Multiprocessor Speed-Up, Amdahl's Law, and the Activity Set Model of Parallel Program Behavior

Erol Gelenbe

December 1988

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 88.37

NASA Cooperative Agreement Number NCC 2-387

RIACS

Research Institute for Advanced Computer Science

Multiprocessor Speed-Up, Amdahl's Law, and the Activity Set Model of Parallel Program Behavior

Erol Gelenbe

December 1988

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 88.37

NASA Cooperative Agreement Number NCC 2-387

Multiprocessor Speed-Up, Amdahl's Law, and the Activity Set Model of Parallel Program Behavior

*Erol Gelenbe**

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 88.37
December 1988

An important issue in the effective use of parallel processing is the estimation of the speed-up one may expect as a function of the number of processors used. Amdahl's Law has traditionally provided a guideline to this issue, although it appears excessively pessimistic in the light of recent experimantal results. In this note we first amend Amdahl's Law by giving a greater importance to the capacity of a program to make effective use of parallel processing, but also recognizing the fact that imbalance of the workload of each processor is bound to occur. We then introduce the **Activity Set Model** of parallel program behavior and the corresponding **parallelism index** of a program, leading to upper and lower bounds to the speed-up.

---

* The author's address is Universite Rene Descartes, Ecole des Hautes Etudes en Informatique, 45 rue des Saints Peres, 75006 Paris, France. This research was supported in part by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA) while the author was visiting the Center for Advanced Architectures at RIACS.

1. Introduction

Consider a program that is to be executed on a multiprocessor containing N identical processors. Each of these processors may itself be equipped with one or more array processors attached for its private use.

Amdahl's law [1,2] estimates an upper bound of $N/\log_2 N$ for the actual speed-up, or ratio of elapsed time with a single processor to the elapsed execution time with N processors. Recent experiments [3] have shown that this may be an unnecessarily pessimistic estimate of speed-up and that values close to N may be obtained for specific applications.

We have recently derived another approach to computing the estimated maximum speed-up [4], assuming an unlimited number of processors, based on a model which considers the density p of precedence relations between tasks in programs. We obtain [4] an approximate formula $(1 + p)/2p$ for the maximum speed-up, on the average, for such a family of programs assuming that the number of processors available is unlimited. When p is close to 1, nearly all tasks are interdependent and the programs will in fact execute sequentially; the speed-up factor itself will be close to just 1. On the other hand, when p is very close to 0 we are dealing with programs composed of many quasi-independent tasks and the maximum speed-up is very large. An "infinite" speed-up merely means that the average execution time for the program family remains nearly constant as the number of individual tasks in the program becomes very large.

In this paper we shall begin by considering Amdahl's law and suggest some modifications or amendments that can serve to explain speed-up factors which are nearly linear in the number of processors. We shall

then return to the intrinsic processor independent behavior of parallel programs and introduce a new model of parallel program behavior, the Activity Set Model,[1] which may be used to describe the behavior of parallel programs and to derive bounds to the speed-up which can be expected when such programs are executed or multiprocessors.

We first consider a simple amendment to Amdahl's law, and derive a speed-up formula with N processors which is of the form

$$\frac{N}{[(1 - \varepsilon)(1 + \delta) + \varepsilon \log_2 N]}$$

where ε is the probability that the programs cannot effectively use N processors. δ is a measure of the unbalance between the workloads of each processor when N processors are used. Indeed, δ/N is the amount of time in excess of the optimistic equal run-time 1/N which the most loaded processor will take to run the task that has been assigned to it. Thus, when the computation has been organized so that ε is very small, the speed-up can be as high as $N/(1 + \delta)$. These results are detailed in Section 2.

In Section 3, we consider an evaluation of speed-up based on intrinsic program behavior which leads to general bounds.

Throughout the discussion we consider a program, or a family of programs, whose average run-time on a single processor is equal to 1.

---

[1]The name of this model is, of course, inspired by Peter Denning's Working Set Model used in paging.

