MetaGrid: A Scalable Framework for Wide-Area Service Deployment and Management

Muthucumaru Maheswaran, Balasubramaneyam Maniymaran, Paul Card, and Farag Azzedin

University of Manitoba and TR*Labs*
Winnipeg, Manitoba
Canada
E-mail: {`mahes, bmmaran, paul, fazzedin`}`@win.trlabs.ca`

Abstract

This paper presents a novel architecture called the MetaGrid based on Grid computing concepts for resource provisioning for wide-area network-enabled applications. Resource provisioning for wide-area applications can involve coordinated allocation of computing and communication resources. A Grid computing systems provides a virtual framework that facilitates controlled resource sharing among different institutions. The MetaGrid extends the Grid computing systems in two major ways: (a) introduces a notion of SubGrid that provides a coarse-grained resource allocation class and (b) introduces a framework for interconnecting Grids by facilitating peering, trading, and brokering among the different Grids. This paper presents (a) the overall architecture of the MetaGrid with a description of the different functional components, (b) the resource allocation model that is introduced by the notion of SubGrids, and (c) strategies of interconnecting Grids.

1. Introduction

The Internet has evolved into a popular communication medium mainly driven by applications such as E-mail, Net News, and World Wide Web. These applications primarily require the Internet to be a best effort data mover to enable “information exchange.” However, for several existing and emerging applications from areas such as business, entertainment, health, and education, plain data moving capabilities of the Internet are proving to be inadequate [2].

As business-critical applications are network-enabled, users will require anytime and everywhere access to the services offered by the applications. Supporting such access with *quality of service* (QoS) constraints in spite of dynamic variations in resource availabilities is a challenging undertaking. The approaches to address this problem are based on either end-to-end resource reservation or resource provisioning through differentiated resource management. The approaches based on former techniques are known to be less scalable. This paper proposes a novel resource provisioning based approach to support wide-area network-enabled applications.

For wide-area network-enabled applications, resource provisioning can involve coordinated allocation of computing and communication resources. The resource provisioning architecture presented in this paper, called the *MetaGrid*, is based on *Grid computing* [6, 9] concepts. Grid computing systems provide a virtual framework that facilitates *controlled* resource sharing among different institutions. The MetaGrid extends Grid computing systems in two major ways: (a) introduces a notion of *SubGrid* that provides a coarse-grained resource allocation class and (b) introduces a framework for interconnecting different Grids by facilitating peering, trading, and brokering among them.

For the purposes of this paper, we define wide-area applications as applications that require geographically dispersed resource allocations. A SubGrid's resource allocation requirements are provided by resource specifications. A Grid instantiates a SubGrid by allocating resources to fulfill the specified requirements. The SubGrid is considered as a single entity for accounting, billing, and authorization. It is the responsibility of the Grid's *resource management system* (RMS) to insulate a SubGrid from another by ensuring that the resource allocations for the different SubGrids continue to adhere to the different SubGrid contracts.

In addition to the SubGrid notion, the MetaGrid introduces the concept of interconnecting Grids. While Grids are virtual systems that are constructed for resource sharing and can scale to large extents, they can evolve differently based on their specialization. For example, we can have specializations such as academic Grids, commercial Grids, non-profit Grids, data Grids, and computational Grids. These different Grids may have various naming, accounting, billing, management, and access policies. Therefore, we require trading among the Grids to fulfill the requirements of the *resource consumers* (RCs). The MetaGrid facilitates inter-Grid resource trading via Grid peering and Grid brokering. Grid peering arrangements are setup among participating Grids through off-line agreements. Whereas Grid brokering involves on-line transactions among the different Grids to decide how the trading should be performed.

We present the MetaGrid system as a resource provisioning mechanism for wide-area networked applications. Currently, resource provisioning for wide-area applications is performed through off-line agreements to acquire the necessary resources. One of the disadvantages of the off-line schemes is the lack of adaptability. The MetaGrid presents a resource provisioning mechanism for on-line adaptability. Resource provisioning alone is not sufficient to provide adaptability. The wide-area application should have sufficient mechanisms to implement adaptation using the resources provided by the MetaGrid. The adaptation process is exemplified using a popular wide-area networked application called the *content delivery network* (CDN) [11].

