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

Evaluating scalability in heterogeneous systems

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

Abstract This paper presents a new expression for an isoefficiency function which can be applied both to homogeneous and heterogeneous systems. Using this new function, called H-isoefficiency, it is now possible to analyze the scalability of heterogeneous clusters. In order to show how this new metric can be used, a theoretical *a priori* analysis of the scalability of a Gauss Elimination algorithm is presented, together with a model evaluation which demonstrates the correlation between the theoretical analysis and the experimental results.

Keywords Heterogeneous Computing · Scalability Analisys · Isoefficiency

1 Introduction and related work

A quick look to the top500 list now, in the age of exascale computing, reveals 6 machines with more than 100,000 processors and 3 of them with more than 200,000. The availability of multiple cores is behind this trend, and from this perspective, aspects such as *performance* loose importance, while *scalability* emerges as one of the key concepts in parallel computing.

The performance of parallel programs must be evaluated together with the computer system on which they run. Otherwise, an algorithm that solves a problem efficiently on a specific architecture and number of processors may perform poorly if the architecture or the number of processors change [3]. For example, common speedup graphs teach us that this index does not grow linearly with the number of processors, but tend to saturate. Nevertheless, a higher speedup can be obtained as the problem size increases while the algorithm runs over the same number of processors [4]. Joining both effects, a system is considered to be scalable if its performance measures remain constant whenever the number of processors is increased by selecting the appropriate

Jose L. Bosque
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

problem size. The system's degree of scalability is given then by the ratio *problem growth to system growth* needed to keep those performance measures constant. It can be said that scalability has been a desired capability that means not just the ability to operate a system, but to operate it efficiently and with an adequate quality of service over the available range of configurations [6].

The study of scalability within homogeneous parallel systems does not count with a unified and generally accepted metric that characterizes the system's behavior [5,8,11]. Among the most representative options, *latency* can be mentioned [13]. Another relevant method is Sun and Rover's *isospeed* [12], extended by Chen and Wu [2] for heterogeneous systems. Its main drawback is that, being the isospeed an *a posteriori* measure, the algorithm has to be implemented first.

Among all of the proposed approaches, the isoefficiency concept [9,3] is the most widely accepted. In this case, the degree of scalability is given by the *isoefficiency function*, that expresses the relationship *problem size/number of processors* needed to keep the efficiency constant. The smaller this ratio, the more scalable a parallel system is. Pastor and Bosque [10] proposed an extension to heterogeneous systems, although their approach lacks generality, since the functions they defined for analyzing the overhead time were linear with respect to the problem size for every case. Kalinov [7] has also extended the isoefficiency model to heterogeneous systems. In his work, the computational power of the slowest processor; the computational power of the fastest processor and the average computational power of the system have to remain unchanged. These are indeed three tight restrictions, difficult to satisfy when a system is upgraded with new nodes, conditioning therefore the range of possible upgrades. In consequence, its definition lacks generality too.

This paper presents a new expression for the isoefficiency function, called *heterogeneous isoefficiency*, which is a more general definition that improves the aforementioned extensions. This new proposal has been applied to some problems, in order to check its validity. The results achieved have always been quite close to those predicted by the model.

The rest of the paper is organized as follows: Section 2 defines the isoefficiency function for heterogeneous systems. Section 3 presents the application of this model to a practical problem. Section 4 shows some experimental results. Finally, Section 5 shows the conclusions and future work.

2 The Isoefficiency Function for Heterogeneous Systems

The isoefficiency function depends only on the number of processors, assuming that all of them have the same computational power [9,3]. This is arguable, since equal processors tend to behave differently, depending on issues such as use of the memory hierarchy, etc. But it certainly does not hold in heterogeneous systems, where the overall response time depends on the number of processors as much as on each processor's particular features.

In order to study heterogeneous systems, it is necessary to define first the notion of *computational power* for the whole system and for each node. In this paper, the computational power of a heterogeneous system (p_T) has been defined as the sum of the computational power of all of its processors (p_i) [10]: $p_T = \sum_{i=1}^{p} p_i$. Also, assuming that W is a parameter that characterizes the problem size, the computational power of each node has been computed using the expression $p_i = W/T_i$, where T_i is the response time for node i. It has to be noted that this definition yields relative estimates which depend on the nature of the problem to be considered.

