Initiating Load Balancing Operations

Marta Beltrán, Jose L. Bosque, and Antonio Guzmán

DIET, ESCET, Rey Juan Carlos University, 28933 Móstoles, Madrid, Spain
{mbeltran,jbosque,aguzman}@escet.urjc.es

Abstract. The initiation rule of a load balancing algorithm determines when to begin a new load balancing operation. Therefore, it is critical to achieve the desired system performance. This paper proposes a generalized procedure for deriving initiation mechanisms or rules based on different objectives for the load balancing algorithm. A new metric, the initiation efficiency, is defined in order to evaluate the initiation performance and to compare the different alternatives.

1 Introduction

Load balancing is critical for achieving high performance in clusters and Grid systems because it enables an effective and efficient utilization of all the available resources ([1],[2]). Dynamic load balancing algorithms can be decomposed in different rules or policies ([3], [4], [5]). But all these decompositions have something in common: it is necessary an initiation mechanism to decide on each system node when to begin a load balancing operation. This mechanism must be efficient, scalable, low overheading, and must be capable of deciding about load balancing operations taking into consideration the available system and workload information.

Different solutions have been proposed for the initiation rule. There are sender-initiated ([5], [6], [7]), receiver-initiated ([8]), symmetric ([9]) and periodic ([10]) rules. On the other hand, some of these solutions are completely local ([5], [7], [11]), i.e, each node evaluates only its own state to determine if a load balancing operations is necessary or not, while other are global ([12]), taking into consideration the global system state.

An exhaustive analysis of all these alternatives allows to conclude that they are designed for a particular load balancing algorithm. The main contributions of this paper are a procedure for deriving initiation mechanisms from general objectives for load balancing algorithms and a performance metric for the initiation rule, the initiation efficiency (ε). It has been defined in order to evaluate the initiation mechanisms performance and to compare the different solutions. For illustration, three example objectives have been proposed to derive their correspondent initiation mechanisms and to compare their performance using the new defined metric.

The rest of this paper is organized as follows. Section 2 proposes the general methodology for obtaining initiation policies from the objectives of the load balancing algorithms. Section 3 illustrates this methodology with three different examples of load balancing objectives. Section 4 defines the initiation efficiency necessary to evaluate these initiation mechanisms performance and to establish comparisons. Section 5 presents some experimental results for the example cases and finally, Section 6 with conclusions.

J.C. Cunha and P.D. Medeiros (Eds.): Euro-Par 2005, LNCS 3648, pp. 292–301, 2005.

2 Generalized Procedure for Deriving Initiation Rules

To implement the initiation mechanism for a load balancing algorithm, it is necessary to decide when a load balancing operation should be requested, considering both its possible benefits, and on the other hand, the overhead it will cause. This section proposes a general procedure for designing initiation mechanisms for load balancing algorithms taking into account their general objectives. The steps of this procedure are the following:

1. Describe quantitatively the requirements for the load balancing algorithm. Let ω be the objective that should be achieved with the algorithm to ensure that these requirements are met.
2. Identify an objective function to quantify the achievement of this objective. Let ϕ^{ω} be the objective function for ω. This objective function must depend on the available information about the local and the global state. Mathematically, it should be expressed as $\phi^{\omega}(I)$ where I is a vector composed of the system nodes load indexes (these load indexes quantify the system nodes computing capabilities and must be updated in all the system nodes with some kind of information policy).
3. Incorporate this objective function to the initiation mechanism in the load balancing algorithm. For this last step two kind of objective functions can be distinguished:
   - **Boundary functions:** In this case the objective function defines an upper or lower bound to a certain magnitude. This condition can be directly transferred to the initiation mechanism.
   - **Optimization functions:** The objective function requires the optimization of a certain magnitude. Even though this kind of functions can be sometimes easily incorporated to the initiation rule, they usually introduce too much overhead in the algorithm. To evaluate the load balancing operation it is necessary to solve an optimization problem and such computation may be very expensive for a dynamic load balancing algorithm. In such cases the optimization objective should be transformed to a boundary one, defining an upper or lower bound for the magnitude that was initially supposed to be optimized.

