Design of a Quality of Service Aware Public Computing Utility

Muthucumaru Maheswaran
School of Computer Science
McGill University
Montreal, QC H3A 2A7
Canada
Email: maheswar@cs.mcgill.ca

Balasubramaneyam Maniymaran
Dept. Electrical & Computer Eng.
McGill University
Montreal, QC H3A 2A7
Canada
Email: bmaniy@cs.mcgill.ca

Shah Asaduzzaman
School of Computer Science
McGill University
Montreal, QC H3A 2A7
Canada
Email: asad@cs.mcgill.ca

Arindam Mitra
Dept. Computer Science
University of Manitoba
Winnipeg, MB R3T 2N2
Canada
Email: arindam@cs.umanitoba.ca

Submitted to the IEEE Symposium on Network Computing Applications (NCA 2004)

Abstract - *This paper describes a design for a quality of service aware public computing utility (PCU). The goal of the PCU is to combine the power of public resources with dedicated (private) resource pools to provide high quality of service to the clients at the least cost. Our PCU design combines peer-to-peer (P2P) and Grid computing ideas in a novel manner to construct a utility-based computing environment. In this paper, we present the overall architecture and describe two major components: a P2P overlay substrate for connecting the resources in a global network and a community-based decentralized resource management system. One of the unique features of our P2P substrate is that it names and locates resources as standardized virtual commodities. This feature is exploited to plugin PCU services as virtual commodities into the P2P substrate. By plugging PCU services into the P2P substrate, we are able to create a community-oriented architecture for the PCU, where services are associated with resources in an on-demand basis. This sharply differs from traditional centralized, hierarchical, and distributed architectures where the services to resources association is fixed. The simulation results comparing different aspects of our design with alternative approaches illustrate the benefits of our design.*

1. Introduction

Grid and utility computing are two major research topics that have captivated the computing systems community in academia and industry over the recent years. While Grid computing [FoK01] and utility computing share the basic idea of pooling resources and services from different providers to create large virtual resource and service bases, they differ in how the virtual pool is managed. Utility computing focuses on providing a *utility* like interface to the virtual pool similar to that provided by common public utilities such as electricity or water. One of the defining feature of a utility is the *commoditization* of the resources. Ideally, the commoditization process

makes the resources provider-neutral and simplifies the services such as metering and billing. In a distributed computing system, commoditization means we categorize the computing resources into virtual resources that provide predefined sets of services. The benefits and challenges of a utility computing lie on efficiently realizing the commoditization process in distributed computing systems.

Given the current trend of outsourcing computing and data processing activities, computing utilities have tremendous scope to be a major part of future computing infrastructures. By employing computing utilities that are run by third parties, organizations can spend more effort on actual "business" logic and less on managing computing infrastructure. Further, computing utilities can provide services with low initial capital investment and running costs by multiplexing the resource demands from different clients.

Typically, computing utilities are built using resources that are supplied by a single or few providers. These resources are installed for the exclusive use of the computing utility. This approach naturally limits the scalability and geographical scope of the computing utility. One of the advantages of this approach is that resource behavior is well managed resulting in predictable *quality of service* (QoS). Here, we consider an extended notion of computing utility called *public computing utility* (PCU) that open up the membership to *public resources* much like the *peer-to-peer* (P2P file sharing systems. This enables the PCU to leverage vast amounts of idle resources spread throughout the Internet. In addition to lowering the cost of participation, the PCU prevents provider monopoly and creates a geographically distributed resource base that is capable of satisfying location specific resource requirements.

Although a PCU provides several benefits by opening the membership to public resources, several key issues should be addressed before this concept can be successfully deployed. One of the key issues is scalability. A PCU is expected to cope with many fold increase in number of resources that are possibly distributed all over the world. In addition, scalability has to be considered in terms of number of users, policies, and applications as well.

Besides scalability, a PCU has to address issues related to trust, security, and incentives. In a private computing utility, resources have high trust among them that are bolstered by off-line agreements. With public resources trust among them should be learned through an online trust modeling process [**?**]. A large-scale system such as the PCU will have resources that are well trusted by every other resource in the system. Our idea is to exploit such global trustworthiness to develop efficient mechanisms to model trust. Induction of public resources also

mandates stronger security mechanisms and better coordination among the trust modeling processes and security provisioning mechanisms such that security breaches by resources are penalized by accordingly lowering their trust values. Because PCUs rely on the cooperation of the public resources, they face the same participation issues as P2P file sharing systems [**?**]. Therefore, designing effective incentive mechanisms to keep the public resources fully engaged with the PCU is very important. The incentive mechanisms should be compatible with the trust and resource management systems that work within the PCU framework. Because the PCU is a generalized computing platform that is targetting different applications, the incentive mechanisms should be application independent.

Finally, it is important to design the PCU such that the overall system will efficiently self-organize as resources frequently arrive and depart the system. The self-organization of the PCU should heal the interruptions that have occured for the management systems within the PCU as well as the resource allocations.

