6th International Conference on Telecommunications
Maja Matijašević and Alen Bažant (eds.)
June 13–15, 2001, Zagreb, Croatia

ConTEL 2001

Different Approaches to Experiments for Load Balancing Method Comparison

Saša Dešić
Sasa.Desic@etk.ericsson.se

Research & Development Centre
Ericsson Nikola Tesla d.d.
Zagreb, Croatia, HR-10000

Abstract

Development of effective techniques for the task distribution on multiple processors or nodes is considered as one of the biggest issues in parallel and distributed operating environments. The purpose of load balancing methods is to assign arriving or internally generated tasks or jobs to processing nodes of a distributed system in a way that provides the highest system utilization and maintains fairness. A great number of load balancing methods has been developed, so far. Each method distinguishes itself in a basic property like scope of activity (local vs. global), cooperation among nodes, distribution (places where decisions are made) (distributed vs. centralized) and optimality (optimal vs. sub-optimal). In order to get the best performance and choose the method that meets application requirements, first the group of methods, and then a particular method have to be chosen according to a specially developed procedure. Such an approach has to be taken because there is an entire group of methods which satisfies the initial proposed requirements.

Several approaches to load balancing methods assessment have been presented worldwide. In addition, experiments with different approaches to performance measurements have been made.

This article deals with approaches to load balancing methods assessment and parameters that are in use. Firstly, load balancing methods are categorized within a taxonomy. Subsequently, the article shows some approaches to load balancing methods comparison and describes two load balancing experiments (conducted at Faculty of Electrical Engineering and Computing, University of Zagreb). Two different approaches to method's performance assessment are applied: real-time and mathematical simulation. Some advantages and drawbacks of selected approaches are discussed. Finally, new parameters for method assessment are introduced.

1. Introduction

In distributed systems, like telecommunications systems, it is very common to have unequal distribution of tasks per node. In that case, it is possible to have one overwhelmed node while at the same time several nodes can be lightly loaded. The reasons for such unfair task distribution may lay in bad system design (inaccurate prediction of demands and respective capacity) or fluctuations in request incoming rate (which may be caused by the change in the number of users or their location). Situation will result in unexpected delays in service activity and availability (overwhelmed nodes will probably expirience problems with servicing all incoming tasks).

To avoid situation described above load balancing methods are introduced. The purpose of load balancing is to assign received tasks to processing nodes in a way that every node has approximately the same amount of tasks to do. Additionally, load balancing methods indirectly decrease the system's response time.

Implementation of the load balancing methods can be considered as a solution to the uneven task distribution problem. Such an approach requires the selection of an appropriate group of task scheduling methods. While task scheduling introduces load balancing as one of its sub-categories, almost the entire taxonomy can be applied to load balancing.

A group of task scheduling methods is to be chosen regarding the most important working characteristics. Section 2 briefly describes different groups of task scheduling methods.

Selection of a particular load balancing method should be made after a method comparison, based on previously developed methods. Chapter 3 presents several applicable methods and results of their comparison.

Experiments, advantages and disadvantages of two opposite methods for load balancing methods assessment are described in section 4. Last section summarizes the described approaches.

2. Taxonomy of task scheduling methods

Due to a wide variety of approaches to the task scheduling problem, it is difficult to meaningfully compare different systems since there is no uniform means for their qualitative or quantitative evaluation. This section introduces taxonomy [1] of approaches to the resource management problem with purpose to provide a common terminology and classification method necessary in addressing this problem.

The presented taxonomy is a hybrid of *flat* and *hierarchical* classification schemes – hierarchical as long as possible in order to reduce the total number of classes, and flat when the descriptors of the system may be chosen in an arbitrary order.

The structure of hierarchical portion is shown in Figure 1.

