AN ANALYSIS OF MULTIPROCESSING SPEEDUP
WITH EMPHASIS ON
THE EFFECT OF SCHEDULING METHODS

Jing-Jang Hwang
National Chiao-Tung University

Yuan-Chieh Chow
University of Florida

Frank D. Anger
Florida Institute of Technology

Abstract

Speedup as a performance measure has become increasingly central with the growing use of multiple processor computing systems and the concomitant interest in parallel and distributed algorithms. This paper reexamines the speedup issue for message-passing multiprocessors and computer networks in which interprocessor communication overhead is considered undesirable but significant. A unified model of speedup is developed for analyzing the system in terms of three major factors: communication overhead, scheduling, and the application algorithm. By so doing, the impact on overall system performance of the interprocessor communication overhead and its interaction with different scheduling methods can be quantitatively assessed. Whereas most authors treat speedup as a measure of improved algorithms or improved systems, the model presented integrates the effects of these factors and that of scheduling. Two concepts, hidden overhead and efficiency loss, are also introduced to clarify the effect, and hence the significance, of the scheduling factor. The concepts are illustrated with examples using two scheduling methods designed for use in systems with significant communication overhead.

1. Introduction

Multiprocessing has long been regarded as a primary method of increasing the speed of computing. It is, of course, common wisdom that several processors work faster on a problem than a single processor works alone. Among hundreds of systems which have been proposed and experimented with, a large number of recent interest can be classified as multiple processor systems, abbreviated as MPSs. An MPS, also called a multiprocessor in the literature, is defined as a multiprocessing system in which all processors have their own local memory, execute instructions asynchronously, and communicate with one another through message transfer. This class of computing systems includes message-passing multiprocessors as well as computer networks. The underlying concept of loosely coupled architecture in such systems leads to the systems' strengths in simplicity, availability, adaptability, and scalability. However, interprocessor overhead in MPSs is generally higher than that in tightly coupled systems and is undesirable when pursuing the speed of supercomputing.

Is the interprocessor communication overhead in MPSs really significant? When does it become significant? How much does it contribute to the degradation of multiprocessing speed? Simple answers to these questions should not be expected since the speed of multiprocessing involves, indeed, all aspects of computing—hardware, system architectures, system software, selection of applications, and algorithms for solving problems. Interprocessor communication overhead is only one of many interdependent factors.

2. Analysis of Speedup

While human beings are pursuing the power of supercomputing, "speedup" is the most salient criterion for evaluating the performance of multiprocessing. Speedup has long been used as a performance measurement in both theoretical analysis and experimental studies on multiprocessing [1, 2, 5, 8, 11]. It is roughly defined as the number of times faster than a single-processor system an n-processor system runs a given problem. Since it involves hardware, software, and algorithm factors, a more precise definition and in-depth analysis is needed.

It is clear that speedup is a function of at least two major variables: the algorithm being used and the system on which it is run. Often the question of the "mapping" of an algorithm onto an architecture is thought of as part of the algorithm itself, while the sequencing of the tasks is treated as part of the system. Those two related aspects of the execution of an algorithm are considered as part of scheduling in this work, and it is argued that the speedup, S, obtained through multiprocessing is a function of three important variables:

\[ S = f(\text{ALGORITHM, SYSTEM, SCHEDULE}) \]
It is understood from the definition given that if the system has only one processor, then $S$ is less than or equal to 1, with $S = 1$ corresponding to the optimal sequential algorithm. The role of the third variable—SCHEDULE—often remains undefined in discussions of multiprocessing speedup.

In order to recognize the different factors affecting speedup and to analyze their influence, three different processing times will be considered for a given problem:

**OSPT** = "optimal sequential processing time"
the best time achievable on a single processor using the best sequential algorithm.

**CPT** = "concurrent processing time"
the actual time achieved on an n-processor system with the concurrent algorithm and specific scheduling method being considered. (The scheduling method being considered is usually non-optimal.)

**OCPT\_ideal**
= "optimal concurrent processing time on an ideal system"
= the best time achievable with the concurrent algorithm being considered on an ideal n-processor system and scheduled by an optimal scheduling policy.

The subscript \_ideal in the term OCPT\_ideal stands for an ideal system, which is an overhead-free system; i.e., a system with no interprocessor communication time. Now, in terms of these theoretical and actual processing times, speedup as defined in the first paragraph of this section, can be written as

$$S = \frac{\text{OSPT}}{\text{CPT}} = \frac{\text{OSPT}_{\text{ideal}}}{\text{CPT}_{\text{ideal}}} \times \frac{\text{OCPT}_{\text{ideal}}}{\text{CPT}} = S_x S_y.$$  

Here, $S_x$ represents the "ideal" speedup obtained by using a multiple processor system over the best sequential time, and $S_y$ gives the degradation of the system due to the actual implementation compared to an ideal system.

