# PERFORMANCE ANALYSIS OF SCHEDULING-BASED LOAD BALANCING FOR DISTRIBUTED AND PARALLEL SYSTEMS USING VISUALSIM

Abu Asaduzzaman, Manira Rani
*CSE Department, Florida Atlantic University, 777 Glades Rd, Boca Raton, FL, USA*

Darryl Koivisto
*Architecture Modeling Group, Mirabilis Design, Inc., 830 Stewart Dr, San Jose, CA, USA*

Keywords: Distributed and parallel system, scheduling algorithm, load balancing, performance analysis, VisualSim.

Abstract: The concurrency in a distributed and parallel system can be used to improve the performance of that system by properly distributing the tasks among the processors. However, the advantage of parallelism may be offset by the increased complexity of load balancing techniques. Scheduling is proven to be an effective technique for load balancing in any distributed and parallel system. Studies indicate that for application-specific systems static scheduling may be the potential choice due to its simplicity. In this paper, we analyze the performance of load balancing by static scheduling for distributed and parallel systems. Using VisualSim, we develop a simulation program that models a system with three processors working simultaneously on a single problem. We obtain the response time and completion time for different scheduling algorithms and task groups. Simulation results show that load balancing by scheduling has significant impact on the performance of distributed and parallel systems.

## 1 INTRODUCTION

In a distributed and parallel system, tasks are simultaneously executed on multiple processors in order to improve performance. Scheduling and load balancing techniques are key areas for obtaining good performance in parallel and distributed applications. Such techniques can be provided either at the application level, or at the system level. In application level, mapping of a parallel computation onto a parallel computer system is one of the most important issues. Similarly, the concept of scheduling and load balancing is very important at system level. System-level simulation using computer programs is an effective technique for performance analysis of complex distributed and parallel systems (Asaduzzaman, 2004), (Amoroso, 2006), (Dunigan, 2005), (Wikipedia, 2007).

Two types of load balancing policies (static and dynamic) are often used for balancing the workload of distributed and parallel systems. Static policies use only the system statistical information in making load balancing decisions, and their principal advantage is their simplicity in mathematical analysis and implementation. They do not, however, adapt to fluctuations in workload. On the other hand, dynamic load balancing policy reacts to the current system state. Dynamic policies attempt to balance the workload dynamically as jobs arrive and are therefore thought to be able to further improve system performance. This makes dynamic policy necessarily more complex than the static one. Studies concerning dynamic load balancing may oversimplify the system and introduce inaccurate results (Renard, 2003), (Zhangt, 1995). Studies show that static scheduling policies are easy to implement and help improving performance. In an application-specific system where the expected workload is almost known, static scheduling may be the potential choice to balance the load.

In this work, performance of schedule-based load balancing is analyzed for distributed and parallel systems using VisualSim. In Section 2, some related articles are presented. Section 3 discusses scheduling and load balancing issues. Simulation details are presented in Section 4. In Section 5, the

Asaduzzaman A., Rani M. and Koivisto D. (2007).
PERFORMANCE ANALYSIS OF SCHEDULING-BASED LOAD BALANCING FOR DISTRIBUTED AND PARALLEL SYSTEMS USING VISUALSIM.
In *Proceedings of the Second International Conference on Software and Data Technologies - PL/DPS/KE/WsMUSE*, pages 106-111

simulation results are discussed. Finally, we conclude our work in Section 6.

## 2 RELATED WORK

A lot of research work has been done and a lot of articles have been published on load balancing in distributed and parallel systems. In this section, we include only those that are very relevant to the work presented in this paper.

In (Renard, 2003), dynamic, static, and a mixture of both techniques are used for load balancing in a distributed and parallel system. In dynamic strategies, data dependencies, in addition to communication costs and control overhead, may well lead to slow the whole process down to the pace of the slowest processors. In static strategies, data redistributions and control overhead are suppressed or minimized during execution. This article concludes that static allocations are necessary for a simple and efficient system.