3 Some Initiation Rule Examples

3.1 Objective 1: Maximize System Load Balance

In this first example the aim of the algorithm is to maximize the balance among the system nodes. This is usually the main objective of any load balancing algorithm but it is not the only one as it will be seen later. Once this objective is identified (step 1 of the proposed procedure), an objective function can be defined for the step 2. Let b denote the system balance, therefore, the objective function is:

$$\phi^{1} : max(b) \tag{1}$$

The larger the b value, the more balanced is the load of the system. This balance can be quantified at a given instant as the ratio of the minimum load index to the maximum.

That is, it can vary from 0 to 1. With this definition, the objective function can be denoted:

$$\phi^1(I) : max\left(\frac{I_{min}}{I_{max}}\right) \tag{2}$$

But this is an optimization function, so every time a new task arrives to a system node, its initiation mechanism must evaluate the b value for each possible allocation for this task, searching in each case the minimum and the maximum load indexes in the system. This process can suppose a great overhead for the load balancing algorithm, specially in systems with a large number of nodes. To overcome this scalability limitation, the objective function is transformed to a boundary one. This function is:

$$\phi^1(I) : \frac{I_{min}}{I_{max}} > \tau \tag{3}$$

Where τ is the algorithm tolerance value, which defines the threshold desired for the system balance. With this objective function, the step 3 of the general procedure is immediate. The initiation rule must try to allocate all the new tasks achieving this objective: the system balance must be always above the τ value after the allocation. This boundary condition can be directly transferred to the load balancing algorithm with a very simple evaluation, without solving the maximization problem.

3.2 Objective 2: Maximize System Throughput

Here the aim of the algorithm is to maximize the system throughput, that is, to minimize the individual processes elapsed time to finish as many tasks per time unit as possible. This objective is typically identified during the step 1 in systems executing independent tasks for high performance computing. It must be achieved using a load balancing algorithm, therefore, assigning each task to the system node in which its elapsed time will be the shortest. Let t_i be the elapsed time of th ith task, therefore the objective function proposed in the step 2 is:

$$\phi^2 : min(t_i) \ \ \forall i \tag{4}$$

But again ϕ^2 is an optimization function that implies evaluating the new task elapsed time for each possible allocation before assigning it to the best system node in terms of this time requirement. Therefore, this objective function is transformed to a boundary one. In this case the bound is referred to the elapsed time of the new task in its *home node*, that is, the node to which this task is initially assigned. The objective function is:

$$\phi^2 : t_i < \tau \cdot \hat{t_i} \ \ \forall i \tag{5}$$

Where $\hat{t_i}$ is the elapsed time of the i*th* task in its home node and τ is the algorithm tolerance to define the threshold desired for this objective. For the step 3, this objective function implies that the initiation rule tries to allocate new tasks to nodes where their elapsed time will not be more than τ times greater than in their home nodes. In this paper the DYPAP monitor ([13]) is used to provide local and global state information to the load balancing algorithm, so the objective function can be based on the load indexes values provided by this tool. The load index provided by the DYPAP monitor is based

on the CPU assignment, defined as the percentage of CPU time that would be available for a new task on a system node, and on the nodes computational powers.

In the home node the objective function is achieved if the CPU assignment for the new task is:

$$\alpha = \frac{1}{\tau} \tag{6}$$

This CPU assignment is the minimum necessary to accomplish the elapsed time requirements for the new task on this node, but due to system heterogeneity, this may not be the minimum in another node (for example, on a node with more computational power). Therefore, the objective function is based on the load indexes values, which take into account the nodes computational powers. The minimum index necessary to achieve the objective is :

$$I_{min} = \alpha_{min} \cdot \frac{P_{home}}{P_{max}} = \frac{1}{\tau} \cdot \frac{P_{home}}{P_{max}} \tag{7}$$

Where P_{home} is the computational power of the home node and P_{max} is the computational power of the most powerful system node. That is:

$$\phi^2(I) : I > \frac{1}{\tau} \cdot \frac{P_{home}}{P_{max}} \tag{8}$$

The initiation mechanism of the load balancing algorithm must look for a node which fulfill this bound to allocate the new task.

