
ELSEVIER

Parallel Computing 25 (1999) 1131–1145

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Modeling performance of heterogeneous parallel computing systems

Andrea Clematis[a,*], Angelo Corana[b]

[a] *IMA-CNR, Via De Marini 6, 16149 Genova, Italy*
[b] *ICE-CNR, Via De Marini 6, 16149 Genova, Italy*

Received 5 January 1998; received in revised form 10 November 1998; accepted 25 May 1999

Abstract

We analyze and model the performance of heterogeneous parallel computing systems, where in general each node has a different computing power.

The main features of our approach are: a simple but quite rigorous analysis; an 'energetic' perspective on performance analysis, using concepts like the useful work carried out by each node, the work lost due to the various sources of overhead, and the local and global efficiencies, both for dedicated and non-dedicated environments.

Although we carry out the analysis having workstation networks in mind, in the first part of the paper we try to maintain maximum generality, without introducing any constraint on the kind of interconnection between nodes and communication speed. This general framework can be applied to different specific situations, provided supplementary assumptions are feasible and values of system and application dependent parameters are available.

In the second part the focus of analysis narrows to consider systems with the same communication speed between each pair of nodes, as it occurs for example with workstations connected by switched networks. We examine in this case a class of problems for which it is possible to define an efficiency worsening factor related to the degree of heterogeneity.

Keywords: Performance analysis; Performance modeling; Heterogeneous parallel systems; Networks of workstations; SPMD applications

* Corresponding author.

0167-8191/99/$ – see front matter
PII: S0167-8191(99)00044-7

1. Introduction

Heterogeneous systems composed of various machines connected by a network have proved to be a suitable platform for the execution of parallel applications [7,11]. They require new approaches to the problem of performance analysis and modeling. Particularly, as pointed out in some recent papers, the concepts of speed-up and efficiency have to be generalized in order to take the heterogeneity of the system into account.

At the highest level there are the 'meta-computers', consisting of distributed, heterogeneous high performance nodes, usually with very different architectures. Such systems are normally most suitable for functional parallelization. Donaldson et al. [5] use a task graph model to obtain the heterogeneous speed-up relative to the fastest node and to show that a speed-up greater than the number of nodes (superlinear in their terminology) is possible, provided the various machines are specialized for the various tasks. A similar approach is used in [9] to analyze performance of a climate modeling application. Actually, for functional parallel decomposition, the speeds of the various nodes vary in general with the mapping of tasks to nodes [8,13]. For this reason some authors prefer to operate directly on execution times. With a different approach, a cost vector (or equivalently a speed vector) can be defined for each node, where the various components are the time costs for the various types of codes [4]. A good mapping is in this case a key issue for improving global performance.

The most widely used heterogeneous systems for parallel applications are however the networks of workstations (NOW), generally available in companies and scientific institutions. Such systems, which provide a very good price/performance ratio, are characterized by the fact that heterogeneity of component nodes [13] can be often described with a good approximation by a single parameter, namely the different computing power [14]. Another important characteristic of NOWs is that normally the various machines are not dedicated.

So, the main features of parallel computing with NOWs are that NOWs are heterogeneous and non-dedicated systems [12], whereas classical parallel machines are usually homogeneous and dedicated.

For such kind of systems it is of primary importance to employ a heterogeneous partitioning of the total work among nodes, possibly through a dynamic load balance strategy to adapt the amount of work handled by each node to the actual load situation [1,13].

In [6,12,14] each node is described by a power weight relative to the fastest workstation in the system. Yan et al. [12] propose a discrete time model with a task graph approach to predict performance of NOWs, taking into account the effects of other workloads and network contention.

In this paper we analyze and model performance of a heterogeneous parallel system under the hypothesis that each node can be described by a computational speed that is independent of the particular splitting of the total computational work involved with the application among nodes. Although the analysis is motivated by the need to understand and model performance of NOWs, we try to maintain

maximum generality. For this reason in the first part of the paper we study performance of a generic parallel system whose nodes have different computing power, without make any assumption about the kind and speed of the interconnection links. In principle our analysis can be applied to systems with distributed and shared memory. Moreover we do not make any hypothesis about the programming paradigm employed.

After a description of the considered computational environment, we carry out a node level analysis, taking into account the useful computational work performed by each node and the work lost owing to the various sources of overhead. Then we show how the local efficiencies contribute to overall performance, both for dedicated and non-dedicated environments; moreover, we define the generalized speed-up, the equivalent global speed and the degree of parallelism. Finally, we consider in more detail a particular class of applications on systems with a homogeneous interconnection network, pointing out the effect of heterogeneity on performance.