A static load balancing scheme for partial differential equation solvers in a distributed computing environment is described in (Ichikawa, 2000). Both communication and computing time are considered to minimize the total execution time. This method is expected to be applicable to a wide variety of parallel processing applications.

(Lee, 2000) proposes a load balancing algorithm for scalable high performance cluster-based shared-memory multiprocessor systems. This algorithm performs load redistribution in a cost-effective way only when the possible savings outweigh the redistribution costs. Results show that this algorithm may enhance performance if using it properly.

A simpler version of single-point algorithms than those of Tantawi and Towsley are proposed in (Kim, 1992) for the models of distributed computer systems with a single communication channel and star network configurations. (Soklic, 2002) introduces a new load balancing algorithm, called diffusive load balancing. The algorithms are tested in three simulated client-server environments – a small-scale, Intranet, and Internet environment. Experimental results are impressive. Various scheduling and load balancing issues related to distributed and parallel systems are addressed in (Kemada, 2000), (Zhang, 1991), (Magee, 2000). The fundamental ideas of their algorithms may be useful for some other related models.

(Anguille, 1995) implements both a static-load balancing algorithm and a receiver-initiated dynamic load-sharing algorithm to achieve high parallel efficiencies on both the IBM SP2 and Intel IPSC/860 parallel computers. Significant speedup improvement was recorded for both methods.

The performances of adaptive and static load balancing policies in a heterogeneous distributed system model are compared using simulation (Zhangt, 1995). Simulation results show that both dynamic and static policies improve performance dramatically. It is also shown that when overheads are non-negligibly high at heavy system loads, static policies can provide performance more stable and better than that provided by dynamic policies.

In this work, we use static scheduling to balance the load of a distributed and parallel system.

## 3 SCHEDULING AND LOAD BALANCING

Scheduling and load balancing techniques are important keys to obtain good performance in distributed and parallel applications.

### 3.1 Distributed System

In a distributed system, different parts of a program run simultaneously on multiple processors that communicate with each other via a network in order to improve the overall system performance. The Berkeley Open Infrastructure for Network Computing (BOINC) is a good example of a distributed system.

### 3.2 Parallel System

In a parallel system, computation of a task is performed simultaneously on multiple processors in order to obtain results faster. According to Flynn's taxonomy, parallel architectures are SIMD (Single Instruction Multiple Data) or MIMD (Multiple Instruction Multiple Data) type.

Distributed computing is a type of parallel processing. Parallel processing requires that a program be parallelized (i.e., divided into parts that can run simultaneously); distributed computing also requires that the division of the program take into account the different environments on which the different parts of the program will be running.

### 3.3 Deadlock and Starvation

Deadlock and starvation are two common problems in distributed and parallel system where many processes share a specific mutually exclusive resource. Deadlock refers to a specific circumstance when two or more processes are each waiting for another to release a resource. Starvation refers to a

situation where a process is continuously denied to acquire necessary resources.

There are four necessary "Coffman" conditions for a deadlock to occur (Wikipedia, 2007).

- **Mutual exclusion (ME)** – there should be at least one non-sharable resource. Only one processor can access/use the non-sharable resource at any time.
- **No pre-emption** – resources cannot be pre-empted. A requesting processor can not have immediate access to the requested resource(s) until the holding processor is done and give up.
- **Hold and wait** – processes already holding resources may request new resources.
- **Circular wait** – two or more processes form a circular chain where each process waits for a resource that the next process in the chain holds. In Figure 1, P0 is waiting on P1, P1 is waiting on P2, …, and finally, Pn is waiting on P0.

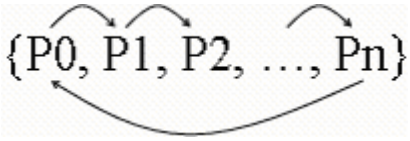



Figure 1: Circular waiting.

In our simulation, we experience both deadlock and starvation. In order to keep the simulation program simple, we avoid deadlock and starvation by using ME, pre-emption, and priority.

