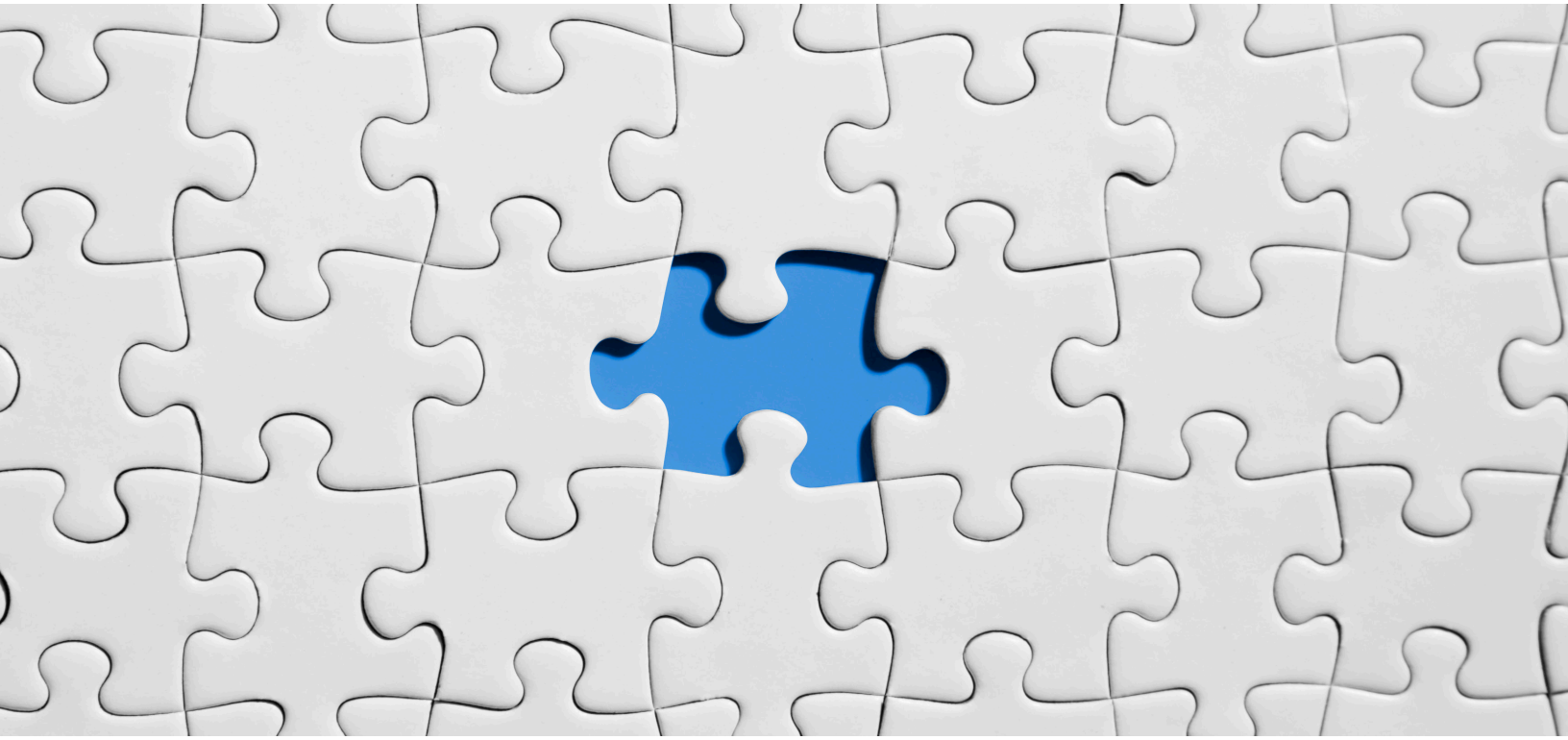
DELL Technologies

OPTIMAL LOAD BALANCING IN MULTI-CLOUD: A JSQLW APPROACH



Shikha Pandey, PhD

Advisor, Learning Design & Development
Dell Technologies | Education Services
shikha.pandey2@emc.com

Knowledge Sharing Article

DELL Technologies Proven Professional

Table of Contents

Abstract

Digital transformation has paved the way for new technologies, cloud being among them, that help accelerate an organization's path toward IT Transformation. To meet business demands and overcome the challenges of public and private cloud, many organizations have started to adopt a multi-cloud approach. A federated infrastructure, multi-cloud is an ephemeral interconnection of clouds, each laying out its own set of resources and exploiting different authentication mechanisms.