The efficiency of a parallel (either homogeneous or heterogeneous) system, denoted by ε, can be redefined as the ratio between the ideal and the real response times achieved while processing a particular problem. While the real one is the time actually spent by the parallel system, the ideal one is the time needed by a single node that has the same computational power as the parallel system for solving the same problem [10]. This is reflected in the following equation:

$$\varepsilon = \frac{\textit{Optimal achievable time}}{\textit{Actual response time}} = \frac{W}{T_R \cdot p_T} \qquad (1)$$

Based on this definition an isoefficiency function for heterogeneous systems, or *H-isoefficiency*, can be defined now. In the heterogeneous case, the key parameter will be the total systems computational power, rather than the number of processors.

Given a heterogeneous parallel system $S(p, p_T, W)$ with p processors, a total computational power p_T and a total amount of work represented by W, and given also $S'(p', p'_T, W')$, a scaled system with $p'_T > p_T$, it can be said that S is a scalable system if, whenever the system is upgraded from S to S', it is possible to select a problem size W' such that the efficiencies of S and S' are kept constant.

Now the *H-isoefficiency* function can be computed, starting from the heterogeneous efficiency definition for S. For that, the response time can be decomposed on execution and overhead times: $T_R = T_{exe} + T_o$. Assuming that the workload is evenly distributed among all of the nodes (proportionally to each node's computational power), T_{exe} will be the same for every node, being given by $T_{exe} = \frac{W}{p_T}$. Then the response time will be given by the expression $T_R = \frac{W}{p_T} + T_o$. Hence, the H-isoefficiency function can be defined as:

$$\varepsilon = \frac{W}{T_R \cdot p_T} = \frac{W}{W + T_o \cdot p_T} = \frac{1}{1 + \frac{T_o \cdot p_T}{W}}$$

For scalable heterogeneous parallel systems, the efficiency can be kept constant if the ratio $\frac{T_o \cdot p_T}{W}$ is also kept constant by changing the problem size:

$$\frac{T_o \cdot p_T}{W} = \frac{1-\varepsilon}{\varepsilon} \Rightarrow W = \frac{\varepsilon}{1-\varepsilon} T_o \cdot p_T$$

Let $K = \frac{\varepsilon}{1-\varepsilon}$. Then the H-isoefficiency function can be written as:

$$W = K \cdot T_o(p) \cdot p_T(p) \qquad (2)$$

This is similar to the expression that gives the homogeneous isoefficiency, although instead of using a single t_c parameter, constant for the whole set of nodes, it includes a new p_T parameter which represents the total aggregated computational power. This expression reflects also the fact that the scalability of heterogeneous environments depends both on the number of nodes and the total computational power of the scaled system.

When scaling a system, the computational power of the new nodes has to be taken into account, in order to increase the problem size accordingly. For example, if a system is scaled up just by upgrading the system nodes, the total response time will decrease, and the total overhead (T_o) will be a larger percentage of the response time, resulting in a smaller efficiency. The problem size will have to be increased in order to keep the efficiency constant.

An important advantage of the proposed approach compared to Kalinov's is that the H-isoefficiency method can be applied whenever new nodes are added *(physical scalability)*, some nodes are upgraded *(power scalability)* or both [14]. On the other hand, Kalinov's method forces to maintain the average computational power and therefore a power scalability study can not be done. With the new approach, *a priori* studies can be carried out, for example, in order to find out whether it is better to execute a particular algorithm on a large number of less powerful processors or on a smaller number of more powerful processors, as shown below.

3 Scalability of the Gauss Elimination Algorithm

Two different implementations of a Gaussian Elimination algorithm for solving linear equation systems have been tested, in order to verify the validity of the proposed approach. Both implementations differ in the way the communication is carried out: the first one uses point-to-point communication, while the second one uses broadcasting. A theoretical analysis is presented here, while the experimental results are shown in Section 4.

