Efficiency of divisible load processing

M.Drozdowski* and Ł.Wielebski

Institute of Computing Science, Poznań University of Technology,
ul. Piotrowo 3a, 60-965 Poznań, Poland
{Maciej.Drozdowski,Lukasz.Wielebski}@cs.put.poznan.pl

Abstract. Effective exploitation of a parallel computer system is a result of cooperation between the communication and the computing parts of a parallel computer system, and the application. These elements are described by various parameters. It is not easy to grasp the connection between the values of particular parameters and the efficiency of parallel computations. In this paper we propose a new way of representing the relations between the parameters of a parallel computer system, and its performance. Results of simulations are presented and discussed.

Keywords: Performance of parallel and distributed systems, divisible load theory, communication delays, clusters, grid.

1 Introduction

Performance of a parallel computer system is guided by complex relations between the computer system components and the application. Understanding the reasons for the limitations in the performance is the first step towards using the power of the hardware effectively.

In this paper we make an attempt to gain new insights into the relationships between the performance of parallel systems and their key parameters. The method we propose is built on the basis of *divisible load theory* (DLT) [2–4], and *isoefficiency function* [6]. DLT assumes that computations (the load) can be divided into parts of arbitrary sizes. The parts can be processed independently by remote computers. DLT appeared to be useful in representing distributed computations in various interconnection topologies. It has become not only theoretically useful, but also practically viable which has been demonstrated by a series of parallel applications [5]. Surveys of DLT can be found in [2–4]. The isoefficiency function has been defined as problem size required to sustain constant efficiency for changing processor number. We generalize this concept.

The rest of this paper is organized as follows. In Section 2 the notion of isoefficiency is presented, and generalized. In Section 3 a DLT model of a distributed computation is presented. Section 4 gives results of performance modeling, and discusses the consequences.

* This research was partially supported by the grant of Polish State Committee for Scientific Research.

2 The concept of isoefficiency

In this section we present, and generalize, the concept of isoefficiency function defined in [6].

Let m denote the number of processors used by a parallel application. We will denote by $T(m)$ the execution time of a parallel application when run on m processors. By speedup we mean ration $S = \frac{T(1)}{T(m)}$. Speedup is the most commonly used measure of a parallel application performance. *Efficiency* $E = \frac{S}{m} = \frac{T(1)}{T(m)m}$ is a measure derived from speedup. Note that both speedup, and efficiency are dimensionless measures. In a sense speedup indicates how much an application can be accelerated by using additional processors. The higher speedup is the better. Efficiency is an indicator showing how well a parallel application uses the processors. Efficiency can be also understood, as an average fraction of all m processors that really work in the parallel application. Thus, the closer efficiency to 1 is, the better.

It has been observed that speedup and efficiency depend on the size of the problem that a parallel application solves. It is hard to maintain high efficiency for small problems and big processor number m. Consequently, with growing processor number m also problem size should increase in order to preserve constant efficiency. This observation resulted in a definition of the isoefficiency function $f(k,m)$ in [6], as the problem size for which efficiency of a parallel application is $E = k$ on m processors. For example, consider an algorithm calculating a minimum spanning tree in a graph with n vertices, where n expresses the size of the problem. There exists an algorithm [1] with running time $T(m) = \frac{c_1 n^2}{m} + c_2 n \log m$, and $T(1) = c_1 n^2$, where c_1, c_2 are constants. Thus, efficiency of this algorithm is $E = \frac{c_1 n^2}{c_1 n^2 + c_2 mn \log m} = k$. In order to maintain constant efficiency value $k < 1$, the size of the problem should be $f(k,m) = \frac{c_2 m \log m}{c_1(\frac{1}{k}-1)}$. Thus, n should be proportional to $m \log m$.

The above idea of isoefficiency function can be further generalized. Efficiency depends not only on the problem size, and the number of processors m, but also on other parameters of a parallel system, e.g., communication, and computation speeds. Therefore, we can define a set of *isoefficient points* in a multidimensional space of the parallel system parameters as the points for which efficiency is constant. It is convenient to think about isoefficient points, or isoefficiency lines, as of analogues of isolines in other branches of science, such as isobars, izohyets, isotherms, isodoses, etc. In order to draw a diagram of the isoefficiency lines a mathematical model linking system parameters and efficiency is needed. We present such a model in the following section.

3 Divisible load model

In this section we present a DLT model for a simple distributed system working in a star interconnection (cf. Fig.1). In the star interconnection the load in amount V initially resides on processor P_0 called originator. The originator sends α_i units

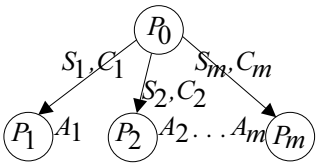

Fig. 1. Star interconnection.