This paper introduces a QoS aware PCU design called *Galaxy*. Delivering QoS while solely relying on public resources is hard to accomplish for the PCU because a public resource working on a client can defect from the PCU at any time. One way to compensate for this uncertainty is to employ redundant public resources. Another approach is to switch the client to dedicated resources once the public resources are determined to perform below the expected performance level. Our PCU design supplements the capacities harnessed from public resources with dedicated resources to meet the performance expectations of the clients.

Section 2 introduces the overall architecture of Galaxy. Our architecture combines ideas from P2P and Grid computing systems in a novel manner to create a utility-based computing environment. The P2P substrate of Galaxy is discussed in Section 3. One of the unique features of our P2P substrate is that it names and locates resources as standardized virtual commodities. This feature is exploited to plugin PCU services as virtual commodities into the P2P substrate. The *Galaxy resource management system* (GRMS) is described in Section 4. The GRMS uses a "community"-based architecture to manage the resources. This differs sharply from traditional resource management architectures such as centralized, hierarchical, or distributed, where a resource is associated with a resource manager in a static manner. In the community-based architecture, a resource can contact any well standing member of the manager community and receive the same service. The community of resource managers are organized in a P2P overlay and can be located and accessed efficiently by all resources. Sections 5 and **??** briefly examine the services and applications that can be supported by the Galaxy architecture, respectively. Other

research works related to Galaxy are presented in Section 7.

2. A Public Computing Utility Architecture

In this section, we present an architecture for the PCU. Our architecture is hinged on the following principles:

- **Localization:** control data related to the operation of a resource is held at the resource itself; there is no system-wide entity that holds or collects global information.
- **Scoped aggregation:** information is aggregated in limited scopes both spatially and temporally; primary purpose of aggregation is to improve the efficiency of management by reusing information; scoping is done to reduce the contension between localization and aggregation.
- **Non-persistence:** it is assumed that any entity can fail and failure of any single entity has a minimal detrimental impact on surviving entities; although the resources that participate in the management process are expected to be relatively more persistent, they can fail too.

Based on these principles, we present the architecture shown in Figure 1 for the PCU. The lowest layer of the architecture is the P2P overlay network called the *resource addressable network* (RAN) [**?**]. All the resources that participate in the PCU plug into the RAN. The RAN provides the resource discovery and access services to the PCU. The next upper layer is the *Galaxy resource management system* (GRMS). Similar to the RAN, the GRMS is also organized as a P2P overlay network. In the RAN, the peers are virtualized resources whereas in the GRMS the peers are *resource brokers* (RBs). The RBs provide several services such as resource allocation, incentive management, and trust management to the PCU. The next upper layer of Galaxy is the Galaxy services. Although the architecture does not impose any restriction on the organization of this layer, it could be organized as a P2P network. Example Galaxy services include application level QoS managers, shells, and network file systems.

The RAN handles resources in distinct resource *types* instead of individual resources. That is, instead of naming and discovering a particular a resource the RAN deals with resource collections that fit the specifications of the resource type. A resource type points to well defined (standardized) virtual resources that deliver certain services. This feature is exploited to plugin RBs into the RAN as virtual resources. Therefore, we implement P2P overlay of RBs using the RAN routing substrate. Each peer provides an *application programming interface* (API) supporting different RAN functions. Some example functions supported by the RAN API include inserting

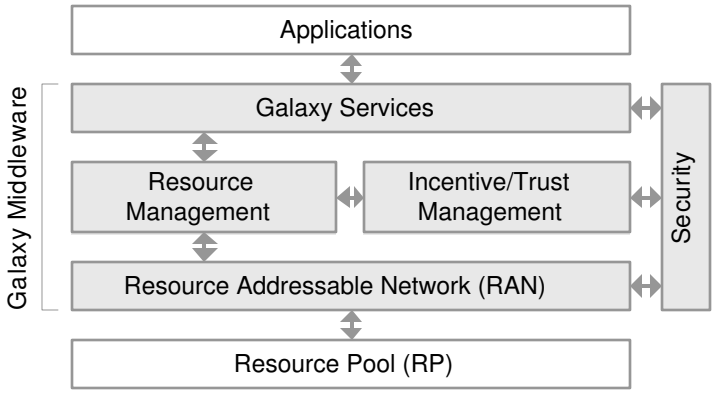

Figure 1: The Galaxy Architecture.

a resource of a given type into RAN, finding a resource instant that satisfies a query specification, and finding a resource instant of a given type at a given location. Internally, the RAN is organized as a P2P network with a routing process similar to Pastry [**?**].

Because the set of RBs provide the management fabric of the PCU, only highly trusted resources participate in providing these services. A decentralized trust modeling process runs among the RBs to isolate malicious RBs from continuing to impart management services. Because RBs are non-persistent services, they maintain only soft-state. Any hard state associated with managing the resources is maintained securely at the resources themselves. As part of the GRMS, the RBs provide a provide a variety of services including: (a) matchmaking to allocate resources to requesters, (b) sign an initial work order between the resources and requesters, (c) arbitrate between the complaints and counter-complaints by the resources and requesters at the completion of the work order, (d) assign increments for incentives based on the behavior during the past work order, and (e) compute the incentive profile for a new resource.

