iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 3 **Issue:** IV **Month of publication:** April 2015

**DOI:**

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

# High Performance Computing Meets Grid and Cloud Computing In Hybrid Computing

Mohana Nagalakshmi.B
*Assistant Professor Department of Information Technology, PSNA College of Engineering and Technology, Dindigul.*

***Abstract:*** ***We analyze a hybrid High Performance Computing (HPC) infrastructure architecture that provides predictable execution of scientific applications, and scales from a single resource to multiple resources, with different ownership, policy, and geographic locations. Identify three paradigms in the evolution of HPC and high-throughput computing: owner-centric HPC (traditional), Grid computing, and Cloud computing. Describe the concept of Elastic Cluster and show how it can be used to achieve effective and predictable execution of HPC workloads. Then we discuss Implementation aspects, and propose a new distributed information system design that combines features of distributed hash tables and relational databases***

## I. INTRODUCTION

High Performance Computing (HPC) has a long tradition of leveraging trends and technologies from the broader computing community to yield greater computational power for solving important computational science and engineering problems. While many computer architecture innovations were first motivated by HPC, many other advances in HPC flowed in the other direction, i.e., from the broader computing community to the world of HPC and scientific applications.

In this paper, we examine how another broad development in computing, namely Cloud computing, can best be combined with traditional HPC approaches to offer greater problem-solving resources for scientific workflows. Important features of Grid computing that complement HPC and Cloud computing,we will think through the relationships among these three paradigms – traditional HPC, Grid and Cloud – in terms of their motivation, strengths and weaknesses.Weseek to combine the best attributes of each, and propose a model for how these paradigms can work together and how scientific workloads can be managed in such a hybrid computing environment. Our key contributions are the following: (1) an architecture for hybrid computing that supports functionality not found in other application execution systems; (2) a mechanism for providing statistical resource reservations for batch jobs; and (3) a distributed information system design that achieves guaranteed data discovery, scalability, and fault resilience. The cornerstone of the proposed hybrid computing.

Architecture is the Elastic Cluster, a model of self-resizable computing resources that make up the emerging hybrid computing environment. The remainder of the paper is organized as follows. we introduce the concept of Elastic Cluster as a general model of the computing resources that make up the emerging hybrid computing environment.

## II. CONVERGENCE OF HPC, GRID AND CLOUD COMPUTING

### *A. Attributes*

Today's hybrid computing ecosystem represents the intersection of three broad paradigms for computing infrastructure and use: (1) Owner-centric (traditional) HPC; (2) Grid computing (resource sharing); (3) Cloud computing (on-demand resource/ service provisioning).Each paradigm is characterized by a set of attributes of the resources making up the infrastructure and of the applications executing on that infrastructure. These attributes include: (1) resource ownership: either locally owned resources or externally owned resources; (2) resource accessibility: either private or public ; (3) resource sizing: either quasi-static or dynamic ; (4) resource allocation policy: either exclusive or shared among organizations, groups, projects, or users; (5) application portability: either tied to a specific platform or platform-agnostic. In Grid computing, resources are both locally and externally owned; accessing external resources provides additional capacity and capability, and the mixed blessing of accessing heterogeneous resources. Resource size is dynamic, growing by way of accessing external, public resources. Allocation builds on the methods used in traditional HPC to differentiate between local and external users; Grid applications are a mix of portable and platform-specific, with platform dependencies for the codes that exploit leading-edge HPC architectures. Well-known Grid computing projects include the Globus Toolkit, UNICORE and the NSF TeraGrid infrastructure. In Cloud computing, resources can be either externally owned,

or internally owned, the former being offered by Cloud providers. Public clouds offer access to external users who are typically billed on a pay-as-you-use basis. The Infrastructure as a Service (IaaS) offered by Cloud providers is well suited for HPC workloads. IaaS services include virtual machines and the management interface.

| Attribute | HPC | Grid | Cloud |
|---|---|---|---|
| Capacity | fixed | average to high; growth by aggregating independently managed resources | high; growth by elasticity of commonly managed resources |
| Capability | very high | average to high | low to average |
| Virtual Machine Support | rarely | sometimes | always |
| Resource sharing | limited | high | limited |
| Resource heterogeneity | low | average to high | low to average |
| Built-in Workload Management | yes | yes | no |
| Distribute Workload Across Resources from Multiple Admin Domains | not applicable | yes | no |
| Interoperability | not applicable | average | low |
| Security | high | average | low to average |

Fig. 1. Comparison of key attributes of traditional HPC, Grid, and Cloud.