3.1 Performance Analysis

The time spent in a single point-to-point communication over an uncontested interconnection network can be well approximated in terms of the *startup latency* (λ), *bandwidth* (β) and message size (m), as $T_M = \lambda + \frac{m}{\beta}$. A broadcast function to p processors requires $\lceil log(p) \rceil$ message-passing steps. Hence the time spent in broadcasting an m word message can be approximated by $T_B = (\lambda + \frac{m}{\beta}) \cdot log(p)$.

The sequential implementation of the Gauss Elimination method needs three nested loops, having a complexity $\Theta(n^3)$. In the parallel implementation, the workload is divided by assigning the rows in an interleaved way, in order to archive a balanced workload distribution. If the average time to perform a basic algorithm step within the whole parallel system is t_c, then the expected computational time of the parallel algorithm is proportional to $T_{CPU} = \left\lceil n^3/p \right\rceil t_c$. Additionally, n communication events are needed, with a 1 to p scheme each one, in order to send a modified row to each node.

Point-to-Point Communication. In this approach, a point-to point communication scheme is used. Each message contains a complete row, and $(p-1)$ messages are needed for each iteration, with an upperbound total communication cost[1] of $T_o^{pp} = n \cdot (p-1) \cdot (\lambda + n/\beta)$. The total response time in this case can be modeled by the expression $T_R = \left\lceil n^3/p \right\rceil t_c + n \cdot (p-1) \cdot \lambda + (p-1) \cdot n^2/\beta$.

Let's determine the isoefficiency function for homogeneous systems. The sequential algorithm has time complexity $\Theta(n^3)$. Each of the p processes executing the parallel algorithm spends $\Theta(n^2 \cdot p)$ time performing communications. Then the total communication overhead across p processors is $\Theta(n^2 \cdot p^2)$.

Therefore the isoefficiency relation is:

$$n^3 \geq K \cdot n^2 \cdot p^2 \Rightarrow n \geq K \cdot p^2$$

where K is a constant. With this isoefficiency function, we can say that the scalability of the point-to-point implementation is poor.

Broadcast Communication. In this case the point-to-point communication primitives are substituted by collective broadcasts. Therefore, the communication time is reduced to $(\lambda + \frac{n}{\beta}) \cdot \lceil log(p) \rceil$ for each iteration. As it is necessary to do n broadcast operations, the response time is: $T_R = \left\lceil n^3/p \right\rceil t_c + n \cdot \lceil log(p) \rceil \cdot (\lambda + n/\beta)$.

Using the same analysis as before, the isoefficiency function for the broadcast-based implementation can be obtained. Each of the p processes executing the parallel algorithm spends $\Theta(n^2 \cdot log(p))$ time performing communications. Then the total communication overhead across p processors is $\Theta(n^2 p \cdot log(p))$. Therefore the isoefficiency relation is:

$$n^3 \geq K(n^2 p \cdot log(p)) \Rightarrow n \geq K(p \cdot log(p))$$

where K is a constant. In this case the scalability is much better.

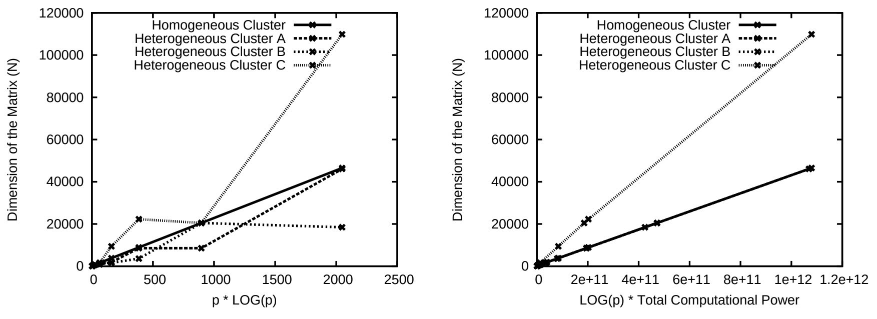
3.2 H-isoefficiency of the Gauss Elimination Algorithm

Now let's determine the H-isoefficiency function of the Gauss' method. As mentioned earlier, the workload is evenly distributed according to each node's computational power ($w_i = \frac{W}{p_T} \cdot p_i$). This way, all the nodes spend the same amount of time performing computations ($\frac{W}{p_T}$). Also, the overhead times in the homogeneous and the heterogeneous system are the same. Therefore it is easy to obtain the H-isoefficiency function for both implementations:

Point-to-Point Communication. Using the H-isoefficiency function given in the previous Section:

$$W = K \cdot p_T \cdot T_o \Rightarrow n^3 = K \cdot p_T \cdot (n \cdot (p-1) \cdot \lambda + (p-1) \cdot \frac{n^2}{\beta})$$

[1] Actually, the communication cost depends on n, which is decremented on each iteration. But this does not affect the expression given for the isoefficiency function.



