed reservation is the total sum of revenue across all its reserved nodes calculated using the assigned price (depending on the specific pricing mechanism) and the reserved duration at each node. Then, the price of a confirmed reservation is computed to reflect the average price across all its reserved nodes.

In Figure 7, out of the four fixed pricing mechanisms listed in Table 3, FixedMax provides the highest revenue (maximum bound), followed by FixedTimeMax, FixedTimeMin, and FixedMin with the lowest revenue (minimum bound). Nevertheless, FixedTime mechanisms is easier to derive and more reliable than Fixed mechanisms since it supports a range of prices across various time periods of resource usage. But, all four mechanisms do not consider service requirements of users such as deadline and budget.

On the other hand, Libra+$ charges a lower price for a request with longer deadline as an incentive to encourage users to submit requests with longer deadlines that are more likely to be accommodated than shorter deadlines. For a request with short deadline, Libra+$Max and Libra+$Min charge a higher price relative to their β in Table 3. Libra+$Max provides higher revenue than Libra+$Min due to a higher value of β.

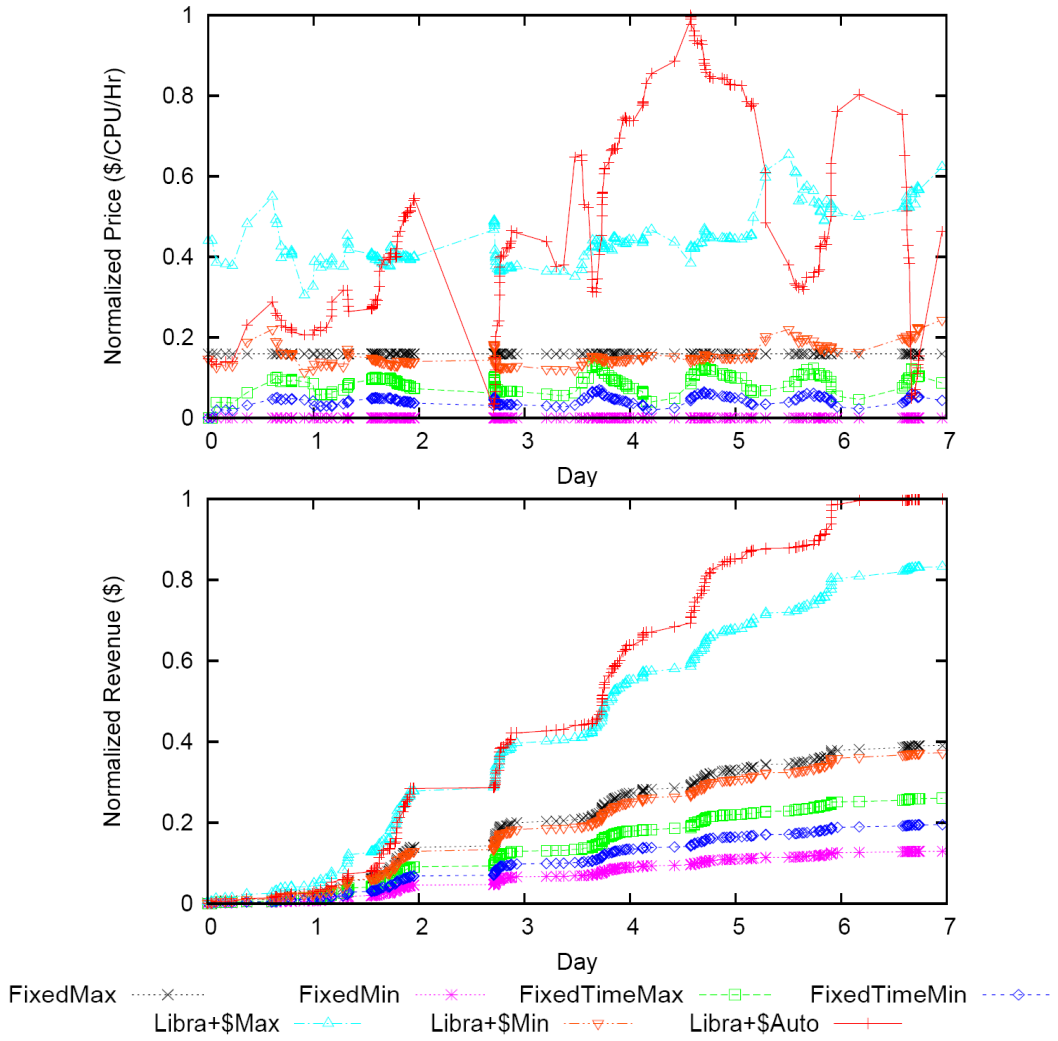


Figure 7: Normalized price/revenue of high urgency requests.

Both Libra+$Auto and Libra+$Max are able to provide a significantly higher revenue than other pricing mechanisms through higher prices for shorter deadlines. Figure 7 shows that Libra+$Auto continues increasing prices to higher than that of Libra+$Max and other pricing mechanisms when demand is high such as during the later half of day 1, 2, 3, and 5. But, when demand is low such as during the early half of day 2, 3, 5, and 6, Libra+$Auto keeps reducing prices to lower than that of Libra+$Max to accept requests that are not willing to pay more. Hence, Libra+$Auto is able to exploit budget limits to achieve the highest revenue by automatically adjusting to a higher β to increase prices when the availability of nodes is low and a lower β to reduce prices when there are more unused nodes which will otherwise be wasted.