Resource pool (RP) is the the collection of physical resources, both public and private, that participate in Galaxy. The resource addressable network (RAN) provides QoS aware naming and discovery mechanisms. It is the fundamental part of the Galaxy and every Galaxy node is installed with this module. The Galaxy resource management system (GRMS) is the workhorse of the Galaxy carrying out resource allocation, aggregation, scheduling, and service level agreement monitoring and management. However, completeness of this module can differ from node to node depending on the need and capability. The incentive/trust management module operates

parallel with the GRMS. It is an important component in a PCU as it is the one that regulates the behavior of the unstable public resources. Galaxy services are set of necessary/common Galaxy tools and their presence in a node is completely on-demand basis. Security is a layer that spans in parallel to all other layers providing security in every Galaxy operations. Above this Galaxy middleware, consumers launch their services.

3. Resource Addressable Network

Resource addressable network (RAN) is the resource naming and discovery substrate of Galaxy. It is completely decentralized and self-organizing. RAN naming is achieved by *profiling* the resources into specific resource *types* based to their attribute-value tuples. Resources of same type are collected together in *type rings* and the type rings are connected by *neighborhood rings*. The RAN rings are built based on *space filling curves* which facilitate an efficient discovery mechanism.

3.1. Space Filling Curves

RAN uses space filling curves in many phases of its design. A *space filling curve* (SFC) [Sag94] is a curve that passes through all the points in a given bounded space. There are a many such curves defined in literature, but the *Hilbert curve* is used in RAN. A real such curve is practically impossible/useless to produce, and mostly approximations of such curve are used (Figure 2).

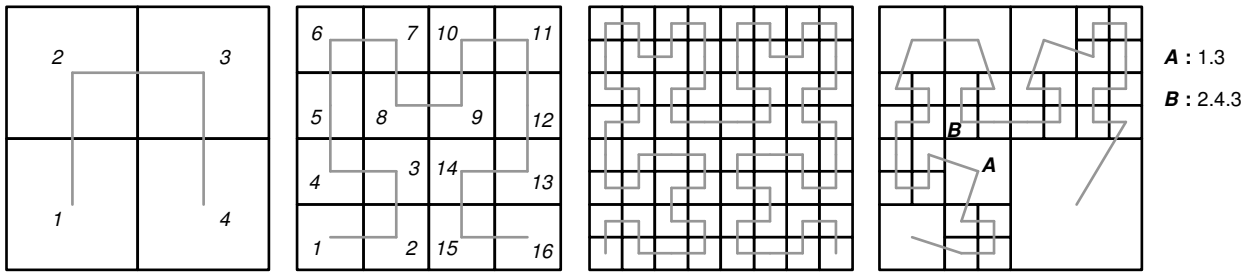

Figure 2: Hilbert space filling curves.

There are three characteristics [ShC02, But71] of the SFCs that make them useful in RAN design:

1. A SFC maps points in multi-dimensional space on to a single dimensional curve: this property is exploited in the creation of type/neighborhood rings (note that the rings are single dimensional).

2. The mapping is such that the points that are closer to each other in the multi-dimensional space are closer on the single dimensional SFC too: exploited in resource discovery.

3. SFCs can be produced recursively and non-uniformly: enables a hierarchical naming of the resources with different *Hilbert index* density in different regions (Figure 2(d)). Hierarchical location-based naming can be effectively used in RAN routing.

3.2. Profile Based Naming

Most of the resource discovery systems name the resources using attribute-value tuples [RaL98, AdS99, DiP01]. While this *description-based naming* provides flexibility in querying resources, it results in complex discovery algorithms and increased overhead in messages in terms of number and size. In contrast, RAN implements a *profile-based naming* where resources are *profiled* into specific resource *types* (Figure 3) which are used in the queries. This approach cuts down the size of the RAN messages, eliminates the necessity of complex resource matching algorithms, and decouples the naming from discovery to increase the scalability.

An analysis on the CNET website [CNET3] demonstrates the validity of argument behind the profile-based naming. CNET is a web site providing review on electronic products. The desktop sections was considered for our analysis. As it is obvious that only the mostly available and mostly sought out products are considered for review in this type of web sites, the products reviewed can be considered as a resource pool for a system like PCU. Among the attributes of the desktops given by CNET, only four attributes are considered for our analysis and as they have different ranges of possible values, there were 2808 combinations (types) possible. However when the population of these types are extracted, they form a cumulative distribution shown in Figure 4. The figure shows that 90% of the total population is contained by only 110 types of resources. It supports our argument that acknowledging only the popular types of resources, a major portion of queries can be satisfied at a very much reduced cost.