3.3 Objective 3: Minimize the Application Elapsed Time

In this last example the aim of the algorithm is to obtain the best elapsed time for the application executing on the system. Once the objective is identified in the step 1, the objective function must be proposed in the step 2. Let T denote this elapsed time:

$$\phi^3 : min(T) \tag{9}$$

Again this objective must be accomplished using a load balancing algorithm, i.e., allocating new tasks in the best way for this objective function. In this case, this function can be easily implemented in the initiation mechanism despite it is an optimization function:

$$\phi^3(I) : max(I) \; or \; min(I) \tag{10}$$

That is, in the step 3, new tasks are always assigned to the system node with the lowest or greatest load index, depending on this index meaning. For example, assuming again the utilization of the DYPAP monitoring tool, the index maximization should be used because the system node with the greatest load index is the one which offers the best compromise between computational power and CPU assignment and can be easily found by the initiation mechanism without evaluating any expression or predicting the system behavior. In this example the optimization only implies the search of the system node with the greatest index value and this does not introduce too much overhead in the algorithm. Therefore, it may not be necessary to transform it into a boundary function.

4 Initiation Efficiency

In order to evaluate the different initiation rules performance and to compare the different alternatives, an initiation performance metric is needed. In this paper, the initiation efficiency (ε) is defined considering two important issues: the ratio of accepted load balancing operations to the requested operations (R) and the degree of achievement for the load balancing algorithm objective (A). Therefore, the efficiency definition is:

$$\varepsilon = R \cdot A \tag{11}$$

The first factor must be taken into account because a good initiation mechanism should begin load balancing operations only when they are going to be accepted. The rejected operations imply an unnecessary overhead to the system, specially to the network. If the mechanism is not efficient or if it is, but it does not have updated information to decide about load balancing operations, some load balancing operations might be rejected in the target node. The ratio of the accepted operations to the requested operations quantifies the efficiency of the initiation rule in this sense (the largest value being 1 in the best case):

$$R = \frac{O_{acep}}{O_{req}} \tag{12}$$

Where O_{acep} is the number of accepted load balancing operations and O_{req} the number of requested operations.

On the other hand, an efficient initiation rule should comply with the objective of the load balancing algorithm. Due to inaccuracies in the state information, to wrong initiation mechanisms or to very demanding requirements this objective might not be achieved. The degree of achievement of the load balancing algorithm objective is quantified in a different way for the boundary and optimization functions:

- **Boundary objective function:** The degree of achievement of the objective can be measured with the ratio of tasks which are assigned accomplishing the proposed objective to the total number of assigned tasks:

$$A = \frac{tasks\ accomplishing\ the\ objective}{N} \tag{13}$$

  Where N denotes the total number of tasks composing the executed application.
- **Optimization objective function:** In this case, the magnitude or attribute that has to be optimized (M) gives the degree of achievement of the objective. Its value can be referred to its optimal value (M_{op}) to quantify how near is the system to the optimal situation:

$$A = \frac{M_{op}}{M}\ or\ A = \frac{M}{M_{op}} \tag{14}$$

  Depending on the kind of optimization the first equation (for a minimization) or the second equation (for a maximization) must be used.