*B. Combining the paradigms*

Each of the three major computing paradigms has its strengths and weaknesses. The motivation for hybrid computing is to combine all three paradigms so that strengths are maintained or even enhanced, and weakness are reduced. The strengths and weakness of each paradigm are well known and much studied. It is not our purpose here to provide a comprehensive evaluation of each individual approach. However, even a high-level summary of the attributes of each paradigm helps identify areas where combining approaches shows promise. Fig. 1 summarizes some key attributes of the HPC, Grid and Cloud approaches. We note that no single paradigm is the best solution from all points of view. For example, there are important differences in the three paradigms with respect to the ‘‘capacity vs. capability’’ distinction. Capability resources are needed for demanding applications, such as tightly coupled, highly parallel codes or large shared memory applications. Capacity resources are well suited for throughput intensive applications, which consist of many loosely coupled processes. A capacity resource is typically an installation with commodity components; a capability resource is an HPC installation with special hardware, such as low-latency interconnects storage-area networks, many-core nodes, or nodes with hundreds of Gigabytes of main memory. The traditional, owner-centric HPC paradigm excels in handling capability workloads in a well-managed, secure environment. However, capacity is fixed in this domain, and there is typically weak support for virtualization and resource sharing. Strengths of Grid computing include access to additional HPC capacity and capability, which also promotes better utilization of resources. Grid computing enables exposing heterogeneous resources with a unified interface, thereby allowing users to access multiple resources in a uniform manner. However, Grids have limited interoperability between different Grid software stacks.

Moreover, the use of an external resource on a non-fee basis is often associated with limited expectations about the reliability and long-term availability of a resource. While Grids enable a wider choice of resources, the amount of resources of each type is still constant within a non-virtualized Grid. A key strength of Cloud computing is on-demand resource provisioning (thanks to virtualization) which enables capacity resizing, workload migration, better availability and performance. Clouds lag in interoperability, security, and built-in workload management services. The idea of federating separate Grid resources to improve

capacity has been used in many projects, such as GridX1 and Grid Way. For capacity-bound workloads, a Grid-of-Clouds can be used, such as a Grid of Nimbus Science Clouds.

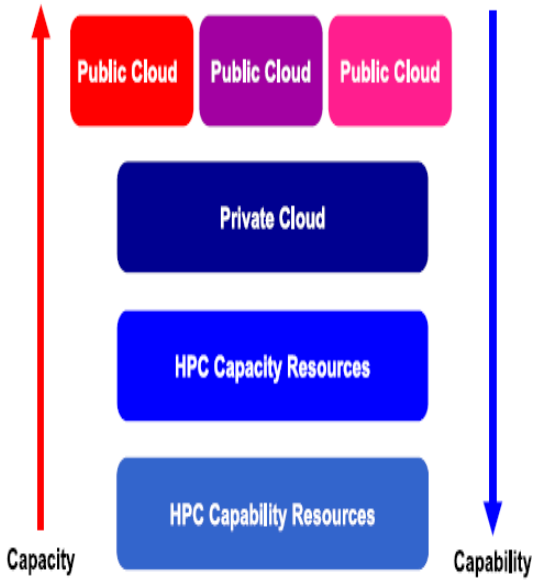



Fig. 2. Hierarchy of resources in the hybrid architecture.

Combining HPC and Grid infrastructures is now common for the execution of many scientific workflows. There are helpful two-way hybrids that result when these paradigms are combined two at a time. Consider again the ‘‘capacity vs. capability’’ question. Traditional HPC clusters are typically used for both types of applications. Hence, HPC, Grid and Cloud should be seen as layers in a hierarchy of resources that may be aggregated, rather than as disparate approaches. Fig. 2 illustrates the hierarchy of resources used in hybrid computing.

However, how to architect a system that combines the strengths of HPC, Grids, and Clouds, it proposes a hybrid infrastructure for the predictable execution of complex scientific workloads across an hierarchy of internal and external resources.