8.2.2 Internet-based Services Workload

MapReduce [40] is one of the most popular programming models designed for data centers. It was originally

proposed by Google to handle large-scale web search applications and has been proved to be an effective programming model for developing data mining, machine learning and search applications in data centers. In particular, MapReduce can enhance the productivity for junior developers who lack the experience of distributed/parallel development. This model is implemented in other distributed computing systems such as Yahoo's Hadoop [41] and our Aneka [42]. Hadoop has been successfully used by many companies [43] including AOL, Amazon, Facebook, and New York Times for running their applications on clusters. For example, AOL used it for running an application that analyzes the behavioral pattern of their users so as to offer targeted services.

Although Hadoop is successful in homogeneous computing environments, a performance study conducted by Matei Zaharia et al. [44] shows that MapReduce implemented in the standard distribution of Hadoop is unable to perform well in heterogeneous Cloud computing infrastructure such as Amazon EC2 [28]. Experimental observations reveal that the homogeneity assumptions of MapReduce can cause wrong and often unnecessary speculative execution in heterogeneous environments, sometimes resulting in even worse performance than with speculation disabled. This evaluation and performance results of their enhanced scheduler in Hadoop demonstrate that Cloud execution management systems need to be designed to handle heterogeneity that is present in workloads, applications, and computing infrastructure.

9. Meta-Negotiation Infrastructure between Aneka Clouds and Brokers

The *Meta-Negotiation Middleware (MNM)* represents the first implementation prototype for the establishment of global Cloud exchange and market infrastructure for trading services. The MNM bridges the gap between different proprietary service interfaces and diverse negotiation strategies used by service providers and consumers [45].

Before committing themselves to a SLA, the consumer and provider may enter into negotiations that determine the definition and measurement of user QoS parameters, and the rewards and penalties for meeting and violating them respectively [46][47]. The term *negotiation strategy* represents the logic used by a partner to decide which provider or consumer satisfies his needs best. A *negotiation protocol* represents the exchange of messages during the negotiation process. Recently, many researchers have proposed different protocols and strategies for SLA negotiation in Grids [12][48]. However, these not only assume that the parties involved in the negotiation understand a common protocol but also assume that they share a common perception about the goods or services under negotiation. But, in reality, a participant may prefer to negotiate using certain protocols (for which it has developed better strategies) over others.

As shown in Figure 8, each meta-negotiation is defined by the means of a *meta-negotiation document* which participating parties may express: the pre-requisites to be satisfied for a negotiation, such as a specific authentication method required or terms they want to negotiate on (e.g. time, price, reliability) (see lines 2–9); the negotiation protocols and document languages for the specification of SLAs (e.g. Web Service Level Agreement (WSLA) [49] or WS-Agreement [50]) that they support (see lines 11–15); and conditions for the establishment of an agreement, such as a required third-party arbitrator (see lines 16–18).

```
 1. <meta-negotiation xsi:noNamespaceSchemaLocation="...">
 2.   <pre-requisite>
 3.     <security>
 4.       <authentication value="GSI location="uri"/>
 5.     </security>
 6.     <negotiation-terms>
 7.       <negotiation-term name="price"/>
 8.     </negotiation-terms>
 9.     ...
10.   </pre-requisite>
11.   <negotiation>
12.     <document name="WSLA" value="u < ri" version="1.0"/>
13.     <protocol name="alternateOffers" schema="uri" version="1.0" location="uri"/>
14.     ...
15.   </negotiation>
16.   <agreement>
17.     <confirmation name="arbitrationService" value="uri"/>
18.   </agreement>
19. </meta-negotiation>
```

Figure 8: Meta-negotiation document.

Before entering a meta-negotiation, a service provider publishes descriptions and conditions of supported negotiation protocols in the registry. Thereafter, service consumers perform lookup on the registry database by submitting their own documents describing the negotiations that they are looking for. The registry discovers service providers who support the negotiation processes that a consumer is interested in and returns the documents published by the service providers. Finally, after an appropriate service provider and a negotiation protocol are selected by a consumer using his/her private selection strategy, negotiations between them may start according to the conditions specified in the provider's document.

In our meta-negotiation architecture (as shown in Figure 9), the service provider role is carried out by Aneka which is a .NET-based service-oriented resource management system. The Gridbus Broker acts as a service consumer and maps jobs to appropriate resources by considering various restrictions specified by terms of *functional* and *non-functional* requirements. *Functional requirements* include but are not limited to task and data dependencies, such as a sequence of tasks required to execute a specific application. *Non-functional requirements* include QoS parameters, such as budget restrictions and deadlines for execution. The broker can guarantee the end-user's deadline requirement only if it is able to reserve nodes on resources in advance. Therefore, in this respect, the broker functions as a consumer that requests reservations from the provider.