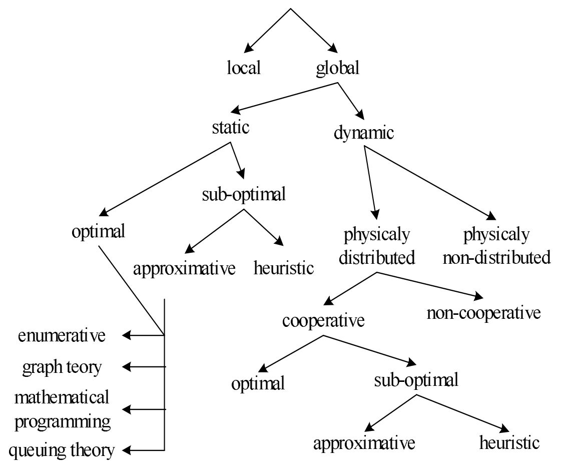

Figure 1. Task scheduling characteristics

Local vs. Global: Local scheduling is involved with the assignment of processes to the time-slices of a single processor. Global scheduling is the problem of deciding where to execute a process (specific network node or specific process in multi-processors system should be selected).

Static vs. Dynamic: In static scheduling, information regarding processes in the system as well as all other system information (like processors power) is known before system startup. On the basis of that information and supposed distribution of load, task scheduling is done prior to start of execution. In dynamic scheduling, information about system is collected during execution time, and it is used in real-time decision-making process.

Optimal vs. Suboptimal: In the case that all information regarding the state of the system as well as the resources needs of a process are known, an optimal assignment can be made based on some criterion function. In the case that these problems are computationally unfeasible, suboptimal solutions may be tried.

Approximate vs. Heuristic: Approximate, instead of searching the entire solution space for an optimal solution, is satisfied when find a “good” one. Heuristic algorithms make the most realistic assumptions about a priori knowledge concerning proves and system loading characteristics. Collected knowledge is used for load balancing decisions. There are four categories of common approaches for optimal and suboptimal-approximate solutions:

- Solution space enumeration and search
- Graph theory
- Mathematical programming
- Queuing theory

Distributed vs. Non-distributed: Responsibility for the task of global dynamic scheduling can physically reside in a single processor (physically non-distributed) or the work involved in making decision can be physically distributed among the processors.

Cooperative vs. Non-cooperative: It may be distinguished between those methods that involve cooperation between the distributed components (cooperative) and those in which the individual processors make decision independent of the actions of the other processors (non-cooperative).

Scheduling algorithms from every group (Figure 1) can furthermore differentiate regarding second, flat classification scheme, as follows:

Adaptive vs. Non-adaptive: An adaptive solution to the scheduling problem is one in which the algorithm and parameters used to implement the scheduling policy change dynamically according to the previous and current behavior of the system in response to previous decision made by the scheduling system.

Load Balancing: This category of policies approaches the problem with the philosophy that being fair to the hardware resources of the system is good for the users of the system. The basic idea is to attempt to balance the load on all processors in such a way as to allow progress by all processes on all nodes to proceed at approximately the same rate.

Bidding: In this class of policy methods, a basic functioning includes sending bids to all nodes. Latter, tasks are scheduled regarding a principle “best offer wins”.

Probabilistic: Sometimes it is very hard (computationally complex) to make deterministic decision. Instead, idea of randomly choosing some node as the next to assign is used. Repeatedly using this method, a number of schedules may be generated, and then this set is analyzed to choose the best from among those randomly generated.

One-time assignment vs. Dynamic Reassignment: Task distribution on task arrival (one-time assignment) looks static when capability of reassignment is possible. Dynamic Reassignment offers opportunity of rearranging initial task distribution.

After describing all these opposite approaches, it can be concluded that problem of choosing appropriate group isn't trivial. But, correct methods group can be selected by making decision, according to the requirements and system architecture, in nodes of taxonomy tree (Figure 1).

3. Parameters for load balancing methods assessment

After selection of one group of load balancing methods, following step is selection refinement to one method. To select the best method, methods capabilities have to be questioned. Desirable effects that load balancing strategy cam have on the system are as following:

- optimal overall system performance – total processing capacity maximizes while retaining acceptable delays,
- fairness of service – uniformly acceptable performance provided to jobs regardless of the source from which the job arrives,
- failure tolerance – robustness of performance maintained in the presence of partial failures in the system.