This Knowledge Sharing paper focuses on the importance of multi-cloud in today's technological world. Also presented is an optimal mechanism of resource allocation and load balancing named as JSQLW (Join Shortest Queue with Left Work) for the heterogeneous cloud environment. The proposed algorithm is designed to schedule the resources based on the estimated left work of the data center and provides an effective mechanism for load balancing.

A new composition and aggregation model is needed because of the diversity and complexity of potential compositions of cloud services. The multi-cloud approach not only provides a common operating environment for both public and private clouds, it also provides insight to monitoring, deployment, migration, automation, and management of applications and data across these clouds. VMware's Cross-Cloud Architecture is one of the best examples which has enhanced enterprise abilities by applying network virtualization technology to public clouds, extending data center networking and security to the cloud, preserving on-premises network rules, and deploying secure network architectures that span multiple clouds.

From a broader perspective, multi-cloud was not only necessary to enable organizations to avoid cloud silos and unify different deployment models, it was also required to:

- Provide a concrete approach capable of large-scale big data processing
- Support application deployment in heterogeneous cloud environments
- Seamlessly use resources and services from multiple providers
- Overcome issues of proprietary vendor APIs and vendor lock-ins
- Address the security challenges
- Meet changes in communication behavior to accommodate different semantics, charging models, and SLA terms
- Solve resource sharing and utilization issues among increasing number of cloud platforms and in aggregated cloud platforms

Though the cloud is a network-based paradigm, network use increases as collaboration between the private cloud and public cloud grows. Changes in communication behavior to accommodate different semantics, charging models, and SLA terms in multi-cloud scenario has created challenges that need to be addressed; key among them being resource sharing and resource utilization. To solve the resource utilization issue, the proposed algorithm JSQLW (Join Shortest

Queue with Left Work) provides an efficient mechanism for scheduling and load balancing across the cloud platforms. The algorithm is designed to schedule the resources based on the estimated left work of the data center which can be determined dynamically based on the environmental condition of the data center and the VMs processing time.

Since the entire process of balancing and resource allocation is considered on the multi-cloud, the left work is assumed to be proportionate work of data centers of different cloud providers. Each server (data center) is also allocated with different weights to meet different requests depending on their costs.

With the main focus on scheduling, not only for resource allocation but also for efficient load balancing, the mechanism can also be applied on cloud infrastructure for management of the heterogeneous VM Images. The proposed work is based on the combination of Transaction-Join-Shortest-Queue (TJSQ) algorithm and Transaction-Least-Work-Left (TLWL) algorithm which mainly achieves its performance by combining the features of dynamic estimates of server load, transactions, and exploiting differences in processing costs for transactions in data centers.

The JSQLW algorithm is a load balancing mechanism unique to all cloud environments, not just multi-cloud environments. It provides finer-grained load balancing than other standard mechanisms, resulting in improved throughput improvements and response-time. We will present a detailed analysis, design, implementation, and evaluation of the mechanism and compare the proposed algorithm to several well-known load balancing approaches.

The proposed mechanism not only provides an efficient mechanism for resource allocation and load balancing across multiple data centers of multi-cloud, it also fulfills the vision of any device, application, and cloud message to simplify IT operations and increase resource sharing and resource utilization agility, efficiency, and productivity.

Introduction

The computing world is changing the way we interact, manage, access, and deliver services. The spotlight has shifted from personal to data center-centric computing and multi-cloud. With wide adoption of cloud by enterprises, multi-cloud has a tremendous impact on IT industries. Along with opportunities due to its diverse characteristics for enterprise and end users, multi-cloud brings implications in perspective of both technical and non-technical environments.

Multi-cloud mostly exhibits dynamic characteristics. To achieve a high user satisfaction, increased throughput, minimum response time, and efficient resource utilization ratio there is a need of proper load balancing process [1][2][3][4][5]. An optimal load balancing mechanism for multicloud environment is proposed in this paper to achieve these objectives. The main aim of the mechanism is to balance load based on the migration of estimated left work of the data center which can be determined dynamically.

Related work and Research perspective

Various scheduling methods have been analyzed in this paper which results in lowering the energy consumption. The analyzed algorithm reduces energy consumption to some extent, so they proposed a" crossbreed algorithm" to minimize energy consumption. Round-robin, FCFS and Priority-based scheduling are the scheduling algorithms which are analyzed [7]

A fine-grained cross-cloud domain trust model with resource-sharing capabilities between domains across distinct homogeneous clouds have been proposed by Pustchi. N et.al. They have implemented a proof of concept with extending OpenStack identity and federation services to support cross-cloud domain trust where the adopted approach does not introduce authorization overhead within current models. Also, the entire mechanism depends upon the Mapping rules. [8].

