Service Centric Computing – Next Generation Internet Computing

Jerry Rolia, Rich Friedrich, and Chandrakant Patel

Hewlett Packard Labs, 1501 Page Mill Rd., Palo Alto, CA, USA, 94304
{jerry_rolia,rich_friedrich,chandrakant_patel}@hp.com
www.hpl.hp.com/research/internet

Abstract. In the not-too-distant future, billions of people, places and things could all be connected to each other and to useful services through the Internet. In this world scalable, cost-effective information technology capabilities will need to be provisioned as service, delivered as a service, metered and managed as a service, and purchased as a service. We refer to this world as service centric computing. Consequently, processing and storage will be accessible via utilities where customers pay for what they need when they need it and where they need it. This tutorial introduces concepts of service centric computing and its relationship to the Grid. It explains a programmable data center paradigm as a flexible architecture that helps to achieve service centric computing. Case study results illustrate performance and thermal issues. Finally, key open research questions pertaining to service centric computing and Internet computing are summarized.

1 Introduction

In the not-too-distant future, billions of people, places and things could all be connected to each other and to useful services through the Internet. Re-use and scale motivate the need for *service centric computing*. With service centric computing application services, for example payroll or tax calculation, may be composed of other application services and also rely on computing, networking, and storage resources as services. These services will be offered according to a *utility paradigm*. They will be provisioned, delivered, metered, managed, and purchased in a consistent manner when and where they are needed. This paper explains the components of service centric computing with examples of performance studies that pertain to resources offered as a service.

Figure 1 illustrates the components of service centric computing. Applications may be composed of application and resource services via open middleware such as Web services. Applications discover and acquire access to services via a grid service architecture. Resource utilities offer computing, network, and storage resources as services. They may also offer complex aggregates of these resources with specific qualities of service. We refer to this as *Infrastructure on Demand* (IOD).

M.C. Calzarossa and S. Tucci (Eds.): Performance 2002, LNCS 2459, pp. 463–479, 2002.

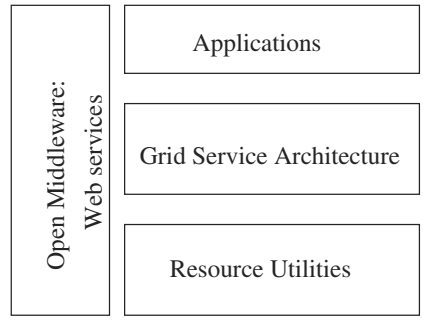

Fig. 1. Service centric computing

Section 2 describes several classes of applications and their requirements on infrastructure for service centric computing. Web service technologies are also introduced. An example of a Grid resource management system and a Grid Service Architecture are introduced in Section 3. Section 4 describes resource utilities, resources as services, and IOD. A programmable data center is introduced as an example of IOD. Next we describe a broad set of technologies that can be integrated to offer IOD with specific qualities of service. Case studies illustrate key issues with regard to IOD. A list of research challenges is offered in Section 5. Concluding remarks are given in Section 6.

2 Applications and Web Services

This section discusses the requirements of applications on service centric computing with a focus on issues that pertain to infrastructure. This is followed by a brief discussion of Web service technologies as an open middleware for service interaction.

2.1 Applications

We consider technical, commercial, and ubiquitous classes of applications and their requirements on service centric computing.

Technical applications are typically processing, data, and/or communications intensive. Examples of applications are found from life and material sciences, manufacturing CAE and CAD, national defense, high-end film and video, electronic design simulation, weather and climate modeling, geological sciences, and basic research. These applications typically present batch style jobs with a finite duration.

Commercial applications may be accessed via Intranet or Internet systems. They often rely on multi-tier architectures that include firewall, load balancer and server appliances and exploit concepts of horizontal and vertical scalability.

Examples of applications include enterprise resource management systems, E-commerce systems, and portals. These applications typically require resources continuously but may require different quantities of resources depending on factors such as time of day and day of week.

With *ubiquitous computing* there is a potential for literally billions of interconnected devices each participating in many instances of value added services. Examples of devices include personal digital assistants, tools used for manufacturing or maintenance, and fixtures. Consider the following example of a value added service for a maintenance process. Instrumentation within tools and an aircraft under service record the steps of maintenance procedures as they are completed to help verify that appropriate service schedules are being followed correctly.