### 3.4 Scheduling

Scheduling is a key factor that refers to the way processes are assigned priorities in a priority queue. This assignment is carried out by software known as a scheduler. The primary goal of the scheduler is to balance loads among the processors and prevent any deadlock or starvation in the system. Some popular algorithms are discussed here.

**First come first serve (FCFS):** FCFS is non-pre-emptive and the simplest scheduling algorithm. It is troublesome for time-sharing systems.
**Priority:** Each process has a priority, the highest priority wins, and the equal-priority follows FCFS
**FCFS + Pre-emption + Priority:** A process must give up resources it may acquire before it completes its task, if higher priority jobs arrive.
**Round robin (RR):** RR is equivalent to FCFS + pre-emption + time-quantum. RR is appropriate for time-sharing systems.

In this work, FCFS, pre-emption + priority, RR, and pre-emption + time-slice algorithms are considered.

### 3.5 Load Balancing

Load balancing is a technique to spread work among many processes/processors in order to get optimal resource utilization and decrease computing time. The load balancing methods help improving performance by selecting the appropriate resources to run the specific tasks.

We use different task groups and scheduling algorithms for balancing the load of in a distributed and parallel system.

## 4 SIMULATION

In this work, we focus on evaluating the performance of scheduling-based load balancing techniques for a distributed and parallel system using VisualSim.

### 4.1 Simulated Architecture

In a distributed and parallel system, a large task is divided into small tasks and assigned among the processors. Figure 2 shows the simulated architecture with three processors working together to solve a single problem and one processor controlling the system.

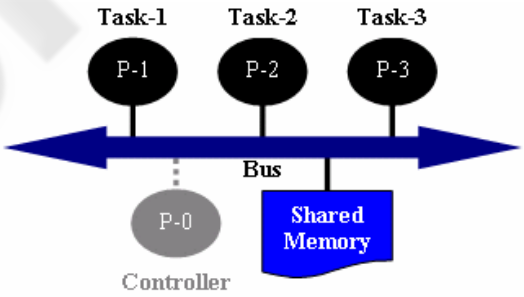



Figure 2: Multiprocessor architecture.

Scheduling activities are maintained by the controlling processor (P-0). Each working processor (P-1 to P-3) needs to access shared memory in order to complete the task. Processors submit their requests for the shared memory to the scheduler and scheduler allow one processor at a time to access the shared memory.

## 4.2 Assumptions

We make the following assumptions to simplify the model and run the VisualSim simulation.

1. Tasks from the same processor are assigned numbers and are executed in order. Tasks from different processors are assigned priorities.
2. For two independent tasks T1 and T2, both orders T1→ T2 and T2 → T1 are okay.
3. Like memory, the bus in the architecture is also shared, but the impact of the shared bus on performance is considered negligible.

## 4.3 System Parameters

Various task groups and scheduling schemes are used to run the simulation. Each task may have start time, mean time (when the next task may generate), priority, and ME indicator. We consider three different task groups based on the task generation criteria as shown in Table 1.

Table 1: Three task groups.

| Task Group | Task-1 Start Time | Task-2 Start Time | Task-3 Start Time |
|---|---|---|---|
| 1 | 0.0 | 0.0 | 0.0 |
| 2 | 0.0 | 2.0 | 4.0 |
| 3 | Random (0.0, 3.0) | Random (0.0, 3.0) | Random (0.0, 3.0) |

We use five different scheduling schemes using FCFS, priority + pre-emption, and RR + time-slicing. All three task groups are tested using these scheduling schemes. Table 2 shows Schedule 1, which is simple FCFS scheme – no priority, no ME are involved.

Table 2: Schedule 1 – FCFS.

| Proc. No. | Task No. | Task Group | Priority (NA) | ME (NA) |
|---|---|---|---|---|
| 1 | 1 | All | NA | NA |
| 2 | 2 | All | NA | NA |
| 3 | 3 | All | NA | NA |

Schedule 2 is FCFS with Pre-Emption (PreE) and priority but no ME as shown in Table 3.

