EFFICIENT RESOURCE AWARE IN DISTRIBUTED COMPUTING USING HYBRID SCHEDULING

Mrs. V. Latha Sivasankari[1], *Mr. S. Rinesh* [2]

ABSTRACT

The resource management system is the central component of distributed network computing systems. There have been many projects focused on network computing that have designed and implemented resource management systems with a variety of architectures and services. Distributed Computing is an efficient paradigm that supports sharing and accessing ahugevolume of enumerating and storingareas which are heterogeneous and geographically distributed. In this paper, an abstract model and a comprehensive taxonomy for describing resource management architectures in distributed computing using hybrid scheduling is developed.

I. INTRODUCTION

Cloud computing is based on a various concepts and the implementations accumulated since the first electronic computer was used to solve computationally demanding problems. Distributed Computing is an efficient paradigm that supports sharing and accessing ahugevolume of enumerating and storingareas which are heterogeneous and geographically distributed. An enormous amount of multifarious data (video, audio, images), methodized in different patterns, is created and requires gathering,processing and then aggregation in a bug-free, high-principled and procure heterogeneous distributed system. Multi Cloud solutions are applied to store, deliver and process this data [1].

Processing of Big Data systems are standardized by a suitable number of components that are implemented in parallel to execute multiple instances of the ideal tasks in order to achieve the needed performance levels in applications described by large amounts of data [2]. Performance prediction for Big Data applications is a powerful tool supporting originators and supervisors to bring about a better purloining of their computing resources. Big Data architectures are not simple, continuously evolving and reconcile, thus a rapid design and verification modeling approach can be adopt to the needs [3].

Design and evaluate the prospectivesafety-aware scheduling architecture, which mainly includes Reliability Analysis, a Global Scheduler, a Programming Queue, a Dispatching Queue, executing on HDC systems. However, it is an expensive approach. Usually, the configuration of anysytem is fixed and it is difficult for us to introduce hardware redundancy [4].

[1]Assistant Professor, Karpagam College of Engineering.
e-mail : ayy.dhinesh@gmail.com
[2]Assistant Professor,Department of Computer Science Engineering, Karpagam University.

Modern HPC systems should be able to managevarious demand loads in an efficient way when referring to the resources utilization [5] (use the less number of resources) or SLA achievement. A resource provisioning mechanism is an important element that should be used to manage utilization of available resourcesand to detect if there has been reached peak demand at a particular circumstance and the system should be expanded by requesting additional resources (from a private Cloud for example), or if the system can compress because few requests and check if the QoS requirements of recent applications will be stillmet [6].

Scheduling is a key aspect in the context of Cloud computing, virtualization and Big Data because of scheduling methodsalso have to evolve along with HPC systems. This means that an optimum format of communications is well-known. Precisely, mappers should finish their computations in the order in which they were enabled. The reducer should read the surveyoroutcome one by one, i.e. parallel reading from many mappers simultaneously is not profitable. In this approach, a divisible load model of the computation, and two load partitioning algorithms are exposed.Novel scheduling approaches have to watching the systemstructure and adjust the planning according to the real-time situation [7].

The remainder of this paper is organized as follows. Section 2 describes the existing systems related works in detail. Section 3 provides the problem statement and gives the implementation of system works and Section 5 experimental analysis of this project. Section 6 concludes the paper. Finally, section 7 reference paper shown in details.

II. Related works

Cloud computing is a model for enabling convenient, on-demand network access to a cumulativeintegration of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be fastly provisioned and released with less management effort or service provider interaction. The problem is that cloud infrastructure is expensive to maintain and also unfriendly to the environment. High energy costs and large carbon copies are incurred due to enormous amounts of electricity needed to power and cool numerous servers hosted in data centers. Cloud environment providers need to adopt measures to ensure that their profit margin is not dramatically reduced due to high energy costs [8].

Recently, researchers have been investigating the usefulness of game theory in solving the task and resource allocation problems in CGs. Using game-theoretic models enables including more resources and features into the computational optimization model for the problem. Meta-heuristics can then be used for interpreting the game to more effectively manage the resolution of the resulting computationally hard problem of finding equilibrium points of the resulting games[9].

SLAs are contractual agreements between application providers and their customers. They are stipulated on a single-customer basis (different customers typically have different SLAs), and govern the minimum quality (e.g., reply time) that customers needed. Violation of SLAs is problematic for application providers, as SLA abuseobtain direct or indirect financial losses for the provider. SLA-bound application providers face an important concessio in IaaS settings. On the one hand, they want to minimize cases of SLA violations, as to minimize amends settlements and customer discomfort, i.e., ultimately, to save money [10].