Applications place many requirements on service centric computing. They include the following.

For technical computing:

- Expensive and/or specialized resources need to be easy to share
- Utilize geographically distributed resources effectively
- Share computing, storage, data, programs, and other resources
- Take advantage of underutilized resources

The above requirements have driven the development of grids for high performance computing. The following additional requirements arise for commercial and ubiquitous computing:

- Automate the deployment and evolution of complex multi-tier applications and infrastructure
- Support applications that execute continuously
- Provide access to resources with some level of assurance
- Scale: enable the deployment of many small distributed services that would not otherwise be possible
- Mobility: take computation/data closer to the client(s)

Technical computing applications express demands for numbers of resources and their capacities. For some applications these demands may constrain the topology of a supporting grid's infrastructure – for example requiring the use of high capacity low latency communication fabrics. However the actual topology of a grid's infrastructure is in general deliberately transparent to such applications. We refer to this as *infrastructure transparency.*

In contrast multi-tier commercial and ubiquitous computing applications can require explicit networking topologies that include firewalls and load balancers. Networking topology and appliance configuration may implement security and performance policies and as a result can be explicit features of such applications. As a result such applications are not necessarily infrastructure transparent. They may require changes in infrastructure in response to changes in workload, device mobility, or maintenance.

To reduce the time and skills needed to deploy infrastructure; avoid the pitfalls of over or under-provisioning; and to enable large scale and adaptive

service deployment (due to changing workload demands and/or mobility), commercial and ubiquitous computing applications need automated support for the deployment and maintenance of both applications and their infrastructure. Furthermore such applications need assurances that they will be able to acquire resources when and where they need them.

2.2 Web Services

Web services [1] are a collection of middleware technologies for interactions between services in Internet environments. They are platform and application language independent. The following Web service technologies are likely to be exploited by applications to discover and bind with other application services and infrastructure services.

- XML, descriptive data
- Messaging (SOAP, etc), message formats
- Description of documents (WSDL, etc), interface/data specifications
- Registry (WSIL, UDDI), Directories/lookup

The extended markup language (XML) [2] describes data. The Simple Object Access Protocol (SOAP) [3] is a mechanism for framing data for remote procedure calls. The Web Service Description Language (WSDL) [4] defines type information for data and interfaces. The Web Service Inspection Language (WSIL) [5] and Universal Description, Discovery and Integration (UDDI) [6] offer registry and lookup services.

A business can become a service provider by encapsulating application functionality as a Web service and then offering that service over its Intranet or the Internet. Another application can reuse that functionality by binding with and then exploiting an instance of the service. In a world of Web services, applications will be an integration of locally managed and outsourced services. This is referred to as *service composition*. Grid computing has a natural relationship to the concept of Web services in that it provides resources as a service.

3 Grids and Grid Service Architectures

Today's grids largely support the needs of the technical computing community. However there are efforts underway to further develop the notion of grids to the level of Grid Service Architectures (GSA) that support both technical and commercial applications. The Global Grid Forum [7] is an organization that promotes grid technologies and architectures. As examples of Grids and GSAs, this section describes Globus [8] and the Globus Open Grid Service Architecture [9].

3.1 Globus

Globus is U.S. government sponsored organization formed to develop Grid solutions for high performance scientific computing applications. The goals are essentially those listed for technical applications in Section 2.1. The initial Grid vision for the Globus Grid development was to enable computational grids:

> A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computing capabilities.
> Ian Foster and Carl Kesselman [10]

Figure 2 illustrates the resource management architecture for the Globus grid infrastructure. The purpose of the infrastructure is to bind applications with resource schedulers. Additional Grid services provide support for large file transfer and access control.

From the figure, applications describe their resource requirements using a Resource Specification Language (RSL). They submit the RSL to Brokers. Brokers interact with Information Services to learn about resource availability. An information service receives resource availability information from resource scheduler systems. Brokers may transform RSL iteratively to find a match between the supply and demand for specific resource types. An application may then use co-allocators to reserve access to the resources managed by resource schedulers via an open Globus Resource Allocation Manager (GRAM) interface.

