Noname manuscript No.
(will be inserted by the editor)

Analyzing Scalability of Parallel Systems with Unbalanced Workload

Jose L. Bosque · Oscar D. Robles · Pablo Toharia · Luis Pastor

the date of receipt and acceptance should be inserted later

Abstract This paper presents a new formulation of the isoefficiency function which can be applied to parallel systems executing balanced or unbalanced workloads. This new formulation allows analyzing the scalability of parallel systems under either balanced or unbalanced workloads. Finally, the validity of this new metric is evaluated using some synthetic benchmarks. The experimental results allow assessing the importance of considering the unbalanced workloads while analyzing the scalability of parallel systems.

Keywords Scalability analysis · Isoefficiency · workload imbalance

1 Introduction

Thanks to the multi-core architecture, the number of cores available in supercomputers has been dramatically increasing during the last years. This fact has turned scalability into a factor of growing importance in the design and implementation of parallel applications, being currently even more important than performance.

All scalability metrics are based on selecting a performance metric (speedup, efficiency, latency,...) and analyze its evolution as it increases the number of processors in the system [3, 1, 5, 4, 6]. If this metric can be kept constant while increasing both the number of processors and the workload, it can be said that the system is scalable. On the other hand, it is widely known that the imbalance in the workload assigned to the processors has a deep impact on

Jose L. Bosque
Dpto. de Electrónica y Computadores
Universidad de Cantabria. Tel.: +34-942-20-15-62
E-mail: joseluis.bosque@unican.es

Oscar D. Robles · Pablo Toharia · Luis Pastor
Universidad Rey Juan Carlos
E-mail: {oscardavid.robles,pablo.toharia,luis.pastor}@urjc.es

performance of parallel systems and therefore in any of the aforementioned metrics [2]. Therefore, if scalability is defined as a function of the problem size, as well as a performance metric, a key aspect to take into account is the effect of load imbalance on the system's scalability.

Up to the present moment all scalability models consider, either implicitly or explicitly, a workload which is perfectly balanced among all of the system nodes [3, 1, 5, 4, 6]. When talking about parallel applications, this is a non-realistic hypothesis due to several reasons: firstly, because it means that the workload can be considered continuous and infinitely divisible; but also because parallel systems have multiple sources of imbalance, since they can be non-dedicated systems or they can have a poor initial workload distribution, for example.

In order to illustrate the problem, let us consider an embarrassing parallel benchmark without communication overhead. Its isoefficiency function should be constant; that is $O(K)$, where $K \in \Re$, and the system should be perfectly scalable [3]. However, once the isoefficiency function has been experimentally measured, the results do not agree with this prediction. Fig. 1 shows the theoretical and real isoefficiency functions of this benchmark, that is, the evolution of the workload in front of the number of processors so as to keep the efficiency constant. The figure shows an isoefficiency function which is linear with respect to the number of processors. Which is the source of this difference between theoretical predictions and real measured results? The answer is simple: the application has a workload showing constant imbalance for all configurations and system sizes evaluated.

Hence, it can be seen the importance of imbalance to obtain an accurate scalability model, since not taking imbalance into account yields poor predictions. This paper presents a new expression for the isoefficiency model that includes a model for unbalanced workload, i. e. a more general isoefficiency function that can be applied to parallel systems with or without load balancing, taking into account theoretical properties of the systems. To the author's knowledge, this is the first work that highlights this problem and suggests a suitable solution. Although this result is eminently theoretical, it has a deep impact in the design and implementation of parallel applications.

Using the new isoefficiency function, a number of theoretical examples are considered, studying different aspects of the scalability of parallel systems which include the communication overhead as well as the overhead originated by workload imbalance. Finally, an experimental validation has been carried out in order to verify the validity and correctness of the proposed model.

2 Scalability of unbalanced parallel systems

2.1 Imbalance workload model

Let us consider a parallel system as a set of m interconnected nodes, $N = n_1, \ldots, n_m$. The workload to be executed can be characterized by a set of

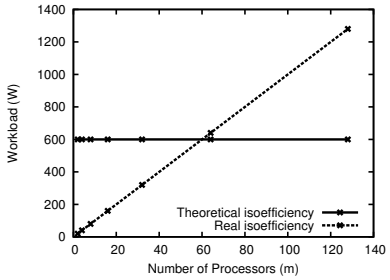


Fig. 1 Theoretical and real Isoefficiency function: unbalanced parallel application

basic operations which can be executed in parallel; it will be assumed that the workload size is W and that it can be decomposed into a set of computational units. The computational power of each node (p) can be defined as the number of basic operations the system is capable of execute per time unit.