To investigate performance, different approaches could be applied. If the goal is to distribute tasks approximately uniquely over nodes, then node's load (number of tasks at node in the moment) can be compared with the average load of all nodes (Figure 2).

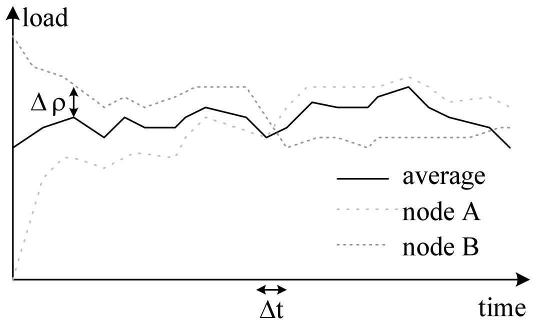

Figure 2. Comparing node load with average load

If the main goal is user satisfaction, the most appropriate parameter to measure is response time. By measuring and comparing response times, it can be concluded which method is most suitable for user.

In this approach system is observed as a black box. After sending a job into the system, it is necessary to wait until report about job's completion is received from the system (Figure 3).

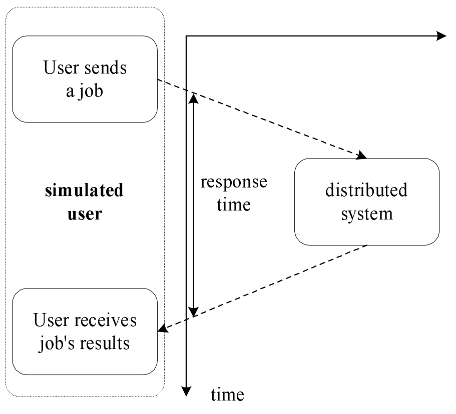

Figure 3. Measurement of system response time

3.1. Performance metric by comparison with predefined method

One example of complete load balancing experiment and measurement is described in [2]. This approach uses one method (FCFS) as a referent method. FCSF (First Come First Served) method is chosen because a good load balancing [2] will tend not to allow any server to be idle while there are jobs awaiting processing in the system. In FCFS the whole system is functioning as a one huge processor. In real life, execution of such a method is very questionable because it requires a lot of resources (communicating cost are crucial). But, in this experiment, it works fine because communication and processing cost for methods are not included.

So, metric for method A can be defined as a ratio between mean response time over all jobs under FCFS method and the highest response time per nodes under method A.

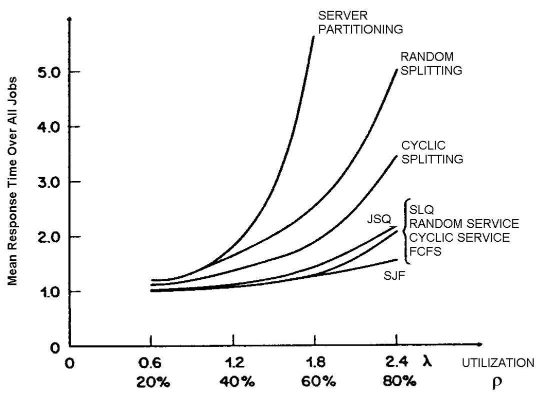

Figure 4. Mean response time (over all jobs) as a function of utilization

Response time was referred to as a very popular parameter for method assessment. In [2], analysis starts with graph that shows response times for different methods (Figure 4).