Table 3: Schedule 2 – FCFS with PreE, and priority.

| Proc. No. | Task No. | Task Group | Priority | ME (No) |
|---|---|---|---|---|
| 1 | 1 | All | 1 | No |
| 2 | 2 | All | 2 | No |
| 3 | 3 | All | 3 | No |

Schedule 3 is a variation of Schedule 2 where ME=Yes for Proc-1 and ME = No for Proc-2 and Proc-3.

Table 4 shows Schedule 4, which is RR with PreE, priority, and no ME. Time slice (TS) is 1 time unit for all tasks.

Table 4: Schedule 4 – RR (TS = 1) with PreE and Priority.

| Proc/ Task | Time Slice | Task Group | Priority | ME |
|---|---|---|---|---|
| 1 / 1 | 1 | All | 1 | No |
| 2 / 2 | 1 | All | 2 | No |
| 3 / 3 | 1 | All | 3 | No |

Schedule 5 is a variation of Schedule 4 where Task-1 is allowed for 3 TS, Task-2 is for 2, and Task-3 is only for 1 TS.

The task groups and schedules are used to run the simulation model.

## 4.4 VisualSim Model

VisualSim is a system-level simulation tool from Mirabilis Design, Inc. The simulated architecture is modelled using VisualSim as shown in Figure 3. In VisualSim, a system is described in three major pasts - Architecture, Behaviour, and Workload.

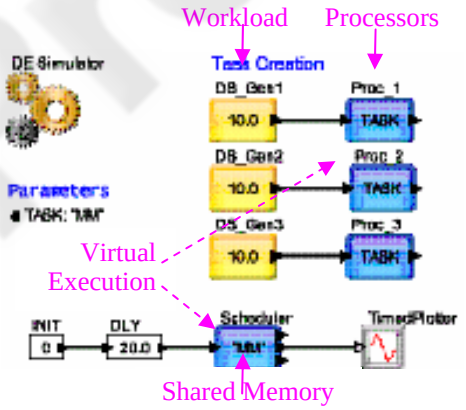



Figure 3: VisualSim model of the architecture.

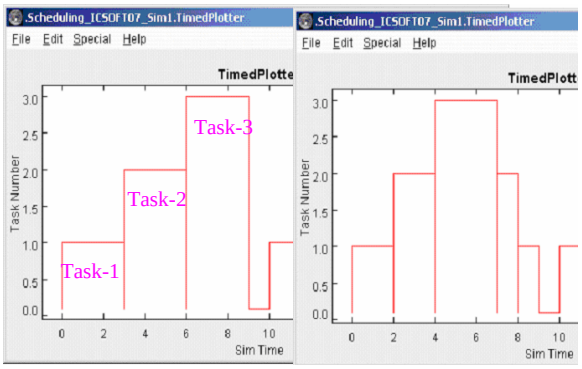
Architecture includes the major elements such as processor and memory. Behaviour describes the actions performed on the system. Workload captures the transactions that traverse the system during the simulation (VisualSim, 2007).

# 5 RESULTS

In this paper, we evaluate the performance of scheduling-based load balancing techniques for a distributed and parallel system. First we discuss the

impact of scheduling on load balancing. Then we present the response time and completion time for different scheduling schemes.

Figure 4(a) shows the simulation output for schedule FCFS (no pre-emption). Task-1 (from Proc-1 with priority 1) starts at 0.0. Task-2 is generated by Proc-2 at time 2.0 with priority 2. Task-1's ME = 'Yes'. So, Task-2 waits (for 1.0 unit of time) until Task-1 is finished at 3.0. Similarly, Task-3 (from Proc-3) is issued at 4.0 with priority 3, waits for 2.0 units of time, starts at 6.0, and completes at 9.0.




(a) ME = Yes (b) ME = No

Figure 4: Task versus time for FCFS; (a) ME = Yes and (b) ME = No.

Now we investigate the impact of FCFS with pre-emption and ME = No. Figure 4(b) shows the simulation output for this schedule. Here, Task-1 should give the resources at time 2.0 even though it is not completed and Task-2 starts. Similarly Task-3 is issued at 4.0 with priority 3 ad starts at 4.0, and completes at 7.0. Task-2 and Task-1 are completed later time (at time 8.0 and 9.0 respectively) based on their priorities.