We discuss the complexity of the scheduling problems in esmating grids. We propose Hierarchic Genetic Strategy (HGS) as a novel GA-based method of scheduling the independent tasks in the batch mode. We consider the bi-objective version of the problem with makespan and flowtime as the main criteria. HGS-based scheduler is evaluated under the heterogeneity, the large scale and dynamics conditions using a grid simulator. HGS implementation outperforms existing GA-based schedulers proposed in the literature [11].

There are several scheduling algorithms and strategies adopted in private Clouds. The scheduling in OpenStack [12] framework is accomplished by the nova-scheduler. The main scheduling phases in the process are: 1. filtering available resources according to users requirements; and 2. weighting phase—the filtered hosts are applied a cost function depending on the input tasks and then sorted from the best to the worst. When using OpenNebula framework [13], the default available scheduling is related to the allocation of VMs on hosts. It uses a Rank Policy for that purpose: the hosts not fulfilling VMs requirements (memory or CPU) are excluded; the remaining hosts are evaluated using a configurable rank function; VMs are allocated to hosts with higher rank [14].

III. Proposed Work

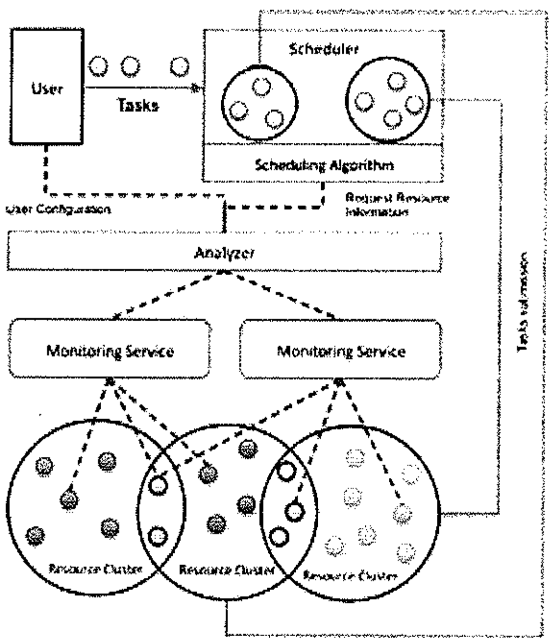
In this paper we present an extension of Hybrid scheduling algorithm for dependent tasks scheduling. This algorithm has three parts: (i) Analyze the available resources and group them into clusters (resource aware algorithm); (ii) Provision different groups of similar tasks to different clusters of resources; and (iii) Schedule the tasks in each cluster of resources. The proposed scheduling algorithm allows a more efficient and exact structure of resources, because the tasks are classified before the scheduling phase and assigned to suitable groups of resources.

Our contributions in this paper can be summarized as follows :

• First, we proposed a hybrid approach for tasks schedulingin HDC considering both tasks and resources clustering. Weacknowledge that there is no solution that could fit all kindsof tasks models. Therefore, our scheduling strategy is based onusing different scheduling strategies, selected by taking intoconsideration both the heterogeneity of computing resources,and application tasks and/or flows. The effectiveness of theutilization of the computing resources is determined by theefficiency of the allocation strategy of the resources.

• We provided a model for adaptive and dynamic clusteringconsidering the abstract modeling of HDC resources. The“distance” computed in this model highlights the similaritybetween resources, so we can group the resources in clustersand then use the formed clusters in Hybrid algorithm.

• We extended CloudSim [15] to consider this approach and weintegrate four scheduling strategies: two for independent tasks(Depth Priority First) and two for Distributed scheduling. The following architecture shown detail of hybrid scheduling algorithm process.



Scheduling

The Scheduler has the role of receiving input tasks and assign them to available resources. The Scheduler module workflow is :

1. The Scheduler receives the input tasks.
2. Next, the Scheduler sends the tasks to the Analyzer for clustering.
3. Further, the Scheduler receives from the Analyzer the clusters of tasks and available cluster of resources.
4. Finally, the Scheduler applies a hierarchical scheduling algorithm and send each task to the identified resource: A. the first step for scheduling a tasks cluster (a set of tasks) is assigning it to a cluster of resources.

B. after that, the set of tasks is scheduled using a classical algorithm specific to selected cluster of resources.

Scheduling algorithms

Independent Tasks approach. Given the fact that the tasks being processed by the scheduler are a set of independent tasks, we apply specific scheduling algorithms. The scheduler implements two scheduling algorithms for independent tasks, and alternates them in each cluster of resources, for analysis and comparison purposes :

1. Shortest Job First (SJF): associates with a task, its estimated CPU processing time (“small” job means having a low processing time), and as soon a resource is available, it assigns on it the shortest task in the waiting list. In order to achieve this more efficiently, the list of tasks is sorted ascending after the CPU processing time (see Algorithm 1).