A CDN is a virtual network formed by interconnecting geographically dispersed virtual "serving" resources such as surrogate servers, caches, and content routers. The content publishers (originators) requiring efficient and timely delivery of their content to the eventual consumers can outsource the delivery process to the CDN. The CDN is responsible for deploying or acquiring enough resources so that the content is delivered to the content consumers in a timely fashion. Suppose demand for a particular content increases in a region, the CDN should reorganize to boost the serving capacity for the content in that region. To boost the serving capacity, the CDN has to increase the resource allocations for the content in demand. This requires the acquisition of new resources by the CDN and/or reassignment of already acquired resources. The MetaGrid enables an application such as the CDN to dynamically acquire additional resources. The effective usage of the resources already allocated to the CDN is left to the "content management" algorithms used by the CDN.

Section 2 presents a description of the MetaGrid architecture. In this section, the components of the architecture, protocols, and a specification language used for defining the SubGrid structure are briefly described. The resource allocation model introduced by the MetaGrid is examined in Section 3. The reasons for interconnecting different Grids and the need for peering and brokering functionalities are investigated in Section 4. Example applications are described in Section 5. The MetaGrid approach is compared with other systems in Section 6.

2. MetaGrid Architecture

2.1. Overview

The Grid computing systems improve upon previous generation *distributed computing environments* (DCEs) by providing extensibility, adaptability, and site autonomy [6]. Although Grids are designed as scalable overlay systems that can potentially span institutions even from different continents, following factors limit their extent: (a) policies and affiliations of the participating and managing entities, (b) specialization of the Grid to improve the efficiency of selected operations, and (c) physical connectivities of underlying resources. To address these issues we propose to interconnect autonomous Grids via peering and brokering arrangements.

The overall MetaGrid architecture is logically segmented vertically and horizontally. Vertically we split the architectural components into four levels. The major motivation for the vertical split is the manageability of the overall MetaGrid architecture. The major motivation of the horizontal split is to provide scalability, extensibility, and adaptability.

Vertically we have MetaGrid specific components, at the first level. These components can be considered as the "glue" that provide the primitive operations necessary for the different Grids to interoperate. The second level contains Grid specific components that provide Grid services such as authentication, naming, allocation, information, and data access. The third level contains local resource specific components and the fourth level contains the SubGrid specific components.

Horizontally we split the MetaGrid into a network of Grids. Each Grid can be split further into domains with each domain being composed of resources. The resources can be further categorized into three different types based on the mode of attachment to the Grid: (a) fully dedicated, (b) partially dedicated, and (c) on demand.

2.2. Basic MetaGrid Components

Figure 1 shows the basic components of a MetaGrid system along with an illustration of the interactions among them for (a) SubGrid creation and (b) SubGrid access. As explained above the components are placed in four different levels: (a) MetaGrid, (b) Grid, (c) local resource, and (d) SubGrid. Following is a description of the major MetaGrid components:

- *Service Originators* (SOs) have the content or service that require wide-area deployment. They out-source the deployment and management of their wide-area services. The wide-area services execute on the resources provided by the MetaGrid.
- *End User* (EU) is the entity that "loads" the wide-area service being deployed by the MetaGrid. The MetaGrid should provide sufficient resources to the wide-area application being hosted so that the application can adequately serve the EU.
- *Virtual SubGrid Manager* (VSM) is a broker contracted by the SO to manage the resource allocation process. The VSM can use other entities that have specialized knowledge regarding the SO's application to determine the resource requirements to deliver the needed QoS.
- *Access Network* is the network used by the EUs to connect to the wide-area services hosted by the MetaGrid. The access network is assumed to have the functionality for service dissemination and discovery.
- *MetaGrid Service Providers* (MGSPs) form the basis of the MetaGrid administrative structure controlling the induction and expulsion of *resource providers*