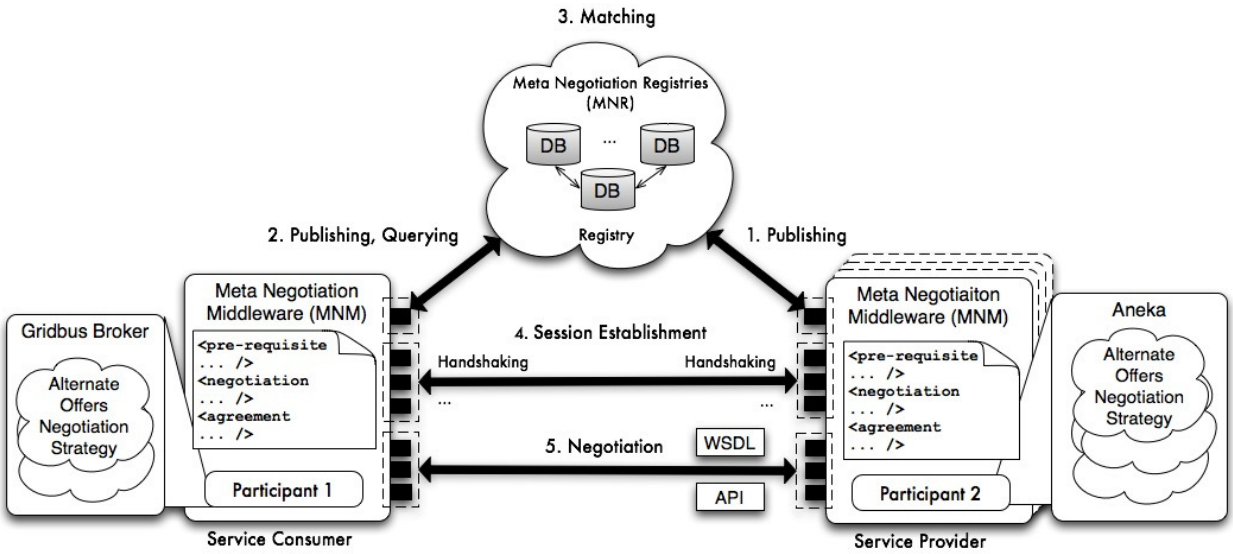


Figure 9: Architecture for meta-negotiations between Aneka Clouds and clients (e.g. Gridbus Broker).

The registry is a searchable repository for meta-negotiation documents that are created by the participants. Currently, this is implemented as a PostgreSQL database with a web service front end that provides the interface to access the registry. However, it is possible to host the registry using a Cloud of databases hosted on a service provider such as Amazon S3 [29] or Google App Engine [30]. When a meta-negotiation document is published, the registry assigns it a unique identifier (ID) that can then be used for subsequent operations. The query call tries to find all the documents in the repository that match closely to the document supplied as the parameter. It returns an array of IDs of these documents to the caller which can then fetch each one.

The MNM facilitates the publishing of the meta-negotiation documents into the registry and the integration of the meta-negotiation framework into the existing client and/or service infrastructure, such as negotiation or security clients. Besides being a client for publishing and querying meta-negotiation documents (Steps 1 and 2 in Figure 9), the MNM delivers necessary information for existing negotiation clients, i.e. information for the establishment of the negotiation sessions (Step 4) and necessary information to start a negotiation (Step 5). As shown in Figure 9, each service consumer may negotiate with multiple service providers concurrently. The reverse may also happen when a consumer advertises a job. In such cases, the providers would negotiate with multiple consumers.

After querying the registry and applying a client-based strategy for the selection of the appropriate service, the

information from the service's meta-negotiation document is parsed. Thereafter, the meta-negotiation information is incorporated into the existing client software using a dependency injection framework such as Spring [51]. This dependency injection follows an Inversion of Control approach wherein the software is configured at runtime to invoke services that are discovered dynamically, rather than known and referenced beforehand. This is suitable in the context of meta-negotiation wherein a participant discovers others at runtime through the registry and has to dynamically adapt based on the interfaces provided by his counterpart (usually through a WSDL document).

10. Creating 3rd Party Cloud Services: MetaCDN – A Case Study of High Performance Content Delivery over Commercial Cloud Storage Services

Content Delivery Network (CDN) providers such as Akamai [52][53] and Mirror Image [54] place web server clusters across the globe in order to improve the responsiveness and locality of the replicated content it hosts for end-users. However, their services are priced out of reach for all but the largest enterprise customers, and typically requiring lengthy contracts and large usage commitments [55]. We have developed an alternative approach to content delivery that leverages existing infrastructure provided by 'Storage Cloud' providers at a fraction of the cost of traditional CDN providers such as Akamai and Mirror Image, with no ongoing contract or usage commitments.

MetaCDN is a system that harnesses global 'Storage Cloud' resources, creating an integrated overlay network that offers a low cost, high performance CDN for content creators. MetaCDN hides the complexity of interacting with multiple storage providers, by intelligently matching and placing users' content onto one or many storage providers based on their quality of service, coverage and budget preferences. A single namespace is provided by MetaCDN that spans all supported storage providers, making it trivial for content creators to integrate into their origin websites, and allows end-users to consume content in a transparent and fast manner. The utility of this new approach to content delivery has been demonstrated by showing that the MetaCDN system (and the participating 'Storage Clouds' used) provides high performance, in terms of throughput and response time, and reliable content delivery for content consumers [56].

Web applications (such as content delivery and video streaming) are particularly well suited to the "Cloud" paradigm. The load on these applications are typically bursty, with periods of low load contrasting with occasional flash crowds caused by an important event (i.e. a large sporting competition of great interest [57][58]) or an unexpected phenomena (such as 9/11 [59] or the 2004 Tsunami) that draws large volumes of traffic to specific websites for the latest information, potentially crippling them in the process. These applications are a perfect fit to be deployed on Cloud Computing infrastructure, which multiplex many users' applications on top of their vast resources, and allow them to expand and contract their resource requirements in a dynamic fashion to address sudden increases in demand or quieter periods of operation. Many websites have successfully utilised individual Storage Clouds to deliver some or all of their content [60], most notably the New York Times [61] and SmugMug [62]. However, there is no general purpose, reusable framework to interact with multiple Storage Cloud providers and leverage their services as a CDN. This gap in functionality has thus motivated the development of the MetaCDN system (described in the next section).