2. Earliest Deadline First (EDF): associates with a task its deadline, and as soon a resource is available, it assigns on it the task with the nearest deadline in the waiting list. The tasks are kept into a priority list, the priorities are the inverse of the deadline, and the tasks with higher priority are scheduled sooner (see Algorithm 1).

Algorithm 1 Earliest Deadline First (EDF) / Shortest Job First (SJF)

```
1: procedure CLASSICAL_SCHEDULING($\mathcal{T}$, $\mathcal{R}$)
2:    sort tasks in $\mathcal{T}$:
      - descending after deadline ($P_4^T$) for EDF OR
      - ascending after CPU processing ($P_1^T$) time for SJF;
3:    while $\mathcal{T} \neq \emptyset$ do
4:       if anyResourceAvailable($\mathcal{R}$) == true then
5:          $R \leftarrow$ getRandomResourceAvailable($\mathcal{R}$);
6:          $T \leftarrow$ popTask($\mathcal{T}$);
7:          execute $T$ on $R$;
8:       end if
9:    end while
10: end procedure
```

We may observe that the difference between the two algorithmsis how the tasks are being sorted in the waiting list. Thelist is being used as a stack.Wefocus on Modified Critical Path (MCP) and Earliest Time First(ETF) because we are oriented towards high performance in thescheduling phase (see Algorithms 2 and 3).

Algorithm 2 Modified Critical Path (MCP) algorithm

```
1: procedure MCP($\mathcal{G}$, $\mathcal{R}$)
2:    for each $n_i \in \mathcal{G}$ do
3:       Compute the $ALAP(n_i)$;
4:    end for
5:    Create a node list $L = \emptyset$;
6:    for each node $n_i$ do
7:       Create a list $L_i = \emptyset$;
8:       $L_i \leftarrow L_i \bigcup \{(n_i, ALAP(n_i))\}$;
9:       Sort $succs(n_i)$ in a descending order with respect to
   ALAP;
10:      for all $n_j \in succs(n_i)$ do
11:         $L_i \leftarrow L_i \bigcup \{(n_j, ALAP(n_j))\}$;
12:      end for
13:      $L \leftarrow L \bigcup L_i$;
14:   end for
15:   Sort these lists ($L$) in an ascending lexicographical order;
16:   repeat
17:      Extract $L_1$ the first node in the node list $L$;
18:      Schedule $L_1$ to a resource from $\mathcal{R}$ that allows the earliest
   execution;
19:                                        ▷ using the insertion approach;
20:      $L \leftarrow L - L_1$;           ▷ Remove the node from the node list.
21:   until $L = \emptyset$;
22: end procedure
```

IV. Experimental analysis

In this section, the simulation results of the proposed algorithm are compared with the some of the existing algorithms which are recent in scheduling with hybrid scheduling system.For the test cases, we considered tasks and resources with various requirements, as support for heterogeneous distributed computing modeling. The characteristics of the processing elements are:

- MIPS: 200, 400, 500, 800, 1000, 2000, 4000, 5000, 8000, 10000;
- RAM dimension: 512, 1024, 2048, 4096, 8192, 16384;
- we vary also the total storage value, but the values are not relevant, because we consider that when a task executes, all data are available on local storage.

The results of hybrid scheduling operation for a system with heterogeneous processing resources are excute processed each task in minimum computation time. Acquired results display that the algorithm has preserved finish time, despite restrictions in resources and the number of processors, while efficiency of processors have increasingly improved.

V. Conclusion

In this paper, a Resource-Aware hybrid scheduling algorithm has exploited for the task scheduling in heterogeneous distributed computing. Most of the existing algorithms are not efficient to achive the processors capabilities for optimizing load balancing and increasing efficiency. We proposed the hybrid scheduling algorithm combined with the (Depth Priority First) and two for Distributed scheduling. In distributed schedule independent tasks are performed efficient using ETF and MCPA algorithms.it increases speedup and efficiency while in most cases reduces finish time. Optimal selection of the available processors for increasing efficiency is of great importance.

AUTHOR'S BIOGRAPHY

Ms.V. Latha Sivasankari, working as Assistant Professor in the Department of MCA, Karpagam College of Engineering for past 5.5 yrs. She has completed BCA in 2003 at Sri Krishna Arts and Science College, also completed MCA in 2006 at Karpagam College of Engineering, then MPhil CS in Vinayaka Mission University and completed ME CSE in 2016 at Karpagam University. She has published 5 papers in international journals.

Mr. S. Rinesh is working as an Assistant professor in the department of CSE at Karpagam University. He has more than 8 years of teaching experience. He has published many papers in various international journals.