*Proceedings of the 10th International Conference*
*on Computational and Mathematical Methods*
*in Science and Engineering, CMMSE 2010*
*27–30 June 2010.*

# H-Isoefficiency: Scalability Metric for Heterogeneous Systems

**Jose Luis Bosque[1], Oscar D. Robles[2], Pablo Toharia[2] and Luis Pastor[2]**

[1] *Dpto. de Electrnica y Computadores, Universidad de Cantabria*

[2] *Dpto. de ATC y CCIA , Universidad Rey Juan Carlos*

emails: joseluis.bosque@unican.es, oscardavid.robles@urjc.es, pablo.toharia@urjc.es, luis.pastor@urjc.es

**Abstract**

Scalability is one of the most important features in exascale computing. Most of this systems are heterogeneous and therefore it becomes necessary to develop models and metrics that take into account this heterogeneity. This paper presents a new expression of the isoefficiency function called H-isoefficiency. This function can be applied for both homogeneous and heterogeneous systems and allows to analyze the scalability of a parallel system. Then, as an example, a theoretical *a priori* analysis of the scalability of Floyd's algorithm is presented. Finally a model evaluation which demonstrate the correlation between the theoretical analysis and the experimental results is showed.

*Key words: Heterogeneous Computing, Scalability Analisys, Isoefficiency*

## 1 Introduction

The performance of parallel programs must be evaluated together with the computer system on which they are run. Otherwise, an algorithm that solves a problem well using a fixed number of processors on a particular architecture may perform poorly if the number of processors changes [3]. Common speedup graphs teach us that the speedup of a system does not grow linearly with the number of processors but tend to saturate. On the other hand, a higher speedup can be obtained as the problem size increases on the same number of processors [4]. Then, a system is considered to be scalable if the performance measures remain constant whenever the number of processors is increased by selecting the appropriate problem size. The system's degree of scalability is given by the ratio problem growth to system growth needed to keep those 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].

But moreover, in the age of exascale computing, a 21st century attempt to push computing capabilities beyond the existing ones, a quick look to the top500 list reveals 6 machines with more than 100,000 processors (3 of them over 200,000). Since processors are affordable and quite powerful nowadays (currently up to 12 cores and growing), other aspects as performance loose importance while scalability emerges as one of the key concepts in parallel computing.

The study of scalability of homogeneous parallel systems has not come up with a unique and common way of evaluation, although the isoefficiency metric [3] is the most accepted and used. In it, the degree of scalability is given by the *isoefficiency function*, that expresses the dependence of the problem size on the number of processors needed to keep the efficiency constant. The smaller the problem size is, the lower is the isoefficiency value and therefore the higher the scalability of the parallel system is.

This paper presents a new expression of the isoefficiency function, called *heterogeneous isoefficiency*. It is a more general definition than the one presented in [10], since the functions they use to analyze the overhead time are always linear with respect to the problem size, so their definition is only valid for the examples they propose. It also improves other existing extensions, as the ones proposed by Kalinov [7] and Chen et al. [1], as it is explained in section 2. In order to prove the use of this model it has been applied to the Floyd algorithm, obtaining from the experiments results quite close to those predicted by the model.

The rest of the paper is organized as follows: Section 2 presents a brief overview of the related work. Section 3 presents the new definition of the isoefficiency function for heterogeneous systems. Section 4 presents the application of this model to a practical problem. Section 5 shows the experimental results achieved. Finally, conclusions and future work are summarized in Section 6.

## 2 Related Work

Despite its importance for parallel and distributed systems, there is no unique and commonly accepted metric for scalability evaluation. A number of techniques have been suggested throughout the years. They are typically based on the selection of a metric which is used for characterizing the behavior of an homogeneous system [5, 8, 12, 14, 18]. Among the most representative works, it can be mentioned the use of latency as a metric to do an experimental measurement and evaluation of the scalability of programs and architectures [15, 17]. The model is based on the average latency, a function of the problem size and the number of processors. It determines the average overhead time needed so that each processor finishes its assigned work. Scalability is then defined as a combination of the machine and the implementation of the algorithm.