2. Amendment to Amdahl's Law

Assume that the program, or family of programs, considered makes full use of the N processors with probability $(1 - \varepsilon)$. It will use only i processors $(1 \leqq i \leqq N - 1)$ with probability ε_i, where

$$\varepsilon = \sum_{1}^{N-1} \varepsilon_i$$

If it uses N processors, one of these will have the greatest workload so that the run-time of the program on N processors is given by the time elapsed for the most heavily loaded one, or

$$\frac{1}{N} + \frac{\delta}{N}$$

Obviously, if all the processors' workload were perfectly balanced, the elapsed time would be simply 1/N.

Similarly, when only i processors are used the elapsed time will be

$$\frac{1}{i} + \frac{\delta_i}{i}$$

These times should be viewed as average values when a family of programs is considered.

Thus, the average elapsed time when N processors are available is given by the formula

$$T(N) = (1 - \varepsilon)(\frac{1}{N} + \frac{\delta}{N}) + \sum_{1}^{N-1} \varepsilon_i (\frac{1}{i} + \frac{\delta_i}{i})$$

So the speed-up $T(1)/T(N) = 1/T(N)$ is simply given by

$$S = \frac{N}{[(1 - \varepsilon)(1 + \delta) + N \sum_{1}^{N-1} \varepsilon_i (\frac{1}{i} + \frac{\delta_i}{i})]} \qquad (1)$$

The Most Favorable Case

Naturally, the greatest speed-up will be obtained if we set $\varepsilon_{N-1} = \varepsilon$, $\varepsilon_i = 0$ for $1 \leqq i \leqq N - 2$.

We then have for large enough N:

$$S \cong \frac{N}{1 + \delta(1 - \varepsilon) + \delta_{N-1}} \qquad (2)$$

This formula explains the quasi-linear speed-ups that can be encountered for some specific applications.

The Least Favorable Case: Amdahl's Law

We consider that the least favorable case for a family of programs is obtained by setting

$$\varepsilon_i = 1 - \varepsilon \simeq 1/N, \text{ for all } 1 \leqq i \leqq N - 1$$

which implies that we are equally likely to make use of any number of processors less than N; this is the assumption made in Amdahl's law. We then have

$$S = \frac{N}{\sum_{1}^{N} \frac{1}{i} + \sum_{1}^{N} \frac{\delta_i}{i}}, \quad \delta_N = \delta \qquad (3)$$

Since $\sum_{1}^{N} \frac{1}{i} \geqq \log_2 N$, we have

$$S \leqq \frac{N}{\log_2 N} \qquad (4)$$

Amendment to Amdahl's Law

Let us consider a family of programs that can fully use all N processors with probability $1 - \varepsilon$. If a program in this family cannot make full use of them, then assume that it is equally likely to use $N - 1$, $N - 2$, ... or just one processor. We then have $\varepsilon_i = \varepsilon/N-1$, $1 \leqq i \leqq N - 1$, so that we obtain

$$S \leqq \frac{N}{[(1 - \varepsilon)(1 + \delta) + \frac{\varepsilon N}{N-1}(\log_2(N - 1) + \sum_{1}^{N-1} \frac{\delta_i}{i})]} \tag{5}$$

For large N, this bound becomes

$$S \leqq \frac{N}{(1 - \varepsilon)(1 + \delta) + \varepsilon \log_2 N} \tag{6}$$

which is a useful compromise between the unnecessarily pessimistic form of Amdahl's Law, and the overly optimistic linear bound.

In (6) program characteristics appear via the parameters ε and δ, where ε is the probability that a program is unable to make use of all N processors, while δ measures the imbalance between the average execution time of parallel tasks. In the sequel we shall consider a simple representation of a parallel program's execution in virtual time.

3. The Activity Set Model of Parallel Program Behavior

Consider a program that is being executed with an **unlimited** number of processors. Its behavior may be characterized by a variable n(t) representing the number of active processes or tasks at some instant t, lying between the instant t = 0 when the program execution is initiated

and the instant t = T when it ends. Such a behavior is shown in Figure 1. n(t) is the size of the Activity Set at time t.