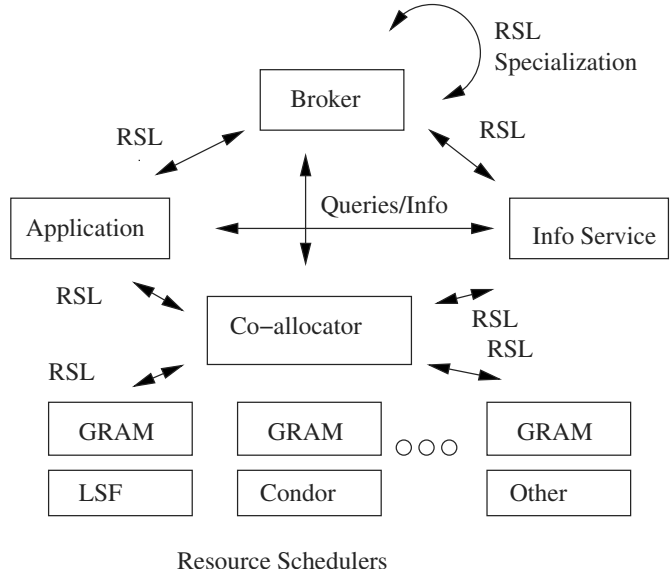

Fig. 2. Globus Resource Management Architecture

There are many examples of resource schedulers including LSF [12], Condor [13] and Legion [14]. A review of scheduling in Grid environments is given in reference [11].

Grids help to enable a utility computing paradigm by providing the mechanisms to match demands for resources with the supply of resources.

3.2 Grid Service Architectures

Recently the notion of integrating the Globus Grid with Web services has developed. The vision for the Grid has evolved to support both e-business and high performance computing applications:

> The Grid integrates services across distributed, heterogeneous, dynamic *virtual organizations* formed from the disparate resources within a single enterprise and/or from external resource sharing and service provider relationships in both e-business and e-science
> Foster, Kesselman, Nick, Tuecke [9]

The integration of Globus and Web services leads to a specification for an Open Grid Services Architecture (OGSA) [9] that treats both applications and resources uniformly as services. This uniform approach is expected to simplify the development of more advanced Grid environments.

Within an OGSA Grid, persistent services accept requests for the creation of service instances. These created instances are referred to as *transient* services. An OGSA governs the creation of and interactions between service instances. A newly defined *Grid service* interface provides a mechanism for service instance creation, registration and discovery, the management of state information, notifications, and management.

A Grid architecture based on Grid services helps to support advanced Grid environments in the following ways. For example, service instances associated with resources may exploit the Grid service interface to implement patterns for joining and departing from resource pools managed by resource schedulers and for propagating resource event information to a resource scheduler. Similarly an application may use the Grid service interface to implement a pattern for asking a resource scheduler for notification of events that pertain to a specific resource.

4 Resource Utilities

In a world of service centric computing, applications may rely on resource utilities for some or all of their resource needs. Today's grid environments offer resources in an infrastructure transparent manner. Yet some styles of applications place explicit requirements on infrastructure topology and qualities of service. In this section we describe programmable data centers as resource utilities that can offer infrastructure on demand. Technologies that contribute to infrastructure on demand are described along with several examples of research in this area.

4.1 Programmable Data Centers

A programmable data center (PDC) is composed of compute, networking, and storage resources that are physically wired once but with relationships that can

be *virtually wired* programatically. Virtual wiring exploits the existing virtualization features of resources. Examples of these features are described in the next subsection.

To support automation and ease of introducing applications into the data center we introduce the notion of virtual application environments (VAE) [15]. A VAE presents an environment to an application that is consistent with the application's configuration requirements. For example, every application has certain requirements regarding server capacity, network connectivity, appliances, and storage services. It can have explicit layers of servers and many local area networks. Network and storage fabrics are configured to make a portion of the data center's resources and back end servers appear to an application as a dedicated environment. Data center management services create these VAEs by programming resource virtualization features. This is done in a manner that isolates VAEs from one another. Later, applications can request changes to their VAEs, for example to add or remove servers in response to changing workload conditions. A PDC is illustrated in Figure 3.