If the workload is continuous and divisible *ad infinitum*, the system can achieve a perfect load balancing by assigning the same workload to every node: $w_i = \frac{W}{m}$. Therefore, assuming that there are not any communication overheads, the execution time of all of the nodes will remain the same, given by the expression: $T_{CPU} = \frac{W}{p \cdot m}$.

However, in real applications the workload can not be divided continuously, due to the intrinsic grain size of the problem. This forces to assign an integer number of jobs to each node, leading to an unbalanced load distribution, where the computation time is not equal for every node. In applications with coarse grain size, these differences are significantly larger.

In consequence, in unbalanced systems there will be some nodes executing less workload while others process more than the optimal one. Let's define Δw_i as the difference between the optimal and the current workload of node i. Note that this value can be either positive or negative, if that node has more or less workload than the optimal. Taking into account this fact, the previous expression can be rewritten as follows:

$$w_i = \frac{W}{m} + \Delta w_i \tag{1}$$

Hence, the execution time of a single node, assuming no communication overhead (i. e., only CPU time), is given by the following expression:

$$T_{CPU_i} = \frac{\frac{W}{m} + \Delta w_i}{p} \Rightarrow T_{CPU_i} = \frac{W}{m \cdot p} + \frac{\Delta w_i}{p} \tag{2}$$

Let T_{qi} represent the time needed by processor n_i for computing the additional workload ($T_{qi} = \frac{\Delta w_i}{p}$). Also, the maximum deviation between ideal and real execution times because of the additional workload will be $T_q = \max_{i=1}^{m}\{\frac{\Delta w_i}{p}\}$. The processor achieving this maximum will be the last one to finish its computation, assuming no communication overhead.

The overhead introduced by this fact, as it will be shown later on, can be a constant value, but it can also depend both on the total workload W and on the number of processors m, so it is redefined as $T_q(W, m)$. In the first case, the imbalance introduced will depend on the total size of the problem to be solved, i. e., increasing the size of the problem will lead to an increase or decrease of the imbalance proportional to W. Let us also consider the communication overhead $T_o(W, m)$. Then, the response time can be expressed as:

$$T_R = \frac{W}{p \cdot m} + T_q(W, m) + T_o(W, m) \tag{3}$$

It is important to emphasize that while T_o increases when the number of processors does (or, at least, it remains constant), T_q might decrease when m increases. This is possible because of the way both grain size and number of processors affect imbalance, and in consequence, T_q.

2.2 Isoefficiency function

As explained above, the response time of a parallel system is: $T_R = \frac{W}{p \cdot m} + T_q(W, m) + T_o(W, m)$. Thus, based on the same parameters, the sequential time needed to solve the same problem with the same input size would be: $T_S = \frac{W}{p}$, which is equivalent to solving the problem by executing the whole workload in a single processor with computational power p. Therefore, this concept can be applied to the efficiency expression:

$$E = \frac{T_S}{T_R \cdot m} = \frac{\frac{W}{p}}{(\frac{W}{p \cdot m} + T_o + T_q) \cdot m} = \frac{\frac{W}{p}}{\frac{W}{p} + m \cdot (T_o + T_q)} = \frac{1}{1 + \frac{m \cdot p \cdot (T_o + T_q)}{W}}$$

Hence, for scalable parallel systems, the efficiency can be kept at a desired value if the ratio $\frac{m \cdot p \cdot (T_q + T_o)}{W}$ in the expression above can be kept at a constant value. In order to keep a specific efficiency figure, the following expression states how large the workload W has to be:

$$\frac{m \cdot p \cdot (T_q + T_o)}{W} = \frac{1 - E}{E} \Rightarrow W = \frac{E}{1 - E} \cdot m \cdot p \cdot (T_q + T_o)$$

Let $K = p \cdot \frac{E}{1-E}$ be a constant depending on the efficiency. Then the isoefficiency function with load imbalance support can be written as:

$$W = K \cdot m \cdot (T_q(W, m) + T_o(W, m)) \tag{4}$$

This means the effect of the workload imbalance can be modeled as an additional overhead of the system, in the same way as communication time can be dealt with. In this case the factor T_q is the additional time that the processor that gets the largest workload chunk needs in order to process it in the CPU, i. e. without taking into consideration communication times.

