Load Balancing for Effective Resource Provisioning in a Heterogeneous Cluster using Machine Learning

Vijayasherly Velayutham, Srimathi Chandrasekaran

Abstract: Compute Clusters are typically installed to increase performance and/or accessibility. Appropriate Resource Provisioning is a key feature in clustered computing environments to avoid provisioning resources lower than the actual requirement and provisioning of resources in excess. In this paper, a load balancing scheme leading to effective provisioning of resources have been proposed. Job History of compute-intensive jobs have been collected by conducting experiments to observe basic parameters of a job in a heterogeneous computing cluster environment. A Machine Learning model using Multi-Layer Perceptron and Support Vector Machine for provisioning of resources has been presented. The prediction model uses the job history collected from the cluster environment to predict the resource that would be appropriate for provisioning in future. The accuracy of the model is computed and the results of experiments show that Multi-Layer Perceptron presents a better performance than Support Vector Machine.

Index Terms: Cluster Computing, Machine Learning, Multilayer Perceptron, Resource Provisioning, Support Vector Machine

I. INTRODUCTION

Alongside the fast improvement of innovative technologies, the computational problems to be solved have become larger in size and highly complicated to compute. Large scale problems can be solved using the Super Computers. But Super computers cannot be extensively employed to solve such problems due to affordability by a common human being. This lead to the deployment of commodity distributed systems. One such system is a high performance cluster computing system which are aimed to create a processing model with single system image [20]. Compute-intensive problems are solved by dividing the given problems into executable tasks [20] which could be processed on a single cluster node. If an appropriate node is not assigned to this process, a user may conclude up with ending the process in the current node and redistributing the process on a different node, consequently reducing the performance with an increase in the response time. Identifying such inappropriate nodes in a massive cluster makes the cluster highly realistic in terms of resource utilization leading to efficient provisioning of resources.

The concept of load balancing is to allocate the workload to two or more computing nodes, network links, hard drives, CPUs demanding to get the maximum throughput, minimize response time and overload. In a load balancing cluster, multiple computers are linked together to save computational workload and function as a single virtual computer. The load balancing algorithm proposed in this paper uses CPU load and Memory usage to derive the load of each compute node and combine this data with attributes of respective job including CPU bound and Memory bound features which are obtained from the preceding runs of those jobs. Network utilization of the node [20] is not used in our algorithm as we are considering only compute-intensive tasks in our experimental study.

As a part of our work presented in this paper, a MPI based cluster has been constructed consisting of five nodes (i.e) one master node with four slave nodes, hosted on CentOS release 5.4. The master node distributes a process to its slave nodes and monitors working of the slave nodes. Slave nodes would execute the processes received by it and send the results to the master node.

Compute-intensive jobs namely Compression and Decompression of humongous files have been run on the cluster. Basic Job History parameters namely Job Name, Job Type, Average memory used, Average CPU utilization, Size of input file, Total job execution time, Memory, Number of Processors, Number of Cores and Node Number were collected and used in our analysis. Four categories of job history data [2] [4] have been collected namely Application Profile, System Status, VM Information and Historical Data. A Machine Learning model using Multi-Layer Perceptrons (MLP) and Support Vector Machine (SVM) for provisioning of resources has been suggested in this paper.

II. RELATED WORKS

Machine learning techniques [1] for time series forecasting and queuing theory have been used to conceptually estimate the appropriate number of resources that must be provisioned by predicting the server’s load in a distributed environment. This might guarantee user’s SLA requirements and optimize the service response time.

Multi-Layer Perceptron has been employed [2] [4] to analyze resource provisioning by profiling scientific applications (CPU-intensive applications) along with job history data in a heterogeneous computing environment. An approach to minimize the total cost of resources in a cluster computing environment [3] used by an application service provider has been presented which might pave way for appropriate provisioning of resources.

Cost-aware and failure-aware provisioning policies [5] are proposed which could be employed on a virtual machine-based cluster. There had been improvements in the response time of user’s requests as demonstrated by
Load Balancing for Effective Resource Provisioning in a Heterogeneous Cluster using Machine Learning

Simulation results. A speculative approach to resource provisioning checks the patterns of past resource allocation [6] leading to the prediction of resource requirements in future. Experimental results have concluded that over or under-provisioning of resources have been avoided. A new approach to resource provisioning [7] has been proposed for applications which are data-intensive additionally constrained on deadlines. Results have shown that for a sample data-intensive application strict deadlines are met incurring minimum cost and total number of instances being launched. A Support Vector Machine (SVM) based scheduling model [13] is presented to balance the load of a cluster of servers. The model could achieve good performance by suitably scheduling the module that balances the load.