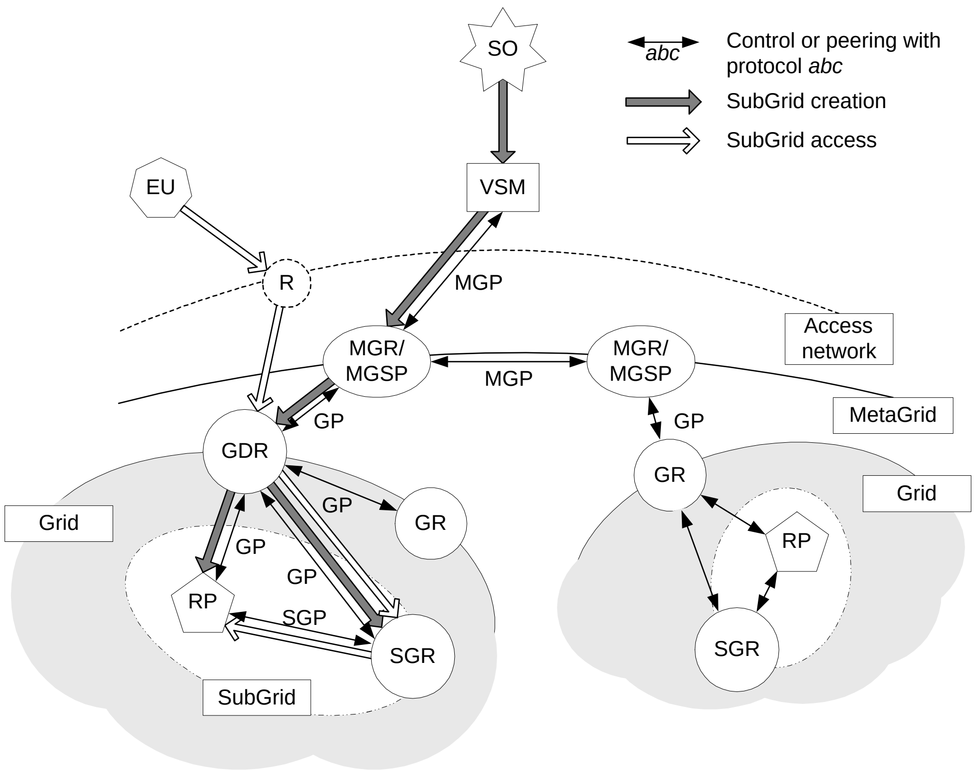

Figure 1. MetaGrid components and typical interoperating scenarios.

(RPs).

- *MetaGrid Resolvers* (MGRs) facilitate peering among the different Grids, gather statistical information on the Grids, and enable the SOs (in general RCs) to contact the most appropriate Grid.
- *Grid Resolvers* (GRs) are resource managers of the MetaGrid. They perform SubGrid resource allocations and continually monitor the resource allocations to ensure that the contracts specified by the SubGrid specifications (as explained below) are not violated.
- *SubGrid Resolvers* (SGRs) are instantiated by the GRs once a SubGrid is created. The SGRs are responsible for service specific resource management.

2.3. MetaGrid Protocols and Operations

2.3.1. MetaGrid Protocols

In MetaGrid, we have three major application-level protocols that are used to exchange messages among the different components. A MetaGrid component should at least speak one of the protocols and some components speak all three protocols. The *MetaGrid Protocol* (MGP), *Grid Protocol* (GP), and *SubGrid Protocol* (SGP) are the three application-level protocols. Following are some of the functions supported by these protocols.

2.3.2. Grid Registration and Selection

As mentioned previously, the MetaGrid facilitates the interconnection of different Grids via peering and brokering. To enable this process, Grids register with the MGRs by providing their attributes and identities. Once the initial registration is performed, the Grids periodically update the MGRs with their status. The status information is used by the MGRs to propose the most suitable candidate Grids when a VSM sends a *request for quote* (RFQ) for the creation of a SubGrid. The RFQ contains specifications for the types resources and the corresponding capacities that a SubGrid should contain.

2.3.3. SubGrid Creation

An SO wanting to host its service on the MetaGrid contacts a VSM. Through a dialogue between the SO and VSM specifications for the SubGrid resource requirements are generated by the VSM. These specifications are expressed in a language described below. Once the *virtual SubGrid specifications* (VSS) are formulated, the VSM negotiates with the MGRs to implement the VSS. The MGRs propose the best set of Grids the VSM should probe further to allocate the SubGrid. The VSM may elect to probe the candidate Grids in any order. The selected Grid is responsible for allocating the resources for the SubGrid.

