Journal of Engineering Sciences
Vol 13, Issue 06, June/2022
ISSN NO:0377-9254

DESIGN OF A VIRTUAL SYSTEM FOR CLOUD COMPUTING USING INVERSE PROPOGATION

[1]Dr. Mohammad Riyaz Belgaum [2]Ediga Divya [3]Bathula Dharani
[4]Yedlapalli Alekya [5]Dayala Jyothirmai

[1]Guide Assosciate Professor [2,3,4,5] U.G. SCHOLAR

G PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY
KURNOOL

Abstract:

For measuring the efficiency of workfow scheduling, determining makespan and execution cost is essential. As estimating makespan and cost is difcult in a Cloud environment, designing an efficient computation of workfow scheduling remains a challenge. The Cloud resources are scaled up and down in accordance with user demand by following a scheduling policy. The scalability of the work environment is achieved through the virtualization process.

Based on system experience, this paper proposes the priority-based backfilling back propagation neural network (PBF-NN) hybrid scheduling algorithm for measuring makespan and execution cost accurately. The backfll algorithm is used to schedule tasks to the available resources. The percentage of migration is reduced when this algorithm is used compared to the First Come First Serve algorithm. Then, the Berger model is used to measure the fairness of resource allocation.

The system decides task reallocation based on the fairness value. The back propagation neural network handles the virtual machine placement process with necessary training and testing. The proposed algorithm dynamically allocates the

tasks and reduces the utilization of resources. We use an experimental study to illustrate how the proposed system enables higher efficiency in cost, makes pan, and performance.

Keywords- cloud computing, Back propagation, virtual flow

Introduction:

Cloud computing has opened up opportunities to apply the notion of physical machines to virtual machines by employing information technology as a service. Complex and highly dependent tasks in a workfow can be accomplished using Cloud computing based on a pay-as-you-use model. Cloud service provider's offer flexible pricing models that help customers to save significant costs while using the Cloud infrastructure to accomplish their tasks. In Cloud computing, user applications come from various heterogeneous environments. is the amount that the service provider charges the user. Recently, computation of the cost of workfow scheduling [3] in infrastructure as a service (IaaS) Cloud has attracted the attention of many researchers to improve

Existing System:

Cloud computing is a dynamic environment. Resources need to be available on demand for the execution of user tasks. If the user tasks are not executed within the allotted time slot, then it becomes a failed execution. To avoid failure in the execution of tasks, the proposed system makes use of dynamic algorithms.

Proposed System:

The proposed algorithm dynamically allocates the tasks and reduces the utilization of resources. We use an experimental study to illustrate how the proposed system enables higher efficiency in cost, makespan, and performance.

A neural network is capable of learning by itself to produce the desired output comparable with the target output. It has a parallel processing ability to speed up job execution. The proposed system that is provided in this paper needs computing power to handle parallel processing.

Advantage:

1. Submitted user applications are sorted based on the dependency value.

2. Sorted tasks are allocated to the resources using the priority-based backfilling scheduling algorithm (static scheduling algorithm).

3. Check the justice function. If it is less than the threshold value, proceed with the migration.

4. The tasks in workflows are classified based on different attributes. The resource utilization attributes are completion time, memory, and bandwidth.

5. The repeated adjustment of weights trains the neural network by using the unseen data set. When it gets the output with less error that output considers a suitable solution for VM assignment (dynamic scheduling with migration).

Software Requirement:

Hardware Requirements:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- RAM : 256 Mb.

Software Requirements:

- Operating system : - Windows 7.
- Front End : - JAVA or JSP
- Database : - SQL SERVER 2008
- Tools :- Eclipse IDE

Module Description:

Dynamic

Prediction-based scheduling algorithms are useful for workfow scheduling in an uncertain situation in a dynamic scheduling environment. The algorithm proposed in this paper uses the prediction technique and finds suitable resources for running user applications through a learning process.

The prediction-based scheduling algorithm determines the unknown future values based on the analysis of known past values for optimizing the scheduling operation.

Migration

The proposed algorithm is also compared with the existing dynamic algorithms such as priority-based backfilling and the traditional neural network separately. In this paper, quantitative parameters such as the number of tasks and the average number of migrations and qualitative parameters such as makespan, cost, and completion time are used for performance analysis. These parameters are analyzed with system load and workfow types.

Neural network

The scalability of the work environment is achieved through the virtualization process. Based on system experience, this paper proposes the priority-based backfilling back propagation neural network (PBF-NN) hybrid scheduling algorithm for measuring makespan and execution cost accurately. The backfll algorithm is used to schedule tasks to the available resources. The percentage of migration is reduced when this algorithm is used compared to the First Come First Serve algorithm.

Scheduling

The service provider charges the user based on the Cloud resources consumed and the time taken to run the applications. Therefore, reducing the cost and makespan in workfow tasks by making use of an efficient scheduling algorithm is the objective of this paper. The makespan is calculated based on computing capacity and size of the task. Cost is the amount that the service provider charges the user.

The multi-task scheduling algorithms are solved in non-polynomial time. Prediction-based scheduling algorithms are useful for workfow scheduling in an uncertain situation in a dynamic scheduling environment. The algorithm proposed in this paper uses the prediction technique and finds suitable resources for running user applications through a learning process.