## III. ELASTIC CLUSTERS

We introduce the Elastic Cluster as a unified model of managed HPC and Cloud resources. Our definition of an Elastic Cluster differs from that used by the Open Nebula virtual infrastructure management service. In Open Nebula an Elastic Cluster is a pool of virtual machines that are managed by Open Nebula, while we extend that meaning to include (1) dynamic infrastructure management services (2) cluster level services such as workload management; (3) intelligent modules that bridge the gap between cluster-level services and dynamic infrastructure management services. Until recently, running HPC workloads on virtual machines accessed via a public Cloud was considered impractical for two key reasons: (1) the overhead of virtualization; (2) the virtual machine instances were not backed by a high performance interconnect and storage. Since that time, Amazon has started offering HPC-type machines, as well as a flavor of network-attached storage called Elastic Block Store (EBS) that provides reliable storage, reusable beyond the lifetime of a virtual machine. VM technology on Intel Nehalem processors, has shown that the performance penalty of virtualization is: (1) less than 4% for CPU and memory; (2) zero for a par virtualized network; (3) significant for a hardware-virtualized network. Virtual machines located in a public Cloud can now run effectively HPC workloads that are latency tolerant. The Elastic Cluster supports both virtualized and physical resources.

### *A. Models of dynamic compute systems*

First, we model existing workload and resource management systems as well as existing dynamic infrastructure management systems. By workload and resource management systems we mean systems that manage resources and workloads so that workloads get executed on compute resources according to a set of scheduling policies defined by the system administrators. Fig. 3 shows a

Volume 3 Issue IV, April 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

general model of existing workload and resource management systems (WRMS). A WRMS supports three kinds of functionality: (1) resource management, including resource status and capability; (2) job management, including creation, queuing, monitoring, and control;(3) job scheduling, i.e., mapping a job to a set of resources and allocating the resources to that job for a certain time. Fig. 4 shows a general model of existing dynamic infrastructure management systems (DIMS).A DIMS supports two kinds of functionality: (1) physical resource management, including resource status and capability; and (2) service management, including service creation, monitoring, migration, and destruction. All these systems provision an infrastructure service, whether that service is implemented using a virtual resource or a physical resource.

At the end of screening one obtains a set of resources that are able to execute the workflow component within a cost limit and a turnaround time limit.

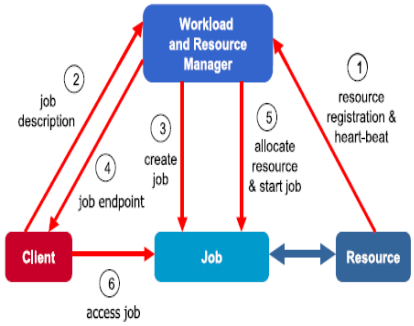



Fig. 3. Model of Workload and Resource Management system

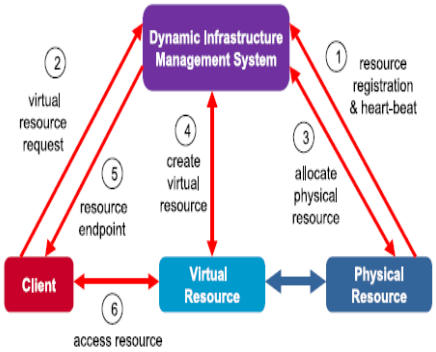



Fig. 4. Model of dynamic infrastructure management systems

In order to obtain workload and resource managers that have the ability to dynamically provision resources, we need a system that combines the features of WRMS and DIMS. A model of such a system is shown in Fig. 5, where the Agent component is responsible for deciding when the WRMS needs more resources in order to satisfy the defined service-level objectives (SLOs). Under the control of the Agent, a DIMS creates and destroys virtual machines. In this paper we present an architecture that provides a robust and efficient implementation of the model shown in Fig. 5. Our architecture is based on a building block called the Elastic Cluster.

*B. Jobs and workflows*

The main goal of the Elastic Cluster is to execute scientific applications such that it satisfies the timing requirements of the applications. The Elastic Cluster supports three kinds of timing constraints: deadlines, advance reservations, and best-effort.A job is the description of a compute or data transfer task, its resource requirements, and its timing constraints. A workflow is a set of jobs with precedence constraints between them.Workflows are abstract representations of scientific applications.