2.4. SubGrid Specification

One of the major benefits of the MetaGrid approach is the isolation of resource allocations for different wide-area applications using the notion of SubGrid. The resource requirements of a SubGrid are specified using a *Virtual SubGrid Specification Language* (VSSL). A portion of the VSSL grammar in BNF notation is shown in Figure 2. The "demand" for the service for which a SubGrid is requested is abstracted by a notion called the *Anchor points* (APs). An AP represents a set of users in a particular geographical span or represents the centroid for multiple users' access patterns. The weight associated with an AP denotes the demand intensity at the geographical span for the service.

```
<anchor> ::=  anchor_points: (<anchor_attrib>
                                <class>)
<class> ::=  class: (<class_attrib>
                      <child>)
<child> := <class> | <class><child> |
<class_attrib> ::=  nodes: (<node_specs>)
                     <link>
<link> ::= <inter_link> |
           <intra_link> |
           <inter_link><intra_link> |
<inter_link> ::=  inter-class_link: (<link_specs>)
<intra_link> ::=  intra-class_link: (<link_specs>)
<anchor_attrib> ::=  nodes: (<node_specs>)
                  distribution: (<dist_function>)
<node_specs> ::= <node_count>
                  <node_type>
<node_type> ::=  type: (<node_type_def>) |
<node_type_def> ::= <cluster> | <node> |
                     <cluster> OR <node>
<node_count> ::= <abs_nodes> | <node_range>
<abs_nodes> ::=  no_of_nodes: <number>
<node_range> ::=  minimum: <number>
                  maximum: <number>
```

Figure 2. Partial BNF specification for VSSL.

The VSSL specification can be considered as a tree rooted on AP nodes. Extending from the AP are the other resource requirements that are expressed as classes. A class denotes a similar set of resource types that may be geographically distributed with the given connectivity constraints. A class specification includes attributes such as machine type, memory capacity, and bandwidth and delay constraints on the intra-cluster connectivities. Further, it specifies the bounds for the inter-cluster link delay and bandwidth. By specifying the bounds on the links connecting the AP set and the SubGrid nodes, we can prevent the GR from concentrating the resource allocation to a particular network vicinity. Further, these constraints on the connectivities among the APs and the SubGrid nodes ensure that the SubGrid nodes are located to cover the demands originating from the APs at a certain QoS level.

3. MetaGrid Resource Management Model

3.1. Overview

The MetaGrid resource management model examined in this section includes hierarchical and peer-to-peer relationships among its entities as shown in Figure 3. The introduction of SubGrids further complicates the resource management model. A resource that is participating in a SubGrid is also by definition part of the "parent" Grid. We assume that resources do not participate in multiple SubGrids simultaneously, unless there is some policy based isolation like hierarchical fair share queuing.

At the lowest level of the resource management model, we have the different resources with their *local resource management systems* (LRMSs) that schedule the tasks on them. The LRMS reserves the right to reject any task assigned by the upper level resource management. Each resource wrapped by its LRMS is attached to the *Grid resource management system* (GRMS) that is implemented by the GRs. In the MetaGrid resource management model, a resource is restricted to attach to only a single GR. A Grid can have several GRs and the GRs are grouped into different topological or administrative sectors called the *Grid domains*. The functions supported by the GRMS include the following:

- *Resource management:* This involves interfacing with the LRMSs and tracking the load levels of the different resources and assigning resources to SubGrids.
- *SubGrid creation:* The GRMS is responsible for allocating the resources to satisfy the VSSL specification given by a VSM. The GRMS maximizes some measure of "importance" while allocating the different SubGrids.
- *Grid status dissemination:* The GRMS at the GRs disseminate information about their respective domains to each other while the GRMS at the GRs disseminate information about the whole Grid to the *MetaGrid resource management system* (MGRMS).

Once a SubGrid is instantiated, the *SubGrid resource management system* (SGRMS) is launched on it. Because the SubGrids are service specific, the SGRMS also includes service specific extensions that are tailored to suit the requirements of the services being hosted by the SubGrid. Although Figure 3 shows both GRMS and SGRMS as controlling the resources, the GRMS and SGRMS manage the resources in different ways. The GRMS administers the resources joining and leaving the SubGrids. The SGRMS manages the resources that are already allocated for the SubGrid.