Elkhatib.Y in his work analyzed the need for development of APIs for cross-cloud. According to him the application deployed in cross-cloud is one that consumes more than one cloud API under a single version of the application[9]

Huioon K. et.al have too laid their focus on the need of integrated APIs. They have developed a mechanism to overcome the issue of heterogeneity of the APIs by developing integrated cloud API to provides a RESTful access.[10].

A resource sharing process between two different tenants (cross-cloud) with cloud resource mediation service has been proposed by Allam Q.et al. The entire mechanism is defined by activation, delegation, forward revocation, and backward revocation along with their formal verification. [11].

An "orchestration algorithm" to reach migration of a component between different providers in an agnostic way has been determined by Dur´an F. and Pimentel E. The main characteristics of this algorithm is that along with the stateless, it is too not bounded to any service level of any particular provider which allows it to overcome vendor lock-in issues. To ensure agnosticity the proposed algorithm is built over the concept of trans cloud, i.e. bidimensional cross-cloud.[13].

EXFed, an efficient cross-federation system for IaaS clouds that "migrates" process/application between different clouds, has been proposed by Pucher A. EXFed also provides ahead-of-time certainty about resource availability despite retaining individual clouds' ability to preempt foreign workload after admission [14].The paper determines the extensions to the Federated CloudSim framework that considerably improve simulation and evaluation of cross-cloud. Service level agreements (SLAs), scheduling and brokering strategies on various levels have been tremendously used for the purpose of evaluation [15].

A cross-layer scheduling framework is proposed for resource allocation. Their approach computes the placement/allocation and routing paths for the new job. The process of routing adopted for determination of routing path is between structured and random topology [16].

The work of Theng D., and Hande P. deals with the issue of VM images scheduling in a cross cloud computing environment. An efficient approach for VM management is proposed and implemented to overcome the issue of cross-cloud scheduling. FSS and HFSS (Factorizing and heterogeneous factorizing self-scheduling) are two scheduling proposed [17],

An approach based on self-organizing multi-agent system to achieving cross-clouds services management, including the service provision at the tenant-end and services aggregation at the cloud-end. Clouds services are managed by a series of autonomous agents capable of autonomously accessing managed services. The paper details the architecture, mechanisms, and algorithms to implement the aggregation and provision of services in cross-clouds. We also develop relevant cross-clouds services management platform, called CCloudMan, with which several experiments based on public data sets have been conducted and the experimental results show the efficiency and usability of our proposed approach [18].

An open source Python library called CloudBridge that provides a simple, uniform, and extensible API for multiple clouds has been proposed by Goonasekera N et.al. As the cloud is extensively used for deploying applications in present day, it is important to seamlessly utilize resources and services from multiple providers. Proprietary vendor APIs make this challenging and lead to conditional code being written to accommodate various API differences, requiring application authors to deal with these complexities and to test their applications against each supported cloud.

Challenges

The concept of “load balancing” dates to the distributed computing era. Represented as a “function f: $T \rightarrow R$ which maps every task $Ti \in T$ on a resource $Rj \in R$” depending upon the availability of resources.

Where Rj = {R1, R2... Rn} is the set of compute resources

Ti={T1,T2…..Tn} is set of task

The process of resource allocation and load balancing face numerous challenges. The granularity of these challenges increases during every stage (such as node load condition after

the definition, acquisition of load information, and selection algorithm) of load balancing in a multi-cloud environment.

Load balancing in multi-cloud is a challenging issue due to:

- **Variability:** Data center network variability induces unpredictable application performance concerns at the time of resource allocation at the dispersed nodes.
- **Flexibility and Portability**: Use of proprietary vendor APIs and vendor lock-in has not only led to limited flexibility and portability but because of it, resource sharing and utilization in aggregated cloud platform has become difficult. It also provides limited flexibility for tenants to manage their communication.
- **Security:** Data center networks do not restrict the communication pattern and bandwidth usage of each application, making the network vulnerable to attacks.
- **Governance, Risk, and Compliance**: Since multi-cloud environments involve communication between different cloud platforms, they must abide to the changes in the communication behavior, semantics, charging models, and SLA terms. The communication becomes important during the acquisition of load information.
- **Heterogeneous nature:** The heterogeneous nature of cloud environment smake allocation and selection decisions troublesome.[6][7]

Proposed Work

In our proposed work, we present JSQLW (Join Shortest Queue with Left Work), an optimal mechanism of resource allocation and load balancing for multi-cloud environments. The algorithm provides an efficient mechanism for scheduling and load balancing across the cloud platforms. The mechanism is designed to schedule the resources based on the estimated left work of the data center which can be determined dynamically based on the environmental condition of the data center and processing time of the VMs.