Moreover, another remarkable fact that can be pointed out from this expression is that in every parallel system, even those which present no communication overheads, the presence of an unbalanced workload means that the system can not be perfectly scaled. Besides, it has an important specific behavior: as overheads always increase with the number of nodes, the imbalance can change the other way around, since an unbalanced system might become more balanced when it scales up.

3 Influence of the unbalanced workload in the scalability

Equation 4 means that the imbalance can vary when the system size m, the problem size W or both of them change, since the overheads T_o and T_q will be affected. Then, a system would be perfectly scalable if there is not any imbalance nor communication overhead, that is $T_q = T_o = 0$. This section presents some interesting examples of the scalability of parallel systems around the variation of T_q. In all cases the starting point is a parallel system defined by $S(m, W)$ and a scaled system $S'(m', W')$, with $m' > m$, and assuming its workload is always distributed proportionally to the number of processors, with an imbalance T_q. Firstly, some cases in which there is not any communication overhead (i. e. $T_o = 0$) will be considered, in order to isolate the effect of imbalance on the theoretical studies. Then, there will be under study some cases where it is considered the combined effect of the two parameters, T_o and T_q.

If each system's workload is distributed among its processors being $T_q = c$, i. e. there is a constant imbalance in both S and S', and it is independent of m and W, then the system is scalable if and only if there is an increase of workload proportional to the number of processors in the system, that is, W is $O(m)$. This situation can happen whenever the problem size is relatively coarse, and the number of available work-packages is not a multiple of the number of processors.

Let us define $T_q = c \in \Re$, constant independent of m and W. In this case the isoefficiency function is:

$$W = K \cdot m \cdot T_q \Rightarrow W = K' \cdot m, \quad \text{where} \quad K' = K \cdot c \tag{5}$$

Also, if the imbalance depends on the number of processors in a linear way, that is $T_q = c_1 \cdot m + c_2$, with $c_1, c_2 \in \Re$, and there is not any additional overhead, the system will be scalable if and only if the problem size grows as a function of $O(m^2)$. The proof is straightforward by replacing T_q in Equation 4 for the new expression.

If the imbalance is inversely proportional to the number of processors, that is $T_q = \frac{c_1}{m}$, with $c_1 \in \Re$, then the system will be perfectly scalable since the isoefficiency function remains constant independently of the number of processors, that is W is $O(K)$. It can be noticed that, when $W > m$, this situation brakes the lower bound established by Grama et al. [3] for the isoefficiency function.

Given $T_q = c_1 \cdot W$, with $c_1 \in \Re$, i. e. the imbalance is linearly dependent of the size of the problem, then the system is non scalable, since an increase in the workload size produces also an increase in the imbalance. In consequence, the efficiency can not remain constant.

If the imbalance is inversely proportional to the size of the problem, that is $T_q = \frac{c_1}{W}$, with $c_1 \in \Re$, it is straightforward to prove that the system will be scalable if an only if the size of the problem grows as a function of $O(\sqrt{m})$. Once again, it brakes the lower bound established by [3] for perfectly balanced systems, since the growth obtained here is under the linear one.

Next, it will be analyzed how the two different factors, communication overhead (T_o) and load imbalance (T_q) affect the overall scalability of the parallel system. For example, if both the imbalance and the overhead are assumed to be constant and independent of m and W, i.e., $T_q = c_1$ and $T_o = c_2$, being $c_1, c_2 \in \Re$, the system is scalable if and only if the size of the problem W grows proportionally to the number of processors, that is, W is $O(m)$. This can be easily demonstrated using the isoefficiency function previously presented in Section 2.2:

$$W = K \cdot m \cdot (T_q + T_o) \Rightarrow W = K \cdot m \cdot (c_1 + c_2) \Rightarrow W = K' \cdot m$$