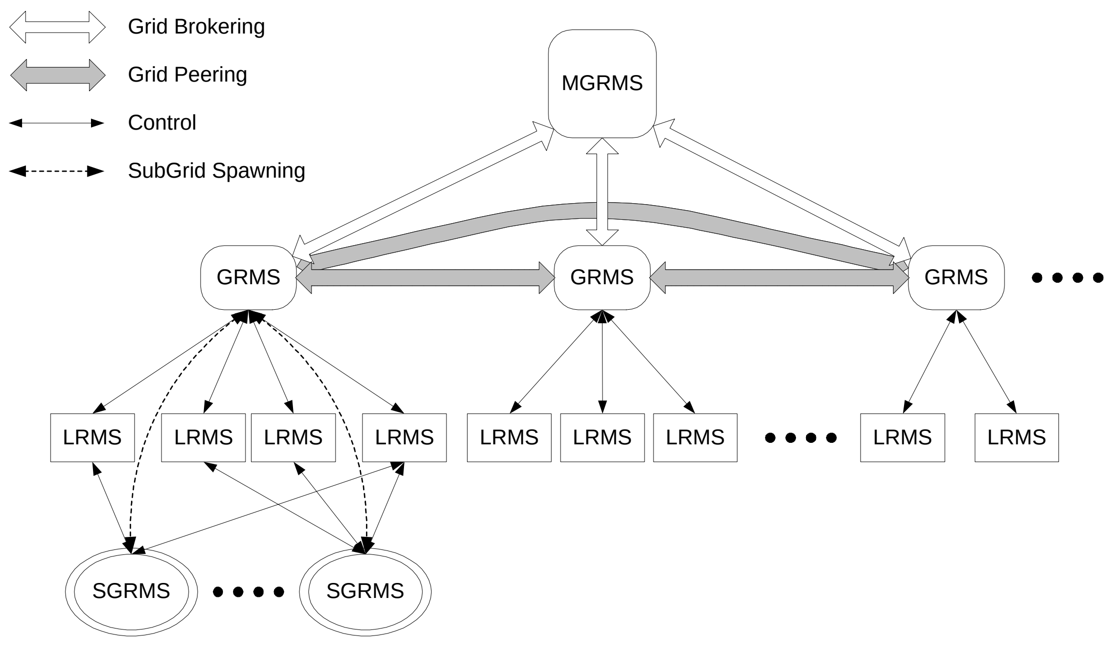

Figure 3. MetaGrid resource management model.

3.2. Constrained Allocation

In the simplest case, resource allocations involve assigning jobs to resources such that the resources are able to complete the jobs in a timely fashion. This process is made harder when the jobs simultaneously require multiple resources. In Grid computing systems, the simultaneous allocation problem is referred to as the co-allocation problem [7]. The co-allocation problem imposes *temporal* constraints among the different resources that are allocated for the set of jobs or set of subjobs that belong to a single job. The *constrained allocation* problem represents a generalized form of allocation where *spatial* constraints are introduced in addition to the temporal constraints that are present in the co-allocation problem.

One of the primary goals of MetaGrid is to provide a framework for resource provisioning for wide-area services. Typically, wide-area services are deployed to serve geographically dispersed demands. Therefore, the MetaGrid should take the locations of the demands for the wide-area services being hosted by the SubGrid while allocating the resources for the SubGrid. Therefore, in the most general case, the constrained allocation problem can include spatial and temporal constraints. However, there can be situations where temporal constraints are not present.

3.3. SubGrid as a Facility Location Problem

VSSL specification for a SubGrid can be considered as a set of constraints that should be satisfied by the resources allocated for the SubGrid. The goal of the allocation process is to find a set of resources with minimum "cost" that satisfies the constraints. Therefore, the SubGrid allocation problem can be posed as an optimization problem or more precisely a *facility location problem* (FLP).

In the general case, the SubGrid allocation problem can differ from the conventional FLP due to reasons including: (a) mutual exclusiveness of the demand nodes (anchor points) and the facility nodes (resources allocated for the SubGrid), (b) intra-resource constraints among the ones allocated for the SubGrid, and (c) layers of resources with different connectivity requirements between among and within layers.