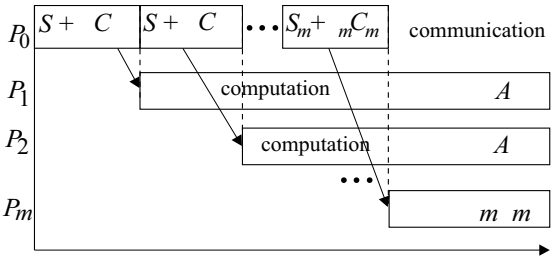

Fig. 2. Gantt chart depicting communications and computation in a star.

of load to processor P_i, for $i = 1, \ldots, m$. Processors receive their load parts one by one from P_1 to P_m, and immediately start processing the load. Messages are sent only between the originator, and the processors. Thus, the star interconnection is equivalent to a bus interconnection, or a master-slave, and client-server systems. For simplicity of presentation we assume that the originator is not computing, but is communicating only. This assumption does not limit the generality of the considerations because otherwise the computing ability of the originator can be represented as an additional processor. For simplicity of the presentation we assume that the time of returning the results is negligible. Also this assumption does not limit the generality of the study, because the process of returning the results can be easily included in the model [2–5]. The time of sending α_i units of load to P_i is $S_i + \alpha_i C_i$, where S_i is communication startup time, C_i is reciprocal of bandwidth. The time of computing on P_i is $\alpha_i A_i$, for $i = 1, \ldots, m$, where A_i is a reciprocal of computing speed. A Gantt chart depicting the communications and computations for the above described system is shown in Fig.2. It is possible to control the computation completion times by the selection of part sizes α_i. When the results are not returned the schedule is the shortest when all processors finish their computations simultaneously [2]. Consequently, the time of computation on a processor activated earlier is equal to the time of sending the load to the next processor and computing on it. Hence, it is possible to formulate the problem of finding the optimum schedule length as a set of linear equations:

$$A_i \alpha_i = S_{i+1} + \alpha_{i+1}(C_{i+1} + A_{i+1}) \text{ for } i = 1, \ldots, m-1 \tag{1}$$

$$V = \sum_{i=1}^{m} \alpha_i, \tag{2}$$

where the length of the schedule is $S_1 + (C_1 + A_1)\alpha_1$. Due to the simple structure, the above equation system can be solved for α_i in $O(m)$ time provided that a feasible solution exists. A solution of (1)-(2) does not exists when $\alpha_i < 0$ for some i. This means in practice that load size V is too small to keep all m processors working, and communication delay incurred in activating all m processors is longer than the time needed to process load V on less than m processors. Using

the solution of (1)-(2) efficiency can be calculated as

$$E = \frac{S_1 + (C_1 + A_1)V}{m(S_1 + (C_1 + A_1)\alpha_1)}. \tag{3}$$

Consequently, we obtained a method of modeling the performance and calculating the isoefficiency diagrams.

4 Performance modeling

In this section we present preliminary results of the simulations performed on the basis of DLT model. The study we present here is limited by admissible size of the paper. For the simplicity of the presentation a homogeneous system was studied. Thus, $\forall_i A_i = A, C_i = C, S_i = S$. The isoefficiency function is depicted in 2-dimensional maps resembling weather maps, e.g., with isobars. The diagrams have two axes depicting two variable parameters, the other parameters are fixed. In the map isoefficiency lines representing points of equal efficiency are shown. The isoefficiency maps separate two areas: the area of high efficiency ($E \approx 1$), and the area of efficiency close to 0 or the equation system (1)-(2) is not solvable. In the latter case the combination of the parameters prevents activating all processors with the given load without idle periods. The area of such points is denoted as $E = 0$. Before discussing the results of the simulations let us comment on the nature of the dependencies presented in the isoefficiency maps. When an isoefficiency line is parallel to one of the axes of the map, then the efficiency changes incurred by this parameter does not influence the efficiency. It also means that one parameter cannot be compensated for by a modification of the value of the other parameter.

In Fig.3 isoefficiency map for variable m, V, and $C = 1, S = 1000, A = 1$ is depicted. As the values of A, C, S in Fig.3 are unitless, these can be, for example, $C = 1\mu s/byte, S = 1ms, A = 1\mu s/byte$. The isoefficiency lines are not smooth which is a result of approximating the hull span on the discrete points for which the efficiency values were calculated. It can be seen in Fig.3 that when m decreases, efficiency grows. Especially for $V <$ 1E2 the increase has a character of a step function which means that only a limited number of processors can be activated for certain problem size V (moving along horizontal lines in the map). The isoefficiency lines have some slope for $V \in$[1E2,1E6]. This means that with growing m, also V should increase to exploit the parallel system efficiently. And vice versa, the number of used processors m should decrease when V is decreasing. Thus, this kind of efficiency behavior confirms observations made in the earlier publications [6]. For big problem sizes $V >$1E6 efficiency no longer depends on V, as the isoefficiency lines are vertical. This phenomenon can be explained in the following way: When V is very big, the ratio between V and α_1 approaches some constant $l(A, C, m) < 1$ dictated by C, A, and m. The formula (3) can be rewritten as: $E = \frac{S_1+(C_1+A_1)V}{m(S_1+(C_1+A_1)Vl(A,C,m))}$. Hence, $\lim_{V\to\infty} E = \frac{1}{ml(A,C,m)}$, and efficiency depends on m, but not on V. It can be concluded that parameters m, V are mutually related in the process of performance optimization.