A predictive model based on a Decision Tree Regression [14] has been presented to compute regional power demand at hourly intervals. The model hence constructed could be used extensively as an application to forecast load, leading to controlled generation and distribution of power. An Ensemble model [16] with Neural Network, K Nearest Neighbour, Support Vector Machine, Naïve Bayes and Decision Tree as base models, has been proposed to predict workload based on stack generalization. Experiments have demonstrated that there had been a reduction in RMSE of predicted CPU usage and memory usage. Long Short-Term Memory Recurrent Neural Network has been used [22] to predict the required resources and automatically scale the virtual resources grounded upon the values predicted. SVM, NN and LR were used on TPC-W benchmark web applications to render robust scaling decisions [23] for the clients on their future resource demands. Experimental results have concluded that the use of SVM is the best prediction model. NN and LR have been used to come up with strategies for resource measurement and provisioning in order to meet future resource demands [24] for applications hosted on cloud. An efficient strategy that integrates Kalman smoother and an improved support vector regression algorithm [25] has been proposed for resource provisioning. Apart from meeting the requirements of service level agreements it could substantially reduce the consumption of resources. A distributed learning mechanism [26] has been recommended to facilitate provisioning of virtual machines. A reinforcement learning algorithm has been developed and tested on Xen-based cloud test bed.

III. PROPOSED WORK

The work presented in the paper consists of the following three tasks.

![Fig. 1: Infrastructure Setup of the MPI based Cluster](image)

A. LOAD BALANCING ALGORITHM

The algorithm that balances the load as given in Algorithm 1 is run on the MPI based cluster setup as given in Fig.1. To run a computationally intensive application on a distributed computing environment, a cluster needs to be setup with one manager node and to a minimum of 4-5 computing nodes. MPICH 3.2 has been employed to construct the heterogeneous cluster. The manager node is the head node of the cluster which is responsible for breaking down the application into executable processes and assigning them to the nodes based on the proposed load balancing policy. On a compute node the real computation takes place after the assignment of tasks by the manager node. The load balancing algorithm runs on manager node and manages all the other compute nodes.

**Algorithm 1: Load Balancing Algorithm**

1. **Step 1:** Calculate the load information of the node and task demand for the resources.
2. **Step 2:** Construct the demand lookup table.
3. **Step 3:** Make decision to assign tasks to the appropriate node based upon load information and the resources demanded by task.

**Step 1: Calculate load information of the node and task demand for the resources**

//Includes the CPU utilization, memory usage

- **CPU Load** is obtained by parsing the contents of “iostat” command
- **Memory Usage** is obtained by parsing the contents of “free” command.
- **Tasks demand** is obtained by parsing the “top” command in batch mode.

**Step 2: Construct the demand lookup table**

//Demand values are obtained from the computing nodes periodically which are stored in a temporary data structure. These values are used to calculate the load parameters which gives a value for a task’s demand at the end of the execution of a task. Load parameters are calculated as given below:

- Load_cpu = Σ CPUusage/k
- Load_mem = Σ MEMusage/k
- Load_cpuNew = Average(Load_cpu, Load_cpuOld)
- Load_memNew = Average(Load_mem, Load_memOld)

The resource demand values are stored in a temporary look-up table maintained by the manager node. ‘k’ is the number of tasks. The demand values are sent by a computing node in specific time interval and kept in the look-up table. The values from the table are used at the end of execution to calculate Load parameters.

**Step 3:** Make decision to assign tasks to the appropriate node based upon load information and the resources demanded by task.

Load_n = A/B
A = Load_cpu * CPUload + Load_mem * MEMusage
B = Load_cpu + Load_mem

Where Load_n is the node n’s load; MEMusage and CPUload are memory usage and CPU load of the node n respectively.

The node with the minimum value of Load_n is the suitable node for the job on which it can get executed with minimum execution time yielding higher performance.