In order to distinguish the role of algorithm, system, and scheduling, the formula for speedup is further refined.

$$S_y = \frac{\text{RC}}{\text{RP} \times n},$$  

where $\text{RP} = \frac{\sum T_i}{\text{OSPT}}$, $\text{RC} = \frac{\sum T_i}{\text{OCPT}_{\text{ideal}} \times n}$.

The factors have been rewritten in terms of a more interesting quantity, $\rho$, the "efficiency loss," which is defined as the ratio of the real system overhead due to all causes to the ideal optimal processing time. As such, $\rho \geq 0$ corresponding to the implementation as good as using optimal scheduling on the ideal system.

In the final formula,

$$S = \frac{\text{RC}}{\text{RP} \times 1 + \rho} \times n,$$  

the causes of the resulting speedup are clearly seen, and analyzing the actual or approximate values of these factors in a given case can give insight into where to concentrate effort if greater speedup is required. Of course, the factors are not at all independent, and simply improving one may make another change for the worse.

### 3. The Concept of Hidden Overhead

One of the interesting results of looking at the interdependence of the factors involved in speedup is the role of "hidden overhead" in the message-passing computer systems. The necessity of communication between separate processors with separate memories gives rise to processing delays, called communication overhead, while one task waits for input from another. Communication overhead can sometimes be eliminated by running communicating tasks on the same processor, and on other occasions it can be hidden by scheduling in such a way that messages are generally sent before the receiver is ready to execute. The existence of this phenomenon underlines the importance of the scheduling factor in the overall speedup of the system and is reflected in the efficiency loss term, $\rho$. The $\rho$ itself, however, cannot be easily broken down into a combination of $\rho_{\text{system}}$ and $\rho_{\text{tasking}}$. For example, if the best possible schedule on a given system has only hidden overhead, then this schedule is running as fast as it could on an ideal system; hence $\rho_{\text{system}}$ should be zero. At the same time, no other schedule does better on this system, so it appears that $\rho_{\text{tasking}}$ is also zero. There may be, nonetheless, a better schedule on an ideal system, indicating that $\rho$ is greater than zero. The opposite can also occur when there is communication overhead that is not hidden: the loss of efficiency may be attributed to both the system and the scheduling. This apparent anomaly arises from the fact that comparisons of two schedules on an ideal system gives different results from their comparison on the real system, and this difference corresponds to overhead produced by the interaction of schedule and system, not by either one alone.

To illustrate the complexity of the interdependence between system factors and the significance of hidden overhead, we consider the precedence graph of a hypothetical algorithm shown in Figure 1-a. Tasks are represented by circles with their processing times included inside the circles. The precedence relationships constitute a tree structure in this case. In addition, a positive integer, called internode communication, defined as $n(T_i, T_j)$, denotes the amount of information that must be passed to task $T_j$ on termination of task $T_i$.
observed in other types of systems as long as the scheduling degradation of speedup is completely caused by the scheduling policy and all communications are hidden.

Figure 1-c, meaning that the real system does not suffer from the communication delays. It can be said that, in this case, the apparent communication overhead is zero. However, reversing the sequence of tasks in the schedule does not affect the task sequence itself and the schedule length in this case. It is important to note that the total schedule length is exactly the same as that in Figure 1-c, meaning that the real system does as well as the ideal system with this sub-optimal schedule, and once again 

Finally, for this algorithm, similar scenarios can be observed in other types of systems as long as the scheduling policy remains the same. It can be said that, in this case, the degradation of speedup is completely caused by the scheduling policy and all communications are hidden.

4. ELS and ETF—Two Scheduling Methods

The phenomenon of hidden overhead indicates that the speedup degradation that might otherwise occur due to interprocessor communication may be covered up to some extent by the inefficiency of scheduling methods. To further analyze the speedup degradation due to these two interdependent factors—communication overhead and scheduling, we introduce two scheduling methods in this section.

Both methods apply to the same computing model, which is represented by a graph 

The first method, ELS (Extended List Scheduling), is a straightforward extension from the well established list scheduling method (LS) in classical multiprocessor scheduling theory [6]. It is a two-phase strategy. First it allocates tasks to processors by applying the LS method and if the underlying system was an ideal (overhead-free) system with the same number of processors. Secondly, it adds necessary communication delays to the schedule obtained in the first phase.