Simulation of resource discovery systems based on profiled-based and description-based naming systems clearly shows the advantage of profile-based system. Figure 5 shows the message overhead with system size on these two systems. (Please not that it is not a simulation on the RAN architecture described in this paper, but on a simple distributed version that just uses the profile-based naming concept).

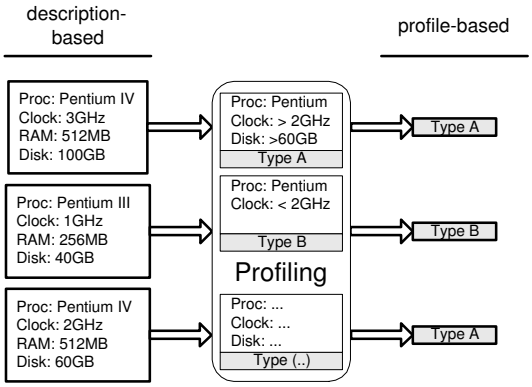

Figure 3: Profiling the resources.

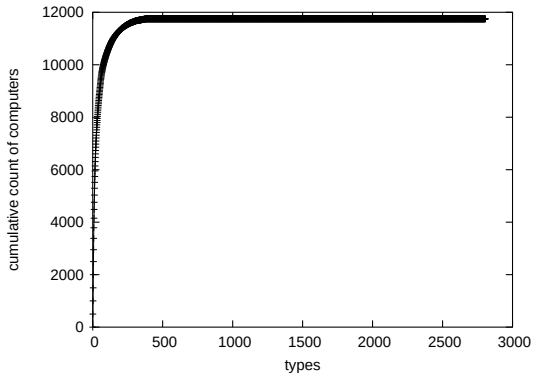

Figure 4: Cumulative distribution of desktop population against resource types.

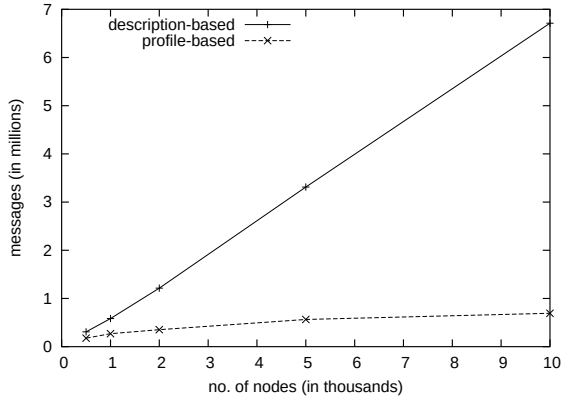

Figure 5: Message complexities of description-based and profile-based naming systems.

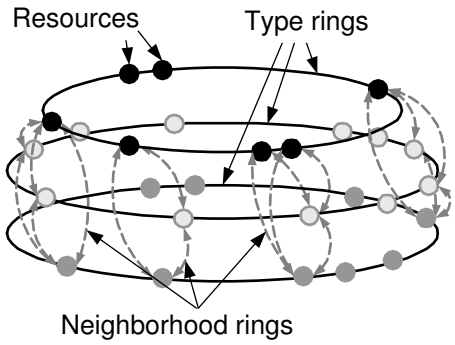

Figure 6: Ring arrangement of the RAN.

RAN recognized resource types are the popular resource descriptions observed in the resource queries. When the resource attributes are considered to be the axis of the resource description space, resource types can be considered as points (with only equalities in the attribute-value tuples) or regions (with inequalities allowed) in the space. When a Hilbert curve is fit into this space, the resource types can be represented by Hilbert indices. Nonuniform approximation levels of the curve, as in Figure 2(d), is used for different levels of resource collection depending on their popularity.

RAN implements also a *profile adaptation engine* which monitors the resource queries, and adds and deletes recognized resource types according to the percentage of accuracy in resource discovery dynamically. Technically, addition and deletion of types are changing the approximation level of an SFC region.

3.3. RAN Rings

Resources of the same type are collected in a *type ring* (Figure 6). It is called as a ring because each resource has at least two routing pointers to its left and right resources. The position of a resource in a type ring is determined by its location in the network. *Landmark aided positioning* is used to determine the location of a resource. In RAN, there are a set of resources called *landmarks*. Upon joining RAN each resource determines their network location in a d-dimensional space by pinging these fixed landmarks. A Hilbert curve is used to name the resources by their location. Therefore basically the types rings are Hilbert curves with the end-points connected together.

In addition to type rings, each resource is a part of at least one *neighborhood ring* (Figure 6 - not all neighborhood rings are shown in the figure). A neighborhood ring composed of one resource from each different resource type which are in the same neighborhood. The neighborhood rings also are constructed as Hilbert curves, this time based on the type names (section 3.2). Neighborhood rings connects the types rings together.

RAN can be considered as yet another *distributed hash table* (DHT) based mechanism where the Hilbert indices are the hash keys. However, not like other DHT based architectures such as Pastry [RoD01], CAN [RaF01], or INS/Twine [BaB02], Galaxy is more decentralized, self-organized, and QoS aware. By default, RAN discovery mechanism selects the closest matching resource for any resource query. This helps to confine the traffic mostly at the network edges reducing the core network congestion.