Another model quite used is the one raised by Sun and Rover [13], who proposed an isospeed scalability metric to describe the scalability of an algorithm-machine combination in homogeneous environments. This model is based on reducing the response time by means of increasing the speed. The execution speed of an algorithm is defined as the amount of work needed to complete its execution divided by the response

time. Ideally, this measure should increase linearly with the size of the system. In general communication and synchronization overheads prevent such a behavior. Chen and Wu [1] extended this model to heterogeneous systems. Its main drawback is that isospeed is an *a posteriori* measure, so it demands to implement the model and obtain the measures empirically to be able to decide about the system's scalability.

The isoefficiency is the most widespread model [9, 3]. It defines scalability as the ability of a parallel system to keep the parallel efficiency constant when both system and problem sizes increase. Then, parallel efficiency is defined as the speedup over the number of processors. Speedup is defined in turn as the ratio sequential execution time to parallel execution time. Pastor and Bosque [10] proposed an extension to heterogeneous systems, although their approach lacks of generality, since the functions they define to analyze the overhead time are linear with respect to the problem size for every case.

Kalinov [7] has also extended the isoefficiency model to heterogeneous systems. In this work it is imposed that the system has to keep constant the computational power of the slowest processor, the computational power of the fastest processor and also the average computational power of the system. These are indeed three tight restrictions quite difficult to satisfy when a system is upgraded with new nodes, conditioning its real heterogeneity.

The model presented in this paper has several advantages over the isospeed model and the extension to the isoefficiency model presented by Kalinov, since as it is explained in the following, it is an *a priori* model that successfully predicts the scalability of a system and also deals with both power and physical scalability.

## 3 The Isoefficiency Function for Heterogeneous Clusters

The isoefficiency function depends only on the number of processors assuming all of them have the same computational power [9, 3]. This is not the case for heterogeneous systems where the performance of a single processor can affect the overall performance of the system. In fact, the response time will depend on the slowest node: $T_R = max_{i=1}^{P} T_i$ — being P the number of processors and $T_i$ the response time for node $i$.

In this way the computational power of a heterogeneous system ($P_T$) can be defined as the sum of the computational power of its processors ($P_i$) [10]:

$$P_T = \sum_{i=1}^{P} P_i \tag{1}$$

$$\tag{2}$$

Assuming $W$ is the size of the problem, in this work the computational power of each node has been computed using the following expression:

$$P_i = \frac{W}{T_i} \tag{3}$$

### 3.1 Heterogeneous Efficiency

As previously mentioned the computational power of a heterogeneous system does not only depend on the number of processors but also depends on each processor's computational power. In order to improve the computational power of this type of systems both the number or the power of some processors have to be increased. The latter is called *physical scalability* while the former is referred as *power scalability* [16].

The efficiency of a parallel (either homogeneous or heterogeneous) system, denoted by $\varepsilon$, can be defined as the ratio between the ideal response time in a single node with the same computational power and the real response time achieved [10]. The best response time is achieved when the workload is evenly distributed and no overhead time is introduced. This is reflected in the following equation:

$$\varepsilon = \frac{Optimal\ achievable\ time}{Actual\ response\ time} = \frac{W}{T_R \cdot P_T} \tag{4}$$

For homogeneous systems, it is easy to demonstrate that $\varepsilon$ becomes the traditional efficiency.

### 3.2 H-Isoefficiency

$T_i$ can be decomposed into computation and overhead times: $T_i = t_c^i + t_o^i$. The isoefficiency function assumes that $t_c$, is constant for all of the processors and therefore it does not affect the scalability of the system. However, this is not true for heterogeneous systems, where not only the number but also the performance of nodes have a high impact on the scalability of the system.