SRS:

Software Environment

1.1 JAVA TECHNOLOGY

Java technology

Conclusion:

The proposed system targets the multi-objective computation of workfow scheduling problem in the Cloud IaaS

environment. The proposed system model offers a solution for the computation of workfow scheduling problem by using three algorithms. The backfilling algorithm is employed for improving parallel processing and to reduce the number of migrations. It also utilizes the intermediate idle nodes available in the initial resource allocation. Then, the Berger model is used to improve user satisfaction.

This model compares the initial population with the expected value and calculates the justice value for measuring the allocation. The back propagation neural network handles the output of the Berger model and utilizes the system experiences for learning. The BPNN network moves in the direction of the system solution converging to a minimum objective value whenever the system function is modeled with constraints and performance information.

Future Work:

The proposed system's performance improves makespan by 8% and total cost by 9% compared to other algorithms. It is proposed to employ an energy consumption model for the proposed system in future work. Additionally, we would also like to apply the artificial neural network concept for solving computation of workfow scheduling problems to derive better solutions.

It utilizes the advantages of both neural network as well as priority-based backfilling algorithms. In this experiment, the scheduler is trained with a dataset that has 100 tasks. Then, the proposed system was tested with 200 tasks dataset. The proposed algorithm learns quickly about resource allocation. So, it outperforms all other algorithms.

The total execution cost versus task completion time for all the three algorithms is compared in the figure. After the system was trained for so many iterations, system performance was found to improve.

Reference:

verify that the system satisfies requirements.

Organize these requirements in a way that works best for your project. See **Error! Reference source not found.Error! Reference source not found., Error! Reference source not found.** for different ways to organize these requirements.

Describe every input into the system, every output from the system, and every function performed by the system in response to an input or in support of an output. (Specify what functions are to be performed on what data to produce what results at what location for whom.)

Each requirement should be numbered (or uniquely identifiable) and prioritized.

See the sample requirements in **Error! Reference source not found.**, and **Error! Reference source not found.**, as well as these example priority definitions:

Priority Definitions

The following definitions are intended as a guideline to prioritize requirements.

Priority 1 – The requirement is a “must have” as outlined by policy/law

Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits

Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS **must be** either A or I" or "It **would be nice** if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

A good requirement is:

Correct

Unambiguous (all statements have exactly one interpretation)

Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)

Consistent

Ranked for importance and/or stability

Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms and specify measurable quantities)

Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)

Does not specify any particular design

Traceable (cross-reference with source documents and spawned documents).

Functional Requirements

In the example below, the requirement numbering has a scheme - BR_LR_0## (BR for Business Requirement, LR for Labor Relations). For small projects simply BR-## would suffice. Keep in mind that if no prefix is used, the traceability matrix may be difficult to create (e.g., no differentiation between '02' as a business requirement vs. a test case)

User Interface Requirements

In addition to functions required, describe the characteristics of each interface between the product and its users (e.g., required screen formats/organization, report layouts, menu structures, error and other messages, or function keys).

Usability

Include any specific usability requirements, for example,

Learnability

The user documentation and help should be complete

The help should be context sensitive and explain how to achieve common tasks

The system should be easy to learn

(See http://www.usabilitynet.org/)

Performance

Specify static and dynamic numerical requirements placed on the system or on human interaction with the system:

Static numerical requirements may include the number of terminals to be supported, the number of simultaneous users to be supported, and the amount and type of information to be handled.

Dynamic numerical requirements may include the number of transactions and tasks and the amount of data to be processed within certain time period for both normal and peak workload conditions.

REFERENCES

[1] B. Zhang, H. Qi, Y. Tao, S.C. Ren, Sun, L.M. Ruan, Application of homogenous continuous Ant Colony Optimization algorithm to inverse problem of one-dimensional coupled radiation and conduction heat transfer, Heat Mass Transf. 66 (2013) 507–516.

[2] H. Qi, L.M. Ruan, L.H. Liu, Study on the imaginary temperature of open boundary wall in cylindrical medium by partition

allocation method, J. Heat Transf. 127 (2005) 791–793.

[3] Y. Yuan, H.L. Yi, Y. Shuai, F.Q. Wang, H.P. Tan, Inverse problem for particle size distributions of atmospheric aerosols using stochastic particle swarm optimization, J. Quant. Spectrosc. Radiat. Transf. 111 (2010) 2106–2114.

[4] Y. Yuan, H.L. Yi, Y. Shuai, B. Liu, H.P. Tan, Inverse problem for aerosol particle size distribution using SPSO associated with multi-lognormal distribution model, Atmos. Environ. 45 (2011) 4892–4897.

[5] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micro Machine and Human Science (1995) 39–43.

[6] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the International Conference on Neural Networks, Australia IEEE 1948, 1942.

[7] M. Taherkhani, R. Safabakhsh, A novel stability-based adaptive inertia weight for particle swarm optimization, Appl. Soft Comput. 38 (2016) 281–295.

[8] Z. Li, T.T. Nguyen, S. Chen, T.K. Truong, A hybrid algorithm based on particle swarm and chemical reaction optimization for multi-object problems, Appl. Soft Comput. 35 (2015) 525–540.