For better understanding, load balancing methods from the graph (Figure 4) are described briefly:

- Source partition – The sources (nodes that receive tasks) are partitioned into groups and each group is served by one server.
- Server partition – The servers are partitioned into groups and each group of servers serves one source.
- Random splitting – Each source distributes jobs randomly and uniformly to each server, i.e. it sends a job to one of the *K* servers with probability *1/K*.
- Random service – Each server visits sources at random and on each visit removes and serves a single job.
- Cyclic splitting – Each source assigns its *i*th arriving job to the *i*(mod *K*)th server.
- Cyclic service – Each server visits the sources in a cyclic manner and removes and serves a single job.
- Join the Shortest Queue (JSQ) – Each source independently sends an arriving job to the server that has the least number of jobs.
- Serve the Longest Queue (SLQ) – A dual to JSQ is the algorithm in which whenever a server becomes available it serves a job from the longest source queue.
- Shortest Job First (SJF) – Servers select the currently waiting job that has the smallest service time requiring.

Pervious graph (Figure 4) can be normalized in respect to FCFS (Figure 5).

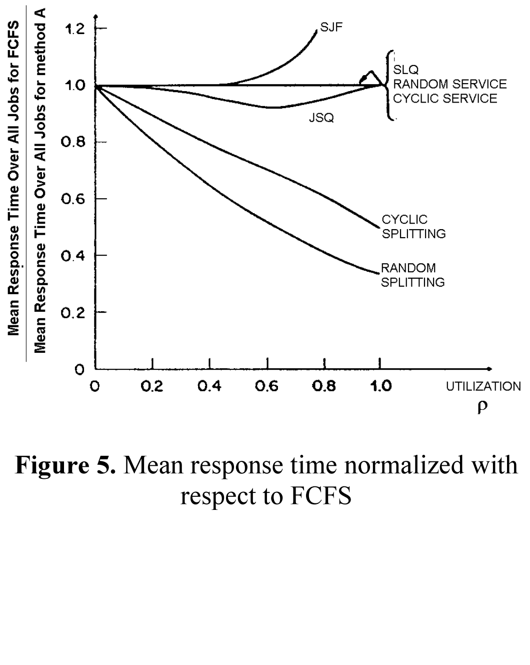

Figure 5. Mean response time normalized with respect to FCFS

This experiment shows experiment which is held in a laboratory environment and which doesn't take care about communication costs. Also, system response time is used for methods assessment.

3.2. Performance metric with response time and communication costs

Measurements and results from previous section do not include computational, communication and other costs that load balancing includes by itself. The main drawback of procedure described in previous section is that the (negative) impact of load balancing method overhead is actually unknown.

Overhead introduced into load balancing experiments by load balancing methods can significantly influence system performance. In [3], communicational and processing costs introduced by load balancing methods are included into elaboration.

The three load balancing methods are included into experiments:

- Random – same as Random splitting from previous section
- Threshold – under this method a node is randomly selected and probed to determine whether the transfer of a task to that node would place it above threshold. If not, then the task is transferred.
- Shortest – same as Join the Shortest Queue (JSQ) from previous section

Described methods are compared to each other and to two "bounding" cases: no load balancing (represented by *K* independent *M/M/1* queues where *K* is number of nodes) and perfect load balancing at zero cost (represented by *M/M/K* queue).

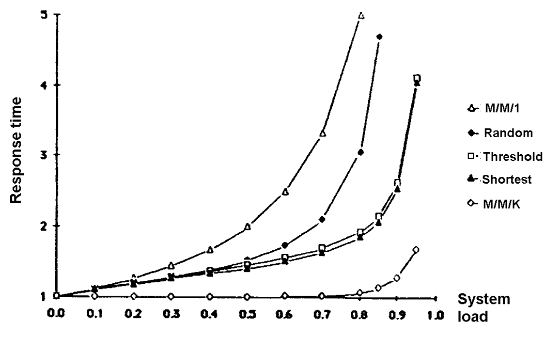

Figure 6. Response time versus load

Figure 6 represents principal performance comparison. Communication costs are included and they are fixed.

Till now communication costs were not included into measurement. But, it is very interesting to see how increase of communication cost influence method's performance. At the beginning, one method could be fixed and communication cost could be raised slowly.