10.1 'Storage Clouds' used by MetaCDN

In recent years, many 'Storage Cloud' providers (or 'Storage as a Service') that provide Internet-enabled content storage and delivery capabilities in several continents, have emerged offering SLA-backed performance and uptime promises for their services. These services follow a *utility computing* [63] model, where customers are charged only for their utilization of storage and transfer of content, with pricing in the order of cents per gigabyte (GB), as depicted in Table 4. This is in stark contrast to typical hosting arrangements that were commonplace in the past, where customers were locked into contracts (with set monthly/yearly fees and excess data charges) on shared hosting services like DreamHost [64]. High–end CDNs like Akamai and Mirror Image, who operate extensive networks of 'edge' servers that deliver content across the globe, were typically utilized by larger enterprise customers who had the means to afford them.

As such, the notion of leveraging Storage Clouds as a low cost CDN is very appealing for smaller organizations. These storage providers offer the ability to rapidly and cheaply scale-out to meet both flash crowds and anticipated or cyclical increases in demand. The most prominent Storage Cloud providers are Amazon Simple Storage Service (S3) [29], and Nirvanix Storage Delivery Network (SDN) [65]. Amazon currently offers storage nodes in the United

States and Europe (specifically, Ireland), whilst Nirvanix has storage nodes in the United States (over two separate sites in California), Germany and Singapore. Another major Storage Cloud provider of note is Mosso CloudFS [66], located in Dallas, Texas, USA, which is expected to launch in the fourth quarter of 2008, and as such has not yet released the full details of their service offerings.

Amazon S3 was launched in the United States in March 2006, and in Europe in November 2007, making available the large infrastructure that they utilize to operate Amazon.com, their highly successful e-commerce company. Amazon provides programmatic access to their storage resources via REST and SOAP interfaces, allowing users the ability to read, write or delete an unlimited amount of objects. S3 users can access storage objects with sizes ranging from 1 byte to 5 GB each. As noted in Table 4, Amazon S3 has a storage cost of $0.15 per GB/month in their USA data center, or $0.18 per GB/month in their EU data center, reflecting the cost of storage and transit in these respective regions. Incoming traffic (i.e uploads) are charged at $0.10 per GB/month, and outgoing traffic (i.e. downloads) are charged at $0.17 per GB/month, from the USA or EU site. For larger customers, Amazon S3 has a sliding scale pricing scheme for its larger customers, with discounts for outgoing data occurring after 10TB, 50TB, and 150TB of data a month has been transferred. Amazon imposes an additional cost per 1,000 PUT/POST/LIST or 10,000 GET HTTP requests, which can add up depending on the type of content a user places on Amazon S3. Indeed, if you are primarily storing and serving a large number of smaller files, you could see significant extra costs on your bill, however these costs are negligible if you are utilizing Amazon S3 to predominantly distribute very large files.

Table 4: Feature and Pricing Comparison of Storage Clouds.

Provider / Feature	Nirvanix US/EU	Nirvanix SDN	Amazon S3 US	Amazon S3 EU	Mosso CloudFS
Incoming Data ($/GB/month)	0.18	0.18	0.10	0.10	Unknown
Outgoing data ($/GB/month)	0.18	0.18	0.17	0.17	Unknown
Storage ($/GB/month)	0.18	0.25	0.15	0.18	0.15
Requests ($/1,000 PUT)	0.00	0.00	0.01	0.012	Unknown
Requests ($/10,000 GET)	0.00	0.00	0.01	0.012	Unknown
SLA	99.9	99.9	99-99.9	99-99.9	Unknown
Max. File Size	256 GB	256 GB	5 GB	5 GB	5 GB
US PoP	Yes	Yes	Yes	N/A	Yes
EU PoP	Yes	Yes	N/A	Yes	No
Asia PoP	No	Yes	No	No	No
Australasia PoP	No	No	No	No	No
Automatic Replication	Yes	No	Yes	No	No
Developer API	Yes	Yes	Yes	Yes	Yes

The Nirvanix Storage Delivery Network (SDN) was launched in September 2007, offering an alternative Storage Cloud to the Amazon S3 service. The Nirvanix service distinguished itself by offering a SLA backed uptime guarantee at a time when Amazon S3 was simply operated on a best-effort service basis. Once Nirvanix launched its SDN, Amazon subsequently added their own SLA-backed uptime guarantees. Nirvanix differentiates itself in several ways (depicted in Table 4), notably by having coverage in more locations (four at the time of writing), offering automatic file replication over two or more sites in the SDN for performance and redundancy, and supporting individual file sizes up to 256 GB. Nirvanix is priced slightly higher than Amazon's service, and they do not publish their pricing rates for larger customers (> 2 TB/month). Developers can utilize SOAP or REST interfaces to the Nirvanix SDN, or utilize the available Software Development Kits (SDKs) in Java, PHP Zend, Python, and C#.

10.2 The MetaCDN System

The MetaCDN service (depicted in Figure 10) is presented to end-users in two ways - via an easy to use web portal, or as a RESTful Web Service. The web portal was developed using Java Enterprise and JavaServer Faces (JSF) technologies, with a clustered MySQL back-end to store user accounts and deployments, and the capabilities,

pricing and historical performance of service providers. Using the web portal, users can create an account on the MetaCDN system, and enter credentials for any Cloud Storage or ancillary providers they have an account with. Once completing the initial setup phase, they can utilize the MetaCDN system to intelligently deploy content onto storage providers according to their performance requirements, coverage needs and budget limitations. The web portal is most suited for small or ad-hoc deployments, and is especially useful for less technically inclined content creators.