Anyway, the A value is always normalized, varying from 0 in the worst case to 1 in the best case. With these definitions, for the initiation rules proposed in the previous section, the initiation efficiency can be measured as:

– **Objective 1:**

$$\varepsilon = \frac{O_{acep}}{O_{req}} \cdot \frac{tasks\ assigned\ with\ b > \tau}{N} \tag{15}$$

– **Objective 2:**

$$\varepsilon = \frac{O_{acep}}{O_{req}} \cdot \frac{tasks\ assigned\ with\ t < \tau \cdot \hat{t}}{N} \tag{16}$$

– **Objective 3:**

$$\varepsilon = \frac{O_{acep}}{O_{req}} \cdot \frac{T_{op}}{T} \tag{17}$$

To evaluate the application elapsed time in the optimum or perfect situation, when all the system load is perfectly balanced, only some information must be known ([14]): the number of tasks that compose the executed application (N), the number of system nodes (q) and their computational powers (P_i with $i = 1, ..., q$). The optimum time value can be obtained supposing that all the workload is sequentially executed on a system with computational power equal to the total computational power of the system (P_T), therefore:

$$T_{op} = \frac{N}{\sum_{i=1}^{q} P_i} = \frac{N}{P_T} \tag{18}$$

With the given definition, a perfect initiation rule would obtain $\varepsilon = 1$. It would request load balancing operations only when they are necessary and can be performed, that is, when they are going to be accepted. And in addition, it would completely achieve the load balancing objective, assigning all the tasks to accomplish this objective or obtaining the desired optimum situation.

5 Experimental Results

This section presents some experimental results to show the influence of the initiation mechanism on the load balancing algorithm performance and to establish the utility of the initiation efficiency in selecting the best initiation rule. These experiments have been performed on a 32 nodes heterogeneous cluster called Medusa. In all the experiments an application composed by 320 tasks is executed on this system. For simplicity, these tasks are independent, i.e. there are no communications between them. In addition it is assumed that they arrive periodically to the cluster, and that they are initially assigned to system nodes between n17 and n31. These assumptions have been made only to simplify the experiments but they are not part of the general formulation presented in previous sections. The computational power (P) for the different system nodes has been computed as the inverse of the elapsed time for this application tasks on each kind of node, being P_T=2.47 the global system computational power.

The elapsed time for the selected application is 436 s without the load balancing algorithm, and with equation 18, the elapsed time with an optimum balance would be T_{op}=129.38 s. In this context, the implemented load balancing algorithm must dynamically balance the system workload. This load balancing is based on the DYPAP model ([13]), therefore, it includes the DYPAP monitoring tool to periodically characterize the system nodes state. An event-driven information policy has been used to exchange this state information between the system nodes. And to evaluate the three proposed initiation mechanisms, different implementations of the load balancing algorithms have been used. But the only difference between all these implementations is the initiation rule, in order to establish fair comparisons and to draw general conclusions from the obtained results.

The implemented load balancing algorithm is based on non-preemptive tasks assignment, thus, the objective functions proposed in equations 3, 8 and 10 have been directly translated to the initiation mechanism:

- **Objective 1:** When a new task arrives to a cluster node, it must be assigned to obtain a balance greater than the algorithm tolerance (τ) after its allocation.
- **Objective 2:** In this case, it is required that the new task allocation achieves an elapsed time for this task no more than τ times its elapsed time in its home node (the node to which it was initially assigned when it arrived to the cluster).
- **Objective 3:** For this last objective function, the new task is always allocated to the node with the largest I value in order to minimize the application elapsed time: it is assumed that this kind of allocation always obtains the best elapsed time for the individual tasks and, thereby, for the global application.

In these three implementations, after checking the local and remote execution of the task, if the achievement of the initial objective is not possible, this requirement is relaxed to avoid blocking a task execution, for example, if it is impossible to comply with this objective in some environment. That is why the algorithm objective achievement not always equals 1. For the objectives 1 and 2, boundary functions have been used, thus, the algorithm tolerance is in both cases an implementation parameter. Tables 1, 2 and 3 show the results obtained for the three proposed initiation mechanisms, and for the two first objectives, different tolerance values have been considered. Each table shows the number of accepted (O_{acep}), requested (O_{req}) and rejected operations (O_{rej}), the application elapsed time (T), the A and R values, and finally, the initiation efficiency (ε) for the different algorithm implementations.