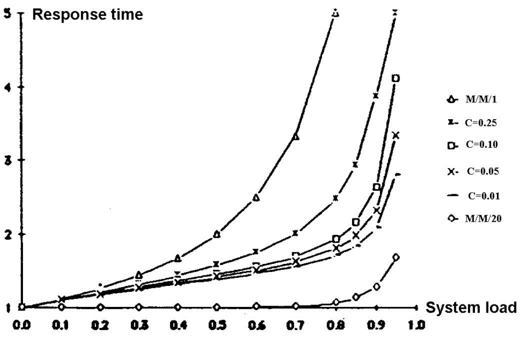

Figure 7. Response time versus load for various transfer costs

Selected method is Threshold method (Figure 7) and it is obvious that transfer costs could significantly decrease system performance by rising response time.

In addition, the impact of increased communication cost to all methods, can be examined, especially the most complex (Figure 8). As it can be expected, highest impact communication costs have on the most complex method – Shortest.

Finally, it can be concluded that this method, which includes communicating and processing cost, is more detailed that method explained in the previous chapter. Communicating costs are included and it is show that communicating costs can strongly influence system performance.

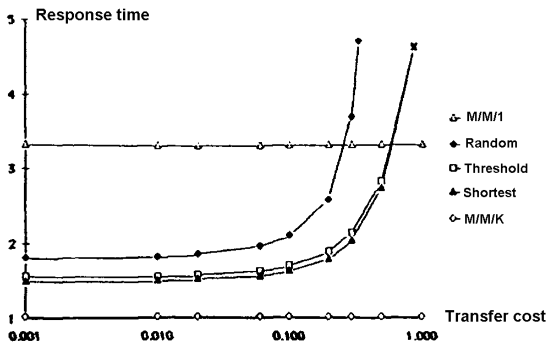

Figure 8. Response time versus transfer costs

4. Experiments for load balancing methods assessment

Assessment of load balancing methods is usually done in laboratory environment by changing parameters that can affect the method's performance. In experimental validation, system performance is observed, and reactions and conclusions regarding changes are noted. Usually, some existing network and nodes are used. However, in particular cases novel networks could be built for the experimental purposes [5].

Common problem for the described type of simulation is a lack of real load. Artificial load generator is included into the simulation. It is required that the generator must have the capability of creating load with different statistical disciplines of tasks arrival and execution times. Load generator can be implemented as a simulation of environment that surrounds the distributed system (Figure 9). Management node is also included into the simulation and it is used as an insight into the system under test.

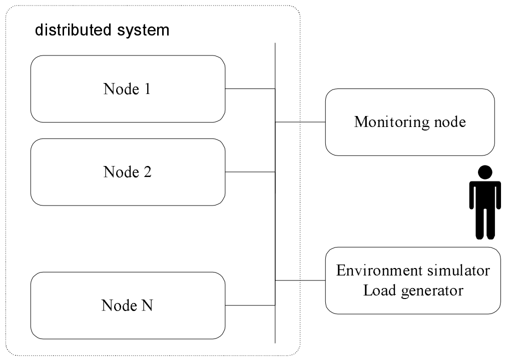

Figure 9. Load generator in experiments

Mathematical simulation raises the level of experiment abstraction. Using mathematical simulation of the distributed system and load balancing methods, it is possible to investigate events that are very hard to observe with the real time simulation. Another advantage of this type of simulation is its low cost. A designer only needs one computer for running simulation instead of a whole network. However, the critical task in the implementation is the translation of a real time system into a mathematical model. Mistakes could dramatically affect results of the investigation.

4.1. Real time simulations for load balancing methods assessment

This section describes the experiments that were conducted at the Department of Telecommunications at Faculty of Electrical Engineering and Computing, University of Zagreb. The real time simulation consists of the following elements (Figure 10):

- environment simulator
- artificial load generator
- system management
- data gathering and presentation
- local node operations

Programming language Erlang [6] was used in the simulations development.

The user can set up, start, stop and monitor the simulation through graphical user interface. During the preparation for the simulation, the user can add nodes to the simulation, choose a statistical distribution for task inter-arrival time and execution length. The load balancing method is selected, too. After simulation start-up, the artificial load generator is also started and it will feed working nodes with tasks, according to the set values of inter-arrival times. Actually, the load generator only sends task type and parameters.