where

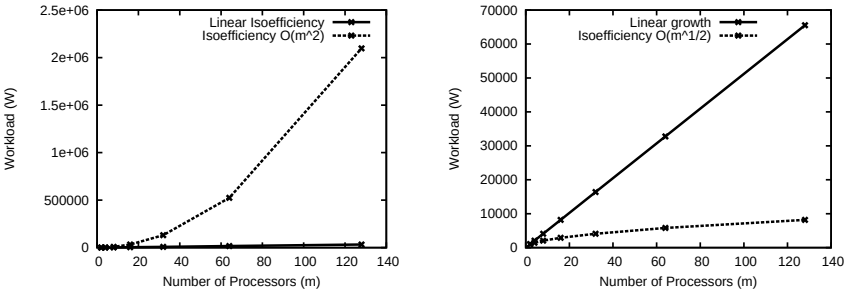
$$K' = K \cdot (c_1 + c_2)$$

Thus, if the system has a communication overhead and the imbalance remains constant with size, then the imbalance has no effect on the system's scalability. It only has impact on the maximum scalability the system can achieve but not on the evolution of the efficiency when increasing the system size. Therefore, in this case W has to grow linearly with the number of nodes, which is the lower bound posed by [3].

4 Model evaluation

A number of experiments was carried out in order to test empirically the validity of the proposed model. The main goals behind the tests were: (1) To verify the hypothesis posed in this paper about the influence of an unbalanced workload in the scalability of parallel systems from an empirical point of view. (2) To validate the scalability model proposed in Section 2 in situations in which there is an unbalanced workload. (3) To verify the correctness of the model stated in Section 3, by showing how a simple application with communication overheads behaves when the proportion of workload imbalance changes.

Altamira, the cluster from the Universidad de Cantabria used in these tests, is composed of 18 eServer BladeCenter, with 256 JS20 nodes (512 processors) linked by a Myrinet network with 1 Gbps of bandwidth. A number of tests have been performed changing the system and workload sizes. For every system's size a curve showing the evolution of the efficiency for different workloads has been obtained.



(a) *proportional_m* benchmark. (b) *inverse_w* benchmark.

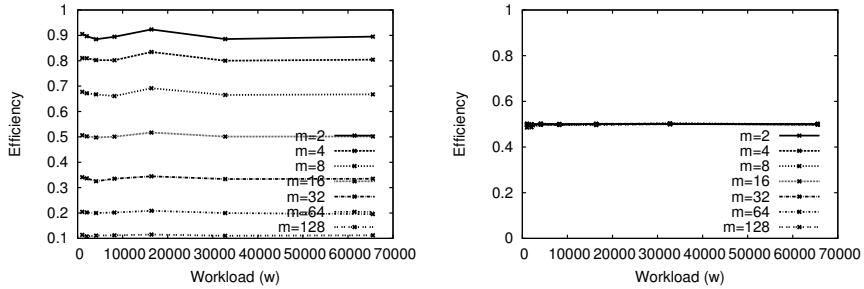
Fig. 2 Isoefficiency function for *proportional_m* and *inverse_w* benchmarks

The starting point is a basic benchmark in which each node works on local data and obtains its own solution, which means that the communication overhead is null and therefore its efficiency is constant, with independence of the workload and the system's size. Then, this benchmark should be fully scalable since its isoefficiency function is $\mathrm{O}(K)$. This statement is true if the workload is perfectly balanced, but the isoefficiency function changes when different levels of imbalance are introduced in the system, as the experimental results show. All the constants used on the experiments depend on the nodes' computational power and on the problem's nature. They have been measured in the sequential implementation of the benchmarks. Four different workload distributions are considered:

- Imbalance proportional to the number of processors, i. e. $T_q = c \cdot m$, where $c \in \Re$. It will be labeled *proportional_m*.
- Imbalance inversely proportional to the number of processors, i. e. $T_q = \frac{c}{m}$, where $c \in \Re$. It will be labeled *inverse_m*.
- Imbalance proportional to the workload, i. e. $T_q = c \cdot W$, where $c \in \Re$. It will be labeled *proportional_w*.
- Imbalance inversely proportional to the workload, i. e. $T_q = \frac{c}{W}$, where $c \in \Re$. It will be labeled *inverse_w*.