The second method, ETF (Earliest Task First), is an intelligent heuristic adopting the following greedy strategy: "The ready task which can be scheduled to start its execution at the earliest time is given the highest priority for the current scheduling decision." A high level version of Algorithm ETF is given in the appendix.

The worst-case performance of ELS and ETF is stated in Theorems 1 and 2 respectively. The proof of these results appears in Hwang [7].

Theorem 1. Any ELS schedule with length \( CPT_{ELS} \) on a system \( S = (n, t_{comm}) \) satisfies the following inequality:

\[
CPT_{ELS} \leq (2 - \frac{1}{n}) OCPT_{ideal} + t_{comm}\max_{\pi} \sum_{T_i, T_j} (T_i, T_j) ,
\]

where \( t_{comm} \) is the maximum of all communication parameters \( t_{comm}(P_i, P_j) \) taken over all pairs of parameters, and \( \Gamma \) is the set of tasks. Moreover, this bound is the best possible.

Theorem 2. Any ETF schedule with length \( CPT_{ETF} \) on a system \( S = (n, t_{comm}) \) satisfies:

\[
CPT_{ETF} \leq (2 - \frac{1}{n}) OCPT_{ideal} + \text{MaxChainComm} ,
\]

where \( \text{MaxChainComm} \) is the maximum sum of the form \( \sum_{T_i, T_j} (T_i, T_j) \), the sum taken along any chain of the system.

The bound shown in Theorem 1 indicates that, under ELS, all communication requirements may turn out to be real communications and yield real delays, even when the schedule length on the corresponding ideal system approaches the worst possible value \( (2 - \frac{1}{n}) OCPT_{ideal} \). In other words, it is not always true that more overhead is hidden when scheduling inefficiency is greater.
5. Decomposition of Efficiency Loss

Now we return to our discussion of speedup. According to our analysis before, the efficiency loss, $\rho$, can be expressed as a function of two variables—system and scheduling. Let $X$ represent a multiple processor system under investigation, $Y$ represent a scheduling policy which is extended for system $X$ from a scheduling policy $Y$ on the corresponding ideal system. (As an example, $Y$ and $Y'$ can be considered as a list scheduling policy defined by a priority list $L$ and its extended policy respectively.) Then

$$\rho = f(X, Y') = \frac{CPT(X, Y') - OCPT_{ideal}}{OCPT_{ideal}}.$$

where $OCPT_{ideal}$ is considered as a constant independent of both system variable $X$ and scheduling variable $Y'$.

We can decompose $\rho$ as the sum of two terms as follows:

$$\rho = \frac{CPT(X, Y') - OCPT_{ideal}}{OCPT_{ideal}} = \frac{CPT(X, Y) - CPT_{ideal}(Y) + CPT_{ideal}(Y) - OCPT_{ideal}}{OCPT_{ideal}} = \rho_{sys} + \rho_{sched},$$

where

$$\rho_{sys} = \frac{CPT(X, Y) - CPT_{ideal}(Y)}{OCPT_{ideal}},$$

$$\rho_{sched} = \frac{CPT_{ideal}(Y) - OCPT_{ideal}}{OCPT_{ideal}},$$

and $CPT_{ideal}(Y)$ is the concurrent processing time of the given algorithm running on the ideal system when the scheduling policy $Y$ is applied.

We can also reverse the decomposition process shown above by analyzing the efficiency loss due to non-optimality of scheduling before analyzing the effect of non-ideal systems. To do so, the intermediate variable $OCPT(X)$, meaning optimal concurrent processing time on system $X$, is introduced instead of $CPT_{ideal}(Y)$. In addition, the scheduling policy under consideration could be either an extended policy as ELS or an entirely new policy such as ETF, and hence is simply denoted as $Z$. Then

$$\rho = f(X, Z) = \frac{CPT(X, Z) - OCPT_{ideal}}{OCPT_{ideal}} = \frac{CPT(X, Z) - OCPT_{ideal}}{OCPT_{ideal}} + \frac{OCPT_{ideal} - OCPT_{ideal}}{OCPT_{ideal}} = \rho_{sched} + \rho_{sys},$$

where

$$\rho_{sys} = \frac{CPT(X, Z) - OCPT_{ideal}}{OCPT_{ideal}},$$

$$\rho_{sched} = \frac{OCPT_{ideal} - OCPT_{ideal}}{OCPT_{ideal}}.$$

The two decomposition processes can be summarized in Figure 2. In this diagram, $\rho$ refers to the efficiency loss due to change of system and schedule both at once, while $\rho_{sys}$ and $\rho_{sched}$ refer to change in system only under the condition of maintaining the same scheduling policy, and $\rho_{sys}$ and $\rho_{sched}$ refer to change in scheduling only in the same system. The $\rho_{sys}$ and $\rho_{sched}$ refers to second change in the analysis process. This diagram demonstrates the complication of efficiency loss of multiprocessing in a real system environment by showing several parameters—$\rho$, $\rho_{sys}$, $\rho_{sched}$, $\rho_{sys}$, $\rho_{sched}$, and $\rho_{sys}$—in a single diagram. In the next section, we shall give numerical examples for the calculations of these parameters.