We approach the SubGrid creation as a *fixed charge capacitated facility location problem*, where a cost (f_j) is incurred for locating a facility j and the facility is capable of covering a given amount of demands. If the focus is on simply reducing the number of nodes in the SubGrid, the location cost can be set to unity (i.e., $f_j = 1$). In SubGrid creation, the nodes to be placed are considered as facilities to be located. The demands at the anchor points are characterized by the product of request rate and request length of the service originating at the points.

Mathematically we can formulate the objective function for the SubGrid creation as

$$\text{minimize} \qquad \sum_j f_j s_j + \sum_i \sum_j h_i c_{ij} a_{ij}$$

where h_i is the demand at AP i and c_{ij} is the cost incurred in covering AP i with SubGrid node j. Further,

$$s_j = \begin{cases} 1 & \text{if node } j \text{ is included in SubGrid} \\ 0 & \text{otherwise} \end{cases}$$

$$a_{ij} = \begin{cases} 1 & \text{if node } j \text{ covers node } i \\ 0 & \text{otherwise.} \end{cases}$$

The cost c_{ij} is defined as

$$c_{ij} = \alpha d_{ij} + \frac{\beta}{b_{ij}}$$

where d_{ij} and b_{ij} are the delay and bandwidth, respectively between AP i and SubGrid node j. The parameters α and β are service specific constants.

Based on the above objective function, the SubGrid creation problem can be further developed into a complete fixed charged capacitated FLP by formulating constraints from the given VSSL specifications.

Solving the general FLP has been proved to be NP-complete [3]. Generally, approximation algorithms are used to solve FLPs in polynomial time by applying LP and/or Lagrangian relaxations. However, here we do not prefer this approach because (a) the additional constraints introduced in the SubGrid problem, as described above, further complicates the solution and (b) the approximation algorithms are generally centralized which, in a way, implies an off-line mechanism. But, the primary aim of the notion of SubGrid is to provide a dynamic resource allocation scheme that requires the SubGrid creation process to be on-line. Therefore, we device a distributed heuristic solution technique that is supported by the MetaGrid resource management architecture. However, the target objective function remains the same as the one formulated above.

In the MetaGrid, a MGR decides the Grid to host a particular service and hence to create a SubGrid for the service. One of the GRs of the selected Grid receives the VSSL from the MGR and disseminates it to all the domains within the Grid. The Grid domains individually bid for hosting parts of the required SubGrid to cover subsets of the APs. Based on the bids, GR decide on which domain to host which part of the SubGrid. Here the SubGrid is created in two level: first, the GR applies a heuristic to form the SubGrid considering the domains as coarse-grained facilities, and then the selected domains applies the same heuristic in finer-grain to locate the actual SubGrid nodes.

4. Networks of Grids

4.1. Overview

Although Grids are designed as scalable overlay systems that can potentially span institutions even from different continents, several factors limit the extent of the individual Grids. These factors include (a) policies and affiliations of the entities creating and managing the Grid, (b) specialization of the Grid to improve the efficiency of selected operations, and (c) physical connectivities of underlying resources. For example, academic and not-for-profit research organizations may participate in a research/academic Grid on which a for-profit organization may not be able to directly participate. Similarly, Grids can be specialized as data or computational. Data Grids may focus on integrating databases or data repositories so that clients can have high-performance access to large volumes of data from different geographical locations. The computational Grids, on the other hand, focus on aggregating the computational capabilities of the participating resources such that computationally intensive applications may use these resources via remote computation.

Given these factors that limit the extent of individual Grids, clients with diverse requirements may have to participate on multiple Grids to fulfill their requirements. This may be undesirable due to the additional overhead incured by the clients. In this paper, we propose that the Grids should have mechanisms to *trade* resources among them. The trading mechanisms enable a Grid to locate foriegn resources that are required by a local client without requesting the client to join the foreign Grid. This way a client can retain the simplicity and reduced operational and management overhead associated with single-point Grid connectivity and fulfill its resource requirements.