Fig. 2 shows the isoefficiency obtained for the *proportional_m* and *inverse_w* tests. As it can be seen, in figure 2(a) the isoefficiency function follows a parabolic curve when in this case the problem size should grow quadratically (W^2); the lower curve shows the issoefficiency growing linearly with the number of nodes. On the other hand, figure 2(b) shows the isoefficiency function with an imbalance that changes inversely to the workload. In this case (*inverse_w* benchmark) it can be seen that the isoefficiency function obtained is $O(\sqrt{m})$, under a linear growth. Both results match the prediction made by the theoretical model in Section 3.

Additionally, figure 3 shows the variation of the efficiency in front of the workload for different system's sizes for the *proportional_w* and the *inverse_m* benchmarks. In the *proportional_w* case, in figure 3(a), it can be seen that



(a) Evolution of efficiency. (b) Isoefficiency function

Fig. 3 Isoefficiency variation vs. workload for different system's sizes for *proportional_w* and *inverse_m* benchmarks.

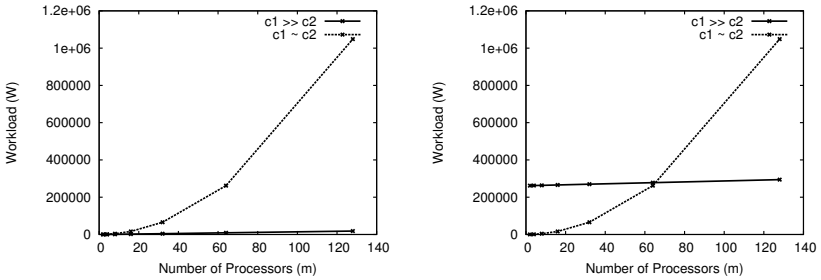
given a certain system's size, the efficiency remains almost constant when the workload grows. On the other hand, it can be seen that when the system's size increases it is not possible to keep the efficiency constant, whatever the workload is. Therefore, the isoefficiency function says the system is not scalable. On the contrary, in the *inverse_m* test the efficiency remains constant, around 0.50, with independence of the problem's size and the number of nodes. Therefore, the isoefficiency function in this situation is constant and the system is fully scalable, as predicted by the model.

Some important conclusions can be extracted from these cases. First, it is clear the great impact that an unbalanced workload has on the scalability of a parallel system. The results obtained allow concluding that the same parallel system can have quite different scalability behaviors, based on the workload distribution, being able to even turn into a non-scalable system, even if it does not have any communication overhead. Additionally, the four cases presented here allow to asses the correctness of the model posed in this paper. They have been analyzed from a theoretical point of view in Section 3, and the results obtained match perfectly the theoretical predictions.

Finally, this section presents some experimental results obtained from a benchmark that introduces both imbalance and communication overheads. The communication overhead is proportional to the number of processors. In this case, the classic isoefficiency function would be $O(m)$, not taking into account the workload imbalance. In this paper two different imbalance scenarios are presented as examples: constant imbalance, i.e, $T_q = c_1$ and imbalance inversely proportional to the number of processors, i.e., $T_q = \frac{c_1}{m}$.

In both cases, the communication overhead is proportional to the number of processors, being $T_o = c_2 \cdot m$. Therefore, taking into account both overhead sources, the scalability model presented in this paper predicts an isoefficiency function $O(m^2)$. Fig. 4 shows the collected experimental results showing the isoefficiency function achieved for both cases.

Fig. 4(a) shows two results achieved for the isoefficiency function regarding the existing relationship between the values of T_o and T_q. These results show



(a) Constant imbalance ($T_q = c_1$)

(b) Imbalance inversely proportional to the number of processors ($T_q = \frac{c_1}{m}$)

Fig. 4 Isoefficiency: benchmark with both communication and imbalance overheads.

the influence of both c_1 and c_2 constants in the scalability properties. If these values are similar (in the same order of magnitude) both communication and imbalance overheads have similar weight in the isoefficiency function. Therefore, it can be seen that the isoefficiency function grows quadratically with the number of processors, i.e., it is $O(m^2)$ as the presented model has predicted. Nevertheless, if $c_1 \gg c_2$ the influence of the communication overhead is almost negligible and has no impact on the isoefficiency function. In this case the isoefficiency function is linear, although the slope is slightly over 1, that would be the ideal scalability. This behavior is due to the relationship between both overheads when $T_q \gg T_o$, which makes the linear growth of T_o negligible compared to T_q, being the value of T_q the one that actually controls the efficiency behavior, and thus the isoefficiency. A similar effect can be observed in Fig. 4(b) for the case where $T_q = \frac{c_1}{m}$. Again if $c_1 \gg c_2$ the obtained isoefficiency remains constant. If both c_1 and c_2 are similar the isoefficiency function is quadratic, as predicted by the theoretical model presented here.