3.4. RAN Routing and Resource Discovery

Each resource in RAN has two similar routing tables, one for type ring and another for neighborhood ring. One such table is shown in Table 7. It is for a 2-d locational space (similar to Figure 2) with maximum approximation level of 5. The *ring pointers* construct the ring and the *jump pointers* ensure that the routing hop complexity is near $\log_4 n$ where n is the number of nodes in the ring. Figure 8 is taken from a simulation, which shows the number of hops each node traversed before at the time of the join process. It can be seen that the hop count increases only logarithmically with the number of nodes in the system because each node acquires more jump pointers as the system grows.

Ring Pointer Section			
2.3.3.0.0		2.3.1.2.0	
Jump Pointer Section			
0	**1**	**2**	**3**
0.2.1.0.0	1.2.3.3.3	x	3.2.2.0.1
2.0.2.3.2	2.1.3.1.2	2.2.3.2.0	x
-	2.3.1.2.0	2.3.2.0.1	x
2.3.3.0.0	x	-	-
x	-	-	-

Figure 7: Type ring routing table of a node with locational name 2.3.3.1.0.

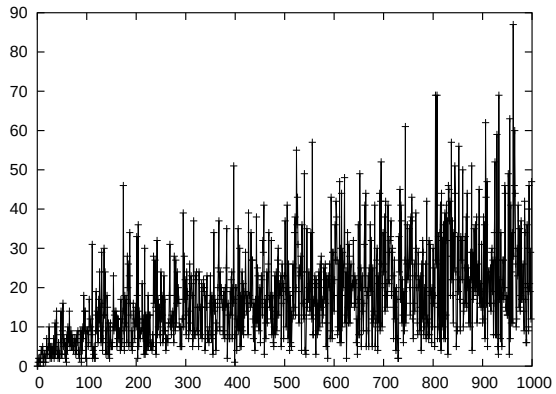

Figure 8: Hop complexity in RAN rings.

RAN is a full decentralized, self-organizing, P2P network where there is no centralized entity to manage the join, departure, discovery, or route creation. When a resource query reaches a RAN node, it first uses its jump pointers to forward the query to the closest node to the required location. If no such selection possible, the query is forwarded to the neighbor that is on the requested side. A new node joining the RAN finds its join position using the RAN discovery mechanism giving its own coordinates as the request. Therefore when a node joins the RAN, it always knows its neighbors (ring pointers). The jump pointers are partially borrowed from the neighbor at the time of join and built up later by observing the query messages that are passing through.

4. Galaxy Resource Management System (GRMS)

The Galaxy Resource Management System (GRMS) provides the allocation and management functionality for the resources addressed and discovered through the RAN. Moreover it controls the incentives to encourage

resource donors and manages their trust values according to their consistency in performance.

4.1. Overview

The GRMS layer sits on top of the RAN substrate which is the collection of all virtual resources in Galaxy. RAN provides the locality aware discovery mechanism for all types of resources. A Galaxy *client, which may be a single physical computer or may be a cluster of computers controlled by a cluster manager, donates and controls the resources. A resource may be a dedicated physical machine, a virtualized slice of a machine or a software entity that provides some Galaxy specific services. In general we call it a virtual resource Each client runs a Galaxy-interface that communicates over the GRMS layer for resource allocation and incentive management.*

A Resource Broker (RB) is a virtual resource that coordinates most of the GRMS layer activities. Being a virtual resource, all the RBs are addressible through RAN and also they constitute a RAN type ring among themselves. The two major functionalities that a resource broker provides are to search and allocate resources as request by Galaxy clients, and to assign, monitor and reassign the trust levels and incentive shares (see section 4.2) of galaxy clients that it interacts with. A client becomes eligible to request for a certain quantity and type of resources depending on the shares it owns. To discover a suitable resource the client always contacts a RB in its proximity. The RB then launches the necessary RAN query, discover the handle of the client that controls the reources, negotiate the allocation of the resource with the client, and returns the resource handle back to the requesting client. At the end of the usage of the resource both the donor and the user clients reports the performance of the resource back to the RB and based on these reports the RB readjusts the incentive shares owned by the donor client.

Resource brokering is a voluntary service and any Galaxy client can get into the RB network as long as they has a certain level of trust. The client of course gets some incentive shares for providing this service. RBs can be refuted or can go down any time, but the clients always can find a RB in its vicinity. RBs can communicate with each other in case of a need to allocate resource from other localities. Some of the clients having very high level of trust may provide a RB+ service, which has a QoS-guaranteed resource scheduler in addition to the usual RB functionalities. The RB+ service is accessible to clients having higher number of shares. When allocatng a resource, the RB+ also installs a probe on the donor client to periodically monitor the preformance of the resource, and in case of fault or unsatisfactory performance it reallocates some other resources in a higher trusted client. The

RB+ actually binds itself with the resource requesting client in a SLA which defines the requirement as quality and duration of allocation of the resources in a certain bounded time.