2. Performance analysis

2.1. Computational environment

Let us consider a parallel system consisting of p nodes, which generally have different computing power. For the moment we do not make any assumption regarding the kind of interconnection and communication speed.

Let us consider a generic application A and let W be the computational work (i.e., the number of atomic application oriented operations) involved with A. Let us consider a reference machine, not necessarily belonging to the processor set. The time needed to execute application A on this reference machine (in a dedicated way) is

$$T_{\text{seq}} = W\tau, \tag{1}$$

where τ denotes an atomic computing time (e.g. the time per operation or per element) on the reference machine. We consider only the computational kernel of the application, e.g. we do not take into account I/O times and similar, and suppose that the application can be fully parallelized.

We now introduce the relative speed s_i of each node, measured with respect to the reference machine

$$s_i = \frac{T_{\text{seq}}}{T^i_{\text{seq}}}, \quad i = 1, \ldots, p, \tag{2}$$

T^i_{seq} being the time for executing A on the single ith node (dedicated). If necessary, for example owing to memory limits, s_i can be measured using a smaller problem size. s_i depends mainly on the clock speed ratio of nodes, but it can also depend, in general, on architectural features and on the kind of application.

We point out that our analysis applies when the speed of each node does not depend on the particular splitting of the total work among nodes. In other words we

assume that s_i does not vary if we measure it using the whole application A or any portion of it. This is true when:

1. The application A is homogeneous, i.e., the various nodes execute essentially the same type of code irrespective of the particular splitting of work. The most common case, and that in which we are mainly interested, is the SPMD (single program-multiple data) programming model.
2. The various processors essentially differ in their clock speed. Some authors [4] identify this situation as 'weak heterogeneity', which occurs when the cost vectors of the various nodes are two by two linear dependent.

We also assume that the relative speed of each node does not vary with the size of the task handled by the node. Indeed, we can sometimes observe a gain in processor speed when the amount of local data decreases, for example due to better use of the hierarchy of memories. In principle we could extend the model by providing expressions able to give the changes in the s_i with the data size for a particular node; however in most cases this effect is negligible, at least for a wide range of intermediate sizes [12,14].

In any case an accurate measurement of the relative speeds is of course essential for a good performance analysis and modeling [8,12]. The investigation of the effects of the relative speed error on the results of the analysis would be of great interest, but it is beyond the scope of the present paper.

Unlike most works on heterogeneous parallel systems [6,12,14], we express the speeds of various nodes relatively to a fixed reference machine [8], and not relatively to the fastest node. Although the latter choice may appear more natural since it makes it possible to obtain the speed-up by comparing performance of the parallel system with that of the fastest single node available, we think that choosing a fixed reference allows clearer performance analysis, especially if we vary the number and/or the power of nodes. In particular, we avoid the sharp changes in speed-up and efficiency which would otherwise arise when the fastest node in the set varies, and that are mainly due to the change of reference machine for the sequential execution.

The total relative speed of the p node system is

$$S = \sum_i s_i \tag{3}$$

and the average relative speed is

$$\overline{s} = \frac{S}{p}. \tag{4}$$

Let us suppose that the application A can be implemented by a parallel program consisting of p tasks A_i $i = 1, \ldots, p$, each requiring a computational work W_i, and that task ith is executed on node ith. We suppose that the parallelization does not modify the total computational work, i.e.,

$$W = \sum_i W_i. \tag{5}$$

We point out that one of the aims of our paper is to consider how, given an application A with a fixed amount of work W, such work splits among nodes, depending on relative speeds and local efficiencies, in order to achieve a good load balancing.

2.2. Node level analysis

Let T_{par} be the parallel execution time measured on the p nodes system.

In general the elapsed time on node i executing task A_i can be expressed as (Fig. 1)

$$T_i = T_i^{\text{comp}} + T_i^{\text{ovh}} + T_i^{\text{oth}}, \tag{6}$$

where T_i^{comp} is the computation time, T_i^{ovh} takes into account all the overheads due to parallelization and T_i^{oth} is the time in which the component process of the parallel application is ready to run but not running since the CPU is busy with other activities (in a dedicated node $T_i^{\text{oth}} = 0$). T_i^{ovh} can be decomposed into various contributions (Fig. 1), namely

$$T_i^{\text{ovh}} = T_i^{\text{comm}} + T_i^{\text{ovhp}} + T_i^{\text{idle}}, \tag{7}$$

T_i^{comm} being the time for communications (to be precise this time takes into account only communications not overlapped with computation), T_i^{ovhp} the possible overhead time due to set-up, etc., and T_i^{idle} the idle time due to unbalancing.