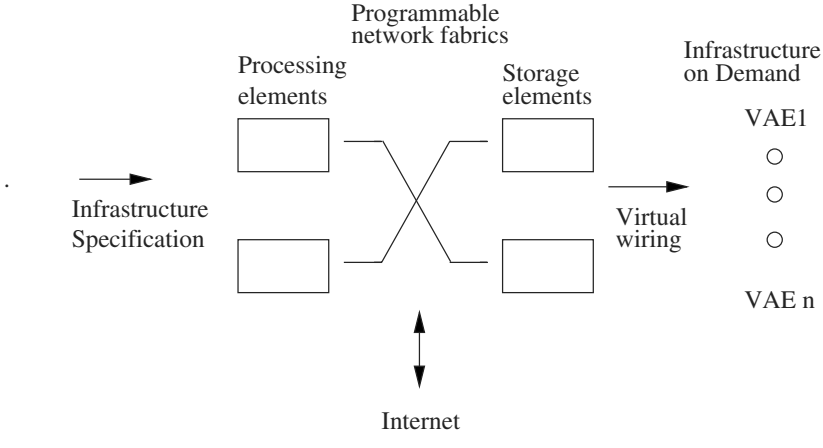

Fig. 3. Programmable Data Center

A PDC accepts a markup language description of the infrastructure required by a VAE. It must perform an admission control test to report whether it has sufficient resources to host the application with its desired qualities of service.

Applications may express requirements for many qualities of service that include: Internet bandwidth, internal communication and storage bandwidths, packet latencies and loss rates, server, storage, and network fabric reliability. They are also likely to require some assurance that it will be possible to acquire resources when they are needed.

Security and reliability are essential features of PDCs. As programmable entities strict controls must be in place to ensure that only valid changes are made to infrastructure and so that VAEs are isolated from one another. Additionally, many applications may rely on shared resources such as PDC network links so the consequences of single failures are larger than in environments without such sharing. For these reasons PDCs must have security and reliability built into their architectures.

Planetary scale computing relies on notions of PDCs and programmable networks. With planetary scale computing PDCs are joined by metro and wide area networking infrastructures providing network capacity on demand. Applications may be deployed across multiple PDCs with resources allocated near to their end-users or where capacity is least expensive. As an application's compute and storage requirements change corresponding changes must be made to network capacities between the PDCs.

4.2 Programmable Infrastructure

This section gives examples of virtualization technologies that can contribute to planetary scale computing. Subsets of these technologies can be combined to provide infrastructure as a service with specific qualities of service.

Programmable Networks. Research on programmable networks has provided a wide range of virtualization technologies. These include:

- *Virtual Private Networks* (VPN)
  - A tunneling mechanism that frames encrypted source packets for transmission to a destination appliance over an insecure network. The encrypted packets are treated as data by the appliance, decrypted, and dropped onto a secure network for delivery to the packet's true destination.
  - Properties: Security isolation
- *Virtual LANs* (VLAN)
  - Ethernet packets are augmented with a VLAN tag header. Ports on appropriate Ethernet switches can be programmed to only accept and forward frames (at line speed) with specific VLAN tags.
  - Properties: Security isolation, classes of service
- *Multiple Protocl Label Switching* (MPLS)
  - Similar to VLAN tags. However these tags can be used by switches/routers to identify specific tunnels of data. Resource reservation, class of service differentiation, and other protocols provide support for fast recovery in the event of network failures and capacity on demand for these tunnels.
  - Properties: Isolation, reliability, capacity on demand
- *Resilient Packet Rings* (RPR)

  - These rings are being used to replace SONET infrastructure in metro and wide area networks. They provide a new frame specification that can carry Ethernet and other traffic, offer security isolation and have special support for fast recovery and capacity on demand.
  - Properties: Security isolation, reliability, capacity on demand
- Optical wavelength switching
  - Optical switches can support switching at the abstraction of wavelengths. In an earlier section we described virtual wiring. Wavelength switching is best described as virtual wire. With appropriate use of lasers wavelengths can each support tens of Gbps of bandwidth and fibers can support hundreds of wavelengths. Lightpaths are end-to-end circuits of wavelengths that provide true performance isolation across optical switching fabrics.
  - Properties: Security and performance isolation, reliability, capacity on demand

Programmable Servers. Since the time of early IBM mainframes server virtualization has been an important feature for resource management. With server virtualization each job or application is isolated within its own logically independent system partition. The fraction of system resources associated with each partition can be dynamically altered, permitting the vertical scaling of resources associated with an application. Server virtualization is a convenient mechanism to achieve server consolidation. Examples of systems that support server virtualization include:

- HP: Superdome and mid-range HP-UX servers [16]
- Sun: Sun Fire [17]
- IBM Mainframes [18]
- Intel [20] processor based servers with VMWare [19]

Server virtualization offers: performance isolation for partitions – when the resource consumption of each partition can bounded; capacity on demand – when the fractions of resources associated with partitions can be changed dynamically; security isolation – depending on the implementation; and can be used to support high availability solutions with redundant, but idle, application components residing in partitions on alternative servers.