(a) Classic Isoefficiency (b) H-isoefficiency

Fig. 1 Comparison of classic and H-isoefficiency figures in broadcast implementation

Analyzing each term of this expression separately, it can be seen that for large values of n, the scalability depends mostly on the term n^2. Therefore, the H-isoefficiency function is given by:

$$n^3 \geq K \cdot n^2 \cdot p_T \cdot p \Rightarrow n \geq K \cdot p_T \cdot p \Rightarrow \Theta(p_T p). \tag{3}$$

Broadcast Communication. Following the same analysis:

$$W = K \cdot p_T \cdot T_o \Rightarrow n^3 = K \cdot p_T \cdot (n \cdot \lceil log(p) \rceil \cdot (\lambda + \frac{n}{\beta}))$$

Therefore, in this case the H-isoefficiency function is:

$$n^3 \geq K(n^2 \cdot p_T \cdot log(p)) \Rightarrow n \geq K \cdot p_T \cdot log(p) \Rightarrow \Theta(p_T \cdot log(p)). \tag{4}$$

Again, the isoefficiency function is better for the broadcast-based approach.

4 Model Evaluation

A number of experiments was carried out in a heterogeneous cluster with up to 512 processors, interconnected with Myrinet. The cluster includes two different kinds of processors, with different computational power, labeled as Fast (NF) with $p_{NF} = 527000000$ and Slow (NS), with $p_{NS} = 207300000$. The rest of the cluster parameters are: ¡¡¡¡¡¡¡ .mine $\lambda = 5\mu s$ and $\beta = 1.5$ *Gbps.* ======= $\lambda = 5\mu s$ and $\beta = 1.5$ *Gbps.* ¿¿¿¿¿¿¿ .r3737

These experiments, together with others presented in [1], show the usefulness of the H-isoefficiency method for studying the scalability of heterogeneous parallel systems. The results presented in this section aim specifically at validating the H-isoefficiency functions presented in Section 3.

Figure 1 shows how much the problem size W has to grow in order to keep the efficiency constant when the parallel system is upgraded. W is in this case characterized by the matrix dimension n, and the system's size by $p_T \cdot log(p)$. Figures 1(a) and 1(b) show the results achieved using the traditional isoefficiency and the new H-isoefficiency approaches respectively, for four different cluster configurations: an homogeneous and three heterogeneous clusters, using different combinations of fast and slow processors and two efficiency values.

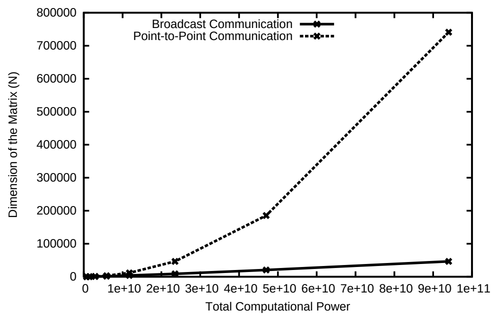


Fig. 2 H-isoefficiency:point-to-point and broadcast implementations.Heterogeneous cluster

Starting with Figure 1(b), this graphic presents some tests performed with different heterogeneous cluster configurations, for two different values of ε (which affects K and the lines slope). It can be seen that the problem size W (given by n) must grow linearly with $p_T \cdot log(p)$ independently of the cluster configuration and the desired ε. This result agrees with the predictions yielded by the theoretical model, proving the validity of the H-isoefficiency model for predicting *a priori* the scalability of an application both in homogeneous and heterogeneous clusters. This result agrees also with previous results [1].