Figure 4 indicates that scheduling has significant impact on load balancing in a distributed and parallel system. The impact of load balancing by scheduling on performance is presented in the following subsections.

## 5.1 Response Time

Response time is a measure of time a system takes to react to a given input (from request to the first react). The average response time versus schedules for task group 1, 2, and 3 are presented in Figure 5.

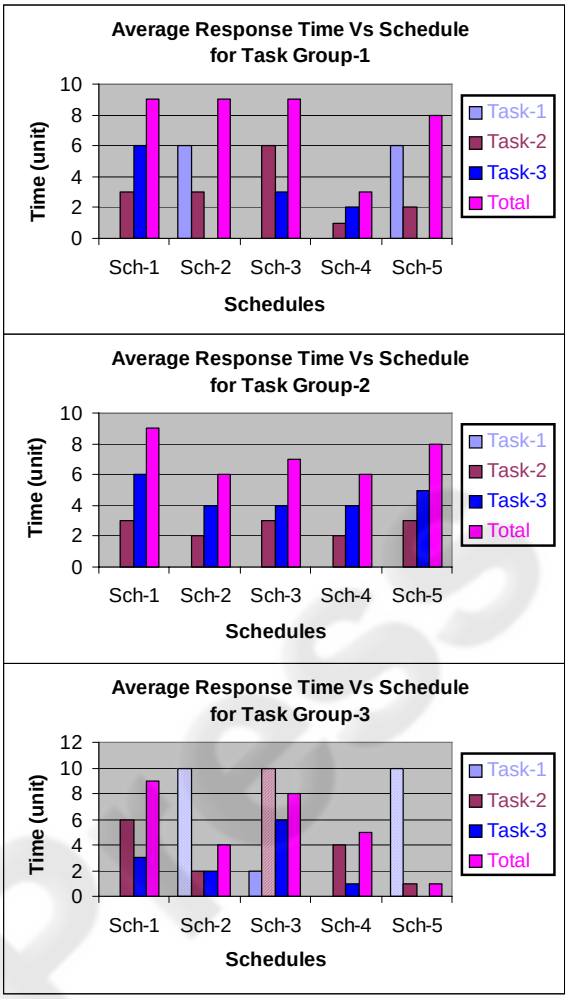



Figure 5: Average response time versus schedules for task groups 1, 2, and 3.

Simulation results show schedule 4 offers the best average response time for task group 1. Similarly, schedules 2 and 4 offer the best performance for task group 2. For task group 3, simulation results show that Task-1 of schedule 2 never starts, even though Task-2 and Task-3 start at 2.0. Similarly, Task-2 of schedule 3 and Task-1 of schedule 5 never start.

## 5.2 Completion Time

Task completion time is the time a system takes to perform a task (from start to finish). Figure 6 shows the completion time required by task groups 1, 2, and 3 for various schedules. Simulation results show that schedule 1 offers the best total completion time for all task groups. For task group 3, some tasks (example: Task-1 of schedule 2) never start.

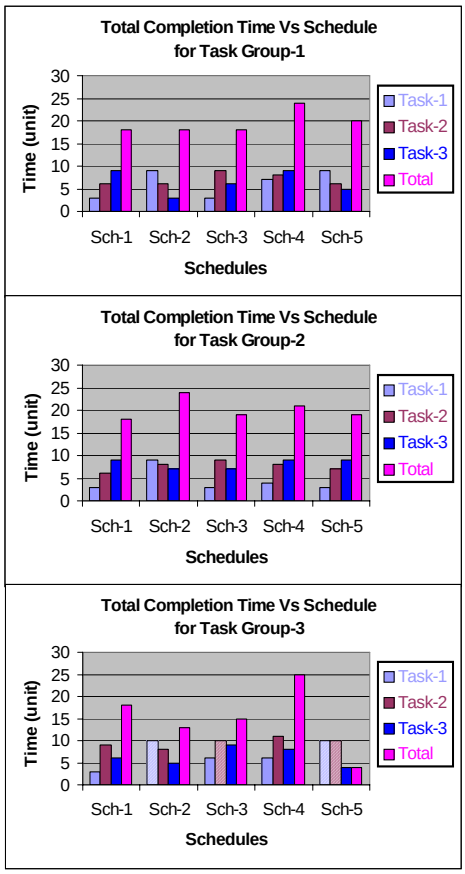



Figure 6: Total completion time versus schedules for task groups 1, 2, and 3.

