International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 3, Issue 5, May 2016

Method for Prediction of Resource Requirements using Clone Detection Mechanism

[1] Mohini Rai, [2] Mrs. Rupali Bhartiya
Shri Vaishnav Institute of Technology and Science, Indore (M.P.)
[1] mohi36.rai@gmail.com, [2] rupalibhartiya@gmail.com

***Abstract*- Clusters, grids, and peer-to-peer (P2P) networks have emerged as popular paradigms for next generation parallel and distributed computing. They enable aggregation of distributed resources for solving large-scale problems in science, engineering, and commerce. In grid and P2P computing environments, the geographically distributed resources in multiple administrative domains introduce a number of resource management challenges. The resource management and scheduling systems for grid computing need to manage resources and application execution depending on either resource consumers' or owners' requirements, and continuously adapt to changes in resource availability. So this offers a challenging field for research work all around the globe. The principle propose is to give powerful asset necessity expectation system for the input guided employment displaying instrument taking into account reproduction recognition. This paper proposes a clone mechanism for prediction of resource requirements. This method compares the codes to find a similar code. [1]**

***Index Terms:* Cryptography System, Distributed Network, Information Security system, Attribute Based Encryption (ABE), Contention Based (CB-ABE)**

I. INTRODUCTION

Grid computing is the collection of computer resources from multiple locations to reach a common goal. The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files. Grid computing is distinguished from conventional high performance computing systems such as cluster computing in that grid computers have each node set to perform a different task/application. Grid computers also tend to be more heterogeneous and geographically dispersed (thus not physically coupled) than cluster computers. Although a single grid can be dedicated to a particular application, commonly a grid is used for a variety of purposes. Grids are often constructed with general-purpose grid middleware software libraries. Grid sizes can be quite large.[2]

So grid can be viewed as a very large-scale, generalized distributed computing system that can scale to Internet size environments with machines distributed across multiple organizations and administrative domains. To meet the needs of newly emerging applications grid must support extensibility, co-allocation of resources, inter-operation of systems with different administrative policies while preserving autonomy and also economy of computations. Supporting these issues lead to efficient data and resource Management mechanisms. The main aim of the resource management system (RMS) in a grid is to manage the pool of resources that are available to the grid, i.e. the scheduling of processors, network bandwidth, and disk storage etc. Regarding the management of resources one major aspect is to generate an efficient mapping between jobs and resources. Jobs need to be assigned to suitable resources. Also a job may be divided into subtasks and they may have varying resource needs and fulfilling their resource requirements needs to be addressed well. [3]

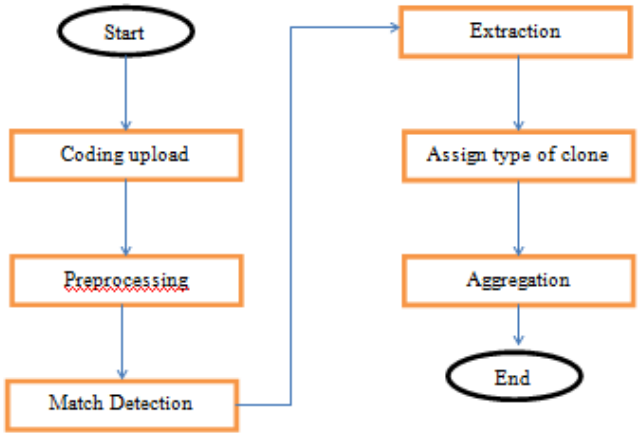

Fig-1: Clone based Flow details

Resource management is one of the important areas in grid computing research. One major objective of resource management in a computational grid environment is to allocate jobs to make efficient use of the computational resources under different resource providers and, thereby, achieve high performance of the jobs. Therefore, performance analysis is also an important issue in grid resource management. An integrated framework for

performance-based resource management in computational grid environment has been developed. The framework is supported by a multi-agent system (MAS) which has been developed using a firm software engineering approach based on Gaia methodology. The MAS initially allocates the jobs onto different resource providers based on a resource selection algorithm. Later, during runtime, if performance of any job degrades or quality of service cannot be maintained for some reason (resource failure or overloading), the MAS assists the job to adapt to the changed environment. [4]

In modern distributed systems like grid jobs are allocated to resources on the basis of the resource requirements of each job. In most resource management systems the task of determining the resource requirements of a job lies with the user who prepares a JRL (Job Requirement List), providing the resources needed by the job. This may often lead to inaccuracy in estimation of resources for a job i.e. it may lead to overestimation or underestimation of the resource requirements as the user may have very little or no idea about the resource requirements of the job. Due to this a job with higher resource requirements may be allocated to low-capacity resources causing degradation in performance or a job with low resource requirements may be allocated to resources with higher capability causing wastage of available resources. Hence, resource requirements of a job must be estimated and resources should be dynamically characterized and allocated to the job on the basis of this estimation. So, an automatic resource requirement prediction technique is presented in this thesis based on feedback guided job modelling scheme and clone detection technique.