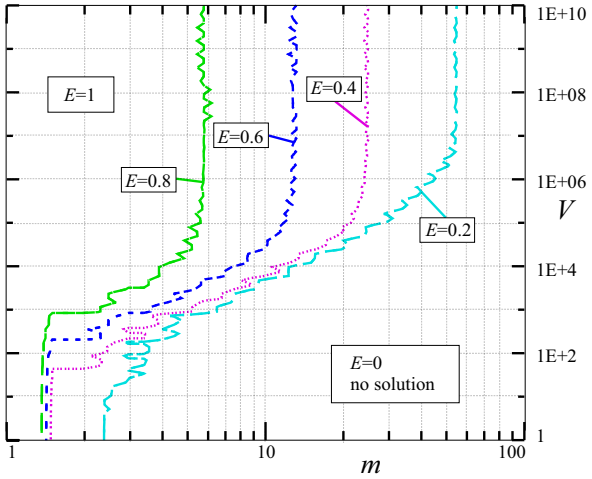

Fig. 3. Isoefficiency map as a function of m, and V.

In Fig.4 the isoefficiency relation for variable C, S, and $m = 10, V =$ 1E8, $A =$ 1E-3 is shown. The dependence of efficiency on C has a character of a step function (moving along vertical lines). For S the situation is similar (moving along horizontal lines). The isoefficiency lines have almost piece-wise linear form separating a rectangular area of high efficiency from the remaining points in the map. This rectangular form of the efficient area means that C, S are unrelated, as far as efficiency is considered. In general, the changes in parameter C (reciprocal of communication speed) cannot be compensated for by startup time S. Only in the narrow north-east corner of the efficient area, where the isoefficiency lines form a kind of a knee, can the changes in S be compensated for by C. The increasing startup time S can be compensated for by decreasing C (i.e. increasing speed). Yet, the range of such a compensation is very limited. The rectangular efficient area determines also the methods of optimizing the performance of parallel computer systems. For example, efficiency of a parallel system with $S \approx$ 1E$-$6, and $C =$ 1E+5 can be improved by reducing C only, which is equivalent to increasing speed. Analogously, a system with $S \approx 1$, and $C =$ 1E$-$5 can be made efficient only by reducing the startup S. From the above discussion we conclude that parameters S, and C are, to a great extent, independent in the process of a parallel system efficiency optimization.

5 Conclusions

In this paper we proposed a new method of representing the relation between the parameters determining the performance of a parallel computer system. Simulations demonstrated that there exist parameters of parallel computer systems, such as m, V, which are mutually related. The changes of one parameter should be accompanied by the changes of the other parameter, in order to maintain constant efficiency. On the other hand there are also parameters which are inde-

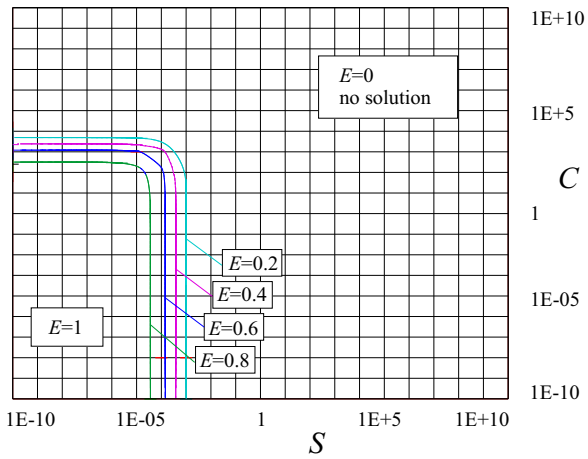

Fig. 4. Isoefficiency map as a function of C, and S.

pendent, e.g. C, S. Future applications of the method may include analysis of the influence of a parallel system heterogeneity, memory limitations, interconnection topology, and load scattering algorithm, on the efficiency of the computations.

References

1. S.G.Akl, *The Design and Analysis of Parallel Algorithms*, Prentice-Hall Int. Inc., Englewood Cliffs, New Jersey, 1989
2. V.Bharadwaj, D.Ghose, V.Mani, T.Robertazzi, *Scheduling divisible loads in parallel and distributed systems*, IEEE Computer Society Press, Los Alamitos CA, 1996.
3. V.Bharadwaj, D.Ghose, T.Robertazzi: Divisible load theory: A new paradigm for load scheduling in distributed systems. *Cluster Computing* **6**, No.1, 2003, 7-18.
4. M.Drozdowski, *Selected problems of scheduling tasks in multiprocessor computer systems*, Series: Monographs, No.321, Poznań University of Technology Press, Poznań, (1997), (see also `http://www.cs.put.poznan.pl/~maciejd/h.ps`).
5. M.Drozdowski, P.Wolniewicz, Experiments with Scheduling Divisible Tasks in Clusters of Workstations, in: A.Bode, T.Ludwig, W.Karl, R.Wismüller (eds.), *Euro-Par 2000, LNCS 1900*, Springer-Verlag, 311-319, (2000).
6. A.Gupta, V.Kumar, Performance properties of large scale parallel systems, *Journal of Parallel and Distributed Computing* **19**, 234-244, (1993).