With this notation we have

$$T_{\text{par}} = T_1 = T_2 = \cdots = T_p. \tag{8}$$

Since $\tau_i = \tau/s_i$ is the atomic CPU time on the ith node, the CPU time in each node is

$$T_i^{\text{comp}} = W_i \frac{\tau}{s_i}, \quad i = 1, \ldots, p. \tag{9}$$

We note from the above expression that $s_i T_i^{\text{comp}}$ is the equivalent CPU time needed by the reference processor to carry out the work W_i performed by node ith in the parallel execution.

Let us define for each node the ratio between the overhead and computation times

$$R_i = \frac{T_i^{\text{ovh}}}{T_i^{\text{comp}}}, \quad i = 1, \ldots, p \tag{10}$$

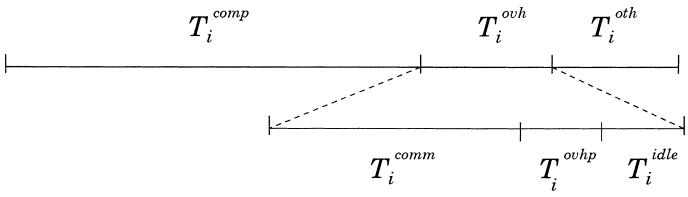

Fig. 1. Schematization of the various contributions to T_i.

and the related efficiency at the node level (node or local efficiency)

$$\eta_i = \frac{1}{1+R_i} = \frac{T_i^{\text{comp}}}{T_i - T_i^{\text{oth}}}, \quad i = 1, \ldots, p \tag{11}$$

which expresses the fraction of the available time on node ith dedicated to computation. By using Eq. (9), η_i can also be viewed as the ratio between the actual work carried out by the ith node and the work that the node could perform in the available time in absence of overheads. If $T_i^{\text{ovh}} = 0$, $R_i = 0$ and $\eta_i = 1$.

We note that η_i and all the other efficiencies we will define in the following can vary, as usually, between 0 and 1, the unitary value being reached in the ideal case.

Defining a load factor for the ith node as

$$\sigma_i = \frac{T_i^{\text{oth}}}{T_i}, \quad i = 1, \ldots, p, \tag{12}$$

we can express the fraction of the elapsed time on node i available for the parallel application

$$\frac{T_i - T_i^{\text{oth}}}{T_i} = 1 - \sigma_i, \quad i = 1, \ldots, p. \tag{13}$$

Combining Eqs. (11) and (13) we obtain the effective node efficiency

$$\eta_i^{\text{eff}} = \frac{T_i^{\text{comp}}}{T_i} = \eta_i(1 - \sigma_i), \quad i = 1, \ldots, p. \tag{14}$$

In the same way, we can define a ratio between any contribution to T_i and the available time at the ith node; for example

$$\rho_i = \frac{T_i^{\text{comm}}}{T_i - T_i^{\text{oth}}}, \quad i = 1, \ldots, p, \tag{15}$$

for the communication time and the analogous quantities δ_i and γ_i for T_i^{ovhp} and T_i^{idle}, respectively. Each of them expresses the fraction of work lost at node i owing to a particular source of overhead. Of course we have

$$\eta_i + \rho_i + \delta_i + \gamma_i = 1. \tag{16}$$

2.3. System level analysis

Let us define the speed-up for application A as the ratio between the sequential time on the reference machine and the elapsed time on the parallel system

$$SU = \frac{T_{\text{seq}}}{T_{\text{par}}}. \tag{17}$$

Using Eqs. (1), (5) and (9) we obtain

$$SU = \frac{\sum_i (s_i T_i^{\text{comp}})}{T_{\text{par}}} \tag{18}$$

which allows estimation of speed-up from a single parallel trial, by measuring the CPU times on the various nodes. Using Eqs. (8) and (14) we may express the speed-up as

$$SU = \sum_i (s_i \eta_i (1 - \sigma_i)) \tag{19}$$

which shows how the local efficiencies contribute to the overall speed-up; $s_i(1-\sigma_i)$ can be considered as the effective relative speed of the ith node. We can see that $SU \leqslant S$.