II. LITERATURE SURVEY

The prediction engine requires two pieces of information- the run time and resource consumption of previously executed jobs that can be acquired using historical data or monitoring services like MonALISA.. Based on these the prediction engine will select the optimum resource that will be least loaded after job scheduling and will take least run time. The prediction engine will tell the planner about the selected site. The planner will then map the abstract work flow on to the selected site.

Many interesting scientific problems require the (often compute-intensive) analysis of large amounts of data. Here, the ability to harness distributed compute and storage resources can be of great value. Furthermore, the naturally parallelism inherent in many data analysis procedures makes it feasible to use distributed resources efficiently. For example, analysis of the many petabytes of data to be produced by future high energy physics experiments will require the harnessing of tens of thousands of processors and hundreds of terabytes of disk space for holding intermediate results. For various technical and political reasons, assembling these resources at a single location appears impractical. Yet the collective institutional and national resources of the hundreds of institutions participating in those experiments can provide these resources. [5]

For a non IT expert to use services in the Cloud is more natural to negotiate the QoS with the provider in terms of service-level metrics –e.g. job deadlines– instead of resource level metrics –e.g. CPU MHz. However, current infrastructures only support resource-level metrics –e.g. CPU share and memory allocation– and there is not a well-known mechanism to translate from service-level metrics to resource-level metrics. Moreover, the lack of precise information regarding the requirements of the services leads to an inefficient resource allocation usually, providers allocate whole resources to prevent SLA violations. According to this, a novel mechanism to overcome this translation problem using an online prediction system is proposed which includes a fast analytical predictor and an adaptive machine learning based predictor. How a deadline scheduler could use these predictions to help providers to make the most of their resources is also showed. Evaluation shows: i) that fast algorithms are able to make predictions with an 11% and 17% of relative error for the CPU and memory respectively; ii) the potential of using accurate predictions in the scheduling compared to simple yet well-known schedulers.[6]

Contribution targets virtualized software as a service (SaaS) providers that handle heterogeneous workloads. These providers dedicate part of their resources to execute web applications, and try to accommodate batch jobs in the remaining resources. A prediction system to determine minimum job resource requirements to be executed before its deadline is proposed. One key innovation of the prediction system is the usage of Machine Learning (ML) to enable the translation from service-level metrics to resource requirements

One of the important issues for proper usage of Grid is selection of suitable resources for jobs. Precise estimation of resource requirements for jobs is important in order to ensure efficient use of Grid resources. This paper gives an overview of an efficient job modeling technique for allocation of jobs onto the resource providers in Grid. The proposed job modeling depends on the feedback gathered from the previous executions of different jobs. The paper mainly focuses on the technical implementation details of collection of hardware performance monitoring data using

the PAPI tool. The performance monitoring data are later used as feedback while analyzing the resource requirements of the job.[7]

PRAGMA is supported by a multi-agent system (MAS). It consists of four components: (i) Resource Broker (ii) Job Controller (iii) Analyzer and (iv) Performance Tuner. Every component consists of one or more agents. These agents can communicate with each other within the component or outside the component, if required. Within PRAGMA, six types of agents are deployed; these are: Broker Agent (BA), Resource Provider Agent (RPA), Job Controller Agent (JCA), Job Execution Manager Agents (JEMA), Analysis Agents (AA) and Tuning Agent (TA). Analysis Agents are organized in a hierarchy and are employed to carry out performance analysis of jobs at various levels of the environment. [8]

III. PROBLEM STATEMENT

In existing system, the system used several approaches. Template based approach was proposed to have similar run times compared to the applications those have nothing in common. Predictions of application run time can be used to improve the performance of scheduling algorithms and to predict how long a request will wait for resources. There are two aspects of this approach- how to define "similar" and how to generate predictions. Here search techniques are used to determine those application characteristics that yield the best definition of similarity for the purpose of making predictions.