Since the entire process of balancing and resource allocation is considered on the multi-cloud, the left work is assumed to be proportionate work of data centers of different cloud providers. Each server (data center) is also allocated with different weights to meet different requests depending on their cost. With its main focus on scheduling, not only for resource allocation but also for efficient load balancing, the mechanism can also be applied on cloud infrastructure for management of the heterogeneous VM Images. The proposed work is based on the combination of Transaction-Join-Shortest-Queue (TJSQ) algorithm and Transaction-Least-Work-Left (TLWL) algorithm which mainly achieves its performance by combining the features of Session Initiation Protocol (SIP), dynamic estimates of server load, transactions, and exploiting differences in processing costs for different SIP transactions in data centers.

Assumptions

- The multi-cloud workload network is assumed as a graph G with set of vertices and edges
- We have divided the entire nodes into sets of overloaded nodes, underloaded nodes and aggregator nodes.
- Migration of resources during the process of load balancing results in formation of tree hierarchy, which is generally an “incremental tree”. The path of migration or transfer represents the edges of a graph.
- The aggregator node processes load consolidation from different overloaded nodes, where the overloaded nodes try to migrate the load on the same physical machine of cloud network.
- Initially there is no element in the newly formed tree but as the machines are balanced subsequent nodes are added.
- The left work is assumed to be proportionate work of the data center which can be determined dynamically based on the environmental condition of the data center and the VMs processing time.
- Each node is allocated different weights to different request depending on their cost.
- Counters are maintained indicating the weighted number of VMs assigned to each server where the new tasks are assigned to servers with the lowest counter.
- A ratio is defined in terms of relative cost of request and response.

Procedure

In a multi-cloud environment each node is also allocated with different weights to meet different requests depending on their cost. After the determination of the underloaded and overloaded nodes, based on the number of VMs and the request process by them, a shortest path is determined and load is migrated taking all assumptions into consideration. The process goes on until all the overloaded nodes, depending upon the capacity of underloaded node of previous determined path, is migrated and there is no work left for the node (server). In the proposed JSQLW algorithm the migration occurs through the aggregated node and the process results in formation of tree hierarchy (Optimal Load Tree), which is generally an “incremental tree”. Counters are maintained indicating the weighted number of VMs assigned to each server (nodes) where the new tasks are assigned to servers with the lowest counter.

CALCULATE Optimum Load TREE ()

Input : Machines(Nodes)

Output: Load Aggregated incremental tree.

K: Set of Nodes

Z ⊆ K: Source Nodes, Sink Nodes, aggregator nodes

S ⊆ Z: Source Nodes

N: Nodes in Tree

N= {}

N' ⊆ Z : Source and Sink nodes in newly Tree

N'= {} //initially there is no element in the new formed tree

z' ∈ S: z' // overloaded node

x' ∈ Z: x is closest to z //determine sink node

Connect (x' ,z') //establish optimum/shortest path between the source and sink

Transfer load z'->x'

N = N , x'z' , path(x',z') // elements in tree

N' =N, z' ,x '//it stores the old tree along with source node and sink node

while N' =! Z

do

x' ∈ Z: a is closest to x' ∈ N //determine the next source (overloaded node)

call aggregator (); //determine the aggregator node

Connect(z',a) //through shortest path

Transfer load z'->a;

N = N , z' + path(z',a)

N' =N' + a

JSQLW Algorithm

Evaluation and Result

The entire evaluation of the proposed work is done on a popular cloud simulation tool known as Cloud Analyst. Possessing the feature of GUI modelling and simulation of applications deployed in cloud, it enables users to analyze and interpret results easily. The Repeated simulations executions are also possible in Cloud Analyst by varying the various parameters such as User base, data center and Internet Cloudlet. On the basis of input data, Cloud Analyst outputs the "response and processing time" of requests, service time and other related metrics. [19] [20]

Testbed configuration

Table 1.1 and 1.2 details configuration for User base parameter and data center parameter respectively used for the purpose of evaluation:

Evaluating measures	**Range**
User Base	B1 to B5
Region	Re1 to Re10
Data size/ request	Default value (100 bytes)
Average peak user	15000-30000
Average off peak	100-30000
Executable instruction/user	150

Table 1.1: User base parameter configurations

Parameter	**Value Range**
Data Center	DC1 to DC5
Region	R1 to R10
Virtual Machine	25 to100
Data size per request	1000 bytes
Virtual machines memory	1GB
Bandwidth	10MB

Table 1.2: Data center Parameter configurations

The machine utilized for the purpose have the following configuration:

Processor: core i3,

Memory: 2 GB of RAM

Operating system: 32-bit Windows 7 operating system and above.

Hard disk:160 GB

The simulation of proposed work is done on the basis of the above configuration and compared with other existing load balancing algorithms such as “Round Robin, Throttled, and Equally spread execution algorithm”.

Considerations

1. Peak hour load

2. Active task is even out of VMs at any given instant of time.

3. SLA between cloud providers and consumers is undertaken for pricing.

4. The entire evaluation of the work is done with respect to Cloud Analyst “Service broker policies” known as optimized response time.

Evaluation

Based on the above consideration and configuration the following results are generated.

Service broker policy: optimized response time

Parameter: Response time(ms)

Value	Round Robin (ms)	Equally spread (ms)	Throttled (ms)	JSQLW (ms)
Minimum	239.17	241.47	237.67	236.44
Average	311.29	311.04	309.26	309.07
Maximum	406.17	403.46	410.38	403.14

Table 1.3: Response time

Service broker policy: Optimized Response Time

Parameter: Data center request Service time(ms)

Algorithm: Round Robin

Data center	AVG(ms)	Min(ms)	Max(ms)

DC1	5.42	0.02	25.95
DC2	7.05	0.39	32.86
DC3	10.09	0.31	58.34
DC5	13.39	0.23	50.47
DC6	7.79	0.68	45.54

Table 1.4: Round robin algorithm

Service broker policy: Optimized Response Time

Parameter: Data center request Service time(ms)

Algorithm: Equally Spread current execution load

Data center	AVG(ms)	Min(ms)	Max(ms)
DC1	5.14	0.56	20.10
DC2	7.13	0.07	49.87
DC3	9.52	0.61	37.01
DC5	13.00	0.35	54.41
DC6	7.33	0.07	24.50

Table 1.5: Equally Spread current execution load

Service broker policy: Optimized Response Time

Parameter: Data center request Service time (ms)

Algorithm: Throttled algorithm

Data center	AVG (ms)	Min (ms)	Max (ms)
DC1	2.82	0.02	12.66
DC2	4.07	0.07	48.54
DC3	8.59	0.61	21.05
DC5	12.10	1.70	38.96
DC6	6.56	0.12	27.63

Table 1.6: Throttled algorithm

Service broker policy: Optimized Response Time

Data Center Request Service time (ms)

Algorithm: JSQLW

Data center	AVG (ms)	Min (ms)	Max (ms)
DC1	4.98	0.05	17.69
DC2	7.26	0.05	55.63
DC3	9.54	0.31	30.85
DC5	12.53	0.12	45.65
DC6	7.06	0.18	44.38

Table 1.7: Data center Request Service Time of JSQLW

Service broker policy: Optimized Response Time

Data Center processing time

Value	Round Robin (ms)	Equally spread (ms)	Throttled (ms)	JSQLW (ms)
Minimum	0.02	0.07	0.02	0.02
Average	8.35	8.11	6.27	4.02
Maximum	58.34	54.41	48.54	45.63

Table 1.8: Data center Processing Time (ms)

With respect to Tables 1.3, 1.4, 1.5, 1.6, 1.7, 1.8 we can see that the proposed “JSQLW algorithm” outperforms other existing algorithms used for load balancing and scheduling in terms of “response time, data center request service time and the processing time” for multi-cloud environment.

Conclusion and Future work

Since the Cloud environment depends upon a large number of customized and interconnected nodes with the supporting feature of scalability and on demand request processing, there is a need for a proper resource allocation mechanism. The proposed mechanism not only efficiently handles the load balancing process but also helps in building a framework for embedding security during the process of load balancing due to involvement of aggregated nodes. How the process impacts the security and evaluation of the algorithm on the various other set of evaluation values (fitness) can be carried out in the future.

