Future Generation Computer Systems 27 (2011) 506–526

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Hybrid reliable load balancing with MOSIX as middleware and its formal verification using process algebra

Shakti Mishra *, D.S. Kushwaha, A.K. Misra

Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology, Allahabad, India

ARTICLE INFO

Article history:
Received 18 December 2009
Received in revised form
11 December 2010
Accepted 13 December 2010
Available online 29 December 2010

Keywords:
Load balancing
Process migration
Cluster computing
Multilevel feedback queue scheduling
Auto-configuration
Process algebra

ABSTRACT

For many years, load balancing in distributed computing environment has been researched to enhance and optimize the scalability of the entire system. Although numerous works have been proposed on the issues of process migration and load balancing, a comprehensive approach still misses out on various fronts. An effort has been made in the present work to cite a reliable and comprehensive load balancing approach for a trusted small scale distributed computing environment based on the priority of the processes. The proposed approach introduces a Process Migration Server (PMS) that also acts as future cluster management server ensuring that the latency time in migrated process execution is reduced along with no starvation policy for any process. We also propose an effective resource sharing mechanism where our system is able to distribute the load evenly among various nodes without much overhead. This in turn increases the reliability of the overall system. To verify our approach, we have implemented hybrid reliable load balancing algorithm where MOSIX is deployed as middleware and significant performance improvement has been observed. We have compared the performance of the proposed algorithm to the default MOSIX. Our results indicate that, hybrid reliable load balancing has lowered the network traffic by 80%–90%, increased CPU utilization by 40%–50%, with 4%–9% lesser memory and computational requirement. A relationship between the formal aspect of hybrid reliable load balancing algorithm and its architectural model JMM is established through process algebra approach.

1. Introduction

In the present computing plethora, the increased need of computing power for computational and commercial applications is coupled with the high cost and low accessibility of traditional supercomputers. This is why recent trends in parallel computing are moving away from supercomputing platforms and have shifted to the use of cheap and low-cost commodity-off-the-shelf (COTS) hardware and software components. Cluster computing has emerged as a result of convergence of several trends including the high performance microprocessors, high speed network, standard tools and their availability with economical COTS components. It leads to the evolution of powerful Computer Supported Cooperative Working (CSCW) environment that enables improving availability of services, sharing computational workload and performing computation intensive jobs.

The cluster technology seems to gain popularity day by day. The prime reason for this popularity is high availability, increased reliability and high performance computing. A cluster is described as a collection of interconnected workstations or PC processors [1]. Cluster based distributed systems provides a cost-effective solution to applications intended for High Performance Computing (HPC). The computational requirement of various applications can be met using cluster technology in an effective manner.

Achieving optimal performance in clusters forces the migration of processes to other nodes. This is where trust comes into play. Establishing trust in a distributed environment is one of the most challenging and important aspects of cluster computing. A secure trusted environment is also needed in order to ascertain the credibility of the participating nodes working together to achieve the goal of Computer Supported Cooperative Working (CSCW). Load Balancing is one of the important aspects of CSCW. It is achieved by evenly distributing the load of heavily loaded nodes to lighter one through process migration [2]. There are instances when majority of the nodes of a cluster may be under loaded and these nodes may accept foreign processes to execute also and there might exist real time process that demands immediate execution. Along with this, these nodes may itself get their own new processes to execute also. This scenario may lead majority of these nodes towards being overloaded. Migration of processes and its execution demands an effective and optimized scheduling approach so that neither foreign nor local of processes may starve.

* Corresponding author. Tel.: +91 9956522703.
E-mail addresses: shaktimishra@mnnit.ac.in (S. Mishra), dsk@mnnit.ac.in (D.S. Kushwaha), akm@mnnit.ac.in (A.K. Misra).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.12.007

For a load balancing system, we propose a combined approach, priority-based run queue management and multilevel feedback queue (MLFQ) scheduling approach for migrated tasks. The proposed scheduling approach tries to resolve following issues:

- To prevent frequent migration of process due to unavailability of underloaded nodes by ensuring proper mixing of local and remote processes in run queue.
- Identification of critical processes by assigning highest priority and scheduling these processes immediately on one of the available processors.
- No starvation policy for any local process while considering the priority and criticality of process.
- Scheduling local and remote jobs in run queue with same priority in round robin fashion so that neither of these processes may starve.
- Boosting priority of computation intensive remote processes in MLFQ to reduce remigration and congestion across the network.