We can define an equivalent relative speed s_{equiv} for the whole parallel system, such that (likewise Eqs. (1) and (9)) we have

$$T_{\text{par}} = W \frac{\tau}{s_{\text{equiv}}}. \tag{20}$$

Using Eqs. (1) and (17) we can see that $s_{\text{equiv}} \equiv SU$; s_{equiv} (see Eq. (19)) can be viewed as the sum of the actual node speeds weighted by the effective node efficiency.

The ideal speed-up for the considered application is reached when $\eta_i = 1,\ i = 1, \ldots, p$, obtaining from Eq. (19)

$$SU_{\text{id}} = \sum_i (s_i (1 - \sigma_i)). \tag{21}$$

We see that the ideal speed-up is equal to the sum of the effective relative speeds. In the particular case of a dedicated system, the ideal speed-up becomes $SU_{\text{id}} = \sum_i s_i$; moreover, for a dedicated homogeneous environment ($s_i = 1,\ i = 1, \ldots, p$) we obtain the usual expression $SU_{\text{id}} = p$.

Hence, considering our domain of application, and in accordance with the related assumptions (speeds s_i independent of task size; $\sum_i W_i = W$; T_{seq} measured on a dedicated reference node) and with our definition of speed-up, we can observe, in a dedicated system, a *sublinear* speed-up if $SU < S$ and a *linear* (or ideal) speed-up if $SU = S$ [14]. We note that with our assumptions we can never obtain $SU > S$, which is usually identified as *superlinear* speed-up. Different application domains and assumptions can lead to different conclusions about superlinear speed-up.

It is now possible to define the global efficiency of the parallel application as the ratio between the actual and ideal speed-up

$$\eta = \frac{SU}{SU_{\text{id}}} = \frac{\sum_i (s_i (1 - \sigma_i) \eta_i)}{\sum_i (s_i (1 - \sigma_i))}. \tag{22}$$

The above equation shows that the global efficiency is a weighted average of the local efficiencies, the weights being the effective relative speeds. We see that to obtain good global efficiency it is necessary to have high local efficiencies on the fastest and/or the most unloaded nodes. Only when the local efficiencies are the same for all nodes, i.e., $\eta_i = \eta_{\text{const.}},\ i = 1, \ldots, p$ we have $\eta = \eta_{\text{const.}}$, irrespective of the relative speeds of nodes. In the particular case of a dedicated homogeneous environment we obtain the usual expression $\eta = \sum_i \eta_i / p$.

Until now we dealt with fixed size problems, with a fixed total computational work W. We can also obtain η with a more direct approach, by considering the ratio between the actual total work and the maximum total work that the system could carry out in the available time. In this case we operate with a fixed elapsed parallel time T_{par} and allow the total work to vary. Using Eqs. (9), (13) and (14) and taking advantage of Eq. (8) we obtain

$$\eta = \frac{\text{actual tot. work}}{\text{max. tot. work (avail.)}} = \frac{\sum_i((s_i/\tau)T_i^{\text{comp}})}{\sum_i((s_i/\tau)(T_i - T_i^{\text{oth}}))} = \frac{\sum_i(s_i(1-\sigma_i)\eta_i)}{\sum_i(s_i(1-\sigma_i))}. \tag{23}$$

To complete the efficiency analysis, we define the global utilization factor of the system

$$\eta_u = \frac{\text{max. tot. work (avail.)}}{\text{max. tot. work (in } T_i)} = \frac{\sum_i((s_i/\tau)(T_i - T_i^{\text{oth}}))}{\sum_i((s_i/\tau)T_i)} = \frac{\sum_i(s_i(1-\sigma_i))}{\sum_i s_i}. \tag{24}$$

In this way we are able to define an effective global efficiency

$$\eta_{\text{eff}} = \eta\eta_u = \frac{\sum_i(s_i(1-\sigma_i)\eta_i)}{\sum_i s_i}. \tag{25}$$

We see how η_u converts the efficiency computed with respect to the available time to the effective one computed with respect to the elapsed time.

$1-\eta$ gives the fraction of the total work (available) lost due to the parallelization overheads; $1-\eta_{\text{eff}}$ gives the fraction of the total work (in the elapsed time) lost owing to the parallelization overheads and other activities.

Similarly, we can express the ratio between the work lost in the system owing to a particular source of inefficiency (e.g. communications) and the total possible work in the available time. We obtain expressions identical to Eq. (22), with the actual ratios ρ_i or δ_i or γ_i instead of η_i, and the corresponding global quantities ρ or δ or γ instead of η.

Also the global quantities satisfy the relation

$$\eta + \rho + \delta + \gamma = 1. \tag{26}$$

As before, η_u is the conversion factor for obtaining the effective global quantities.