The alternative method of accessing the MetaCDN service is via RESTful Web Services, which expose all the critical functionality of the MetaCDN system. This access method is most suited for customers with more complex and frequently changing content delivery needs, allowing them to integrate the MetaCDN service in their own origin web sites and content creation workflows, and manage their deployment in a dynamic fashion.

The MetaCDN system works by integrating with each storage provider via connectors (depicted in Figure 10) that provides an abstraction to hide the complexity arising from the unique interfaces each provider utilizes for accessing their systems. An abstract class, DefaultConnector, encapsulates the basic functionality that each provider could be expected to support, as well as more advanced facilities that may only be supported by some providers. This class must be implemented by all existing and future connectors, imposing a common interface. The basic operations include creation, deletion, and renaming of replicated files and folders. If an operation (i.e. an advanced storage or delivery feature) is not supported on a particular service, then the connector for that service throws a FeatureNotSupportedException. This is crucial, as whilst the providers themselves have very similar functionality, there are some key differences, such as the largest allowable file size, their coverage footprint or a particular delivery mechanism. This exception also assists the MetaCDN system to match a user's deployment to a particular provider that can meet their specific feature requirements.

10.3 Critical Functionality of the MetaCDN Platform

The MetaCDN service depends on a number of core components (depicted in Figure 10) that encapsulate the logic and management layers required to harness the capabilities of different upstream storage providers, and present a consistent, unified view of the aggregated services available to end-users. These components include the MetaCDN Allocator, which selects the optimal providers to deploy content to, and performs the actual physical deployment; the MetaCDN QoS Monitor, which tracks the current and historical performance of participating storage providers; the MetaCDN Manager, which tracks each user's current deployment and performs various housekeeping tasks; the MetaCDN Database, which stores important information needed by the MetaCDN system; and the MetaCDN Load Redirector, which directs MetaCDN end-users to the best file replica, ensuring fast and reliable service at all times.

The MetaCDN Allocator allows users to deploy files either directly (uploading a file from their local file system) or from an already publicly accessible origin website (sideloading the file, where the backend storage provider pulls the file). Given that not all providers support sideloading, the MetaCDN system can perform this feature on behalf of the user and subsequently upload the file manually. When accessing the service via the web portal or Web Services, MetaCDN users are given a number of different deployment options depending on their needs, including:

- Maximize coverage and performance, where MetaCDN deploys as many replicas as possible to all available providers and locations,
- Deploy content in specific locations, where a user nominates regions and MetaCDN matches the requested regions with providers that service those areas,
- Cost optimized deployment, where MetaCDN deploys as many replicas in the locations requested by the user as their transfer and storage budget will allow, or
- Quality of Service (QoS) optimized deployment, where MetaCDN deploys to providers that match specific QoS targets that a user specifies, such as average throughput or response time from a particular location, which is tracked by persistent probing from the MetaCDN QoS Monitor.

Once a user deploys using the options above, all information regarding their deployment is stored in the MetaCDN database, and they are returned a set of publicly accessible URLs pointing to the replicas deployed by MetaCDN, and a single MetaCDN URL that can transparently redirect end-users to the best replica for their access location.