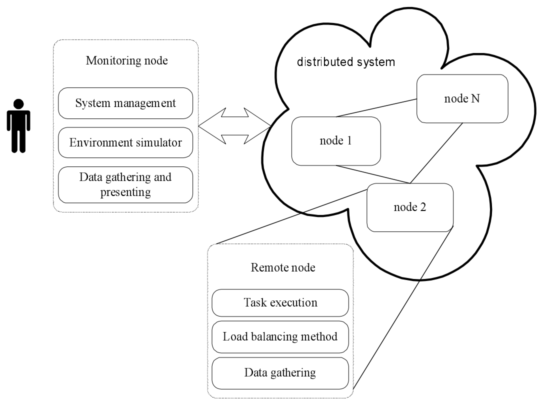

Figure 10. Simulation configuration and basic elements

On the other side, working node has several processes. All processes are started after the user adds a node into the simulation. When the node receives a task request, load balancer is consulted for a location of the task execution. If the node load is low, the task will be executed locally. In the other case, the load balancer will search for a node that is ready to receive a task. If it succeeds a task request will be sent, or else the task must be executed locally or otherwise rejected.

The four load balancing methods are included into the experiments: Random, Joint Shortest Queue (JSQ), Serve Longest Queue (SLQ) and Bidding.

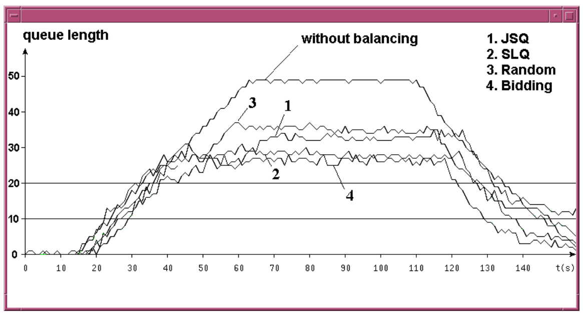

Figure 11. Queue length as a parameter for comparison

As a comparison parameter, the length of processor's queue on a specific node was selected. Changes of queue length are observed through time for all methods (Figure 11). Measuring of queue length indirectly includes all types of communicating costs and also the overhead which methods involve. Mean inter-arrival rate was set to 1 second, mean execution time was set to 100 seconds and total number of jobs was set to 45.

Results from the experiment prove that using any load balancing method could significantly improve system performance. System performance can be compared with the situation where the load balancing method has not been activated. Of course, the random method resulted in the worst performance.

Furthermore, by changing the parameters of generated load, some interesting results were achieved. For example, the following graphs show the method's behavior in case of a double increase (Figure 12) and a double decrease (Figure 13) of task arrival intensity. Mean inter-arrival rate were set to 2 and 0.5 seconds, mean execution time was set to 70 seconds and total number of jobs was set to 45.

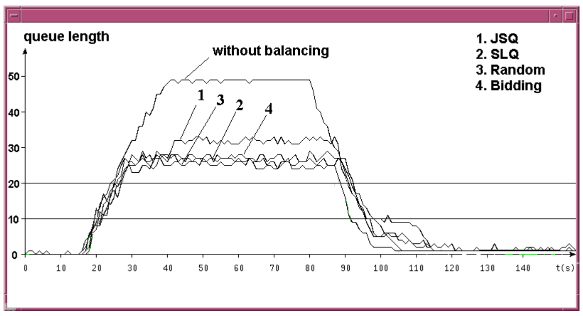

Figure 12. Methods performance at high task arrival rate

The change of load arrival intensity showed that all methods had almost the same results (only JSQ has distinguished performances). Even the simplest method (random) had a performance that is very close to the most complex method.