Volume 3 Issue IV, April 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

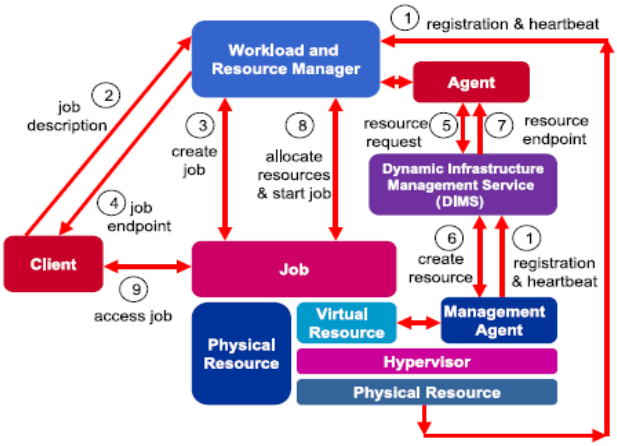



Fig. 5. Model of WRMS with dynamic capacity.

Definition (1) is general, in the sense that it covers deadline constrained jobs, as well as jobs requiring advance reservation and best-effort jobs.

Best-effort jobs can be modeled as jobs whose maximum start time is not guaranteed, but may satisfy some quality-of-service requirement such as the job slowdown not exceeding a threshold. If the job slowdown is guaranteed not to exceed the threshold Slowdown ax then the maximum start time of a best effort job is T s

$j = tR$

$j + Dj \times (SlowDownmax - 1)$.

*C. The concept of Elastic Cluster*

An Elastic Cluster, shown in Fig. 6, is a unified model of managed HPC and Cloud resources. It is a self-resizable resource that adapts to the workload to meet the timing requirements of the applications. Unlike traditional HPC clusters, where the worker nodes are statically provisioned, the Elastic Cluster automatically provisions additional resources leased from a private or public Cloud. The Elastic Cluster is an extension of the model shown in Fig. 5 that features predictable job execution and support for integration with higher-level services for federating Elastic Clusters. The WRMS provides two key services: (1) job submission, monitoring and control; and (2) resource monitoring and control. The worker nodes of the Elastic Cluster can run either traditional physical machines denoted by PM (for Physical Machine) in Fig. 6, or virtual machines, denoted by VM (for Virtual Machine) in Fig. 6.

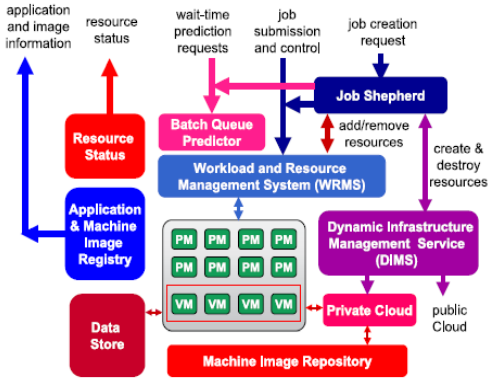



Fig. 6. The Elastic Cluster architecture.

Jobs enter the Elastic Cluster through a Job Shepherd who puts them in a submit queue that it maintains, decides when to dispatch them to the WRMS, and manages them in order to achieve two goals: (i) meeting the timing constraints of the jobs; and (ii) enforcing the job dependencies. To achieve the first goal, the Job Shepherd creates statistical reservations, and controls the dynamic

resizing of the cluster relying on the DIMS to provision resources by leasing them. To achieve the second goal, the Job Shepherd controls the time when job j becomes ready for execution tR j : An Elastic Cluster can be a building block for a multi-site workflow planning and execution system. To enable this, the Elastic Cluster provides information about what applications and machine images are available locally via the Application and Machine Image Registry, which describes the contents of the Machine Image Repository.Each Elastic Cluster has a Data Store, which is a high performance data storage that offers a standard API or protocol for remote access. It can be implemented as a centralized (e.g., NFS) or distributed (e.g., GPFS or Lustre) file system, and it is exposed to remote clients with a secure protocol such as GridFTP, or for better performance, striped GridFTP .

*D. Predictable execution*

The Elastic Cluster provides predictable start time for the jobs and enforces the job dependencies specified by the workflows.When we say ‘‘predictable start time’’ we mean the start time that meets the requirements of each type of job. Our approach combines statistical reservation or advance reservation with on-demand provisioning of resources leased from the Cloud.Upon getting the endpoints of the resources from the DIMS, the Job Shepherd adds them to the reservation and submits job j to the WRMS.