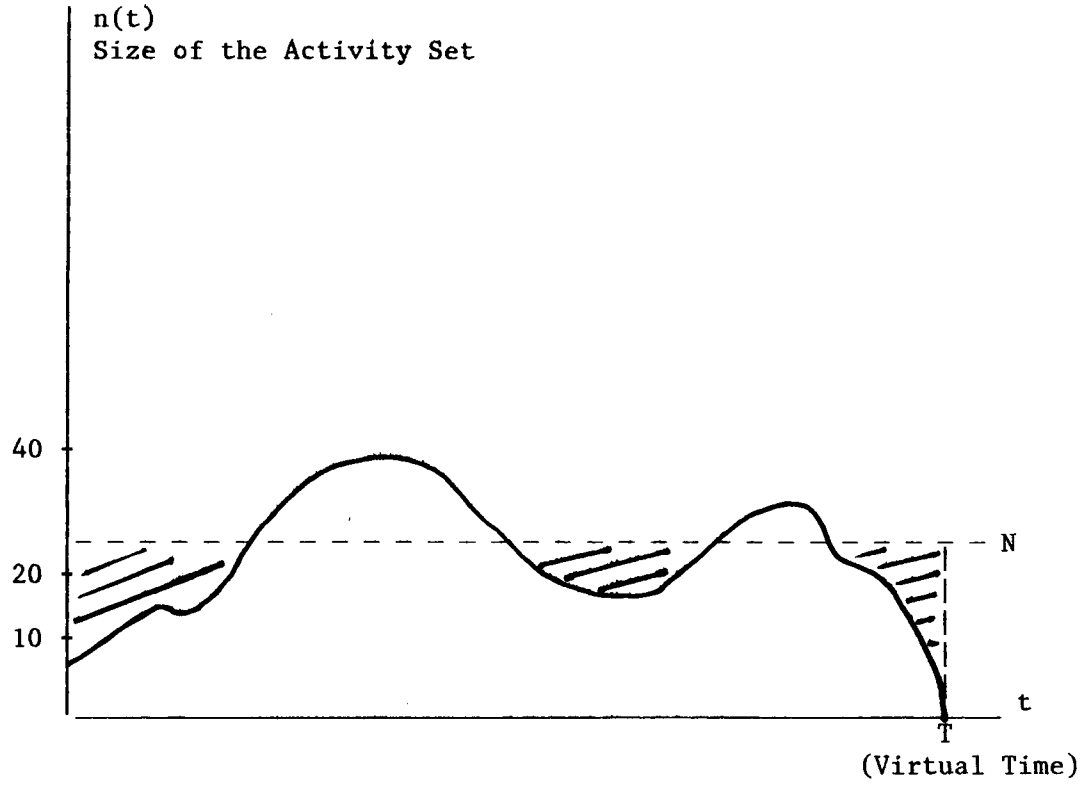

Figure 1. Size of the Activity Set of a parallel program as a function of virtual time t, between t = 0 when the program execution begins and t = T when it ends.

The total work

$$W = \int_0^T n(t)dt \qquad (7)$$

represents the amount of computational effort that must be accomplished by any set of processors in order to execute the program.

The Activity Set of the program at time t is the set of parallel processes or tasks that are running simultaneously at time t.

The parallelism index N_0:

$$N_0 = \frac{1}{T}\int_0^T n(t)dt = W/T \qquad (8)$$

is the average number of active processes in the program, or the average number of processors that it can use simultaneously.

Time t in the expressions above should be considered to be virtual time, or program execution time from which the time spent for all interruptions (paging, I/O, other programs' execution time, etc.) has been deleted.

If the program is to be run on a single processor whose speed is c times faster than any one of the initial set of processors, the program will now run sequentially in time

$$T(c,1) = W/c = N_0T/c \qquad (9)$$

If, on the other hand, we are limited to running the program on N processors, each of which has the same power as one of the processors in the initial unlimited set, we can derive some bounds on the execution time.

Consider the quantity

$$C(N) = \int_0^T (N - n(t))^+ dt$$

where $(x)^+ = x$ if $x \geq 0$ and $(x)^+ = 0$ if $x < 0$.

C(N) denotes the amount of additional work that N processors could provide during the interval [0,T], or their excess capacity, when the program is executed with an unlimited number of processors. It is shown as the shaded area of Figure 1. Similarly,

$$D(N) = \int_0^T (n(t) - N)^+ dt$$

is the work accomplished by the other processors being used when the total number of processors available is unlimited.

When the total number of processors is limited to N, the total execution time T(N) cannot be smaller than

$$T(N) \geq T + [D(N) - C(N)]^+/N$$

since in the best case the excess capacity C(N) will be used to accommodate as much excess work D(N) as the processors can take.