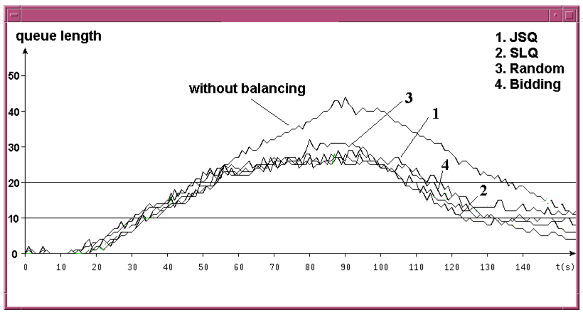

Figure 13. Methods performance at low task arrival rate

This briefly described methodology and its results produce parameters for methods assessment. A great advantage of the approach with real time simulations is a system testing in conditions of real environment that have characteristics of the target system. Also, the communicating cost and overhead that load balancing involves are included into measurements.

However, the main disadvantage of this approach is in the difficulty in extracting some additional information about the simulation when the simulation is finished. In order to give extra information, the simulator has to be preprogrammed. In addition, the simulation designer is usually concentrated on problems of simulator design and functioning. The designer has to take care of the data gathering, transferring etc.

4.2. Mathematical simulations for load balancing methods assessment

The mathematical simulation of a distributed system has one great advantage compared to the real time simulations – everything we want to know about the system can be extracted easily from a mathematical model. System is defined through matrices that describe parameters of the real system.

Approximation of the system with a mathematical model looks very convenient but it can be very dangerous. Misinterpretation of a system's parameter could produce the wrong mathematical model and differences regarding basic model. Finally, the wrong conclusions could be drawn. Another potential problem is the obligation to model some common things from real life i.e. communication between nodes and node processing.

Experiments with mathematical simulations were made at Department of Telecommunications at Faculty of Electrical Engineering and Computing, University of Zagreb [4].

The two load balancing methods are included into experiments: Random and Joint Shortest Queue (JSQ). Results are compared with the system without load balancing method.

As parameters to compare, two different parameters are measured: response time and average deviation from mean number of jobs per node.

Average number of jobs per node can be defined as:

$$\overline{q}_j = \frac{\sum_{i=1}^{M} q_{ij}}{M} \qquad (1)$$

where M is number of nodes, and q_{ij} is a number of jobs at the moment j (time interval). Average deviation from mean number of jobs per node Q can be defined as:

$$Q = \max_{i=1,M} \sum_{j=1}^{P} \left| \overline{q}_j - q_{ij} \right| \qquad (2)$$

where P is number of time intervals occupied by simulation.

Performance is observed regarding the time, but more interesting information could be extracted by considering the influence of several other parameters. In order to emphasize the positive influence of load balancing methods, the experiment can be arranged so that a great number of jobs additionally slows the node on which they reside. This can be implied through "slowing" factor (SF). "Slowing" factor increases execution time (ET) of every job on node by value proportional to number of jobs (NJ) on the node. Higher "slowing" factor produces longer execution time:

$$ET = ET_{normal} \cdot SF_i \cdot NJ \qquad (3)$$

Reactions of load balancing mechanisms are shown at following graphs: response time (Figure 14) and average deviation from mean number of jobs per node Q (Figure 15).

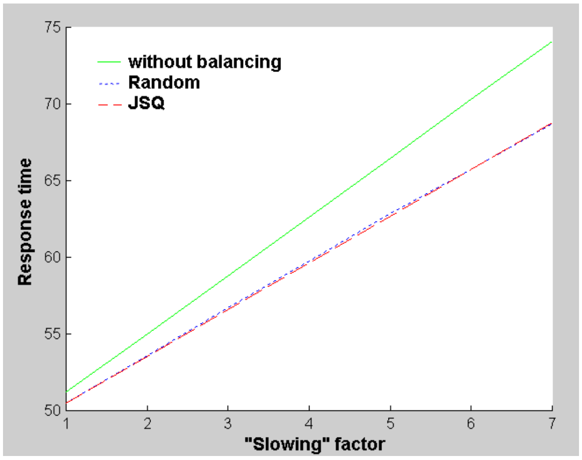

Figure 14. Response time against "slowing" factor

Furthermore, some other parameters could be changed in order to examine the method's behavior [4]: maximum number of jobs per node, threshold L that is used in deciding to move a task to an other node, transferring costs, communication cost etc.