Programmable Storage. Today's storage systems are utilities in and of themselves. They can contain thousands of physical disks and have sophisticated management services for backup, self-tuning, and maintenance. Storage virtualization offers the notion of virtual disks (logical units). These virtual disks are striped across a storage system's physical disks in a manner that supports the service level requirements and workload characteristics of application loads.

All major storage vendors support storage virtualization for storage systems accessed via storage area networks. Storage virtualization mechanisms support the management of performance, capacity, availability, reliability, and security.

Programmable Cooling Systems. Data centers contain thousands of single board systems deployed in racks in close proximity which results in very high heat density. Thermal management aims to extract heat dissipated by these systems while maintaining reliable operating temperatures. Air conditioning resources account for 30% of the energy costs of such installations [32].

Today's data centers rely on fixed cooling infrastructures. The PDCs of tomorrow will exploit programmable cooling controls that include:

- Variable capacity air movers
- Variable capacity compressors in air conditioners
- Variable capacity vents and air distribution systems

These features can be used to dynamically allocate cooling resources based on heat load while operating at the highest possible energy efficiency. This type of on-demand cooling is expected to reduce cooling costs by 25% over conventional designs [32].

Programmable Data centers. Programmable data centers provide infrastructure on demand for complex application infrastructures. They exploit the virtualization features of other resources to render multi-tier virtual infrastructure. This permits applications to automate the acquisition and removal of resources in proportion to their time varying workloads.

Examples of programmable data centers include:

- HP Utility Data Center with Controller Software [21], provides integrated computing, networking, and storage infrastructure as a service
- Terraspring [22], provides integrated computing, networking, and storage infrastructure as a service
- Think Dynamics [23], provides a scripting environment to enable the implementation of infrastructure on demand
- IBM Research, The Oceano Project [24], an E-business utility for the support of multi-tier E-commerce applications

These systems help to provide infrastructure on demand. For the commercial PDCs, solutions must be engineered to offer security isolation and specific internal network qualities of service that are required by their customers.

4.3 Case Studies on Infrastructure on Demand

This subsection offers several examples of research and results on infrastructure on demand. We consider *server consolidation*, presenting some otherwise unpublished results that demonstrate opportunities for resource sharing in a commercial data center environment. Next we illustrate the resource savings offered by a utility environment to two horizontally scalable Web based applications along with a mechanism to achieve those savings. We note that commercial applications are unlikely to rely on utilities unless they can receive some assurance that resources will be available when they need them. We describe some recent work

on admission control for PDCs. Next, an example of a self-managing storage system is given.

Planetary scale computing relies on the co-allocation of resources, for example wide area networking as well as data center resources. Mechanisms to achieve co-allocation in Grid environments are described. The concepts of programmable data centers, programmable networks, and co-allocation help to enable wide area load balancing. Control issues regarding wide area load balancing are introduced.

Finally, we present results that pertain to the thermal management of data centers. We describe the importance of thermal management on resource reliability and on overall energy costs.

Server consolidation. Figure 4 presents the results of a server consolidation exercise for 10 identically configured 6-cpu servers from an enterprise data center. Consolidation is based on cpu utilizations as measured over 5 minute intervals for nearly 2 months. An off-line integer programming model is used to map work from the *source* servers onto as few consolidated *target* servers as possible such that the work of only one source server is allocated to a target server or the total per-interval mean cpu utilization on the target server does not exceed 50%. The following factors are considered:

- Number of CPUs per server - The number of cpus per target server: 6, 8, 16, 32
- Fast migration - whether the work of a source server may migrate between target servers at the end of each interval without penalty.
- On-line server migration - whether the pool of target servers can vary in size.

The figure shows the peak number of cpus required with and without fast migration and the mean number of servers required with fast migration and on-line server migration. In the figure, FM represents the case with fast migration while fm represents no fast migration.