In table 1, results for the first objective are shown. The larger the value of the algorithm tolerance, the more restrictive is the initiation mechanism: more load balance is required in the system. This is why for the largest τ values, more load balancing operations are requested, because they are needed to achieve these exigent load balance requirements. But it can be seen that the increase of τ leads to a decrease of both A and R, due to the difficulty in finding a proper allocation to achieve the algorithm objective. Therefore, the initiation efficiency decreases when the τ value increases. The intuitive explanation for this behavior is that the more difficult is to find a good allocation the less efficient becomes the initiation mechanism. The best elapsed time for the application is obtained with the medium tolerance values. With low τ values, the load balance

Table 1. Experimental results with the objective 1

τ	O_{acep}	O_{req}	O_{rej}	T (s)	R	A	ε
0.20	291	328	37	304	0.89	0.96	0.85
0.30	286	335	49	289	0.85	0.86	0.74
0.40	313	395	82	276	0.79	0.82	0.65
0.45	318	403	85	272	0.79	0.80	0.63
0.50	325	402	77	269	0.81	0.74	0.60
0.60	318	410	92	286	0.78	0.63	0.49
0.70	315	419	104	297	0.75	0.55	0.41

Table 2. Experimental results with the objective 2

τ	O_{acep}	O_{req}	O_{rej}	T (s)	R	A	ε
1	315	423	108	372	0.74	0.92	0.68
2	313	420	107	311	0.75	0.98	0.73
3	240	332	92	259	0.72	1	0.72
4	197	286	89	292	0.69	1	0.69
5	184	263	79	316	0.70	1	0.70
6	163	234	71	329	0.70	1	0.70
10	142	210	68	359	0.68	1	0.68
15	121	177	56	388	0.68	1	0.68

Table 3. Experimental results with the objective 3

Version	O_{acep}	O_{req}	O_{rej}	T (s)	R	A	ε
Simple	300	340	40	323	0.88	0.40	0.35
Modified,F=1.2	288	318	30	320	0.91	0.40	0.37
Modified,F=1.8	258	315	57	280	0.82	0.46	0.38

required in the system is too low to give good elapsed times, but with the largest values, the tasks assignment becomes too complicated and this has a negative influence on the elapsed times.

For the second objective (table 2), similar conclusions can be derived. But in this case low tolerance values imply more restrictive requirements, therefore, more requested load balancing operations. The main difference with the first objective is that in this case the influence of the τ value on the A, R and ε values is not so significant. Similar efficiency values can be obtained with all the considered algorithm tolerances. This is due to the specific features of the performed experiment, that is, with this system-application combination, it is easier to achieve the objective 2 than the objective 1 even with the more restrictive requirements.

Finally, in table 3 the results for the objective 3 are shown . The 'simple' implementation is based on the objective function proposed in equation 10. But this optimization function leads to a poor system performance, in terms of elapsed time and initiation efficiency. So, an easy modification is proposed (the 'modified' version), to assign new tasks to the system node with the largest load index only when this index is F times greater than in the local node. The utilization of this threshold does not affect the objec-

tive achievement and allows to avoid unnecessary load balancing operations, improving the algorithm performance specially for F values significantly different from 1.

In the proposed context, the three implementations can obtain similar elapsed time values: 269, 259 and 280 s respectively. But the third objective must be rejected due to its efficiency value, only 0.38 for this best elapsed time. For the first and second objectives, similar initiation efficiencies can be obtained, 0.60 and 0.72, therefore both objective functions could be used for this algorithm, being a little best the second objective performance for this experiment.

6 Conclusions

This paper proposes a general procedure to methodically obtain initiation mechanisms for load balancing algorithms. This methodology implies choosing a general objective for the load balancing algorithm. This objective is mathematically expressed with an objective function, which can be an optimization or a boundary one, depending on the available system state information. And finally, this function is directly translated into an initiation mechanism for the load balancing algorithm. In addition, a performance metric for this mechanism, the initiation efficiency, has been defined.

For illustration, three example objectives have been presented to derive their initiation mechanisms using the proposed methodology and to evaluate their performance with the defined metric. The presented experiments for these three objectives show the utility of the proposed procedure in implementing initiation policies and of the initiation efficiency in selecting the best alternative.