Based on the definition of the efficiency given in the previous section, an isoefficiency function for heterogeneous systems, H-isoefficiency, can be defined. In the heterogeneous case the key parameter will be the total computational power of the system, instead of 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 *heterogeneous isoefficiency function, H-isoefficiency* for heterogeneous parallel systems can be computed, starting from the heterogeneous efficiency definition for S. Furthermore we define the response time of the parallel algorithm as the sum of the execution time plus the overhead time: $T_R = T_{exe} + T_o$. Assuming that the workload is evenly distributed among the nodes proportionally to each node's computational power, $T_{exe}$ will be the same for all of the nodes, and it can be determined as $T_{exe} = \frac{W}{P_T}$. Then the response time will be given by the following expression: $T_R = \frac{W}{P_T} + T_o$. Hence, the H-Isoefficiency function is 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 maintained at a desired value if the ratio $\frac{T_o \cdot P_T}{W}$ in the expression of the efficiency is maintained at a constant value. To maintain a certain efficiency we can do:

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

Let $K = \frac{\varepsilon}{1-\varepsilon}$ be a constant depending on the efficiency. Then the H-isoefficiency function can be write as:

$$W = K \cdot T_o(P) \cdot P_T(P) \tag{5}$$

From this expression can be pointed out that the scalability of heterogeneous environments depends on both, the number of nodes and the total computational power of the scaled system.

This expression is similar to the one proposed for homogeneous isoefficiency. The main difference between both is that instead of using a single $t_c$ parameter, which remains constant for the whole set of nodes of the system and which is included in the $K$ parameter, a new $P_T$ parameter is introduced to represent the total aggregated computational power of the system.

Therefore, when scaling a system, the computational power of new nodes has to be taken into account in order to increase the size of the problem in a proportional way. This problem can be seen from a qualitative point of view: if a system is scaled using nodes more powerful than the system's, the total response time would be lower than the time achieved if the nodes had the same computational power. In this way the total overhead ($T_o$) would be a bigger percentage of the response time and the efficiency would be decreased. This fact makes necessary to increase the size of the problem in order to achieve the same efficiency as is represented in Eq. 5.

A big advantage of the proposed approach compared to Kalinov's generalization of the isoefficiency function [7] is that our approach allows to study the behavior of the system both when scaling by the number of processors *(physical scalability)* and when scaling increasing the power of some of them *(power scalability)* Kalinov's work forces to maintain the average computational power and therefore a power scalability study can not be done. With our approach *a priori* studies can be carried out in order to analyze if a specific algorithm is more suitable for its execution using a large number of less powerful processors or a lower number of more powerful ones. This is demonstrated in the experimental results section.

## 4 Scalability of the Floyd Algorithm

Once the H-isoefficiency has been defined it becomes necessary to experimentally validate and verify it. In this way, as an example, Floyd's algorithm has been chosen. This algorithm solves the all-pairs shortest-path problem. In this Section the performance of a parallel implementation of this algorithm is analyzed in order to obtain both the isoefficiency and H-isoefficiency. In Section 5 experimental results are presented.

### 4.1 Performance Analysis

Let's assume a model for communication cost in parallel programs. The time spent in a single point-to-point communication over an uncontested interconnection network can be well approximated in terms of *startup latency* ($\lambda$) and *bandwidth* ($\beta$). Then the time to communicate a $m$ word message can be approximated by: $T_M = \lambda \cdot \frac{m}{\beta}$.

A broadcast function to $p$ processors requires $\lceil logp \rceil$ message-passing steps. Hence the time spent in broadcasting a $m$ word message can be approximated by $T_B = \lambda \cdot \frac{m}{\beta} \cdot logp$.

The sequential implementation of the Floyd algorithm is composed by three nested loops, from 0 to $n-1$, where $n$ is the dimension of the adjacency matrix. Therefore the complexity of the sequential Floyd algorithm is $\Theta(n^3)$.