We found that fast application migration enables a more effective consolidation but is sensitive to number of cpus per server. As the number of cpus per target server increases fast migration offered no benefit for the system under study. On-line server migration permits us to reclaim unused capacity for other purposes. It permits us to meaningfully characterize a system based on its average number of target servers required because we can use these resources for another purpose. Last, the figure also shows that for the system under study many small servers can be nearly as effective as fewer large servers.

Horizontal Scalability. Ranjan *et. al.,* characterize the advantages of on-line server migration for commercial E-commerce and search engine systems using a trace driven simulation environment [25]. In each case the simulation exploits an algorithm named Quality of Infrastructure on Demand (QuID) that attempts to maintain a target cpu utilization for application servers by adding and removing servers in response to changing workload conditions. The simulation takes into account the time needed to migrate servers into an application (including server

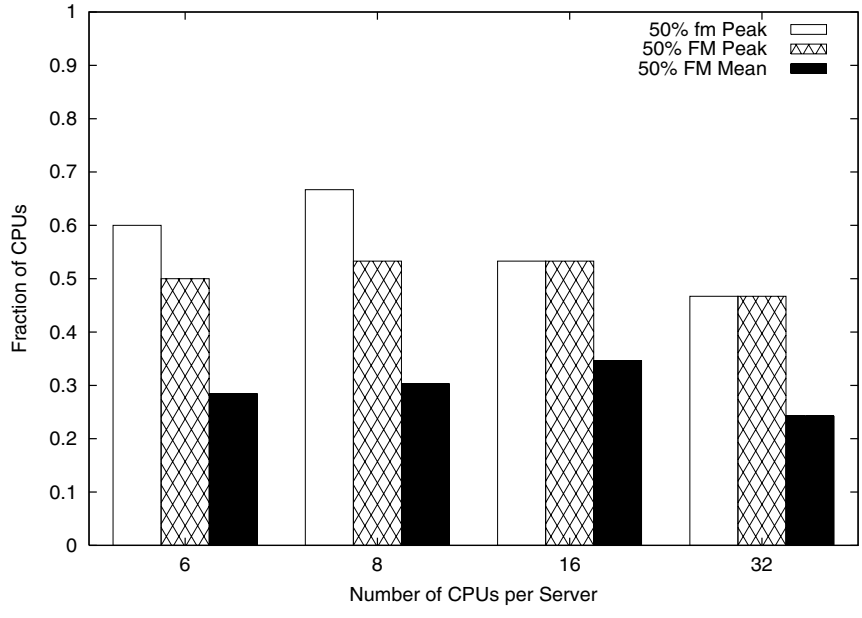

Fig. 4. Fraction of Original Number of CPUs

boot time) and time needed to drain user sessions prior to removing a server. The results show resource savings, with respect to a static allocation of resources, of approximately 29%. In both cases the peak to mean ratio for resource demand was approximately 1.6. A transformed version of the E-commerce trace with a peak to mean of 5 offered resource savings of 68% with respect to a static allocation of resources. We note that the resource savings are expected to be higher in practice since the static cases would have to be further over-provisioned.

Admission control for PDCs. Commercial applications that exploit infrastructure on demand will expect some assurance that resources will be available when they need them. An admission control approach is presented in [26]. The approach characterizes demand profiles for applications based on factors such as time of day. An aggregate of the demand profiles along with application demand correlation information are used to estimate the number of resources needed to satisfy requests for resources with a specific probability θ. Simulations suggest that the technique is relatively insensitive to correlations in application demands as long as they are taken into account when estimating the number of resources required.

Automated storage management. Reference [27] describes tools for automated storage management. Methods are used to automate an initial storage design – the layout of virtual disks over physical disks. Iterative techniques then

exploit online measurements to improve the design while the system operates. Experiments showed performance results within 15% of that achieved by expert administrators.

Co-allocation. A resource co-allocation technique is described in reference [29]. The method extends the Globus resource management architecture with a component for acquiring groups of resources from multiple resource schedulers. A two phase commit protocol can be used to ensure that either all or none of the required resources are reserved.

Such a co-allocation mechanism is particularly important in the context of service centric computing as it may be necessary to acquire resources from multiple resource utilities that include network providers and programmable data centers.

Wide area load balancing. A wide area load balancing system is described in reference [30]. A goal of the work is to balance application demands over servers within and across utilities using distributed, collaborative, self-control mechanisms.