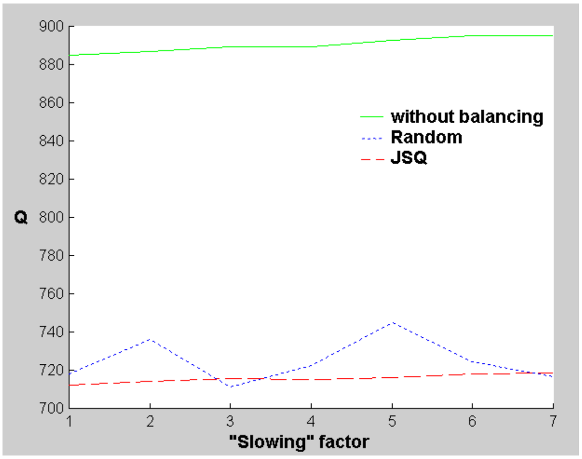

Figure 15. Average deviation from mean number of jobs per node Q against "slowing" factor

Finally, it can be concluded that a mathematical model offers a great number of varieties for experimenting with the load balancing mechanism. Some of these parameters could not be changed anyway in real time simulations. By choosing different parameters for analysis, some different aspect of load balancing methods performance could be emphasized and analyzed.

5. Summary and Conclusions

This article describes different approaches to load balancing methods assessment. Additionally, some experimental results are presented and some parameters for methods assessment (e.g. response time or queue size) are introduced.

The main aim of this article was neither a particular evaluation of methods specification, nor the results of experiments. The selected graphs only emphasize the different approaches and parameter evaluation.

Real time and mathematical simulations of a distributed system that includes load balancing methods are created and some experiments are made.

The presented analysis shows that both mathematical and real time simulations have their advantages and disadvantages. Real time simulations describe the real system well. However, they have a complex task in getting some values from of the system. Mathematical simulations can emphasize some aspects that are unseen or too fast to be noticed when using real time simulations. Furthermore, they are suitable for different types of experiments enabling the adjustment of crucial parameters for system description.

In conclusion, particular application determines the measurements in order to select the most appropriate and successful approach in solving the load balancing problem.

References

[1] T.L. Casavant, J. G. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems" *IEEE Transactions on Software Engineering*, Vol. 14, pp. 141-153, 1988.

[2] Y. T. Wang, "Load Sharing in Distributed Systems", *IEEE Transactions on Computers*, Vol. C-34, pp. 204-217, 1985.

[3] D. L. Eager, "Adaptive Load Sharing in Homogeneous Distributed Systems", *IEEE Transactions on Software Engineering*, Vol. SE-12, No 5. pp. 662-675, 1986.

[4] S. Desic, "*Comparison and experimental proof of methods for load balancing in distributed systems*" (croatian only), Master thesis, Faculty of Electrical Engineering and Computing, University of Zagreb, 1999.

[5] M. Hailperin, "*Load Balancing Using Time Series Analysis for Soft Real Time Systems with Statistically Periodical Loads*", Ph.D. thesis, Department of Computer Science, Stanford University, 1993.

[6] J. Armstrong, R. Virding, C. Wikstrom, M. Williams, "*Concurrent Programming in Erlang*", Prentice Hall, 1996.

[7] T. Kunz, "The Influence of Different Workload Description on a Heuristic Load Balancing Scheme", *IEEE Transaction on Software Engineering*, vol 17, no. 7, 1991.

[8] S. P. Dandamudi, "Sensitivity Evaluation of Dynamic Load Sharing in Distributed Systems", *IEEE Concurrency*, 1998.

[9] M.G. Sriram, M. Singhal, "Measures of the Potential for Load Sharing in Distributed Computing Systems", *IEEE Transaction on Software Engineering*, vol 21, no. 5, 1995.