The general approach to defining similarity taken in this method is to use characteristics such as job-type, queue-type, number of nodes, user, maximum run time, submission time, start time, run time etc to define templates that identify a set of categories to which jobs can be assigned. The characteristics include physical characteristics of the job itself and also characteristics like "maximum run time" where information is to be provided by the user. Once a set of templates is defined (using a search process described later), a set of applications is categorized by assigning each application to those categories that match the characteristics of a template. Categories need not be disjoint, and hence the same job can occur in several categories. If two jobs fall into the same category, they are judged similar; those that do not coincide in any category are judged dissimilar. [9]

Prediction method is used for generating runtime predictions. The input to this method is the set of templates T and a set of jobs W for which runtime predictions are required. In addition to the characteristics described in the preceding section, a maximum history, type of data to store, and prediction type are also defined for each template. The maximum history indicates the maximum number of data points to store in each category generated from a template. The type of data is either an actual run time, denoted by act, or a relative run time, denoted by rel. A relative run time incorporates information about user-supplied run time estimates by storing the ratio of the actual run time to the user-supplied estimate. [10]

IV. PROPOSED SOLUTION

In proposed system, the system using clone detection for prediction of resource requirements. Clone is a method for object duplication. In this system, first the coding will be uploaded. After uploading the code prerpocessing will be done. Then it compares the coding to find similar code. Then excracts the code. And then it aggregates it. Classes that want copying functionality must implement some method to do so. To a certain extent that function is provided by "Object. Clone ()".

Recognize the clones is not just satisfy of programming advancement what's more, support. At that point alter the clones utilized the refactoring strategy is one of the proficient component for code clones. In this exploration at first identify the clones in programming framework and at that point utilized refactoring strategy to settle the clones. Refactoring result is created in code with upgraded viability and it is see as a preventive support action. The refactoring strategy have five stages of techniques, these are obliged to perform before applying refactoring in programming framework. clone() acts like a copy constructor. Typically it calls the clone() method of its super class to obtain the copy, etc. until it eventually reaches Object's clone() method. The special clone() method in the class Object provides a standard mechanism for duplicating objects. The class Object's clone() method creates and returns a copy of the object, with the same class and with all the fields having the same values. The default implementation of Object. Clone() performs a shallow copy. When a class desires a deep copy or some other custom behavior, they must perform that in their own clone() method after they obtain the copy from the super class.

There are several different approaches towards clone detection. Three of them are discussed here very briefly. Metric based clone detection technique: In metric-based approaches, different metrics (such as, number of lines of source code, number of function calls contained) for code fragments are retrieved and these metrics are compared instead of comparing codes directly. An allowable distance

(for instance, Euclidean distance) for these metrics can be used to detect similar code. PDG based clone detection technique: Here, the control flow graph and the data dependency graph are used to generate the Program Dependency graph (PDG). Then isomorphic sub graph matching is done to detect clones from the PDGs. AST based clone detection technique: In this technique, the source file is parsed to generate its Abstract Syntax Tree using a parser of the target language. All information of the job is contained in the nodes of the parse tree. The AST matching is then done using some tree-matching technique.

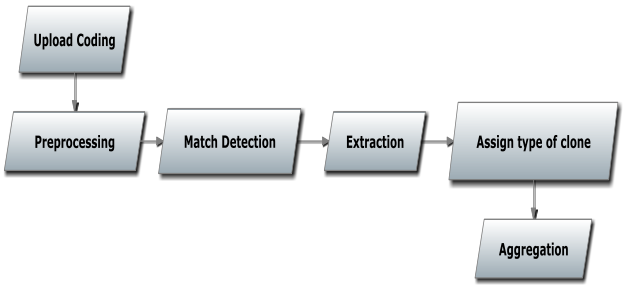

Fig-2: Proposed Architecture

In upload and Pre-processing, the system uploads the code first. Then it starts the pre-processing stage. This is the first phase of any clone detection process. It involves removing non-useful parts from the source code like comments and determining the comparison units of the target source code. In Match detection phase, the transformed code i.e. the metrics, PDGs and AST are used as input to a suitable comparison technique to find a match between code fragments. This comparison technique is done through three stages 1) metrics matching, 2) PDG matching and finally 3) AST matching. After this matching process, the system extracts the code. This module retrieve the similar coding out of total coding for further process that assigning the type and for aggregation. Then the system assigns the type of the coding. Then it aggregates the code. If a class has an entity reference, it is known as Aggregation. Aggregation is used for Code Reusability.