Following [14] we can define the parallelism degree P_{deg} (of application A on the considered parallel system) as the ratio between s_{equiv} and $\overline{s}$, and using Eqs. (4), (19) and (25) and remembering that $s_{\text{equiv}} = SU$ we obtain

$$P_{\text{deg}} = \frac{s_{\text{equiv}}}{\overline{s}} = p\frac{\sum_i(s_i(1-\sigma_i)\eta_i)}{\sum_i s_i} = p\eta_{\text{eff}}. \tag{27}$$

We see that $P_{\text{deg}} \leqslant p$; P_{deg} can be viewed as the effective number of active processors in the parallel execution; using P_{deg} we can express the speed-up as $SU = P_{\text{deg}}\overline{s}$.

2.4. Node contribution to total work

It may be interesting to obtain the ratio of the work carried out by each node and the total work; from Eqs. (9) and (1) we obtain

$$\frac{W_i}{W} = \frac{s_i T_i^{\text{comp}}}{T_{\text{seq}}} \tag{28}$$

which, using Eqs. (17), (19) and (14), can be expressed as

$$\frac{W_i}{W} = \frac{s_i(1-\sigma_i)\eta_i}{\sum_k (s_k(1-\sigma_k)\eta_k)}, \tag{29}$$

showing that the fraction of the whole work performed by each node is proportional to its relative speed, to the local efficiency and to the utilization factor. In the case of an ideal parallelization ($\eta_k = 1,\ k = 1,\ldots,p$) on a dedicated system we have

$$\frac{W_i^{(\text{id})}}{W} = \frac{s_i}{S} \tag{30}$$

i.e., each node performs a work exactly proportional to its speed.

From Eqs. (29), (30) and (25) we can obtain for each node the ratio between the actual work and the work in the ideal case; of course we must deal with the same application A requiring the same total work W, and the ideal case means an ideal system/parallelization, i.e., infinite communication speed or communications perfectly overlapped with computations, absence of other overheads and perfect load balancing.

$$\frac{W_i}{W_i^{(\text{id})}} = \frac{(1-\sigma_i)\eta_i}{\eta_{\text{eff}}}. \tag{31}$$

3. Using performance metrics for a class of applications

The analysis proposed henceforth is quite general since we do not pose any restriction on the way the processors communicate or on the computational model.

Given a particular parallel system/environment and an application, the analysis can be specialized by considering the various contributions to execution time, possibly providing analytical expressions able to give the various times as a function of the variables involved (e.g. CPU time per element, communication parameters, subdomain size, etc.) [10]. In this way we are able to obtain the overhead to computation time ratios R_i at the node level, and hence the local efficiencies η_i; from these local quantities we derive global performance metrics like speed-up and efficiency at the system level. In this section we consider an example of a more specific analysis.

Let us suppose that our system is a distributed memory multiprocessor with communications based on message passing, and that the time to send/receive a message of length N from/at the node i to/from the node j can be modeled as [10,12]

$$t_{\text{c}} = \alpha_{i,j} + N\beta_{i,j}, \tag{32}$$

where $\alpha_{i,j}$ is the latency and $\beta_{i,j}$ is the time to transmit a byte.

In the most general case the communication speed between each pair of nodes can be different. We do not consider, on the contrary, the possible variations in

communication time depending on the actual interconnection path, if messages are routed, and on the traffic on the various links. As already stated we only consider communications not overlapped with computation.

Eq. (32) can also be applied to a virtual shared memory (VSM) environment such as Linda [2], since read and write from/to VSM are actually carried out with message passing between pairs of nodes.

Since we are mainly interested in NOWs, the most common case is a network with all nodes connected by links of the same bandwidth (for example switched Ethernet or Fast Ethernet). Of course it is also possible to have a mixed network (for example a subnetwork connected by Ethernet and another one connected by Fast Ethernet or FDDI). In any case, it is interesting to examine systems with a homogeneous interconnection network in more detail; in this situation $\beta_{i,j} = \beta$ for any pair of nodes, and as a first approximation the latencies can also be considered the same for all pairs ($\alpha_{i,j} = \alpha$) [12,14].

With these assumptions the communication time can be expressed as

$$T_i^{\mathrm{comm}}(A_i) = \sum_{l=1}^{n_m(i)} (\alpha + N_l(i)\beta) = n_m(i)\alpha + C_i\beta = f_{\mathrm{c}}(C_i, n_m(i), \alpha, \beta),$$
$$i = 1, \ldots, p, \tag{33}$$

where $n_m(i)$ is the total number of messages that processor i must handle, $N_l(i)$ is the length of the lth message handled by processor i, C_i is the total amount of communications involved in A_i.

f_{c} points out that communication time on node i depends on C_i and $n_m(i)$ (which in turn are related to the particular application and to the task size), and on the network speed (modeled by α and β).