Load values are maintained in a flat file system and in the forthcoming runs of the MPI job updated load values are considered. The tasks resource...
demand parameters and load information are given by the computing node to the manager node.

![Image](image1.png)

**Fig. 2: Flow in the Load Balancing Algorithm**

**B. DATASET GENERATION**

The methodology used in creating the training dataset for our Machine Learning Model is presented in Algorithm 2.

**Algorithm 2: Generating the dataset**

- **Input:** JobName, JobType, sizeOfInputFile
- **Output:** Dataset D

Dataset D = {Φ}
1. Spawn Compute-intensive jobs
2. For each job of (JobName,JobType) do
3. Construct the 10-tuple
   {JobName,JobType,AverageCPUUtilization,AverageMemoryUtilization,SizeOfInputFile,TotalJobExecutionTime,NumberOfProcessors,NumberOfCores,NodeNumber}
4. 1. AverageCPUUtilization ← Parse the result of “top” command to obtain the CPU utilization of the job.
5. 2. AverageMemoryUtilization ← Parse the result of “top” command to obtain the memory utilization of the job.
6. 3. TotalJobExecutionTime ← time the job through “time” command or a customized code to get execution time.
7. 4. The subset of the 10-tuple
   {Memory,NumberOfProcessors,NumberOfCores,NodeNumber} describes the details on which the job was provisioned to run by the load balancing module.
8. Add the current 10-tuple to D
9. end for
10. 6. return D

**C. RESOURCE PROVISIONING MODEL**

The dataset thus constructed is linearly inseparable. Hence SVM and MLP are the best suited models to perform further analysis. The model as presented in Fig. 3 has a configuration setup which sets the kernel for SVM or number of hidden layers in MLP. The last member of the 10-tuple dataset, namely the NodeNumber is the output of our classification problem rendered using SVM and MLP. The model could help us predict the right node (in the cluster) on which a new task could be run. Broadly, our model could function as a basic model to provision resources for a variety of computationally intensive applications.

**Fig. 3: Machine Learning Model**

**IV. EXPERIMENTAL SETUP**

Firstly, on the MPI cluster setup as in Fig. 1, the load balancing algorithm has been run as a daemon. Compute intensive tasks have been spawned to test the working of the load balancing module. Secondly, the data generation module as given in Algorithm 2, was run periodically to capture the dataset for our machine learning model. Python’s sklearn library has been used to implement the model.

**V. RESULTS AND DISCUSSION**

Experimental results have shown that MLP has been performing higher than SVM. The accuracy of MLP was 74.5% and that of SVM was 70.1%. The SVM model could be tuned by defining an application specific kernel. The performance metrics are tabulated as in Table 1.

**Table 1: Performance parameters**

<table>
<thead>
<tr>
<th>Metric</th>
<th>SVM</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity</td>
<td>65.26</td>
<td>68.78</td>
</tr>
<tr>
<td>Specificity</td>
<td>75.45</td>
<td>80.35</td>
</tr>
<tr>
<td>Accuracy</td>
<td>70.1</td>
<td>74.5</td>
</tr>
</tbody>
</table>

**VI. CONCLUSION AND FUTURE ENHANCEMENT**

A model to provision resources based on SVM and MLP is recommended in the paper. It considers the job history dataset of two categories of compute-intensive applications run on a heterogeneous cluster. The number of compute nodes can be increased in an exponential order of 2. Comprehensively, our model could serve as a generic model to provision resources for a variety of computationally intensive applications run on a heterogeneous cluster. Further, the accuracy of the SVM model can be increased by defining application specific kernels.

**REFERENCES**

Load Balancing for Effective Resource Provisioning in a Heterogeneous Cluster using Machine Learning


12. Gaith Rjoub, Jamal Bentahar, Cloud Task Scheduling based on Swarm Intelligence and Machine Learning, IEEE 5th International Conference on Future Internet of Things and Cloud, 2017


14. Dhiman Chowdhury, Minmoy Sarkar, Mohammad Zakaria Haider, Tauque Alam, Zone Wise Hourly Load Prediction Using Regression Decision Tree Model, International Conference on Innovation in Engineering and Technology (ICET) 27-29 December, 2018


16. Tajwar Mehmood ,Dr. Seemab Latif , Dr. Sherezyaar Malik, Prediction Of Cloud Computing Resource Utilization, 2018, 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT, 2018