6. Efficiency Loss of ELS and ETF

The difficulty of the calculation of any of the efficiency loss parameters centers in the difficulty of obtaining two schedules—the optimal schedule of concurrent processes on an ideal system and the optimal schedule of concurrent processes on a non-ideal system. The lengths of these two optimal schedules are abbreviated as $OCPT_{ideal}$ and $OCPT(X)$ respectively in the several formulæ on efficiency loss. For $P$ (polynomial time) problems with restricted assumptions, both $OCPT_{ideal}$ and $OCPT(X)$ can be attained, and then each of the efficiency loss parameters can be calculated accordingly. For general problems, algorithms with polynomial running times for the calculations of these two variables are not available now and are also unlikely to be found in the future since the problems involved are NP-complete. Nevertheless, Theorems 1 and 2 allow us to establish upper bounds for efficiency loss of both ELS and ETF on real systems.

In the following propositions, a system $S = S(n, t_{com})$ or its corresponding ideal system $S = S(n, 0)$ are considered as the underlying system. The total number of processors in the system, $n$, is considered as a constant. $T$ is the set of tasks to be scheduled.

Proposition 1. For any list scheduling policy applied on an ideal system, the efficiency loss parameter $\rho_{sys}$ is bounded above by $1-(1/n)$. Moreover, this bound is the best possible.

Proof: It immediately follows from Graham's theorem on list scheduling [8] and the definition of $\rho_{sys}$.

Proposition 2. The efficiency loss due to interprocessor communication overhead alone, assuming that an LS policy and its associated ELS policy are applied to the ideal system and the real system respectively, is bounded by $n$ times the ratio of total maximum communication requirement over total computation requirement. In symbols,

$$\rho_{sys} = n \frac{\text{TotalComm}_{max}}{\text{TotalComp}},$$

where

$$\text{TotalComm}_{max} = \sum_{T \in T} \sum_{T' \in T'} \eta(T, T'),$$

$$\text{TotalComp} = \sum_{T \in T} \text{Comp}(T).$$

Furthermore, this bound is the best possible.

Proof: According to the definition of ELS, the total communication delay which makes the ELS schedule longer than the corresponding LS schedule is always less than or equal to the
This inequality, together with the fact that the total maximum communication requirements of the whole program graph, that is,

\[ CPT_{ELS} - CPT_{Ideal}(LS) \leq \text{TotalComm}_{\text{max}}. \]

This inequality, together with the fact that \( \text{TotalComm} \leq n \times CPT_{\text{Ideal}} \), yields the bound in this theorem. The proof is straightforward using the definitions. The example in Figure 3 shows that the bound is asymptotically achievable as \( n \) approaches zero. \( \Box \)

**Proposition 3.** The combined efficiency loss due to both the ELS scheduling and the interprocessor communication overhead is bounded by the summation of the upper bounds in above two theorems, in symbols,

\[ \rho \leq (1 - \frac{1}{n}) + n \times \frac{\text{TotalComm}_{\text{max}}}{\text{TotalComp}}. \]

Proof: This result follows from \( \rho = \rho_{\text{cpu}} + \rho_{\text{sched}}. \) \( \Box \)

**Proposition 4.** The combined efficiency loss due to both the ETF scheduling and the interprocessor communication overhead satisfies the following inequality:

\[ \rho \leq (1 - 1/n) + \Theta_{\text{path}}. \]

The parameter \( \Theta_{\text{path}} = \frac{\text{MaxChainComm}}{\text{MaxChainComp}} \) is called path communication/computation ratio and is defined by:

\[ \Theta_{\text{path}} = \frac{\text{MaxChainComm}}{\text{MaxChainComp}} \]

where the maximum chain communication, \( \text{MaxChainComm} \), has been defined in Theorem 2 and the maximum chain computation, \( \text{MaxChainComp} \), is defined in a similar way.