4.2. Grid Peering

Grid peering is one of the inter-Grid trading mechanisms proposed in this paper. Similar to the peering arrangement in the Internet, Grid peering is defined as the arrangement among different Grids to exchange resource usage rights. In general, the peering arrangements can (a) provide a settlement-free relationship that facilitates controlled access to the resources or (b) provide a controlled but payment based approach to accessing remote resources.

In our Grid peering model, we define a fixed number of *access classes*, $\Lambda_0, \Lambda_1, \ldots \Lambda_{n-1}$. The access classes are ordered such that Λ_i provides more priviledges than Λ_j for all $i > j$. Each resource specifies the minimum access class necessary to access its different functionalities. For example, a supercomputer may limit the number of processors acquired by requests that belong to access class Λ_3 to four while requests belonging to access class Λ_4 may acquire up to 16 processors. The peering arrangement specifies the maximum access class a Grid can use when launching requests onto its peers and the maximum weighted outstanding requests it can have at any given time. By imposing the limit on the weighted outstanding requests, each Grid is motivated to use the minimum access level required for the access.

A Grid may have two limits on the weighted outstanding requests. First limit is for settlement-free access and the second limit is for payment based approach. The limits imposed on the weighted outstanding requests prevent a Grid from excessively loading other peering Grids. This is especially important for preventing a subverted Grid from being used to launch denial-of-service attacks against other Grids.

Grids with less priviledges will end up paying the access fee to other Grids. To bill the appropriate Grids some form of metering should be performed to determine the usages.

4.3. Grid Brokering

Grid brokering is another mechanism for trading resource usage rights among the different Grids. Grid brokering allows the different Grids to interoperate by providing them an intelligent way of routing the requests to the most appropriate target Grid. The Grid peering achieves the same functionality by working agreements among the different Grids that were made offline. The Grid brokering, on the other hand, is an online mechanism for routing the resource requests.

Because Grid peering is based on agreements made offline it is targeted to handle expected average load conditions. Whereas, Grid brokering is meant to handle unexpected overload or unique resource requirements.

5. Example Applications

The MetaGrid characteristics such as horizontally integrated resource managment, service segregation, and time-dependent resource allocations make it an attractive infrastructure for hosting applications that require coordinated allocations in wide-area networks. Examples of applications that can benefit from MetaGrid include CDN, *video-on-demand* (VoD), and *virtual storage wide-area networks* (VSWAN).

CDNs are an emerging paradigm for delivering content (both streaming and non-streaming) to end users such that some measure of QoS is met. The basic idea behind the current CDN deployments is to acquire sufficient resources at the "edge" of the Internet and deliver the content from these resources. To achieve this, the CDN providers install resources at selected locations over the Internet and interconnect these resources so that delivery of the required content is performed from the most appropriate location. The MetaGrid can provide the "server" resource for a CDN. The requirements of the CDN providers should be specified by a VSSL specification and the MetaGrid will allocate the necessary resources. Dynamically obtaining the resources through the MetaGrid can be significantly more cost-efficient for a CDN provider than placing their own resources. Further, it enables the CDN providers to relinquish the usage rights when the resources are not needed.

VoD is another wide-area application that can benefit from the MetaGrid. A VoD deployment needs generous amounts of storage and server capacity to play back the streams as needed. A typical VoD system might require other components such as stream patching nodes besides the storage and play back servers that the MetaGrid can provide on-demand and at time-dependent locations.

Other applications of MetaGrid may include distributed gaming where users can form a SubGrid for themselves to reserve bandwidth among them, virtual private networks where the SubGrid can provide a wide-area autonomous pool of resources, peer-to-peer computing, and distributed interactive simulations.

6. Related Work

The 2K [5] is a network operating system in which all entities (users, devices, etc) exist in the network and are represented by CORBA objects. One of the unique features of 2K is it reconfigures automatically such that at any given time only components absolutely needed by the applications are loaded into the system.

Globus [9, 6] is a toolkit for Grid computing that is being widely used for implementing *computational Grids*. The Globus toolkit provides a set of "core" services on top of which higher level services can be built. The *application level scheduling* (AppLeS) [4] is a network-enabled scheduler that works with a Grid computing toolkit such as Globus to schedule individual applications. AppLeS agents use static and dynamic application and system information while selecting a viable set of resources and resource configuration.