V. RESULT ANALYSIS

In this case the running time varies mainly due to change in operators. So, a set of abstract operations or AbOp (e.g. +, -, *, /, =) is created and it is attempted to compute number of machine cycles needed to execute each abstract operation [20]. This is done by timing a loop both with and without the AbOp of interest; the change in the total number of machine cycles is due to that AbOp. Now, computing the number of cycles for each unit AbOp (CYCAbOp), it is possible to predict total machine cycles for the submitted new job. To elaborate, let us have new job NJ and history job HJ' that is operator mismatch clone with NJ and in Execution History it

Also contains Total Cycle Counts (TOT_CYC) for history jobs acquired using code profiling tool TAU [21]. If there are k operator mismatches between NJ and HJ' namely operator HJ'opi in HJ' is replaced with NJopi in NJ for i=1 to k then,

NJopi

HJ op NJop

*NJ HJ HJ op TOT _CYC TOT _CYC freq *CYC freq *CYC*

Where,

TOT_CYCNJ= Predicted CPU cycle count of job NJ

TOT_CYCHJ'=CPU cycle count of history job HJ' provided by TAU

freqHJ'op=Number of times operator HJ'op executes in HJ'

freqNJop= Number of times operator NJop executes in NJ

CYCHJ'op=Number of cycles needed to complete single HJ'op operation

CYCNJop=Number of cycles needed to complete single NJop operation Thus, the total cycle count of program NJ on machine M is just the linear combination of the number of times each abstract operation is executed which depends only on the program, multiplied by the number of cycles it takes to execute each operation which depends only on the machine CYCAbOp of each AbOp on machine M has already been calculated. From TOT-CYC the processor time can be computed using the well known equation

CPU time = TOT-CYC / CPU clock frequency

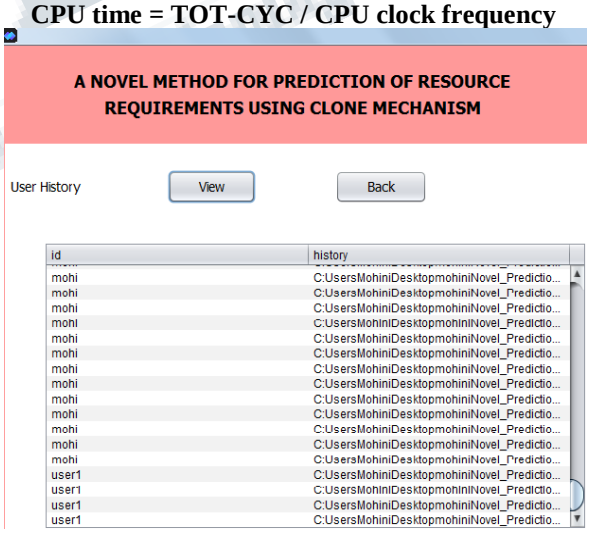

Fig- 3: User record Page

Fig: 3 shows record of the users who have used the system.

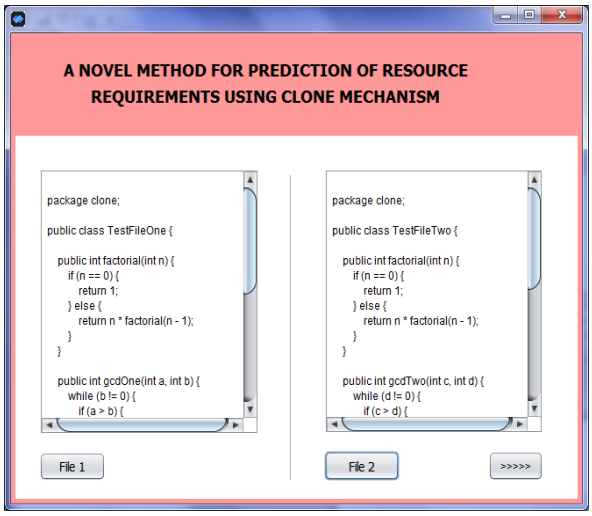

Fig- 4: Clone matching window

Fig: 4 shows matching of two codes after they have been uploaded.