With respect to the parallel algorithm the innermost loop has complexity $\Theta(n)$. Given a row-wise block-striped decomposition of the adjacency matrix, each process executes at most $\lceil \frac{n}{p} \rceil$ iterations of the middle loop. Hence the complexity of the inner two loops is $\Theta(\frac{n^2}{p})$. Immediately before the middle loop is the broadcast step. Passing a single message of length $n$ from one processor to another has time complexity $\Theta(n)$. Since broadcasting to $p$ processors requires $\lceil logp \rceil$ message-passing steps, the overall time complexity of broadcasting each iterations is $\Theta(nlogp)$. The outermost loop executes $n$ times. Hence the overall time complexity of the parallel algorithm is:

$$\Theta(n(nlogp + \frac{n^2}{p})) = \Theta(\frac{n^3}{p} + n^2 logp)$$

Now let's come up with a prediction of the response time of the parallel algorithm. The parallel Floyd program requires $n$ broadcasts, each of them with $\lceil logp \rceil$ steps. Each step involves passing messages that are $4n$ bytes long. Hence the expected communication time of the parallel program is:

$$n\lceil logp \rceil(\lambda + \frac{4n}{\beta})$$

If $t_c$ is the average time needed to update a single cell (a basic algorithm operation), then the expected computational time of the parallel algorithm is:

$$n^2 \left\lceil \frac{n}{p} \right\rceil t_c$$

However it is possible to overlap communication and computation operations. The computation time per iteration exceeds the time needed to pass messages. For this reason after the first iteration each process spends the same amount of time waiting for or setting up messages: $\lceil logp \rceil \lambda$. If $\lceil logp \rceil \frac{4n}{\beta} < \lceil \frac{n}{p} \rceil n t_c$, the message transmission time after the first iteration is completely overlapped by the computational time and should not be counted toward the total execution time. Hence a better expression for the execution time of the parallel program is:

$$T_R = n^2 \left\lceil \frac{n}{p} \right\rceil t_c + n\lceil logp \rceil \lambda + \lceil logp \rceil \frac{4n}{\beta}) \tag{6}$$

### 4.2 Isoefficiency Function

Let's determine the isoefficiency function for the parallel implementation of the Floyd's algorithm. The sequential algorithm has time complexity $\Theta(n^3)$. Each of the $p$ processes executing the parallel algorithm spends $\Theta(nlogp)$ time performing communications. Therefore the isoefficiency relation is:

$$n^3 \geq K(nplogp) \Rightarrow n \geq K(plogp)^{\frac{3}{2}}$$

where K is a constant.

### 4.3 H-isoefficiency of the Floyd Algorithm

Now let's determine the H-isoefficiency function of the Floyd's algorithm. For obtaining the execution time in a heterogeneous environment, we have to take into account that the workload is evenly distributed according to each node's computational power. Then each node has a computational workload given by $w_i = \frac{W}{P_T} \cdot P_i$.

In such a way, the middle loop executes $w_i$ times per each node, but all the nodes spend the same amount of time $\frac{W}{P_T}$. A performance analysis similar to the one presented in Section 4.1 gives the following response time and complexity expressions:

$$T_R^H = \frac{n^3}{P_T} + n \cdot \lceil logp \rceil \cdot (\lambda + \frac{4n}{\beta}) \Rightarrow T_o = n \cdot \lceil logp \rceil \cdot (\lambda + \frac{4n}{\beta}) \quad (7)$$

$$\Theta(\frac{n^3}{P_T} + n^2 logp) \quad (8)$$

Then the H-isoefficiency function is:

$$W = KP_T T_o \Rightarrow n^3 = KP_T(n\lambda\lceil logp \rceil + \frac{4n}{\beta}\lceil logp \rceil) = KP_T n\lambda\lceil logp \rceil + KP_T\frac{4n}{\beta}\lceil logp \rceil \quad (9)$$