All the above work focuses the need of resolving following issues:

- Frequent migration of process.
- Real time process criticality.
- Elimination of requirement of specialized utility running on each node.
- Single point failure recovery mechanism.
- Optimized load balancing strategy with reduced network traffic load.

Before implementing a system to a large scale domain, mathematical modeling is necessary, in order to ascertain architecture and framework. System evaluation is one of the most important aspects during design, development and operational phases. Distributed systems are composed of various sequential components which synchronize with each other on the basis of predefined protocol. Specifying operational and transitional characteristics of distributed systems is one of the prime issues in modeling. Modeling pertains to capturing and analyzing the dynamic behavior of any system. This dynamic but complex behavior of distributed systems can be analyzed by underlying communication protocol and characteristics of their components.

Calculus of Communicating System (CCS) [3] is a formal language for describing patterns of interaction in the concurrent systems. CCS is used as a tool for specifying and verifying the concurrent aspects of variety of systems. It allows the description of system in terms of sub processes that includes primitives for describing parallel composition and interacts with each other through message passing. Therefore, the motivation for using process algebra is to simplify the specification part and to verify the design structure of model while meeting its ultimate goal i.e. load balancing. In this paper, we use CCS to validate the design of JMM with hybrid reliable load balancing approach.

The rest of the paper is organized as follows. The next section discusses various related work done so far. Section 3 proposes Jingle–Mingle Model and Hybrid Load balancing algorithm with its performance evaluation. Section 4 proposes scheduling of migrated tasks in hybrid load balancing approach along with its evaluation. Section 5 proposes Inter-cluster load balancing trough self-organizing cluster approach and its analysis. Section 6 carries out the stability analysis of the proposed Jingle–Mingle Model through process algebraic approach. The final section concludes the work followed by references.

2. Related work

The DP Model [4] proposes an approach to divide an application into many independent processes, exploit one single program to execute on multiple data and get the partial solution. The mechanism provides dynamic load balancing through process migration [5] or data migration according to the property of application, but authors have not defined the property of application in their proposal.

Charlotte [6] is a message based distributed operating system which detaches a running process from its source environment and attaches it with a new environment using starter utility process. This approach requires a special kernel utility running on each node participating in CSCW. However, the author has not proposed any solution for source and destination failure.

Hectiling System [7] follows a hierarchical architectural model and crash at any single point may lead to the system failure. The proposed approach in Condor [8,9] is to combine the Remote UNIX (RU) remote execution facility with the UP–Down Algorithm for the fair assignment of remote capacity to a needy system. However, the proposal does not elaborate issues of high priority real time critical process and the case of frequent migration of processes. BAG [10] is a heterogeneously distributed operating system designed for real time applications. Yet another approach has been proposed in [11] wherein the job arrivals are characterized by correlation among their dependencies. The author's have proposed a policy wherein the migrated processes are queued based on the size of the processes and even considers the autocorrelation among these. The proposed approach does not consider the priority of processes, which is the major limitation of the cited approach.

Zhang and Pande [12] discusses about minimization of migration cost and defines strategy as to which parts of the program should migrate. Many of the researchers [13] have tried to resolve issues like longer freeze time that may be due to unavailability of competing resources but their approach requires pre-fetching of memory pages for process migration. Some researchers [14] have also proposed virtualization of process migration. The author in [15] proposes to create a uniform virtual runtime environment on top of different hardware and software platform and this is the point of contention because every remote node has to be fine tuned to the kind of distributed clusters and Grids.

A hybrid load balancing policy underlying grid computing environment [16] proposes a dispatcher and agent based approach. The dispatcher performs maintenance, status monitoring, node selection and assignment and adjustment task for each node. The author's consideration of load balancing restricts the system to the "join and leave" decision of nodes. This suits P2P system but not CSCW.

Dynamic Load Balancing (DLB) [17] provides application level load balancing for parallel jobs using system agents and DLB agent. The approach requires a copy of system agents on all the system so that DLB agent may collect load information from these systems and perform load balancing. The other contemporary work includes grid load balancing using Intelligent Agents [18], that proposes a combined approach using intelligent agents and multi-agents for effectively scheduling the local and global grid resources that also incorporate peer to peer advertisement and service discovery to balance the workload. The approach requires a copy of system agents on all system so that DLB agent may collect load information from these systems and perform load balancing.

Yagoubi and Slimani [19] puts forward a dynamic tree based model to represent grid architecture and proposes Intra-site, Intra-cluster and Intra-grid load balancing. The work in [20] addresses issues to balance the load by splitting processes in to separate jobs and then distributing them to nodes using Mobile Agent (MA). The authors propose a pool of agents to perform this task. The authors