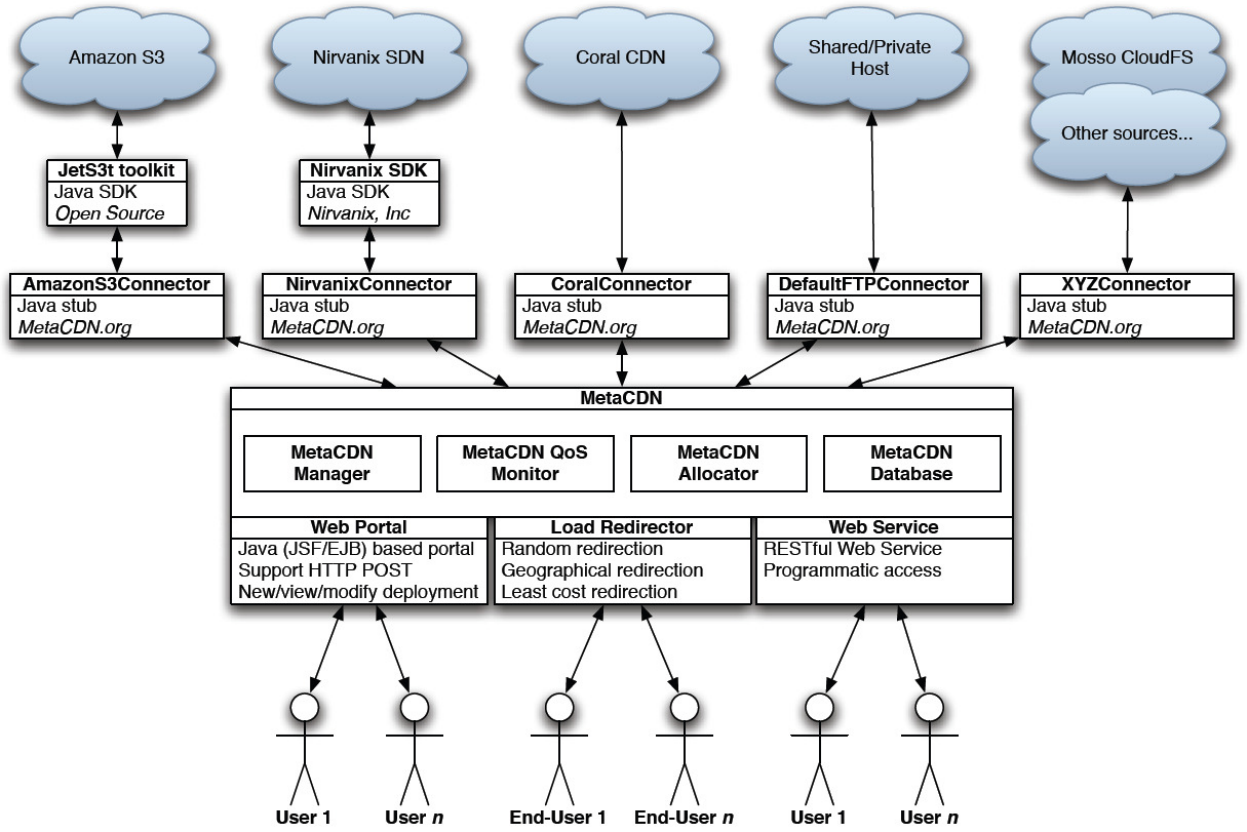


Figure 10: The MetaCDN system.

The MetaCDN QoS Monitor is responsible for tracking the performance of participating providers at all times, monitoring and recording throughput, response time and availability from a variety of locations, which is used for QoS optimized deployment matching. This component also ensures that upstream providers are meeting their SLAs, and provides a logging audit trail to allow end-users to claim credit in the event the SLA is violated. This information is necessary in order to make a claim in the event of an outage. This persistent monitoring keeps the providers 'honest', and also provides signals to the MetaCDN system, which can redeploy content with minimal effort to alternative providers that can satisfy a user's QoS constraints, if available.

The MetaCDN Manager ensures that all current deployments are meeting their QoS targets for users that have made QoS optimized deployments. It also ensures that replicas are removed when no longer required (i.e. the 'deploy until' date set by the user has expired), guaranteeing that storage costs are minimized at all times. Finally, for users that have made cost optimized deployments, it ensures a user's transfer budget has not been exceeded, by tracking usage (i.e. downloads) from auditing information provided by upstream providers, and removing replicas when their budget has been exhausted.

The MetaCDN database ensures persistent and reliable day-to-day operation of the MetaCDN system, by storing important information like MetaCDN user accounts, their credentials for Storage Cloud and other alternative providers, information on users' deployments and their utilization. This database also stores logistical details regarding the storage providers used, such as their pricing structures, SLAs, coverage locations and their historical performance.

The MetaCDN Load Redirector is the only component that end-users (i.e. consumers) of the MetaCDN system interact with. The Load Redirector is responsible for directing end-users (via a combination of DNS and HTTP Redirection) to the "best" replica. Which replica is the "best" depends on the preference of the MetaCDN user who made the deployment – it could be the closest geographical replica, the cheapest replica, or simply a random replica. All of these load redirection capabilities are supported by the MetaCDN Load Redirector.

12. Conclusion and Future Thoughts

Cloud computing is a new and promising paradigm delivering IT services as computing utilities. As Clouds are designed to provide services to external users, providers need to be compensated for sharing their resources and capabilities. In this paper, we have proposed architecture for market-oriented allocation of resources within Clouds. We have also presented a vision for the creation of global Cloud exchange for trading services. Moreover, we have discussed some representative platforms for Cloud computing covering the state-of-the-art. In particular, we have presented various Cloud efforts in practice from the market-oriented perspective to reveal its emerging potential for the creation of third-party services to enable the successful adoption of Cloud computing, such as meta-negotiation infrastructure for global Cloud exchanges and provide high performance content delivery via 'Storage Clouds'.

The state-of-the-art Cloud technologies have limited support for market-oriented resource management and they need to be extended to support: negotiation of QoS between users and providers to establish SLAs; mechanisms and algorithms for allocation of VM resources to meet SLAs; and manage risks associated with the violation of SLAs. Furthermore, interaction protocols needs to be extended to support interoperability between different Cloud service providers. In addition, we need programming environments and tools that allow rapid creation of Cloud applications.

Data Centers are known to be expensive to operate and they consume huge amounts of electric power. For example, the Google data center consumes power as much as a city such as San Francisco. As Clouds are emerging as next-generation data centers and aim to support ubiquitous service-oriented applications, it is important that they are designed to be energy efficient to reduce both their power bill and carbon footprint on the environment. To achieve this at software systems level, we need to investigate new techniques for allocation of resources to applications depending on quality of service expectations of users and service contracts established between consumers and providers [67].

As Cloud platforms become ubiquitous, we expect the need for internetworking them to create market-oriented global Cloud exchanges for trading services. Several challenges need to be addressed to realize this vision. They include: market-maker for bringing service providers and consumers; market registry for publishing and discovering Cloud service providers and their services; clearing houses and brokers for mapping service requests to providers who can meet QoS expectations; and payment management and accounting infrastructure for trading services. Finally, we need to address regulatory and legal issues, which go beyond technical issues. Some of these issues are explored in related paradigms such as Grids and service-oriented computing systems. Hence, rather than competing, these past developments need to be leveraged for advancing Cloud computing. Also, Cloud computing and other related paradigms need to converge so as to produce unified and interoperable platforms for delivering IT services as the 5th utility to individuals, organizations, and corporations.