Proof: Due to the precedence relation in a chain, we have

\[ \text{OCPT}_{\text{Ideal}} \geq \sum_{i=1}^{n} \mu_{i} \]

for any chain \((T_{1}, ..., T_{n})\), where \( \mu \) is the function of task computation times. By applying this inequality, this theorem can be easily derived from Theorem 2 of Section 4 and the definitions. \( \Box \)

It should be noted here that the chains taken to calculate \( \text{MaxChainComp} \) and \( \text{MaxChainComm} \) could be different. This bound is obviously better than that stated in Proposition 3 since the multiplier \( n \) disappears from the expression. We conclude this section by calculating values of all parameters of efficiency loss with one example. For this example, both \( \text{OCPT}_{\text{Ideal}} \) and \( \text{OCPT}(X) \) can be obtained by exhaustively comparing all feasible schedules.

Example: The weighted DAG shown in Figure 4-a is to be scheduled on a 3-processor system illustrated in Figure 4-b. An LS schedule (on the corresponding ideal system) and its corresponding ELS schedule are shown in Figure 4-c. An ETF schedule is given in Figure 4-d. For this example, we get \( \text{OCPT}_{\text{Ideal}} = 16 \) and \( \text{OCPT}(X) = 18 \). All parameter values for both ELS and ETF are shown in Table 1. The optimality of ETF in this example just happens incidentally; ETF does not guarantee optimality in general. The values of \( \rho_{\text{cpu}} \) and \( \rho_{\text{sched}} \) for ETF are not shown in the table since our first decomposition scheme cannot be applied to a method such as ETF which is designed to deal with communication delays.

<table>
<thead>
<tr>
<th>parameter</th>
<th>ELS</th>
<th>ETF</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \rho )</td>
<td>0.75</td>
<td>0.125</td>
</tr>
<tr>
<td>( \rho_{\text{cpu}} )</td>
<td>0.75</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>( \rho_{\text{sched}} )</td>
<td>0</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>( \rho_{\text{sent}} )</td>
<td>0.625</td>
<td>0</td>
</tr>
<tr>
<td>( \rho_{\text{recv}} )</td>
<td>0.125</td>
<td>0.125</td>
</tr>
</tbody>
</table>

**Table 1**

7. Conclusion

The question of the real impact of interprocessor communication overhead on multiprocessing speed has been the central theme of the analytical investigation performed in the foregoing sections. New insight into this issue has been gained through showing scheduling to be a key factor in determining the multiprocessor speedup in an MPS. The advantage of ETF over ELS, clearly illustrated by the comparison of Propositions 3 and 4, shows that a good scheduling method may sharply reduce the impact of communication overhead. The upper bound of the efficiency loss \( \rho \) established in Proposition 4 for ETF is obviously less than 1 + \( \Theta_{\text{path}} \) which is independent of the size of the system. This nice property is made possible by the intelligence built in the scheduling method as well as the system’s property of contention-free communication. It is clear that the contention-free property does not guarantee the same bound if a good method like ETF is not applied.

The concept of hidden overhead and the formulas on efficiency loss point out new directions for studying multiprocessing speedup. The speedup model presented brings together a number of algorithm and system characteristics which can be useful in determining the speedup achieved by a system. If parallelism is to be exploited on loosely-coupled systems, the interaction of the principal parameters of the speedup model must be recognized and studied for the systems and applications in question.

REFERENCES


---

(a) A Weighted DAG

(b) Optimal Schedule

---

(c) CP-schedule on a 4-Processor Ideal System

(d) A 4-Processor System Model

---

Figure 1. An Example for Showing the Concept of Hidden Communication

Figure 2. Decomposition of Efficiency Loss

---

1 The scale in Figure 1(b) is different from Figures 1(c) and 1(e).
Figure 3. A Worst-case Example for Proposition 2

Algorithm ETF (high-level version):
1. Initialization.
2. Repeat the following steps until all tasks have been scheduled.
   If no unscheduled ready tasks available then do
   begin
   2.1 Let the scheduling process proceed (move CM forward).
   2.2 Update related information.
   end
   else do
   begin
   2.3 Find a task which can be started at an earliest time. Let $T$ be the selected task. Let TestableStart($T$) be the earliest starting time. Let TestableAlloc($T$) be the processor on which $T$ can be started at TestableStart($T$).
   2.4 If TestableStart($T$) $\leq$ NM then do
   begin
   2.4.1 Schedule $T$.
   2.4.2 Update related information.
   end
   else do
   begin
   2.4.3 Postpone the scheduling decision and let the scheduling process proceed.
   2.4.4 Update related information.
   end
   end.