Similarly, an upper bound to T(N) can be derived by considering that all of the work D(N) will be accomplished on the N processors after time T:

$$T(N) \leq T + D(N)/N$$

The speed-up obtained by using N processors instead of a single processor (of speed c = 1) can now be estimated from these bounds. It is given by the formula S = T(1)/T(N) so that

$$\frac{N_0 T}{T + \frac{(D(N)-C(N))^+}{N}} \geq S \geq \frac{N_0 T}{T + D(N)/N} \qquad (10)$$

or

$$\frac{N_0 N}{N + \frac{(D(N)-C(N))^+}{T}} \geq S \geq \frac{N_0 N}{N + D(N)/T} \qquad (11)$$

From these inequalities we see quite clearly that S can never exceed N_0, which should be obvious, but also that $\frac{S}{N_0}$ is bounded from above by the multiplicative factor

$$\frac{1}{1 + \frac{(D(N)-C(N))^+}{TN}}$$

and from below by the multiplicative factor

$$\frac{1}{1 + \frac{D(N)}{NT}}$$

3.1 A Statistical Interpretation

Clearly, the precise behavior of a program as given by the function n(t) is in general very difficult to predict since it will obviously be data dependent. Thus, it is quite natural to treat n(t) as a random function so that N_0, D(N)/T, C(N)/T will now have convenient statistical interpretations in terms of the statistical average or expected value $E[\cdot]$ taken over the finite time interval [0,T]:

$$N_0 \to E[n(t)]$$

$$\frac{D(N)}{T} \to E[(n(t) - N)^+]$$

$$\frac{C(N)}{T} \to E[(N - n(t))^+]$$

We shall now write in particular

$$\frac{S}{N_0} \geq \frac{1}{1 + \frac{1}{N}E[(n(t) - N)^+]} \tag{12}$$

3.2 A Numerical Example

Within the framework of the statistical interpretation of n(t), consider the case where it is time independent and its distribution function is geometric over the interval [0,T]:

$$P[n(t) = i] = q^{i-1}(1 - q), \quad i \geq 1$$

This example describes a situation in which the program is always more likely to have a smaller number of active parallel tasks. Clearly, q must be chosen so that $E[n(t)] = N_0$, hence $N_0 = 1/(1 - q)$ or $q = (N_0 - 1)/N_0$.

We then have

$$E[(n(t) - N]^+] = \sum_{i=N}^{\infty} (i - N)q^{i-1}(1 - q)$$

$$= q^N/(1 - q) = \left(\frac{N_0 - 1}{N_0}\right)^N \cdot N_0$$

Hence

$$\frac{S}{N_0} \geq \frac{1}{1 + \frac{N_0}{N}\left(\frac{N_0 - 1}{N_0}\right)^N} \tag{13}$$

On Figure 2 we show the lower bound (13) to the speed-up S as a function of the number of processors N for various values of N_0 (the parallelism index).