Algorithm 1. Provision and Submit an Advance Reservation Job

Input: Job description Desc(j) = ・Resj, Execj, Envj, Depj, T s j , Dj・
Output: The ID of the job obtained from submission to the WRMS

1. Requestj = ・Resj, T s j , T s j + Dj・
2. reservation_ID = get reservation( Requestj )
3. if (reservation_ID < 0) then // need to lease Cloud resources
4. count = get_resource_count ( Requestj )
5.(count_avail, reservation_ID) = get_partial_reservation (Requestj)
6. count_leased = count − count_avail
7.res_endpoints=lease_resources (Resj,count_leased, T sj , T sj + Dj)
8. add_resources_to_reservation(reservation_ID,
res_endpoints )
9. end if
10. job_ID = submit_reservation_ID(j, reservation_ID)
11. return job_ID

If a job is deadline-constrained or best-effort, then the Job Shepherd determines a statistical reservation for the job and submits it to the WRMS.

Algorithm 2. Provision and set submit time for deadline-based and best-effort jobs

Input: Job description Desc(j) = ⟨Resj, Execj, Envj, Depj, T s j , Dj⟩ and Plb
Output: Job description augmented with the job submission time tsub j

1. Requestj = ⟨Resj, T s j , T s j + Dj⟩ // tsub j is the maximum submit time such that p(T s j , tsub j ) ≥ Plb
2. tsub j = get_submit_time_from_queue_predictor( Requestj )
3. if (tsub j < 0) then // need to lease Cloud resources
4. count = get_resource_count ( Requestj )
5.(count_avail, tsub j ) = get_submit_time_partial_resources (Requestj)
6. count_leased = count − count_avail
7.res_endpoints = lease resources(Resj, count_leased, 0, T s j + Dj)
8. add_resources_to_job(j, res_endpoints )
9. end if

10. set_job_submit_time(j, tsub).

In Algorithm 2 is that a job should be submitted to the WRMS as late as possible – to allow other jobs with earlier deadlines to start – but not too late, in order to be able to start it by the required deadline.

Algorithm 3. Auto-release Virtual Machines
Input: Virtual machines that are currently up
Output: Virtual machines that are kept up for each scheduling cycle

1. while ( true ) do
2. now = get_time ();
3. jobs = get_jobs_with_start_time(now, now + S);
4. free_vms = get_idle_virtual_machines();
5. for vm in ( free_vms ) do
6. if (!vm_can_be_reused(vm, jobs)) then
7. Puse(vm) = vm.stat_use × S/(now + S −
vm.create_time);
8.Cup=(now−vm.idle_time+S(1−Puse(vm)))×vm.rate;
9. $\tau = 3S/2 + vm.SU - \Delta s\ avg$ ;
10. if ($\tau \leq 0$) then
11. Cdown = 0;
12. else
13. Puse(vm) = vm.stat_use × S/(now + S −vm.create_time);
14. Cdown = $\tau \times (C - C') \times vm.cores \times Puse(vm)$;
15. end if
16. if (Cup > Cdown) then
17. shut_down(vm);
18. end if
19. end if
20. end for
21. suspend_until(now + S);
22. end while

## IV. RELATED WORK

We review related work, starting with the two approaches most similar to ours, and compare existing approaches with the approach that we propose. The SDM module can manage a variety of services. For example, it interfaces to the Amazon EC2 service by way of an adapter. Using a set of Service-Level Objectives (SLOs) about the quality of service provided by SGE, SDM makes decisions about what EC2 resources to create, and acts upon these decisions using the EC2 adapter. Both the Elastic Cluster and SGE provide a WRMS with dynamic capacity. Unlike SGE, where the SDM module is responsible both for deciding when resources are to be acquired or released and for dynamically acquiring or releasing resources, our architecture assigns these two responsibilities to different components: the Job Shepherd and DIMS. This allows us to support a wide range of dynamic infrastructure management systems and to create multiple Job Shepherds that interface to a given DIMS. Furthermore, we support creating a personal WRMS in order to effectively manage high-throughput workloads. Essentially, OpenNebula is a solution to managing a dynamic pool of virtual resources, called a virtual cluster; it is able to dynamically provision virtual resources such as virtual machines. OpenNebula can be used as the DIMS component of the model of WRMS with dynamic capacity depicted in Fig. 5, but does not address the issue of integrating infrastructure management with workload management, e.g., when should a virtual cluster grow with new resources. Some HPC sites may want a dynamic infrastructure, but may not be willing to resort to virtualization. For these sites, the Extreme Cloud Administration Toolkit (xCAT) an open-source project with major contributions from IBM, is an alternative to