There are application specific QoS requirement for different classes of applications. QoS management for some generic classes of applications is provided as Galaxy services on top of the GRMS layer. Details of QoS management is discuessed in section 4.3

4.2. Incentive Management in Galaxy

To encourage continuous connectivity among resources in Galaxy, we adopt a shared-based incentive mechanism. Shares are alloted to resources based on the capacities (e.g., processing capacity, memory) each contributes to the PCU for executing client jobs. We address such shares as *ca*pacity *sh*ares or CASH. The incentive is to perk the resources with CASH for their periods of capacity contributions and sack them for defecting from the PCU. The CASH earned will be used by the resources to maximize their individual utility. For instance, the CASH earned by a resource may be exchanged in lieu of financial benefits or prioritizing job executions in the PCU. We use the capacity portion of the profile-names created by RAN, shown in Figure (3), for measuring the contribution of resources to the PCU.

In a PCU, resources can change their behavior and enter and leave the system independently and continuously. To maximize insurance against such random behavior, we have modeled our incentive mechanism to operate in epochs. At the beginning of each epoch, the PCU will compute the maximum CASH, called *maxCASH*, that a participating resource can earn based on the capacity it is willing to contribute during that epoch. Whenever during that epoch, the resource successfully completes some job, the PCU rewards the resource with some CASH. The CASH earned in one epoch may be the starting CASH value for next epoch, but the maximum accumulated CASH during a epoch will not exceed its maxCASH computed at the beginning of that epoch. Assuming maxCASH computation is non-biased, then this mechanism will ensure fair allocation of shares among participating resources. Non-biased maxCASH for resources may be computed by one of the following methods: (a) normalizing a resource's contributing capacity against a base reference capacity for the PCU or (b) considering the ratio of a resource's contributing capacity and the total available capacity of the PCU (or portions of the PCU). Whenever a resource defects from the PCU, it will be penalized by decaying its CASH by some amount. For more details, see [MiM04].

4.3. Quality of Service Management

For any service in a computing utility to have any market value, some parameters of the service must have some quality guarantee. Depending on the business models at different levels of the software stack, mechanisms for Quality of Service (QoS) guarantees at different levels may be required by the system. Since Galaxy is a general purpose resource provisioning system, the GRMS concentrates on the QoS management of resources and some generic application level QoS is supported in terms of galaxy services. The main substrate of Galaxy is publicly owned autonomous resources that are guided by incentives and reputation. But, it has been shown [**?**] that the probability of successfully guaranteeing QoS with this type of uncontrolled resource only is very minimal. The QoS guarantees can be significantly improved if the public resources are augmented with a fully controlled and reliable pool of resources and a resource manager manages the allocation of the resources to compensate the unpredictability of the public resources.

For uniformity and simplicity, all resources, whether public or private, are discovered through the common P2P discovery network RAN. The subscribers of the Galaxy may access some best of service from the resources by directly getting allocation of resources through resource brokers on the RAN. The guaranteed quality service is a value added service that may be delivered by some resource manager who owns some dedicated private resources. In the latter case the resource allocation and failure management is controlled by the resource manager.

To ensure a guaranteed quality of service there must be a service level agreement (SLA) between the parties involved about the parameters of the service. When a public resource is discovered through the RAN and bound to the requesting user, the user may define some quality metrics like available time, minimum capacity or share of the resource etc. Since public resources participate on a voluntary basis, there is no financial transaction involved here, but the user monitors the performance it receives and the result affects the credential given for the resource accordingly. In case of a resource manager who deploys a dedicated pool of resources may come to service level agreements with its application level clients to guarantee some application level quality metrics, such as throughput or response time. Support for creating SLAs and monitoring of performance for some generic applications are provided as galaxy services.

For efficient management of the shared public resources and the dedicated private pool of resources in order to deliver guaranteed quality service, the resource manager has to use some online heuristic algorithm to route

and re-route the resource requests onto proper pool. We have developed an on-line algorithm that can manage both type of resources targeting maximum compliance with the agreed-upon quality guarantees. Results obtained from some sample experiments demonstrating the performance of the algorithm are described in section **??**. To allow maximum level of autonomy of the public resources and to minimize the overhead of communication, the resource manager exerts almost no monitoring or control of the public resources. Instead, all the public resources are modeled to have identically distributed stochastic characteristics and the distribution parameters are estimated from history data. Although bringing all the resources causes loss of information about their characteristics, it simplifies the algorithm radically and minimizes the overhead. Our results show that significant gain in performance in terms of both application throughput and compliance with agreement can be achieved through this technique.

5. Galaxy Services

Galaxy services are some tools provided by the Galaxy infrastructure that assist the consumers of the Galaxy in monitoring and managing the resources they acquired or in launching applications on them. These tools comes in a modular fashion so that the users can load them on demand basis. Galaxy shell and Galaxy file system are two of the important Galaxy services.