## 6 CONCLUSIONS

In a distributed and parallel system, performance can be improved by properly distributing the concurrent tasks among the processors. However, the advantage of parallelism may be offset by the increased complexity of load balancing techniques. Scheduling is proven to be an effective technique for load balancing in distributed and parallel systems. Studies indicate that when the expected workload is (almost) known, static scheduling may be a potential choice to balance the load in such a system (Renard, 2003), (Zhangt, 1995). Therefore, static scheduling may be used in order to improve the overall system performance by balancing the load. In this paper, we analyze the performance of load balancing technique by static scheduling for a distributed and parallel system. We develop a simulation program using VisualSim. Simulated architecture includes three processors working on a single task simultaneously using the shared memory (and one controlling processor to run the scheduler). We obtain the performance in terms of the average response time and the total completion time for different scheduling algorithms and task groups. Simulation results show that scheduling technique has significant impact on load balancing. Simulation results also show that load balancing by scheduling can be used to improve the performance of distributed and parallel systems.

We plan to evaluate the performance of dynamic scheduling and load balancing in distributed and parallel systems in our next endeavour.

## REFERENCES

Asaduzzaman, A., Mahgoub, I., 2004. Evaluation of Application-Specific Multiprocessor Mobile System, SPECTS'04

Amoroso, A., Marzullo, K., 2006. Multiple Job Scheduling in a Connection-Limited Data Parallel System, IEEE Tran on Parallel & Distributed Systems

Dunigan, T., Vetter, J., White, J., Worley, P., 2005. Performance Evaluation of the Cray X1 Distributed Shared-Memory Architecture, IEEE Computer Society

Wikipedia, 2007. Distributed and Parallel Computing, http://en.wikipedia.org/wiki/

Renard, H., Robert, Y., Vivien, F., 2003. Static load-balancing techniques for iterative computations on heterogeneous clusters,

Zhangt, Y., Hakozakit, K., Kamedat, H., Shimizu, K., 1995. A Performance Comparison of Adaptive and Static Load Balancing in Heterogeneous Distributed Systems, IEEE

Ichikawa, S., Yamashita, S., 2000. Static Load Balancing of Parallel PDE Solver for Distributed Computing Environment, Proceedings ISCA 13th Int'l Conf. on Parallel and Distributed Computing Systems

Lee, S., Yang, C., Tseng, S., Tsai, C., 2000. A Cost-Effective Scheduling with Load Balancing for Multiprocessor Systems

Kim, C., Kameda, H., 1992. An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems, IEEE Transactions on Computers

Soklic, M., 2002. Simulation of load balancing algorithms: a comparative study, ACM Press, New York

Anguille, L., Killough, J., LI, T., Toepfer, J., 1995. Static and dynamic load-balancing strategies for parallel reservoir simulation, Symposium on reservoir simulation, TX

Kameda, H., Fathyy, E., Ryu, I., Li, J., 2000. A Performance Comparison of Dynamic vs. Static Load Balancing Policies in a Mainframe - Personal Computer Network Model, CDC00-INV1601

Zhang, Y., Kameda, H., Shimizu, K., 1991. Parametric analysis of optimal static load balancing in distributed computer systems. J. Inf. Process.

Magee, J., Kramer, J., 2000. Concurrency: State Models & Java Programs, John Wiley & Sons Publisher, London, 2Rev Ed edition.

VisualSim, 2007. System-level simulator from Mirabilis Design, Inc. http://www.mirabilisdesign.com