The conclusions obtained after inspecting Figure 1(a), using the traditional isoefficiency, are quite different. This figure shows that n has to be increased linearly with the product $p \cdot log(p)$, as predicted by the isoefficiency function, but *only for the homogeneous cluster*. The results for heterogeneous clusters are erratic, not showing the linear trend they should. Clearly, the classical isoefficiency can not be applied to heterogeneous clusters.

Finally, Figure 2 shows the experimental results obtained regarding the H-isoefficiency function for both Point-to-Point and Broadcast implementations. It can be seen that the Point-to-Point communication approach is much less scalable than the Broadcast one, since the problem size has to grow quite faster with respect to the number of processors. It therefore confirms the theoretical analysis obtained in Section 3, supporting the claim that H-isoefficiency is an appropriate model for predicting *a priori* the scalability of different algorithms and implementations, saving efforts at the implementation stage.

5 Conclusions

This paper presents a new expression of the isoefficieny function, called H-isoefficiency, which can be applied to homogeneous and heterogeneous systems and which can be used for predicting algorithm scalability without needing the actual implementation in the selected architecture. Comparing this model with others, it presents several advantages, just like the isoefficiency model proposed by Kumar and Rao [9] does. Its most remarkable advantage is that, being an *a priori* method, it does not require the implementation of the algorithm in the selected architecture. Additionally, it deals with both power and physical scalability without imposing any restriction on the system's setup.

The experiments performed here and in [1] show that the proposed method yields results quite close to the values theoretically predicted. This shows that the H-isoefficiency function is an accurate model that allows performing scalability analysis both for homogeneous and heterogeneous systems. The results have also verified the strong impact that different configurations have on the scalability of a heterogeneous environment.

Future work will focus on the ability to deal with cases where balanced workload distributions among the different nodes can not be made.

Acknowledgements

This work has been partially funded by the the Spanish Ministry of Education and Science (grants TIN2010******, TIN2010-21291-C02-02 and Consolider CSD2007-00050), as well as by the HiPEAC European Network of Excellence.

References

1. Jose L. Bosque, Oscar D. Robles, Pablo Toharia, and Luis Pastor. H-isoefficiency: Scalability metric for heterogeneous systems. In *10th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2010*, pages 240–250, Almeria, Spain, June 2010. CMMSE - J. Vigo-Aguiar.
2. Yong Chen, Xian-He Sun, and Ming Wu. Algorithm-system scalability of heterogeneous computing. *Journal of Parallel and Distributed Computing*, 68(11):1403–1412, 2008.
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. John L. Gustafson. Reevaluating amdahl's law. *Communications of the ACM*, 31(5):532–533, May 1988. Sandia NL.
5. John L. Gustafson, Gary R. Montry, and Robert E. Benner. Development of Parallel Methods for a 1024-Processor Hypercube. *SIAM Journal on Scientific and Statistical Computing*, 9(4):609–638, 1988.
6. P. Jogalekar and M. Woodside. Evaluating the scalability of distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 11(6):589–603, June 2000.
7. A. Ya. Kalinov. Scalability of heterogeneous parallel systems. *Programming and Computer Software*, 32(1):1–7, 2006.
8. Alan H. Karp and Horace P. Platt. Measuring parallel processor performance. *Communications of the ACM*, 22(5):539–543, May 1990.
9. V. Kumar and V. N. Rao. Parallel depth-first search on multiprocessors: Part II: Analysis. *International Journal of Parallel Programming*, 16(6):501–519, December 1987.
10. 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.
11. Xian-He Sun and John L. Gustafson. Sizeup: a new parallel performance metric. In *Proceedings of the 1991 International Conference on Parallel Processing*, volume II, Software, pages II–298–II–299. CRC Press, August 1991.
12. 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.
13. Y. Yan, X. Zhang, and Q. Ma. Software support for multiprocessor latency measurement and evaluation. *IEEE transactions on Software Engineering*, 23(1):4–16, 1997.
14. X. Zhang and Y. Yan. Modelling and characterizing parallel computing performance on heterogeneous networks of workstations. *Proc. 7th IEEE Symp. on Parallel and Distributed Processing*, pages 25–35, 1995.