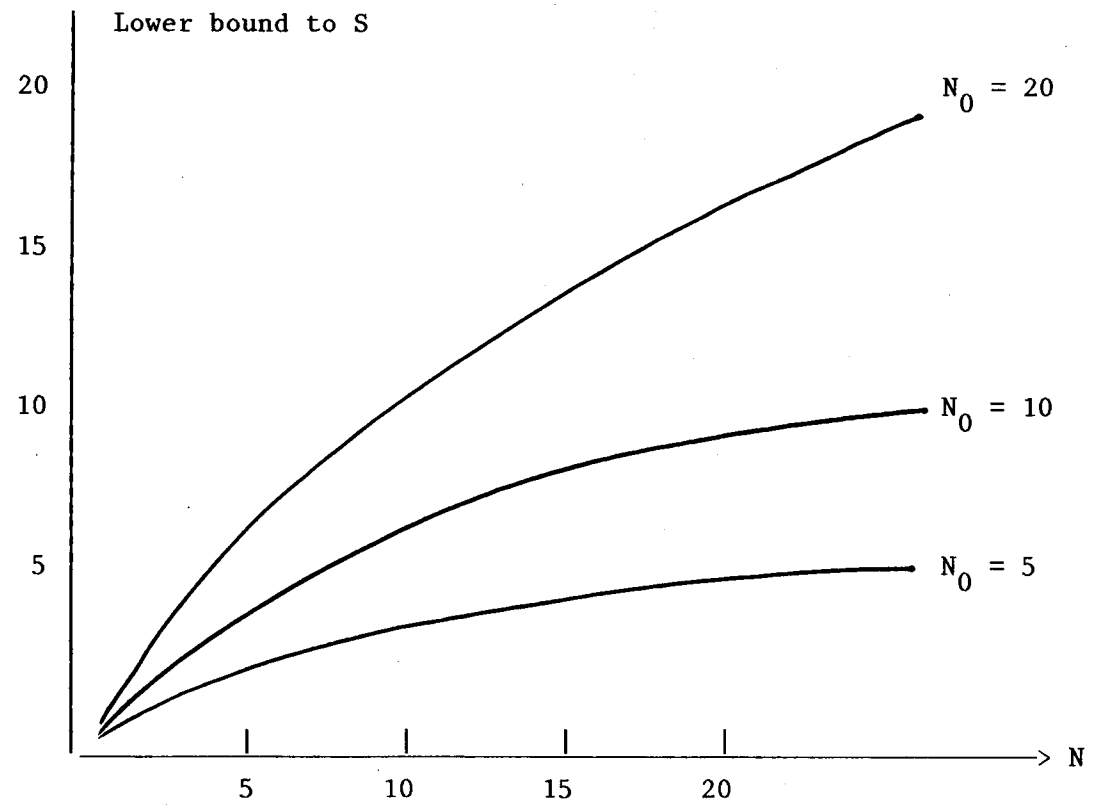

Figure 2. The lower bound for the speed-up S as a function of N for various values of N_0 provided by equation (13).

4. Conclusions

In this note we have considered some amendments to Amdahl's law which take into account the fact the programs may be able to make effective use of all of the processors which are available to them, but which also recognize the fact that imbalance in the partition of the workload between processors reduces the speed-up one could expect.

We have then suggested examining the speed-up issue in terms of a representation of the intrinsic behavior of parallel program execution which we call the Activity Set Model. This model describes the set of simultaneously active parallel processes as a function of program virtual time. We show how the size of the Activity Set can be used to derive

bounds on the speed-up of the program as a function of the number of processors which are available to it.

References

Amdahl's law has been introduced in [1]. A more recent discussion of the problem and some pertinent modifications can be found in [2]. Other relevant references and extensions can be found in [5,6,7,8,9].

[1] Amdahl, G. M. "Validity of the Single-Processor Approach to Achieving Large-Scale Computing Capabilities," AFIPS Conference Proceedings 30, AFIPS Press, 483-485 (1967).

[2] Gustafson, J. L. "Reevaluating Amdahl's Law," Communications of the ACM. 31, 5, 532-533 (1988).

[3] Cosnard, M., Robert, Y., Tourancheau, B. "Evaluating Speed-ups on Distributed Memory Architectures," to appear in Parallel Computing.

[4] Gelenbe, E., Nelson, R., Philips, T., Tantawi, A. "Asymptotic Processing Time of a Model of Parallel Computation," Proceedings National Computer Conference USA, November 1986.

[5] Gustafson, J. L. "The Scaled-Sized Model: A Revision of Amdahl's Law," ICS Supercomputing '88. L. P. Kartashev and S. I. Kartashev eds., International Supercomputing Institute Inc., II, 130-133 (1988).

[6] Gustafson, J. L., Hawkinson, S., Scott, K. "The Architecture of a Homogeneous Vector Supercomputer," Proceedings of ICCP 86, IEEE Computer Society Press, 649-652 (1986).

[7] Moler, C. "Matrix Computations on Distributed Memory Multiprocessors," Hypercube Multiprocessors 1986, M. T. Heath ed., SIAM Editions, 161-180 (1986).

[8] Robert, Y., Tourancheau, B. "LU and QR Factorization on the FPS T Series Hypercube," CONPAR 88, Manchester, U.K., September 1988.

[9] Saad, Y. "Gaussian Elimination on Hypercubes," Parallel Algorithms and Architectures, M. Cosnard et al. eds., North-Holland, 5-17 (1986).