Galaxy shell or GShell is the Unix shell like interface to the Galaxy. Users can launch interactive sessions with Galaxy through GShell, querying and acquiring resources for their need and launching jobs on the acquired resources. GShell provides one simple way of utilizing Galaxy. In addition to the ordinary Unix commands, GShell commands include commands for querying available types, querying resources of desired properties, acquiring required type of resources, launching jobs on the acquired resources, and setting up Galaxy environment variables like the scope (time and domain) of the session.

6. Galaxy Applications

As it is established, Galaxy is a general purpose resource provisioning infrastructure. It can host many different types of applications such as content delivery, high performance computations, or online-gaming. Galaxy is also designed such that it well suits a commercial model with provisions for SLA management and revenue handling. Whoever wants to host a service on Galaxy contacts the *Galaxy service provider* (GSP) via a web interface.

Following the online registration, the SLA is created through a web-form interface (simple) or through some SLA specification creation software (complex). The specification describes the intended application in details such as cost agreed upon, QoS expected, application model to be hosted, and type and number of resources required. Once the SLA is created, a virtual resource cluster or simply a *virtual cluster* (VC) will be allocated to the consumer which he/she can use as it is a privately owned cluster. The virtualization hides the technicality behind the implementation of the VC from the consumer.

Consider a case of a content delivery application: at the time of SLA creation, the the content service provider, a consumer of the Galaxy, provides details of size of the content, expected bandwidth usage, distribution of the end users (in time and geography), and the expected satisfaction of the end users. Here *end users* are the ones who consume the content hosted on Galaxy. If the service provider does not explicitly mention the type of machines on which the content to be hosted, Galaxy, to be precise the GRMS, can decide the appropriate number and type of machines based on the content, expected bandwidth, and the distribution of the end users. The RAN returns a set of required type of machines at suitable locations that matches the geographical distribution of the end users. The selection is then fine-tuned by the incentive/trust mechanisms at the GRMS level. Once the resources are selected, it is completely under the responsibility of the GRMS which provides the virtualization and maintains the agreed QoS. The Galaxy services assist the content service provider to use the VC. For example, he/she can use the GNFS to move the content onto the acquired resources. The GRMS can often change the virtual-to-real resource mappings as the allocated real resources can leave the system or get overloaded affecting the QoS. But this is transparent to the consumer.

7. Related Work

The research works related to the PCU concept can be categorized into three classes: (1) the works carried out on or around Grid environment; (2) the works concentrated on realizing “utility” computing; and (3) the works related to the real PCU environment. However, as can be seen below, the boundary between these categories are not clearly defined as they try to exploit the power of the other.

Grid related researches are based on the concept of Grid found in [FoK01] and carried out mostly with the Globus toolkit [Glo04]. Grid is a distributed enterprise solution where different institutions pool their resource together to build a high performance computing platform. Even though Grid concept originated as a computing

architecture, it is now evolved into *open grid services architecture* (OGSA) [FoK02] which is more of a service-oriented system. One of the important drawbacks of the Grid approach is its poor scalability [FoI03]. Due to the manual interaction involved in the building up the base resource pool, the size of the Grid always restricted to a few institutions. It also implies that the availability of the Grid power is restricted to tight communities. These are major differences the Grid have from a PCU architecture. There are projects that augment the Grid environments with other techniques such as P2P and utility-oriented management to move the Grid environment towards a PCU system. *OurGrid* [AnC03] is one such approach which interconnects multiple Grid environments in a P2P structure and automates the resource trading. The fairness in resource sharing is administered by a simple incentive-based mechanism.

As mentioned in Section 1, utility computing is a concept of commoditizing computer resources. With heterogeneous resources and different applications/services present in a system, utility computing becomes a complex design problem. However, as it produces a unified interface between different services and heterogeneous resources, it simplifies the resource management in the managerial level and increases the usage of individual resources thus increasing profit per capita. There are a lot of recent interests towards this area: HP's *utility data center* (UDC) [UDC04] is one such project. It pools together the resources of an institution and provides a controller that can create service-specific resource *farms* according to the user specification. Recent effort in the UDC project is to combine the UDC technology with Grid architecture. This provides inter-enterprise resources sharing by connecting the multiple UDCs with Globus toolkit [GrP03] and a control architecture to provide utility-oriented view in such a system [GrK02]. *Cluster on demand* (COD) [ChG03] is a similar project as UDC which can produce service-specific isolated resource collection (called *virtual clusters*) on-demand basis. This project also moves towards combining COD with Grid technology. *Virtual appliances and Collective* [SaL03] from Stanford system lab is another project towards utility computing. It proposes application-specific virtual appliances which are guaranteed-to-work collections of software and hardware. Even though these collections need not be physically together, the virtual appliances can create an illusion as they are.