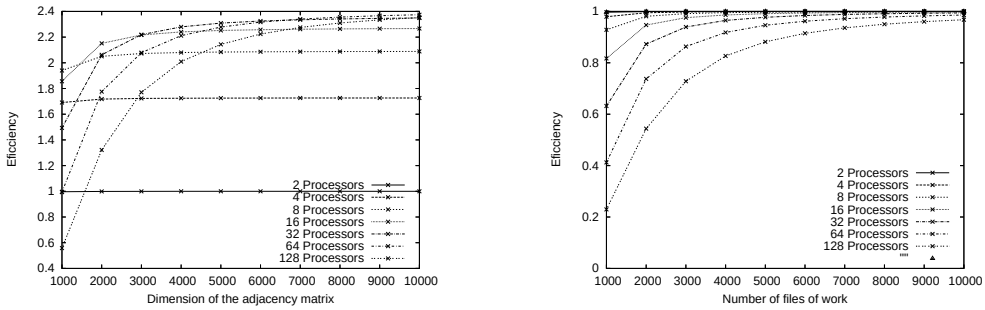
Analyzing each term independently we reach the same expression for the H-isoefficiency:

$$\Theta((P_T logp)^{\frac{3}{2}}). \quad (10)$$

## 5 Model Evaluation

A practical experiment was set up in order to test the analytical results presented in the previous sections and to show how the heterogeneous isoefficiency model could be applied. The tests were performed on a heterogeneous HP Cluster with up to 142 processors, interconnected with Gigabit Ethernet network. The cluster is composed of the following resources:

- 20 HP Proliant DL 145 with two 1.8 GHz Dual Core AMD Opteron 265 processors, labeled as Node Slow (NS) in this paper.




(a) Efficiency in a heterogeneous cluster

(b) Heterogeneous efficiency in a heterogeneous cluster.

Figure 1: Comparison of classic and proposed efficiency figures in a heterogeneous cluster

- 25 HP Proliant DL 160 with two 3.0 GHz Quad Core INTEL Xeon 5472 processors, labeled ad Node Fast (NF).

In this cluster the parameters of the model have been measured and they are listed in Table 1. The application was developed using GNU tools and the MPICH 1.2.7 library [2, 11].

| Label | $P_i$ | $tc_i$ | $\lambda$ | $\beta$ |
|---|---|---|---|---|
| Node Fast (NF) | 83.988.126 | 0,0000000119 | 25 | 2.000.000.000 |
| Node Slow (NS) | 34230899 | 0,0000000292 | | |

Table 1: Parameters of the cluster

The first experiment compared the values of classical and proposed efficiency in a heterogeneous cluster under variable node count and workload conditions (figures 1(a) and 1(b)). In all the configurations, the cluster is composed of 2 NS processors while the rest of processors are NF. It can be seen that the efficiency figures computed using the classical efficiency definition on the heterogeneous cluster are not consistent. On the other hand the proposed efficiency yields results very close to those obtained in the homogeneous cluster.

In the second experiment we have measured the heterogeneous efficiency value for a cluster composed by two nodes and a dimension of the adjacency matrix of 128. Then we have estimated the workload needed to maintain constant this efficiency when the number of nodes is increased in a heterogeneous cluster and compared with the real measured values. Table 2 presents the results achieved, where P is the number of nodes, $P_T$ is the total computational power, $N$ is the measured workload and H-isoefficiency is the theoretical computed value for the workload. Additionally the values of heterogeneous and classical efficiency are shown.

Table 2 presents the workload computed through Equation 10 (labeled as H-isoefficiency), necessary to keep the heterogeneous cluster's efficiency constant at a value around 0.842. The column $N$ shows the workload actually measured. Keeping the heterogeneous efficiency constant requires that the problem size must be increased according to the H-isoefficiency expression rather than with respect to homogeneous one. The errors between the estimated and measured workload values are very small, validating the assumption that the H-isoefficiency function provides an accurate, *a priori* method for analyzing scalability in heterogeneous clusters.