Let us suppose now that T_i^{ovhp} and T_i^{idle} are negligible, i.e.,

$$T_i^{\mathrm{ovh}} = T_i^{\mathrm{comm}}. \tag{34}$$

Using Eqs. (10), (34), (9) and (33) we obtain

$$R_i = \frac{T_i^{\mathrm{comm}}}{T_i^{\mathrm{comp}}} = s_i \frac{f_{\mathrm{c}}(C_i, n_m(i), \alpha, \beta)}{W_i \tau}. \tag{35}$$

We shall now focus on a particular class of problems in which the communication time $f_{\mathrm{c}}(C_i, n_m(i), \alpha, \beta)$ is proportional to the computation time $W_i\tau$ on the reference node; in this case we can write

$$R_i = s_i R, \tag{36}$$

where R is the constant (independent of W_i) ratio of communication to computation times on the reference node. Eq. (36) shows that R_i increases (i.e., worsens) as the node speed increases.

If we disregard the communication latency, we see that Eq. (36) holds for applications with the amount of communications C_i proportional to the amount of computational work W_i.

A particular case is the one of a data parallelization obtained subdividing the application into a number of subtasks a_j of equal size (task farm paradigm). Each node processes n_i of these subtasks ($A_i = \{a_{j_1}, a_{j_2}, \ldots, a_{j_{n_i}}\}$) in a time

$$T_i = n_i\left(\frac{t}{s_i} + t_c\right), \tag{37}$$

t being the CPU time to process a subtask on the reference node and t_c the time needed for the communications involved in the subtask. In this case Eq. (36) holds exactly, irrespective of the particular forms assumed by the times t and t_c at the subtask level [3].

Let us now examine the implications of expression (36). By substituting Eq. (36) into Eq. (11) we obtain

$$\eta_i = \frac{1}{1 + s_i R}. \tag{38}$$

So, owing to the constant communication speed, the fastest nodes are penalized, and show a lower local efficiency.

Given the ratio R and a system of p nodes with a fixed total power S, it would be interesting to find the values of s_i which maximize the global efficiency. We observe that, since the total power is constant, when the efficiency is maximum also the speed-up is maximum.

Considering, for the sake of simplicity, a dedicated system ($\sigma_i = 0,\ i = 1, \ldots, p$), substituting Eq. (38) into Eq. (22) and using the method of Lagrange multipliers, we easily find that these optimal values are

$$s_i = \frac{S}{p}, \quad i = 1, \ldots, p, \tag{39}$$

i.e., the optimal configuration is the homogeneous one. Using Eqs. (39) and (38), we obtain from Eq. (22) the efficiency for the homogeneous system

$$\eta_{\text{hom}} = \frac{1}{1 + R(S/p)}. \tag{40}$$

We are therefore able to introduce an efficiency worsening factor (ewf) to quantify the effect of heterogeneity on performance; ewf is defined as the ratio between the actual efficiency obtained with the actual speeds and the efficiency in the homogeneous case

$$\text{ewf} = \frac{\eta}{\eta_{\text{hom}}} = \frac{\sum_i ((s_i/\overline{s} + s_i R)/(1 + s_i R))}{p}. \tag{41}$$

This equation shows that ewf $\leqslant 1$ and that it is unitary (a) in the homogeneous case; (b) if $R = 0$, irrespective of the s_i values.

To perform our analysis we chose to express the degree of heterogeneity of the system through a single parameter, i.e., the standard deviation $H = \left(\sum_i (s_i - \overline{s})^2/p\right)^{1/2}$.

In order to obtain some numerical values, we consider a system of six nodes, with constant total power, and ten different configurations with increasing heterogeneity, as reported in Table 1.

Fig. 2 shows ewf, computed from Eq. (41), versus the degree of heterogeneity H for different R ratios. We may observe that efficiency decreases as H increases, and the effect is stronger when R is greater. There is one exception, namely the configuration with $H = 0.43$, which performs better than the one with $H = 0.40$, the reason is that the latter has a single node with a low speed in a otherwise homogeneous system, so H is not so high and does not capture completely the impact of heterogeneity on performance.