All these results highlight the fact that it is possible to find different tasks allocations with similar elapsed times values but very different values for the initiation efficiency. And in this situation, the implementation with the best initiation efficiency must be always selected because it implies a better resources utilization (less rejected load balancing operations) and a better degree of the algorithm objective achievement. And of course, it can be seen with the different examples that the best elapsed time does not necessary imply the best initiation efficiency for the algorithm, because for this performance metric the important issue is the degree of achievement for the algorithm objective and the resources utilization efficiency to obtain this degree, and not the elapsed time.

Furthermore, a general observation can be made based on all the performed experiments: the boundary objectives are the best solution for load balancing algorithms, their performance always improve the obtained with optimization objectives. The explanation for this behavior is that the optimization objectives always introduce more overhead in the initiation mechanism and are not scalable, while a good selected boundary objective can obtain better results without causing this overhead due to its simplicity. An example of this behavior is that the mechanism 3 achieves a worse elapsed time for the global application although it is its main objective, due to the utilization of an optimization function.

On the other hand, for these boundary objective functions the algorithm tolerance must be tuned, taking into account that too demanding requirements can have a negative influence on the system performance.

References

1. Rajkumar Buyya. *High Performance Cluster Computing: Architecture and Systems, Volume I*. 1999. Prentice-Hall.
2. Gregory Pfister. *In search of clusters: The Ongoing Battle in Lowly Parallel Computing*. 1998. Prentice-Hall.
3. Jerrell Watts; Mark Rieffel and Stephen Taylor. Dynamic management of heterogeneous resources. In *High Performance Computing: Grand Challenges in Computer Simulation*, pages 151–156, 1998.
4. Chengzhong Xu and Francis C. M. Lau. *Load Balancing in Parallel Computers: Theory and Practice*. 1997. Kluwer Academic Publishers.
5. Ashok Rajagolapan and Salim Hariri. An agent based dynamic load balancing system. In *Proceedings of the International Workshop on Autonomous Decentralized Systems*, pages 164–171, 2000.
6. Manish Arora; Sajal K. Das and Rupak Biswas. A de-dentralized scheduling and load balancing algorithm for heterogeneous grid environments. In *Proceedings of the International Conference on Parallel Processing Workshops*, 2002.
7. Ron Lavi and Ammon Barak. The home model and competitive algorithms for load balancing in a computing cluster. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 127–134, 2001.
8. L.M. Ni; C. Xu and T.B. Gendreau. A distributed drafting algorithm for load balancing. *IEEE Transactions on Software Engineering*, 11(10):1153–1161, 1985.
9. Raymond Chowkwanyun and Kai Hwang. Multicomputer load balancing for LISP execution. In *Parallel Processing for Supercomputers and Artificial Intelligence*, pages 325–366, 1989. McGraw-Hill.
10. Rami G. Melhem; Kirk R. Pruhs and Taieb F. Znati. Using spanning-trees for balancing dynamic load on a multiprocessor. In *Proceedings of the Sixth Distributed Memory Computing Conference*, pages 233–237, 1991.
11. M. Beltrán; A. Guzmán and J.L. Bosque. Dynamic tasks assignment for real heterogeneous clusters. *Parallel Processing and Applied Mathematics:5th International Conference, Lecture Notes in Computer Science*, 3019/2004:888–895. Springer Verlag.
12. Michael Mitzenmacher; Balaji Prabhakar and Devavrat Shah. Load balancing with memory. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.
13. M. Beltrán and J.L. Bosque. Estimating a workstation CPU assignment with the DYPAP monitor. In *Proceedings of the 3rd IEEE International Symposium on Parallel and Distributed Computing*, 2004.
14. Luis Pastor and Jose L. Bosque. Efficiency and scalability models for heterogeneous clusters. In *Proceedings of the 3rd IEEE International Conference on Cluster Computing,*, pages 427–434. IEEE Computer Society Press, 2001.