Thermal management for PDCs. Programmable cooling is a smart cooling proposition achieved through modeling, metrology and controls - by charting real-time temperature distribution through a distributed sensor network and modulating the cooling. The cooling resources are dynamically provisioned based on distributed sensing (power, air flow and temperature) in the data center and numerical modeling [31][32][33]. The capacity of compressors, condensers, and air moving devices are varied commensurate with the heat loads present in the data center. An example is an instance when high heat loads prompt an increase in the opening of "smart" cool air inlet vents and a change in speed of air movers and compressors in air conditioners to address a specific area in the data center.

In addition to this dynamic variation in cooling, distributed measurements and thermal resource allocation policies may guide the provisioning of workload within the data center. As an example we may choose to provision a resource that results in the most efficient utilization of cooling resources. In yet another example, workload allocation policies programmatically move workload and shut some systems down in response to the failure of certain air conditioning infrastructure thereby maintaining the reliability of the overall data center. Furthermore, in the context of Grid computing, workload may be provisioned in a global network of data centers based on the most cost effective energy available e.g. on diurnal basis based on the climate - e.g. Bangalore, India at night to provide a more energy efficient condensing temperature for the air conditioning vapor compression cycle. Additionally, the economics of energy production around the globe may be used to drive the choices for global load distribution.

5 Research Challenges

This section offers several performance related research challenges for service centric computing. They address issues of data center design and resource management, resource management for planetary scale computing (federations of infrastructure providers), and general control and validation for these large scale systems. We use the term resource broadly to include information technology and energy.

What is the most efficient, economical data center design?

- What high density, low power, high performance computing architectures most economically support infrastructure as a service?
- What are the simple building blocks of processing, communications and storage that support dynamic allocation at a data center level of granularity?
- What are the implications of commodity components on multi-system designs?

What are the most effective performance management techniques for utility computing?

- What measurement techniques/metrics are appropriate for large scale distributed environments?
- What automated techniques are appropriate for creating models of applications and infrastructure?
- How are models validated for this very dynamic world?

What are the most efficient dynamic resource management techniques?

- What techniques are appropriate for ensuring qualities of service within layers of shared infrastructures?
- What to do about federations of providers (co-allocation)?
- What techniques are appropriate for making good use of resources?

What control system techniques can be applied effectively to this scale and dynamism?

- What automated reasoning techniques can eliminate the complexity of controlling large scale systems?
- What control theoretic techniques are applicable to reactive and predictive events?
- What time scales are appropriate for control?
- How are control measures and decisions coordinated across federated systems?

What is the science of large scale computing that provides probabilistic assurance of large scale behavior based on small scale experiments?

- What is the equivalent to the aeronautical engineer's wind tunnel?
- What behaviors scale linearly?

6 Summary and Remarks

This paper motivates and explains the concept of service centric computing as an approach for next generation Internet computing. With service centric computing: applications, resources, and infrastructure are offered as services. This is essential for the support of commercial and ubiquitous computing applications as it enables the reuse of application functions, server consolidation, and large scale deployment.

As an architecture, service centric computing relies on middleware such as Web services and Grid service architectures as open mechanisms for service creation, interactions, discovery, and binding. Resource utilities offer access to resources and infrastructure. Today's resource utilities in Grid environments typically offer resources in an infrastructure transparent manner. Commercial applications can have explicit dependencies on networking topologies and their relationship with servers, storage, and appliances. These require resource utilities that offer infrastructure on demand.

We believe that there are many opportunities for research in this area. When realized service centric computing will enable new kinds of applications and reduce barriers to market entry for small and medium sized organizations.

7 Trademarks

Sun and Sun Fire are trademarks of the Sun Microsystems Inc., IBM is a trademark of International Business Machines Corporation, Intel is a trademark of Intel Corporation, VMware is a trademark of VMware Inc., HP Utility Data Center with Controller Software is a trademark of Hewlett Packard Company, Terraspring is a trademark of Terraspring, and Think Dynamics is a trademark of Think Dynamics.

Acknowledgements. Thanks to Xiaoyun Zhu, Sharad Singhal, Jim Pruyne, and Martin Arlitt of HP Labs for their helpful comments regarding this tutorial paper.