Acknowledgements

This work is partially supported by the Australian Department of Innovation, Industry, Science and Research (DIISR) and the Australian Research Council (ARC) through the International Science Linkage and the Discovery Projects programs respectively. Ivona Brandic was a visitor to the GRIDS Lab while performing this work. This paper is a substantially extended version of a keynote talk [68] presented at the *10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008)*. We would like to thank Marcos Assunção and anonymous reviewers for their constructive comments that greatly improved the quality of this paper.

References

[1] L. Kleinrock. A vision for the Internet. *ST Journal of Research*, 2(1):4-5, Nov. 2005.

[2] S. London. INSIDE TRACK: The high-tech rebels. *Financial Times*, 6 Sept. 2002.

[3] I. Foster and C. Kesselman (eds). *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann, San Francisco, USA, 1999.

[4] M. Chetty and R. Buyya. Weaving Computational Grids: How Analogous Are They with Electrical Grids? *Computing in Science and Engineering*, 4(4):61-71, July-Aug. 2002.

[5] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. Technical Report HPL-2002-57R1, HP Laboratories, Palo Alto, USA, 3 July 2003.

[6] D. Abramson, R. Buyya, and J. Giddy. A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker, *Future Generation Computer Systems*, 18(8):1061-1074, Oct. 2002.

[7] A. Weiss. Computing in the Clouds. *netWorker*, 11(4):16-25, Dec. 2007.

[8] Twenty Experts Define Cloud Computing. http://cloudcomputing.sys-con.com/read/612375_p.htm [18 July 2008].

[9] G. F. Pfister, *In Search of Clusters*, *2nd Edition*, Prentice Hall, Upper Saddle River, USA, 1998.

[10] R. Buyya (ed.), *High Performance Cluster Computing: Architectures and Systems*, *vol. 1*, Prentice Hall, Upper Saddle River, USA, 1999.

[11] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE*, 93(3):698-714, March 2005.

[12] S. Venugopal, X. Chu, and R. Buyya. A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol. In *Proceedings of the 16th International Workshop on Quality of Service (IWQoS 2008)*, Twente, The Netherlands, June 2008.

[13] B. Van Looy, P. Gemmel, and R. Van Dierdonck (eds). *Services Management: An Integrated Approach*. Financial Times, Prentice Hall, Harlow, England, 2003.

[14] B. Schneider and S. S. White. *Service Quality: Research Perspectives*. Sage Publications, Thousand Oaks, USA, 2004.

[15] C. S. Yeo and R. Buyya. Integrated Risk Analysis for a Commercial Computing Service. In *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007)*, Long Beach, USA, Mar. 2007.

[16] M. Crouhy, D. Galai, and R. Mark. *The Essentials of Risk Management*. McGraw-Hill, New York, USA, 2006.

[17] R. R. Moeller. *COSO Enterprise Risk Management: Understanding the New Integrated ERM Framework*. John Wiley and Sons, Hoboken, USA, 2007.

[18] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003)*, Bolton Landing, USA, Oct. 2003.

[19] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: an architecture for secure resource peering. *ACM SIGOPS Operating Systems Review*, 37(5):133-148, Dec. 2003.

[20] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent and Grid Systems*, 1(3):169-182, 2005.

[21] A. AuYoung, B. Chun, A. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In *Proceedings of the 1st Workshop on Operating System and Architectural Support for the Ondemand IT Infrastructure (OASIS 2004)*, Boston, USA, Oct. 2004.

[22] D. E. Irwin, J. S. Chase, L. E. Grit, A. R. Yumerefendi, D. Becker, and K. Yocum. Sharing networked resources with brokered leases. In *Proceedings of the 2006 USENIX Annual Technical Conference (USENIX 2006)*, Boston, USA, June 2006.

[23] D. Hamilton. 'Cloud computing' seen as next wave for technology investors. *Financial Post*, 4 June 2008. http://www.financialpost.com/money/story.html?id=562877 [18 July 2008]

[24] Morgan Stanley. Technology Trends. 12 June 2008. http://www.morganstanley.com/institutional/techresearch/pdfs/TechTrends062008.pdf [18 July 2008]

[25] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the Grid. *Scientific Programming*, 13(4):265-275, October 2005.

[26] OpenNebula Project. http://www.opennebula.org/ [23 July 2008]

[27] Reservoir Project. http://www.reservoir-fp7.eu/ [30 Oct. 2008]

[28] Amazon Elastic Compute Cloud (EC2). http://www.amazon.com/ec2/ [18 July 2008]

[29] Amazon Simple Storage Service (S3). http://www.amazon.com/s3/ [18 July 2008]

[30] Google App Engine. http://appengine.google.com [18 July 2008]

[31] Microsoft Azure. http://www.microsoft.com/azure/ [30 Oct. 2008]

[32] Sun network.com (Sun Grid). http://www.network.com [18 July 2008]

[33] X. Chu, K. Nadiminti, C. Jin, S. Venugopal, and R. Buyya. Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications. In *Proceedings of the 3th IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*, Bangalore, India, Dec. 2007.

[34] S. Venugopal, R. Buyya, and L. Winton. A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids. *Concurrency and Computation: Practice and Experience*, 18(6):685-699, May 2006.

[35] A. Chien, B. Calder, S. Elbert, and K. Bhatia. Entropia: Architecture and Performance of an Enterprise Desktop Grid System. *Journal of Parallel and Distributed Computing*, 63(5):597-610, May 2003.