These results point out the importance of taking into account the workload imbalance when estimating the scalability of a parallel system. Additionally, it remarks the importance of the constant parameters. In general, when complexity studies are carried out, these parameters are assumed to have a small influence. Nevertheless, in these cases two different effects have to be added up and it is important not to forget the weight of each of them in the final efficiency value. This weight is determined by the complexity of the expression but also by the constant values. Not taking into account this values might lead to an unexpected system behavior.

5 Conclusions and Future Work

Since long ago there is evidence that unbalanced workloads are one of the aspects that have the biggest impact on the performance of parallel applications. This paper gives the proof for the very first time, as far as the authors know, that this statement can also be applied to scalability. Thus, the main contribution of this paper is that the evaluation of the scalability of a parallel system without considering the workload imbalance leads to potentially erro-

neous predictions. Although many authors have proposed scalability models before, none of them considers load imbalance.

This paper proposes a simple mathematical model for node imbalance in parallel systems. The model allows considering unbalanced workloads as another overhead in the system, similar to the communication overhead. This way, it is quite simple to introduce this factor in the isoefficiency function, in order to successfully predict the scalability of unbalanced parallel systems.

The new isoefficiency function proposed here makes it possible to perform a deep analysis of the influence of imbalance on the scalability of parallel systems, as well as its relationship with the communication overhead. This way, a number of theoretical analysis have been presented with one remarkable result: if the variation of imbalance is inversely proportional to the workload or to the number of processors, the system's scalability can be under linear. This result brakes the lower bound established by Grama et al. [3].

Both the model and its application to scalability studies have been experimentally validated using some synthetic benchmarks. In all of the experiments carried out the correlation between theoretical predictions and empirical results is excellent, and therefore, it can be stated that the model is quite accurate. Another important conclusion from the experiments is the importance of the relative value of both communication and imbalance overheads. Thus, since the isoefficiency function is a complexity analysis, it only collects the function's tendency as the number of processors grows. This is valid if all sources of imbalance are homogeneous and therefore they all present similar constants. But the results achieved show that if one of the overheads is much bigger than the others then the first one can clearly control the systems' behavior.

Finally, the next step is the use of this model with real applications in order to obtain a methodology that allow modeling unbalanced workloads. Also, an extension of this model to heterogeneous computing systems in which the nodes have different capabilities has to be done.

Acknowledgements This work has been partially supported by the Spanish Ministry of Education and Science (grants TIN2010-21289, TIN2010-21291-C02-02, Consolider CSD2007-00050 and Cajal Blue Brain project)

References

1. J. Chen and V. Taylor. Mesh partitioning for distributed systems. *Proceedings of Seventh IEEE International Symposium on High Performance Distributed Computing*, July 1998.
2. Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar. *Introduction to Parallel Computing (Second Edition)*. PEARSON - Addison-Wesley, Redwood City, CA, 2003.
3. Ananth Y. Grama, Anshul Gupta, and Vipin Kumar. Isoefficiency: measuring the scalability of parallel algorithms and architectures. *IEEE parallel and distributed technology: systems and applications*, 1(3):12–21, August 1993.
4. P. Jogalekar and M. Woodside. Evaluating the scalability of distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 11(6):589–603, June 2000.
5. Luis Pastor and Jose L. Bosque. Efficiency and scalability models for heterogeneous clusters. In *Third IEEE International Conference on Cluster Computing,*, pages 427–434, Los Angeles, California, Octubre 2001. IEEE Computer Society Press.
6. Xian-He Sun and D. T. Rover. Scalability of parallel algorithm-machine combinations. *IEEE Transactions on Parallel and Distributed Systems*, 5(6):599–613, June 1994.