The other set of projects try to exploit the scalability of the P2P systems in the utility computing arena. However, with P2P architecture, the resource management becomes much more complex and the incentive/trust management becomes vital. With these issues addressed, these projects comes more closer to the concept of

PCU. *Cluster computing on the fly* (CCOF) [LoD04, ZhL04] is such a complete system. It provides mechanisms for organizing the resources in P2P groups, resource discovery, creating utility-type resource collection, and incentive/trust management. However, it is not as decentralized and self-organizing as the Galaxy. *Opus* [BrK02] is another P2P based utility architecture. It creates a dynamic and self-healing P2P *service overlay* over the participating resources which is responsible and capable of creating dynamic *application overlays*. However Opus concentrates more on resource allocation and hence overlook other issues like incentive/trust management which is essential for a PCU made up of public resources.

Despite the increasing number of research works in the area of utility computing, Galaxy excels in many aspects such as looking at the issue in a bottom-up architecture providing fully decentralized self-organizing base architecture, providing layered architecture to handle the issues in modular fashion, and importantly concentrating on exploring the behavior of including public resources. Further, Galaxy is appropriately structured to be marketed in the real-world scenario.

8. Conclusion

References

[AdS99] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," *Operating Systems Review*, Vol. 34, No. 5, Dec. 1999, pp. 186–201.

[AnC03] N. Andrade, W. Cirne, F. Brasileiro, and P. Roisenberg, "Ourgrid: An approach to easily assemble grids with equitable resource sharing," *The 9th Workshop on Job Scheduling Strategies for Parallel Processing*, June 2003.

[BaB02] M. Balazinska, H. Balakrishnan, and D. Karger, "INS/Twine: A scalable peer-to-peer architect ure for intentional resource discovery," *International Conference on Pervasive Computing*, (Zurich, Switzerland), Aug. 2002.

[BrK02] R. Braynard, D. Kostic, A. Rodriguez, J. Chase, and A. Vahdat, "Opus: an overlay peer utility service," *the 5th International Conference on Open Architectures and Network Programming (OPENARCH)*, June 2002.

[But71] A. R. Butz, "Alternative algorithm for hilbert's space filling curve," *IEEE Transactions on Computers*, April 1971.

[CNET3] "CNET.com web page, Desktop reviews," http://www.cnet.com, 2003.

[ChG03] eff Chase, L. Grit, D. Irwin, J. Moore, and S. Sprenkle, "Dynamic virtual clusters in a grid site manager," *the 12th International Symposium on High Performance Distributed Computing (HPDC-12)*, June 2003.

[DiP01] P. Dinda and B. Plale, "A unified relational approach to grid information services," Feb. 2001.

[FoI03] I. Foster and A. Iamnitchi, "On death, taxes, and the convergence of peer-to-peer and grid computing," *2^{nd} International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Feb. 2003.

[FoK01] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *International Journal on Supercomputer Applications*, 2001.

[FoK02] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," *Open Grid Service Infrastructure WG, Global Grid Forum*, June 2002.

[Glo04] "The globus alliance," Home Page, 2004.

[GrK02] S. Graupner, V. Kotov, A. Andrzejak, and H. Trinks, "Control architecture for service grids in a federation fo utility data centers," HP Labs, Aug. 2002. Tech Report.

[GrP03] S. Graupner, J. Pruyne, and S. Singhal, "Making the utility data center a power station for the enterprise grid," HP Labs, Mar. 2003. Tech Report.

[LoD04] V. Lo, D. Zappala, D. Zhou, Y. Liu, and S. Zhao, "Cluster computing on the fly: P2p scheduling of idle cycles in the internet," *The 3^{rd} International Workshop on Peer-to-Peer Systmes (IPTPS 2004)*, Feb. 2004.

[MiM04] A. Mitra and M. Maheswaran, "Resource Pariticipation Models for Public-Resources in Computing Utilities," School of Computer Science, McGill University, May 2004.

[RaF01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," *ACM SIGCOMM*, Aug. 2001.

[RaL98] R. Raman, M. Livny, and M. H. Solomon, "Matchmaking: Distributed resource management for high throughput computing," *The 7^{th} IEEE International Symposium on High Performance Distributed Computing*, (Chicago, IL), July 1998.

[RoD01] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Nov. 2001.

[SaL03] C. Sapuntzakis and M. S. Lam, "Virtual appliances in the collective: A road to hassle-free computing," *The 9th Workshop on Hot Topics in Operating Systems (HOTOS IX)*, May 2003, pp. 55–60.

[Sag94] H. Sagan, *Space-filling curves*, 1994.

[ShC02] S. Shekhar and S. Chawla, *Spatial Databases : A Tour*, 1 Edition, 2002.

[UDC04] HP Labs, "HP utiltiy data center: Transforming data center economics," http://www.hp.com /go /udc, 2004. White Paper.

[ZhL04] D. Zhou and V. M. Lo, "Cluster computing on the fly: Resource discovery in a cycle sharing peer-to-peer system," *the 4^{th} International Workshop on Global and P2P Computing (GP2PC'04)*, Apr. 2004.