[36] Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/ [18 July 2008]

[37] D. E. Irwin, L. E. Grit, and J. S. Chase. Balancing Risk and Reward in a Market-based Task Service. In *Proceedings of the 13th International Symposium on High Performance Distributed Computing (HPDC 2004)*, Honolulu, HI, June 2004.

[38] C. S. Yeo and R. Buyya. Pricing for Utility-driven Resource Management and Allocation in Clusters. *International Journal of High Performance Computing Applications*, 21(4):405-418, Nov. 2007.

[39] C. S. Yeo, S. Venugopal, X. Chu, and R. Buyya. Autonomic Metered Pricing for a Utility Computing Service. Technical Report GRIDS-TR-2008-16, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 28 Oct. 2008.

[40] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2004)*, San Francisco, USA, Dec. 2004.

[41] Hadoop. http://hadoop.apache.org/ [6 Nov. 2008]

[42] C. Jin and R. Buyya. MapReduce Programming Model for .NET-based Distributed Computing. Technical Report GRIDS-TR-2008-15, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 17 Oct. 2008.

[43] Hadoop, Hadoop Applications. http://wiki.apache.org/hadoop/PoweredBy [6 Nov. 2008]

[44] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica. Improving MapReduce Performance in Heterogeneous Environments. In *Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2008)*, San Diego, USA, Dec. 2008.

[45] I. Brandic, S. Venugopal, M. Mattess, and R. Buyya. Towards a Meta-Negotiation Architecture for SLA-Aware Grid Services. Technical Report GRIDS-TR-2008-10, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 8 Aug. 2008.

[46] I. Brandic, S. Pllana, and S. Benkner. Specification, Planning, and Execution of QoS-aware Grid Workflows within the Amadeus Environment. *Concurrency and Computation: Practice and Experience*, 20(4):331-345, 25 Mar. 2008.

[47] E. Elmroth and J. Tordsson. A Grid Resource Broker Supporting Advance reservations and Benchmark-based Resource Selection. In *Proceedings of the 7th Workshop on State-of-the-art in Scientific Computing (Para 2004)*, Lyngby, Denmark, June 2004.

[48] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In *Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2002)*, Edinburgh, Scotland, July 2002.

[49] Web Service Level Agreement (WSLA). http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf [18 July 2008]

[50] WS-Agreement. http://www.ogf.org/documents/GFD.107.pdf [18 July 2008]

[51] Spring. http://www.springframework.org [18 July 2008]

[52] B. Maggs. Global Internet Content Delivery. In *Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)*, Brisbane, Australia, May 2001.

[53] A. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante. Drafting behind Akamai (travelocity-based detouring). *ACM SIGCOMM Computer Communication Review*, 36(4):435-446, Oct. 2006.

[54] Mirror Image. http://www.mirror-image.com [18 July 2008]

[55] D. Rayburn. CDN pricing: Costs for outsourced video delivery. In *Streaming Media West 2008: The Business and Technology of Online Video*, San Jose, USA, Sept. 2008. http://www.streamingmedia.com/west/presentations/SMWest2008-CDN-Pricing.ppt [7 Nov. 2008]

[56] J. Broberg, R. Buyya, and Z. Tari. MetaCDN: Harnessing 'Storage Clouds' for High Performance Content Delivery. Technical Report GRIDS-TR-2008-11, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 15 Aug. 2008.

[57] M. Arlitt and T. Jin. Workload Characterization of the 1998 World Cup Web Site. *IEEE Network*, 14:30-37, May 2000.

[58] A. K. Iyengar, M. S. Squillante, and L. Zhang. Analysis and characterization of large-scale Web server access patterns and performance. *World Wide Web*, 2(1-2):85-100, Jan. 1999.

[59] V. N. Padmanabhan and K. Sripanidkulchai. The Case for Cooperative Networking. In *Proceedings of the 1st International Workshop on Peer-To-Peer Systems*, Lecture Notes In Computer Science, 2429:178-190, Mar. 2002. Springer-Verlag, London.

[60] J. Elson and J. Howell. Handling Flash Crowds from your Garage. In *Proceedings of the 2008 USENIX Annual Technical Conference (USENIX 2008)*, Boston, USA, June 2008.

[61] D. Gottfrid. Self-service, Prorated Super Computing Fun!. *The New York Times*, 1 Nov. 2007. http://open.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/ [7 Nov. 2008]

[62] D. MacAskill. Scalability: Set Amazon's Servers on Fire, Not Yours. In *O'Reilly Emerging Technology Conference (ETech 2007)*, San Diego, USA, Mar. 2007. http://blogs.smugmug.com/don/files/ETech-SmugMug-Amazon-2007.pdf [7 Nov. 2008]

[63] J. Broberg, S. Venugopal, and R. Buyya. Market-oriented Grids and Utility Computing: The State-of-the-art and Future Directions. *Journal of Grid Computing*, 6(3):255-276, Sep. 2008.

[64] DreamHost. http://www.dreamhost.com [18 July 2008]

[65] Nirvanix Storage Delivery Network (SDN). http://www.nirvanix.com/sdn.aspx [18 July 2008]

[66] Mosso CloudFS. http://www.mosso.com/cloudfs/ [18 July 2008]

[67] K. H. Kim, R. Buyya, and J. Kim. Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)*, Rio de Janeiro, Brazil, May 2007.

[68] R. Buyya, C. S. Yeo, and S. Venugopal. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008)*, Dalian, China, Sept. 2008.