Table 2 also presents the values of efficiency obtained measuring the sequential time in both the fast and the slow processors. It has to be noted that these results are not consistent with the efficiency definition, in the NF processor. On the other hand the heterogeneous system is not scalable which is not reasonable.

| P | $P_T$ | N | $\varepsilon$ | E (NF) | E (NS) | H-isoeff. |
|---|---|---|---|---|---|---|
| 2 | 68461798 | 128 | 0,842 | 0,842 | 0,343 | 128 |
| 4 | 236438050 | 336 | 0,841 | 1,453 | 0,592 | 336,40 |
| 8 | 572390554 | 640 | 0,841 | 1,758 | 0,717 | 641,05 |
| 16 | 1244295562 | 1088 | 0,841 | 1,910 | 0,779 | 1091,38 |
| 32 | 2588105578 | 1760 | 0,842 | 1,989 | 0,811 | 1759,79 |
| 64 | 5275725610 | 2752 | 0,842 | 2,027 | 0,826 | 2752,34 |
| 128 | 10650965674 | 4216 | 0,841 | 2,045 | 0,833 | 4224,05 |

Table 2: H-isoefficiency for a E=0.842

Finally Table 3 shows the H-isoefficiency and the H-efficiency values for different cluster configurations with the same number of nodes. It has been to highlight that H-isoefficiency predicts different values of workload to different configurations, depends on the total computational power. This predictions are consistent with the measured values and the H-efficiency can be maintained constant. Again, the figures presented by classical efficiency are not consistent.

| Configuration | $P_T$ | H-isoeff. | E(NF) | E(NS) | $\varepsilon$ |
|---|---|---|---|---|---|
| 128 NF | 10750480128 | 3680,60 | 0,800 | 1,964 | 0,800 |
| 114 NF 16 NS | 10122340748 | 3571,45 | 0,754 | 1,849 | 0,800 |
| 96 NF 32 NS | 9158248864 | 3397,12 | 0,682 | 1,673 | 0,801 |
| 64 NF 64 NS | 7566017600 | 3087,72 | 0,563 | 1,382 | 0,800 |

Table 3: H-isoefficiency for a E=0.80, P=128 with different cluster configurations

# 6 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 of the algorithm in the selected architecture. Comparing this model with others previously described in the literature, it presents several advantages, just like the isoefficiency

model proposed by Kumar and Rao [9] does. Its most remarkable advantages are that, on one hand, being an *a priori* method it does not require the implementation of the algorithm to be studied in the selected architecture. On the other hand, it deals with both power and physical scalability without imposing any restriction on the system's setup.

The experiments performed have shown 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 includes the development of more systematic and precise methods for estimating both overhead and relative node computational power. Additionally the assumption of the "workload evenly distributed according to each's node computational power" will be removed from the scalability theorems.

## Acknowledgements

This work has been partially supported by the Spanish Ministry of Education and Science under the contracts TIN2007-68023-C02-01, TIN2007-67188 and CSD2007-00050, by the HiPEAC European Network of Excellence as well as by the Cajal Blue Brain project.

## References

[1] 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.

[2] MPI Forum. A message-passing interface standard. 1995. http://www.mpi-forum.org.

[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] Vipin Kumar and V. Nageshwara 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] Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, and Jack Dongarra. *MPI: The Complete Reference*. The MIT Press, 1996.

[12] 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.

[13] Xian-He Sun and Diane T. Rover. Scalability of parallel algorithm-machine combinations. *IEEE Transactions on Parallel and Distributed Systems*, 5(6):599–613, June 1994.

[14] Frederic A. Van-Catledge. Towards a general model for evaluating the relative performance of computer systems. *International Journal of Supercomputer Applications*, 3(2):100–108, 1989.

[15] 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.

[16] 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.

[17] Xiaodong Zhang, Yong Yan, and Keqiang He. Latency metric: An experimental method for measuring and evaluating parallel program and architecture scalability. *Journal of Parallel and Distributed Computing*, (3), February 1994.

[18] J. R. Zorbas, D. J. Reble, and R. E. VanKooten. Measuring the scalability of parallel computer systems. In *Proceedings, Supercomputing '89: November 13–17, 1989, Reno, Nevada*, pages 832–841. ACM Press, 1989.