OpenNebula. xCAT can provision dynamically both physical and virtual resources. A special case implementation of the managed computation model is the personal cluster. In a personal cluster, the worker nodes are allocated to the user's workflow for the entire duration of the workflow, and these worker nodes are managed by a resource manager instantiated for the user's workflow. Hence a personal cluster gives an illusion to the user as if the instant cluster is dedicated to the user for a certain period of time .A managed computation factory is often backed by a resource manager that is responsible for creating and managing computations.

Virtual clusters, unlike personal clusters, can dynamically change their size it have proposed a virtual cluster approach called Cluster-on-Demand (COD). COD partitions a physical cluster into multiple virtual clusters configured according to the application requirements. Each virtual cluster runs a clusterlevel service, which for HPC workloads is a WRMS. COD interacts with the WRMS via a Virtual Cluster Manager (VCM) assigned to each virtual cluster. The VCM decides when worker nodes should be added or removed from the virtual cluster, and sends the corresponding requests to the COD. In response to requests from the VCM or because of site policy, the COD provisions nodes for a virtual cluster or reclaims nodes and instructs the VCM to carry out the WRMS configuration actions for adding or removing worker nodes.

## V. MANAGING HPC WORKFLOWS ACROSS ELASTIC CLUSTERS

The maximum size of an Elastic Cluster is determined by several factors related to capacity, performance, and cost constraints. High availability and security requirements may also lead to using multiple Elastic Clusters. Since not all the services of an Elastic Cluster may be highly available, having multiple Elastic Clusters effectively replicates the service of job execution. Second, an Elastic Cluster will often accept a certain set of ''trusted'' machine images, thereby limiting the set of applications that can be run on that Elastic Cluster. Hence, a complex scientific workflow may have to execute on more than one Elastic Cluster.

In this section we show how to build on top of Elastic Clusters an infrastructure for the management and execution of workflows across multiple resources. Workflows are submitted to a workflow queue from which they are fetched by the workflow manager.

The Workflow Manager processes the components of a workflow in order to map them to the resources on which the components will execute. The super scheduler receives the description of a workflow component from the workflow manager, and determines the best resources to which to map the workflow component such that: (1) the cost of running the workflow component does not exceed the limit specified by the user ; and (2) the turnaround time of the workflow component does not exceed the limit required by the user. In order to make this decision, the super scheduler uses information about the resources, the applications to be executed, as well as statistics gathered from previous executions of the applications in the workflow, and a resource-specific cost model.The process of resource mapping can be divided in two stages:(1) resource screening: finding the resources capable of executing a workflow component; (2) resource selection: selecting the best resources out of those determined during screening. To perform screening, the super scheduler consults the Application and Machine Image Catalog which aggregates the information provided by the Application and Machine Image Registries associated with the Elastic Clusters and the Cloud providers. In the resource selection phase, the super scheduler selects the most suitable resource mappings for the workflow component. To achieve redundancy, and because of the imprecision in the estimates, the super scheduler is configured to present multiple mappings for a workflow component. Once the workflow manager has mapped a workflow component, it inserts it into the queue of a (partially) mapped workflow. The Workflow Engine is responsible for fetching these mapped workflow components from this queue and sending them to a message service which acts as the communication layer between the workflow management system and the Elastic Clusters.

## VI. IMPLEMENTATION

Upon request from the Job Shepherd, the DIMS allocate and instantiates the requested number and type of virtual machines (VMs).The DIMS provisions resources by leasing them from a Cloud. A lease is defined by a start time, an end time, and the properties of the leased resources. For the duration of the lease, the virtual machines appear as being part of the cluster subnet. In order to make the VMs appear as part of the cluster network, the DIMS configures a Virtual Private Network (VPN) server. Describe a new design of a distributed information service that achieves effectiveness, scalability, and reliability.