References

1. www.webservices.org.
2. www.w3.org/XML.
3. www.w3.org/TR/SOAP.
4. www.w3.org/TR/wsdl.
5. www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html.
6. www.uddi.org.
7. www.globalgridforum.org.
8. Czajkowski K., Foster I., Karonis N., Kesselman C., Martin S., Smith W., and Tuecke S.: A Resource Management Architecture for Metacomputing Systems. JSSPP, 1988, 62-82.

9. Foster I., Kesselman C., Nick J., and Tuecke S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. www.globus.org, January, 2002.
10. The Grid: Blueprint for a New Computing Infrastructure, Edited by Ian Foster and Carl Kesselman, July 1998, ISBN 1-55860-475-8.
11. Krauter K., Buyya R., and Maheswaran M.: A taxonomy and survey of grid resource management systems for distributed computing. Software-Practice and Experience, vol. 32, no. 2, 2002, 135-164.
12. Zhou S.: LSF: Load sharing in large-scale heterogeneous distributed systems, Workshop on Cluster Computing, 1992.
13. Litzkow M., Livny M. and Mutka M.: Condor - A Hunter of Idle Workstations. Proceedings of the 8th International Conference on Distributed Computing Systems, June, 1998, 104-111.
14. Natrajan A., Humphrey M., and Grimshaw A.: Grids: Harnessing Geographically-Separated Resources in a Multi-Organisational Context. Proceedings of High Performance Computing Systems, June, 2001.
15. Rolia J., Singhal S. and Friedrich R.: Adaptive Internet Data Centers. Proceedings of the European Computer and eBusiness Conference (SSGRR), L'Aquila, Italy, July 2000, Italy, http://www.ssgrr.it/en/ssgrr2000/papers/053.pdf.
16. www.hp.com.
17. www.sun.com.
18. www.ibm.com.
19. www.vmware.com.
20. www.intel.com.
21. HP Utility Data Center Architecture, http://www.hp.com/solutions1/infrastructure/solutions /utilitydata/architecture/index.html.
22. www.terraspring.com.
23. www.thinkdynamics.com.
24. Appleby K., Fakhouri S., Fong L., Goldszmidt G. and Kalantar M.: Oceano – SLA Based Management of a Computing Utility. Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, May 2001.
25. Ranjan S., Rolia J., Zu H., and Knightly E.: QoS-Driven Server Migration for Internet Data Centers. Proceedings of IWQoS 2002, May 2002, 3-12.
26. Rolia J., Zhu X., Arlitt M., and Andrzejak A.: Statistical Service Assurances for Applications in Utility Grid Environments. HPL Technical Report, HPL-2002-155.
27. Anderson E., Hobbs M., Keeton K., Spence S., Uysal M., and Veitch A.: Hippodrome: running circles around storage administration. Conference on File and Storage Technologies (FAST3902), 17545188 - 284530 January 2002, Monterey, CA. (USENIX, Berkeley, CA.).
28. Borowsky E., Golding R., Jacobson P., Merchant A., Schreier L., Spasojevic M., and Wilkes J.: Capacity planning with phased workloads, WOSP, 1998, 199-207.
29. Foster I., Kesselman C., Lee C., Lindell R., Nahrstedt K., and Roy A.: A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation. Proceedings of the International Workshop on Quality of Service, 1999.
30. Andrzejak, A., Graupner, S., Kotov, V., and Trinks, H.: Self-Organizing Control in Planetary-Scale Computing. IEEE International Symposium on Cluster Computing and the Grid (CCGrid), 2nd Workshop on Agent-based Cluster and Grid Computing (ACGC), May 21-24, 2002, Berlin.

31. Patel C., Bash C., Belady C., Stahl L., and Sullivan D.: Computational Fluid Dynamics Modeling of High Compute Density Data Centers to Assure System Inlet Air Specifications. Proceedings of IPACK'01 The Pacific Rim/ASME International Electronic Packaging Technical Conference and Exhibition July 8-13, 2001, Kauai, Hawaii, USA.
32. Patel, C.D., Sharma, R.K, Bash, C.E., Beitelmal, A: Thermal Considerations in Cooling Large Scale High Compute Density Data Centers, ITherm 2002 - Eighth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems. May 2002, San Diego, California.
33. Sharma, R.K, Bash. C.E., Patel, C.D.: Dimensionless Parameters for Evaluation of Thermal Design and Performance of Large Scale Data Centers. Proceedings of the 8th ASME/AIAA Joint Thermophysics and Heat Transfer Conf., St. Louis, MO, June 2002.