Bibliography

1.Bhujbal A. et.al, “Load Balancing Model in Cloud Computing”, International Journal of Emerging Engineering Research and Technology, Volume 3, Issue 2, PP 1-6, ISSN 2349-4395 ,2015.

2.Patel D. et.al, “Efficient Throttled Load Balancing Algorithm in Cloud Environment”, International Journal of Modern Trends in Engineering and Research, p-ISSN: 2393-8161, pp 463-480,2015.

3.Saba N. and Song A. et.al, “Grammatical Evolution Enhancing Simulated Annealing for the Load Balancing Problem in Cloud Computing”, Proceedings of the Genetic and Evolutionary Computation Conference 2016, pp.997-1003, ISBN: 978-1-4503-4206-3, ACM,2016.

4. Gangwar I.et.al ,”Juxtaposition of Load Balancing Algorithms in Cloud Computing using Cloud Analyst Simulator”, International Journal of Computer Applications (0975 – 8887) Volume 97– No.2, pp 21-26, 2014

5. Paya A. and Marinescu A., “Energy-aware Load Balancing and Application Scaling for the Cloud Ecosystem”, IEEE TRANSACTIONS ON CLOUD COMPUTING, Issue: 99, ISSN: 2168-7161,2015.

6. Upadhyay A.et.al, “The-impact-of-network-parameters-in-cloud-environment-A-detailed-analysis”, International Journal of Scientific & Engineering Research, Volume 7, Issue 9, ISSN 2229-5518, pp 613-618 ,2016

7. Upadhyay A.et.al, “Suboptimal mechanism for load balancing in cloud environment”, 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), 2017 Chennai

7.A cross cloud scheduling algorithm at SaaS level on Rack Space, Go Daddy and Azure platforms

2016 International Conference on Computing, Communication and Automation (ICCCA)

8.Multi Cloud IaaS with Domain Trust in OpenStack

Navid Pustchi, Farhan Patwa ,Ravi Sandhu, CODASPY’16 March 09-11, 2016, New Orleans, ACM ISBN 978-1-4503-3935-3/16/03. DOI: http://dx.doi.org/10.1145/2857705.2857745,pp 121-123

9. Elkhatib Y, Mapping Cross-Cloud Systems: Challenges and Opportunities, https://www.usenix.org/node/196337

10 Kim H. et.al, Experience in Practical Implementation of Abstraction Interface for Integrated Cloud Resource Management on Multi-Clouds, KSII Transactions on Internet & Information Systems. Jan2017, Vol. 11 Issue 1, p18-38. 21.

11. Alam Q et.al, A Cross Tenant Access Control (CTAC) Model for Cloud Computing: Formal Specification and Verification IEEE Transactions on Information Forensics and Security ,Volume: 12, Issue: 6, 2017.

12.www. Vm ware blog com. https://www.vmware.com/radius/cross-cloud-introduction

13. Dur´an F. and Pimentel E., Component-wise Application Migration in Bidimensional Cross-Cloud Environments.

14. A Pucher, EXFed: Efficient Cross-Federation with Availability SLAs on Preemptible IaaS Instances, IEEE International Conference on Cloud Engineering (IC2E), IEEE,2017.

15. Andreas Kohne et.al, Financial Evaluation of SLA-based VM Scheduling Strategies for Cloud Federations**.** Crosscloud'17 Proceedings of the 4th Workshop on Cross Cloud Infrastructures & Platforms Article No. 1 ISBN: 978-1-4503-4934-5,2017

16. Alkaff H et.al, Cross-Layer Scheduling in Cloud Systems Cloud Engineering (IC2E), 2015 IEEE International Conference on

17. Theng D et.al, VM Management for Cross-Cloud Computing Environment, pp 731-735.IEEE ,2012 International Conference on Communication Systems and Network Technologies,2012.

18. Goonasekera N, Cloud Bridge: A Simple Cross-Cloud Python Library (ACM),2015XSEDE16 Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale.

19.http://www.dummies.com/programming/cloud-computing/hybrid-cloud/types-of-workloads-in-a-hybrid-cloud-environment/

20. Wei Yi et.al, “Cost-optimal service selection approach for collaborative workflow execution in clouds”, IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, ISBN: 978-1-5090-1915-1 ,2016.

Appendix

1.1 User base parameter configurations

1.2 Data Center Parameter configurations

1.3 Response time

1.4 Data center request service time Round robin

1.5 Data center request service time Equally Spread current execution load

1.6 Data center request Service time of Throttled algorithm

1.7 Data center request Service time of JSQLW algorithm

1.8 Data center Processing Time