Table 1
Configurations with increasing heterogeneity used in Fig. 2; $p = 6$; $S = 6$

s_i						H
1.0	1.0	1.0	1.0	1.0	1.0	0.00
0.6	0.8	1.0	1.0	1.2	1.4	0.26
0.5	0.7	0.9	1.1	1.3	1.5	0.34
0.1	1.18	1.18	1.18	1.18	1.18	0.40
0.4	0.6	0.8	1.2	1.4	1.6	0.43
0.5	0.5	0.5	1.5	1.5	1.5	0.50
0.1	0.4	0.7	1.3	1.6	1.9	0.65
0.2	0.2	0.8	0.8	2.0	2.0	0.75
0.2	0.2	0.6	0.6	2.2	2.2	0.86
0.1	0.1	0.1	1.9	1.9	1.9	0.90

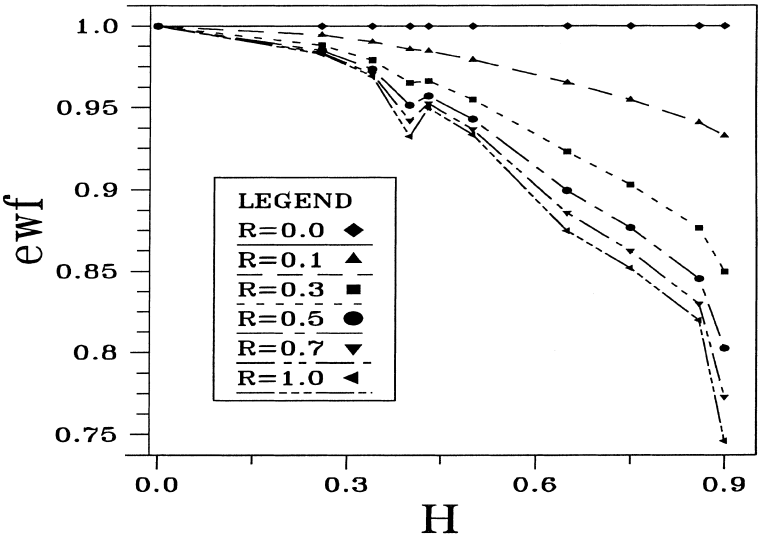

Fig. 2. Efficiency worsening factor vs. the degree of heterogeneity H (standard deviation of speeds), for various values of R; $p = 6$; $S = 6$.

A similar procedure can be applied for non-dedicated systems, obtaining more involved expressions which take into account the dependence on the σ_i values.

Using the assumptions stated in this section we can also derive a specialized version of Eq. (31), obtaining

$$\frac{W_i}{W_i^{(id)}} = \frac{1}{1 + s_i R} \frac{\sum_k s_k}{\sum_k (s_k/(1 + s_k R))} = \eta_i/\eta. \tag{42}$$

By definition, this ratio is unitary if $R = 0$, irrespective of the s_i values; moreover it is unitary in the homogeneous case, irrespective of R. Table 2 reports $W_i/W_i^{(id)}$ for the configurations of Table 1 and various values of R. It turns out that, owing to communication time, the fastest nodes carry out an amount of work lower than in the ideal case, whereas the lowest nodes carry out an amount of work higher than in the ideal case; this effect increases with R, i.e., with the communication time.

Since the impact of heterogeneity on performance, keeping the total power and the number of nodes fixed, is a quite controversial point, some remarks are needed.

In the situation examined in this section the optimal configuration is the homogeneous one because it yields the most uniform distribution of the total communications among the available links. Our result is in accordance with the work of [12,14], where it is affirmed that an increase in heterogeneity worsens performance.

Mazzeo et al. [8] on the other hand found that a higher degree of heterogeneity may yield a benefit for overall performance, since a concentration of the computational work on the most powerful nodes should produce a reduction in

Table 2
Actual to ideal work ratio on each node for various configurations with increasing heterogeneity and different values of R; $p = 6$; $S = 6$

H	s_i	$R = 0.1$	$R = 0.3$	$R = 0.5$	$R = 0.7$	$R = 1.0$
0.26	0.6	1.04	1.11	1.17	1.22	1.27
	0.8	1.02	1.06	1.09	1.11	1.13
	1.0	1.01	1.01	1.02	1.02	1.02
	1.0	1.01	1.01	1.02	1.02	1.02
	1.2	0.99	0.97	0.95	0.94	0.93
	1.4	0.97	0.93	0.90	0.87	0.85
0.43	0.4	1.07	1.20	1.31	1.39	1.50
	0.6	1.05	1.14	1.21	1.26	1.32
	0.8	1.03	1.08	1.12	1.14	1.17
	1.2	1.00	0.99	0.98	0.97	0.96
	1.4	0.98	0.95	0.92	0.90	0.88
	1.6	0.96	0.91	0.87	0.84	0.81
0.75	0.2	1.13	1.36	1.56	1.73	1.96
	0.2	1.13	1.36	1.56	1.73	1.96
	0.8	1.07	1.16	1.22	1.26	1.30
	0.8	1.07	1.16	1.22	1.26	1.30
	2.0	0.96	0.90	0.86	0.82	0.78
	2.0	0.96	0.90	0.86	0.82	0.78