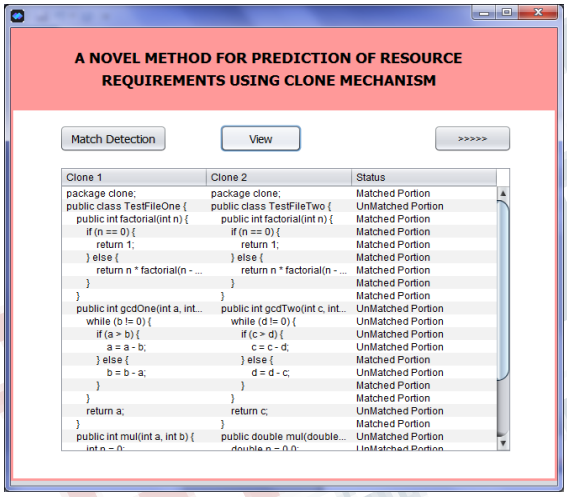

Fig-5: Comparision Window

Fig: 5 shows comparison between the clones obtained after the match detection phase. It shows the entire matched and unmatched portion between the clones.

VI. CONCLUSION

In this project, the system has proposed a powerful asset necessity expectation system for the input guided employment displaying instrument taking into account reproduction recognition. Based on these the prediction engine will select the optimum resource that will be least loaded after job scheduling and will take least run time. The prediction engine will tell the planner about the selected site. The planner will then map the abstract work flow on to the selected site. Cloning unnecessarily increases program size. Since many maintenance efforts correlate with program size, this increases the maintenance effort. A minimum requirement is usually applied to the quantity of code that must appear in a sequence for it to be considered duplicate rather than coincidentally similar. Sequences of duplicate code are sometimes known as code clones or just clones; the automated process of finding duplications in source code is called clone detection. The prediction technique works on top of the hybrid clone detection system and is entirely dependent upon the clone based similarity relation among the newly submitted job and the jobs already executed and saved in execution history. The proposed system can effectively clone the coding using prediction method.

REFERENCES

[1] MadhulinaSarkar, Sarbani Roy, Nandini Mukherjee, "Feedback-guided Analysis for Resource Requirements in Large Distributed System", published in 2010 10th IEEE/ACM International Conference on Cluster, Cloud andGrid Computing (CCGrid 2010).

[2] Ian Foster, "The Grid: A New Infrastructure for 21st Century Science", Physics Today. \Globus Toolkit, www.globus.org/toolkit.

[3]MadhulinaSarkar, RupamMukhopadhyay, DibyajyotiGhosh, Sarbani Roy, Nandini Mukherjee, "Feedback Guided Job Modeling In PRAGMA Environment". GCA 2010: 115-121.

[4] MadhulinaSarkar, SameetaChudamani, Sarbani Roy, Nandini Mukherjee, "A Hybrid CloneDetection Technique for Estimation of Resource Requirements of a Job", Accepted in International Conference on Advanced Computing & Communication Technologies, ACCT-2013, April 2013.

[5]Shaneel Narayan (Member IEEE), Shailendra S. Sodhi, Paula R. Lutui, Kaushik J. vijaykumar "Network Performance valuation of Routers in IPv4/IPv6 Environment A testbed analysis of software routers" 978-1- 4244-5849-3/10/$26.00 ©2010 IEEE

[6] M. Bohlouli, M. Analoui, "Grid-HPA: Predicting Resource Requirements of a Job in the Grid Computing Environment", World Academy of Science, Engineering and Technology 42, 2008.

[7] Swarna M, P. S. SitharamaRaju, NageshVadaparthi, "Memoir: A History based Prediction for Job Scheduling in

Grid Computing", International Journal of Computer Applications (0975 – 8887) Volume 46– No.10, May 2012.
[8] MadhulinaSarkar, TriparnaMandal, Sarbani Roy, Nandini Mukherjee, "Resource requirement prediction using Clonedetection technique", published in journal, Future Generation Computer Systems, Volume 29 Isuue 4, June 2013, Pages 936-952.

[9] C. K. Roy and J. R. Cordy (2007), "A Survey on Software CloneDetection Research Techniques", 115(2007-541), 115. Citeseer.

[10] MadhulinaSarkar, RupamMukhopadhyay, DibyajyotiGhosh, Sarbani Roy, Nandini Mukherjee, "Feedback Guided Job Modeling In PRAGMA Environment". GCA 2010: 115-121.