Our distributed information service design, called Key-Partitioned Database (KPAD), can be described as follows: (1) Information is distributed across multiple relational database tables, each table being managed by a host; (2) A distributed hash table (DHT) scheme is used to locate the host corresponding to the table; In KPAD, we use a lookup mechanism that is both structured and symmetric. Unlike purely symmetric approaches such as super-peers , KPAD achieves both guaranteed data discovery and fault resilience.

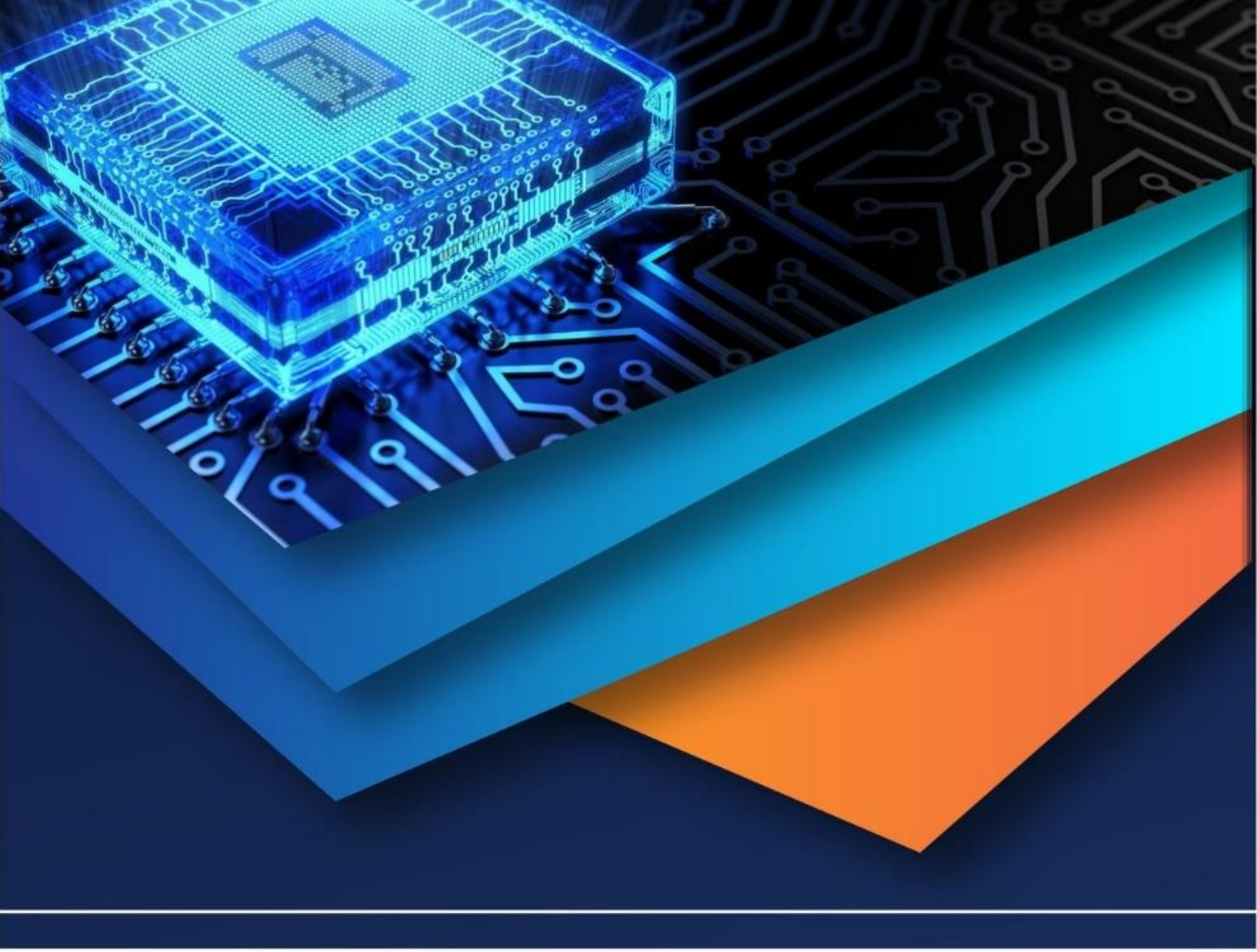
One can think of KPAD as a generalization of the DHT technique.In a DHT, the basic operations are: (1) name resolution (lookup):given a key k find the machine node that stores the key-value pair,i.e., node = lookup(k); (2) get value by key: get from machine node the value v associated with k: v = get(node, k); and (3) write the value v associated with key: put(node, k, v).DHTs provide an effective solution for managing large keyvalue stores. However, DHTs only supports exact-match queries rather than the kinds of queries often applied to information systems, such as finding the resources that have the value of a certain attribute in a specified range.To provide the full power of SQL, we use DHT as the name resolution layer on top of a distributed relational database. The information stored in the database is about entities and consists of one or more attributes associated with each entity. A database table is created for each entity, and the name of the table is the name of the entity. Table names are the keys passed to DHT to locate the host (or hosts, in the case of replication) that stores the tables.