The Jini [10] is a set of protocols for managing dynamic network computing based on Java. It allows services to join a network and discover other services that are available in the network. However, the clients should be Jini-enabled to use this framework.

The Portolano Project [12] proposes developing the following set of enabling technologies for invisible computing: (a) user interfaces, (b) horizontally integrated distributed services, and (c) infrastructure. The MIT Oxygen Project [8, 13] is a large-scale initiative to build a future computing infrastructure that is pervasive, embedded, nomadic, and eternal. One of their approaches is to use a self-organizing network that uses "intent" to locate resources and services.

The WebOS [1] is a network-computing system that provides basic operating system services to build applications that are geographically distributed, highly available, incrementally scalable, and dynamically configurable. The benefits of WebOS were demonstrated by an application called "Rent-A-Server" that used WebOS services to implement dynamic replication of overloaded web services.

While MetaGrid shares the same vision as most of the above mentioned researches, the MetaGrid approach is unique for the following reasons. First, the MetaGrid provides a QoS-aware resource management framework that enables the "staging" of wide-area services to provide better end user experience. Second, the MetaGrid presents a framework for deploying adaptive wide-area services. In essence, MetaGrid proposes a "Rent-A-Grid" approach, which is similar to the "Rent-A-Server" concept [1]. While the "Rent-A-Grid" is harder from a resource management perspective, it addresses issues such as co-allocation, co-reservation, and QoS.

7. Conclusions

In this paper, we presented a novel architecture called the MetaGrid based on Grid computing concepts for resource provisioning for wide-area network-enabled applications. Resource provisioning for wide-area applications can involve coordinated allocation of computing and communication resources. The MetaGrid extends the Grid computing systems in two major ways: (a) introduces a notion of SubGrid that provides a coarse-grained resource allocation class and (b) introduces a framework for interconnecting Grids by facilitating peering, trading, and brokering among the different Grids.

In this paper, we presented the overall MetaGrid architecture with a description of the various components. We also introduced the resource allocation model and defined the resource allocation problem as introduced by the notion of SubGrid. Finally, we examined two different strategies for interconnecting the Grids.

Currently, we are focusing on developing a proof-of-concept prototype of the MetaGrid.

References

[1] A. Vahdat, T. Anderson, M. Dahlin, and C. Yoshikawa. WebOS: Operating system services for wide area applications. In *7th IEEE Symposium on High Performance Distributed Computing*, July 1998.

[2] M. Blumenthal and D. Clark. Rethinking the design of the Internet: The end-to-end arguments vs. The brave new world. *ACM Transactions on Internet Technology*, Aug. 2001.

[3] M. S. Daskin. *Network and Discrete Location – Models, Algorithms, and Applications.* John Wiley & Sons, Inc., 1995.

[4] F. Berman and R. Wolski. The AppLeS project: A status report. In *8th NEC Research Symposium*, May 1997.

[5] F. Kon, R. Campbell, M. Mickunas, and K. Nahrstedt. 2K: A distributed operating system for dynamic heterogeneous environments. In *9th IEEE International Symposium on High Performance Distributed Computing*, Aug. 2000.

[6] I. Foster and C. Kesselman. *Blueprint for New Comupting Infrastructure*. Morgan Kaufmann, San Fransisco, CA, 1999.

[7] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservation and co-allocation. In *Seventh IEEE International Workshop on Quality of Service (IWQoS 99)*, May 1999.

[8] H. Balakrishnan, S. Seshan, P. Bhagwat, and F. Kaashoek. Self-organizing collaborative environments. In *NFS/DARPA/NIST Workshop on Smart Environments*, July 1999.

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

[10] K. Arnold, B. O'Sullivan, R. W. Scheifler, J. Waldo, and A. Wolrath. The Jini specification. In *Addision-Wesley, Reading, MA*, 1999.

[11] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *ACM SIGCOMM Interent Measurement Workshop*, 2001.

[12] M. Esler, J. Hightower, T. Anderson, and G. Borreillo. Next century challenges: Data-centric networking for invisible computing: Portolano project at the University of Washington. In *5th ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 1999)*, Aug. 1999.

[13] M. L. Dertouzos. The future of computing. In *Scientific American*, July 1999.