communications. In this sense the most effective configuration is that with one processor with the total power and all the others with null power; in this case of course the system reduces to a sequential machine. Apart from this trivial case, we think that a general treatment is not possible since the behaviour depends on the interconnection topology and on the communication pattern of the considered application.

4. Conclusions

We have presented a simple but effective methodology for analyzing and modeling the performance of heterogeneous parallel systems.

The analysis can be applied to various systems, provided the speed of each node does not depend on the particular splitting of total work among nodes. This holds in particular for homogeneous applications (e.g. SPMD), irrespective of the node characteristics.

We do not pose any restriction on the way the processors communicate or on the programming paradigm. Actually, the main architectural class we are interested in, and which motivated our analysis, is the network of workstations.

Starting from the general model, we present a more specific analysis which applies to homogeneous interconnection networks and to problems with a parallelization overhead proportional, at least as a first approximation, to the computation time on the reference node. For such class of systems/applications we derive the effect of the heterogeneity degree on performance.

In a similar way the general analysis can be applied to other situations.

References

[1] C.H. Cap, V. Strumpen, Efficient parallel computing in distributed workstation environments, Parallel Comput. 19 (1993) 1221–1234.

[2] N.J. Carriero, D. Gelernter, T.G. Mattson, A.H. Sherman, The Linda alternative to message-passing systems, Parallel Comput. 20 (1994) 633–655.

[3] A. Clematis, A. Corana, Performance analysis of SPMD algorithms on a network of workstations with virtual shared memory, in: Parallel Computing Fundamentals, Applications and New Directions, E.H. D'Hollander et al., eds (1998), Elsevier Science, 657–664.

[4] L. Colombet, L. Desbat, Speedup and efficiency of large-size applications on heterogeneous networks, Theoretical Computer Science 196 (1998) 31–44.

[5] V. Donaldson, F. Berman, R. Paturi, Program speedup in a heterogeneous computing network, J. Parallel and Distributed Computing 21 (1994) 316–322.

[6] D. Grosu, Some performance metrics for heterogeneous distributed systems, in: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA'96, vol. III, CSREA Press, 1996, pp. 1261–1268.

[7] A.A. Khokhar, V.K. Prasanna, M.E. Shaaban, C. Wang, Heterogeneous computing: challenges and opportunities, Computer 26 (1993) 18–27.

[8] A. Mazzeo, N. Mazzocca, U. Villano, Efficiency measurements in heterogeneous distributed computing systems: from theory to practice, Concurrency: Practice and Experience 10 (1998) 285–313.

[9] C.R. Mechoso, J.D. Farrara, J.A. Spahr, Achieving superlinear speed-up on a heterogeneous distributed system, IEEE Parallel and Distributed Technology 2 (1994) 57–61.
[10] B.K. Schmidt, V.S. Sunderam, Empirical analysis of overheads in cluster environments, Concurrency: Practice and Experience 6 (1994) 1–32.
[11] V.S. Sunderam, Methodologies and systems for heterogeneous concurrent computing, in: G.R. Joubert, D. Trystram, F.J. Peters, D.J. Evans (Eds.), Parallel Computing: Trends and Applications, Elsevier, Amsterdam, 1994, pp. 29–45.
[12] Y. Yan, X. Zhang, Y. Song, An effective and practical performance prediction model for parallel computing on non-dedicated heterogeneous NOW, J. Parallel and Distributed Computing 38 (1996) 63–80.
[13] M.J. Zaki, W. Li, M. Cierniak, Performance impact of processor and memory heterogeneity in a network of machines, in: Proceedings of the Fourth Heterogeneous Computing Workshop, Santa Barbara, California, 1995.
[14] X. Zhang, Y. Yan, Modeling and characterizing parallel computing performance on heterogeneous networks of workstations, in: Proceedings of the Seventh IEEE Symposium on Parallel and Distributed Processing, IEEE Computer Society Press, 1995, pp. 25–34.