## VII.CONCLUSION

We have presented a hybrid computing architecture and a set of strategies for achieving effective and predictable execution of HPC workloads on a hierarchy of resources. The proposed architecture is an HPC-oriented tailoring of Cloud computing that enables harnessing resources from multiple providers into a cohesive aggregate. Our approach recognizes that IaaS is the most flexible Cloud service model, and that access to this service is done via Cloud management middleware.We have described three kinds of strategies: (1) scale-up and scale-down, e.g., increasing and decreasing the size of an Elastic Cluster to adapt to changes in workload and to meet quality of service objectives; (2) scale-out, e.g., distributing the workload across multiple Elastic Clusters; and (3) personal virtual cluster, e.g., setting aside resources for a user or workflow. The first strategy is also called cloud bursting, and is currently used by providers of hybrid Cloud environments. However, elasticity is more than cloud bursting; we have shown that there are other techniques, which originate from cluster and Grid computing, that contribute to creating an elastic environment.We have sketched the implementation of the proposed architecture.For the distributed information service, we have described a new robust and scalable design.

## REFERENCES

[1] E. Deelman, D. Gannon, M. Shields, I. Taylor, Workflows and e-science: an overview of workflow system features and capabilities, Future Generation Computer Systems 25 (5) (2009) 528–540. doi:10.1016/j.future.2008.06.012.
[2] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organizations, International Journal of Supercomputer Applications 15 (3) (2001).
[3] I. Foster, C. Kesselman, Globus: a metacomputing infrastructure toolkit, The International Journal of Supercomputer Applications and High Performance Computing 11 (2) (1997) 115–128.
[4] Globus Alliance, Globus Project, December 2010. URL: http://www.globus.org/.
[5] A. Streit, UNICORE: getting to the heart of grid technologies, in: UNICORE Project, 2010.
[6] TeraGrid, TeraGrid project, December 2010. URL: http://www.teragrid.org/.
[7] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (6) (2009) 599–616.
doi:10.1016/j.future.2008.12.001.
[8] Amazon.com., Amazon Elastic Compute Cloud (EC2), December 2010. URL:
http://aws.amazon.com/ec2/.
[9] Amazon.com, Amazon Simple Storage Service (S3), December 2010. URL: http://aws.amazon.com/s3/.
[10] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-
degree compared, in: Grid Computing Environments Workshop, 2008, GCE-08,
2008, pp. 1–10. doi:10.1109/GCE.2008.4738445.
[11] G. Mateescu, A. Agarwal, R. Sobie, B. Arnaud, GridX1: a Canadian computational grid, Future Generation Computer Systems 23 (5) (2007) 680–687. doi:10.1016/j.future.2006.12.006.
[12] E. Huedo, R. Montero, I. Llorente, The gridway framework for adaptive scheduling and execution on grids, Scalable Computing—Practice and Experience 6 (3) (2005) 1–8.
[13] K. Keahey, M. Tsugawa, A. Matsunaga, J. Fortes, Sky computing, IEEE Internet Computing 13 (5) (2009) 43–51. doi:10.1109/MIC.2009.94.
[14] R. Moreno-Vozmediano, R.S. Montero, I.M. Llorente, Elastic management of cluster-based services in the cloud, in: First Workshop on Automated Control for Datacenters and Clouds, 2009, pp. 19–24. doi:10.1145/1555271.1555277.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY