An Execution-Time Estimation Model for Heterogeneous Clusters

Yoshinori Kishimoto and Shuichi Ichikawa

Department of Knowledge-based Information Engineering
Toyohashi University of Technology
1-1 Hibarigaoka, Tempaku, Toyohashi 441-8580, Japan
yoshy@ich.tutkie.tut.ac.jp, ichikawa@tutkie.tut.ac.jp

Abstract

Heterogeneous clusters are flexible and cost-effective, but entail intrinsic difficulties in optimization. Although it is simple to invoke multiple processes on fast processing elements (PEs) to alleviate load imbalance, the optimum process allocation is not so obvious. Communication time is another problem. It is sometimes better to exclude slow PEs to avoid performance degradation, but it is generally difficult to find the optimal PE configuration. In this study, the execution time is first modeled from the measurement results of various configurations. Then, the derived model is used to estimate the optimal PE configuration and process allocation. We implemented the models from HPL (High Performance Linpack benchmark) of N = 400–6400, and estimated the optimal configuration for N = 3200–9600. The execution time of the estimated optimal configuration was 0%–3.6% longer than the actual optimal configuration. The models derived from N = 1600–6400 were also constructed, and their errors were 0%–4.3% for N = 1600–9600.

1 Introduction

It is reasonable to enhance the performance of an existing PC cluster by adding the latest high-performance processors. The resulting cluster becomes heterogeneous, consisting of a range of processing elements (PEs) from fast to slow. However, it is well known that heterogeneous clusters inherently entail difficulties in optimization and suffer from load imbalance.

Although it is simple to invoke multiple processes on fast PEs to alleviate load imbalance, this approach (*multiprocessing*) has some drawbacks. The first problem is the overhead to execute multiple processes on the same processor. Another problem is that the ratio of PE performance is not always an integer, while the number of processes invariably is. Thus, the best process allocation among PEs is far from obvious.

Communication time is also very important. It is not always preferable to use all PEs, since superfluous communications can make the total execution time longer. In particular, a slow PE can create a performance bottleneck in computation and communication. The total performance can be improved by excluding slow PEs and instead using the best subset of PEs. However, it is generally difficult to find the best subset of available PEs, i.e., the best PE configuration of a heterogeneous cluster.

Many applications for parallel computers or homogeneous clusters are written to distribute workloads equally on PEs. Although it is desirable to rewrite the application for heterogeneous clusters, it requires much time and effort to polish it up for a heterogeneous environment. Moreover, the effort must be repeated for each application.

The purpose of this study is to execute conventional parallel applications efficiently on heterogeneous clusters without rewriting them. Our study adopts a multiprocessing approach, providing an effective way to estimate the best PE configuration and process allocation based on an execution-time model of the application. Our method does not aim to extract the maximum performance from a heterogeneous cluster, but rather to offer an easy and simple way to accelerate a wide range of conventional parallel applications in heterogeneous clusters. We examine HPL (High Performance Linpack benchmark) [8] as a sample application in this study. However, our approach is not limited to HPL but it is widely applicable to many other applications.

Section 2 introduces related works, and then briefly summarizes the background of this work. In Section 3, the computation and communication times are modeled from the measurement results of various configurations. The derived models are then used to estimate the optimal PE configuration and process allocation. The evaluation results will be also found in Section 4.

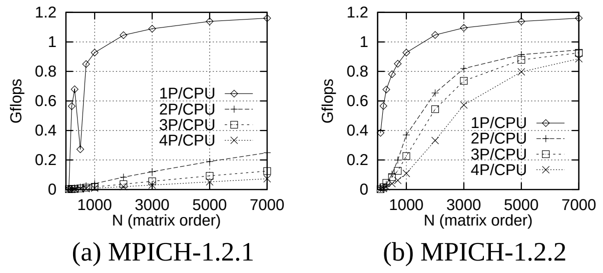

Figure 1. Multiprocessing performance of an Athlon using two MPICH versions

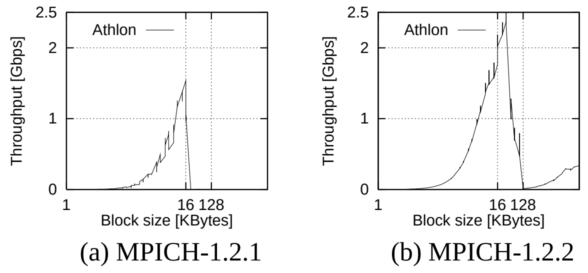

Figure 2. Communication throughput of two MPICH versions

2 Background and Related Works

Though a block cyclic distribution is very popular in the load-balancing of matrix-matrix multiplication and LU decomposition, such a distribution in its original form is not suited to a heterogeneous environment. Therefore, many researchers have studied alternative load-balancing schemes. For example, Kalinov and Lastovetsky [7] presented a "heterogeneous block cyclic distribution" for the Cholesky factorization of square dense matrices. Beaumont et al. [1] presented a "2D heterogeneous grid allocation" for the heterogeneous cluster ScaLAPACK [2], while Sasou et al. [9] modified the HPL source code for heterogeneous clusters to dispatch multiple panels to fast PEs in LU decomposition.

In all above studies, the original source codes were rewritten for heterogeneous computing environments, while the present study aims to find a way to execute the existing applications as they exist in heterogeneous clusters. In preceding studies, computational workloads are distributed according to PE performance, but communication is not given enough attention. However, our method considers communication time quantitatively in the optimization. The abovementioned studies use all PEs but lack a viewpoint from which to select the best set of processors among those available. In contrast, our method estimates the PE configuration most likely to yield the optimal execution time.

Sasou et al. [9] found the performance of the multiprocessing approach to be rather poor. They speculated that the communications between processes on the same PE had been obstructed by the process scheduling of OS. However, this should not pose a serious obstruction if the communication is adequately buffered and managed. We replicated the situation, examined the problem, and found that the communication library strongly affects performance.

Figure 1 illustrates the performance of HPL on a single Athlon 1.33 GHz processor. The X-axis is the size N of HPL, and the Y-axis is the total performance reported by HPL. In the figure, "nP/CPU" means that n processes were simultaneously executed on that processor. Owing to the overhead of multiprocessing, the performance decreases as n increases. We used MPICH [5] for communication, and compared two versions, i.e., MPICH 1.2.1 and 1.2.2. With MPICH-1.2.1, the multiprocessing approach shows a drastic performance degradation as Sasou reported [9] (Fig. 1 (a)). However, the performance loss is much smaller with MPICH-1.2.2 (Fig. 1 (b)).

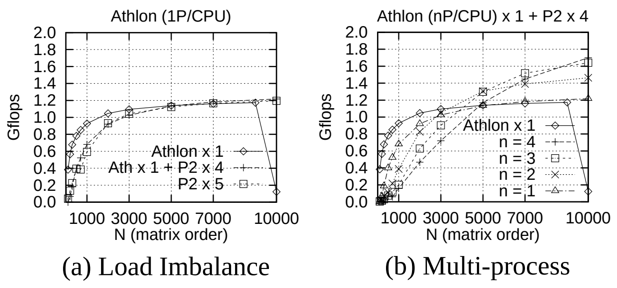

Figure 3. HPL performance of a heterogeneous cluster of various configurations

Figure 2 displays the communication throughput between two processes on the same Athlon processor measured using NetPIPE [6]. MPICH-1.2.2 shows a much better throughput than 1.2.1, and this fact explains the differences in Figure 1.

Figure 3 summarizes the HPL performances of various subsets of a heterogeneous cluster, which consists of an Athlon 1.33 GHz node, four Pentium-II 400 MHz dual-processor nodes, and a 1000base-SX network. The details of this cluster will be found in Table 1.

In Figure 3 (a), "Athlon x 1" designates the performance of a single Athlon processor. Though the peak performance of an Athlon is about 1.2 Gflops, its performance degrades in case $N \geq 10000$ for the shortage of main memory. "P2 x 5" designates the performance of five Pentium-II processors, which is almost the same performance as for a single Athlon. As the total memory capacity of five Pentium-II nodes is much larger than that of an Athlon, the performance does not degrade even for $N = 10000$ in this configuration. "Ath x 1 + P2 x 4" designates the performance of a heterogeneous configuration, which consists of an Athlon and four Pentium-II processors. The performance of this configuration is practically the same as "P2 x 5". An Athlon 1.33 GHz is about 5 times faster than a Pentium-II 400 MHz, and thus the peak performance of "Ath x 1 + P2 x 4" should be nearly twice that of "P2 x 5". However, the load imbalance degrades the performance of this heterogeneous configuration. Since the computational workload of HPL is equally distributed, the Athlon must wait for synchronization after finishing its computation. To alleviate this load imbalance, we attempt to invoke multiple processes on fast PEs.

Figure 3 (b) shows that the HPL performance of the heterogeneous configuration (Ath x 1 + P2 x 4) can be improved by adopting a multiprocessing approach. In this figure, "n = 2" means that two processes are executed on an Athlon, while each Pentium-II invokes a single process (6 processes in total). "Athlon x 1" designates the performance of a single Athlon, which is shown for a contrast.

From Figure 3 (b), it is readily seen that the load imbalance can be dissolved by multiprocessing. In the case of $N = 10000$, 77% of the peak performance (2.2 Gflops) can be utilized by executing four processes on an Athlon (n = 4). On the other hand, for $N < 5000$, "n = 4" demonstrates lower performance than "n = 1", owing to the multiprocessing overhead. For better performance, the choise should be "Athlon x 1" for $N \leq 3000$, "n = 2" for $3000 < N \leq 5000$, "n = 3" for $5000 < N \leq 8000$, and "n = 4" for $8000 < N \leq 10000$, judging from Figure 3 (b).

It is no easy task to find the best way of execution in general cases. The next section discusses how to optimize the multiprocessing execution in a heterogeneous cluster.

3 Construction of Estimation Model

3.1 Assumptions

To optimize the multiprocessing approach for heterogeneous clusters, it is necessary (1) to select the optimal subset of PEs and (2) to determine the optimal number of processes on each PE. This problem is modeled as a combinatorial optimization problem to minimize the total execution time, where one must construct an objective function that estimates the total execution time from the given PE set and the given number of processes.

In this section, we construct the estimation model from some small HPL trials. As the orders of computation and communication are derived from the algorithm, we can assume the approximation formula of the total execution time. We then extract constant factors from measurement results by the least-squares method.

This kind of modeling technique is very common in various applications. For example, in MOS transistor modeling for circuit simulations [3], many fitting parameters are introduced from the measurement results into analytical models based on device physics. Although such a semi-empirical model is not always elegant, it may contain several factors that were unknown or neglected in the modeling process. In our case, the semi-empirical model may include the miscellaneous overhead caused by cache misses, communication buffer management, etc.

Let N be the size of HPL, M_i the number of processes on PE_i, and $P = \sum M_i$ the total number of processes in the cluster. The execution time T_i of PE_i consists of the computation time Ta_i and the communication time Tc_i. The purpose of the model is to estimate T_i from N, P, and M_i. Clearly, T_i depends on the process grid. Though we examine only the case of a 1-by-P process grid (one-dimensional block cyclic distribution) in this study, our scheme is universally applicable to any other process grid.

We make the following assumptions here to simplify our model:

- Ignore the overlap of computation and communication, and assume $T_i = Ta_i + Tc_i$.
- Ignore the network topology, and assume that the network is homogeneous.
- Assume that the communication time is independent of the sender/receiver.
- Apply the same M_i to PEs of the same specification.

Such simplification may possibly lead to a slight discrepancy with reality, but such a discrepancy proved to be modest in our study. The evaluation results will be found in Section 4. If the discrepancy was unacceptably large, we had to rebuild our models on other assumptions.

3.2 N-T Model

To approximate Ta_i and Tc_i separately, we have to measure the execution time item by item. Figure 4 summarizes the items included in the total execution time of HPL. The term $rfact$ represents the time for recursive panel factorization, which includes $pfact$ (panel factorization) and $mxswp$ (max row swap communication). The term $update$ is the time required for the update phase, which includes $laswp$ (row interchange communication). The term $uptrsv$ indicates the time required for backward substitution, and $bcast$ is broadcast communication. All these items (except for $bcast$) can be measured by defining

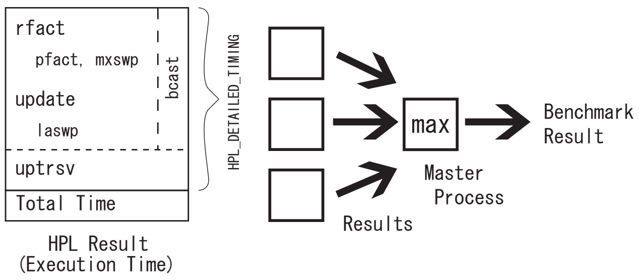

Figure 4. Detailed Timing Measurement

HPL_DETAILED_TIMING in compiling HPL. To measure $bcast$, we had to add some lines to the original source code.

Finally, the computation time Ta_i and the communication time Tc_i are estimated by the following equations:

$$\begin{aligned} Ta_i &= (rfact - mxswp) + \\ &\quad (update - laswp) + uptrsv \\ Tc_i &= mxswp + laswp + bcast \end{aligned}$$

The order of computation is determined from the algorithm, and is summarized by the following equations [9]:

$$\begin{aligned} rfact &= \frac{3}{2} \cdot N^2 + O(N) \\ update &= \frac{2N^3}{3P} + \frac{P+1}{P} \cdot O(N^2) + O(N) \\ uptrsv &= \frac{1}{P} \cdot O(N^2) \end{aligned}$$

Thus, Ta_i is estimated as $O(N^3)$. According to actual measurement results, $update$ is approximately 100 times more than both $uptrsv$ and $rfact$ for $N = 9600$. Since the equation for $update$ can include $rfact$ and $uptrsv$, Ta_i is estimated based on $update$ in the following discussion.

The order of communication is estimated and summarized as follows.

$$\begin{aligned} mxswp &= O(1) \\ laswp &= \frac{1}{P} \cdot O(N^2) \\ bcast &= (P-1) \cdot O(N^2) \end{aligned}$$

Consequently, Tc_i is estimated as $O(N^2)$.

The following equations approximate Ta_i and Tc_i for a given set of P and M_i. The constant factors k_0–k_6 are determined from measurement results by the least-squares method. This model is called *N-T model* in the following discussions.

$$\begin{aligned} Ta_i(N)|_{P,M_i} &= k_0N^3 + k_1N^2 + k_2N + k_3 \\ Tc_i(N)|_{P,M_i} &= k_4N^2 + k_5N + k_6 \end{aligned}$$

As these equations are linear functions of k_0–k_6, the coefficients k_0–k_6 can be extracted by using the `gsl_multifit_linear()` function of GSL (GNU Scientific Library) [4]. We have to measure $Ta_i(N)$ and $Tc_i(N)$ of (at least) four different N for each configuration to extract coefficients, because $Ta_i(N)$ includes four coefficients and $Tc_i(N)$ includes three coefficients.

3.3 P-T Model

Since it is not sophisticated to manage many N-T models for each set of P and M_i, we tried to integrate N-T models for the same set of M_i into a new model, which includes P as a variable (*P-T model*). P-T models are represented by the following equations, where the constant factors k_7–k_{11} have to be extracted from the corresponding N-T models by the least-squares method.

$$\begin{aligned} Ta_i(N,P)|_{M_i} &= k_7 \cdot \frac{Ta_i(N)|_{P,M_i}}{P} + k_8 \\ Tc_i(N,P)|_{M_i} &= k_9 \cdot P \cdot Tc_i(N)|_{P,M_i} + \\ &\quad k_{10} \cdot \frac{1}{P} \cdot Tc_i(N)|_{P,M_i} + k_{11} \end{aligned}$$

As these equations are linear functions of k_7–k_{11}, these coefficients are also extracted by using the `gsl_multifit_linear()` function [4]. We have to measure (at least) three different P for each configuration to extract coefficients, because $Ta_i(N,P)$ includes two coefficients and $Tc_i(N,P)$ includes three coefficients.

3.4 Binning

N-T models and P-T models are selectively used according to the circumstances. Figure 5 summarizes the selection of models. The “X” in Figure 5 means that there are no such cases.[1]

When HPL is executed on a single PE_i (i.e., $P = M_i$), *no* inter-PE communication emerges. This case is distinct from the execution with multiple processors (i.e., $P > M_i$). It is illogical and imprecise to handle these two cases equally. Thus, the N-T model is used for $P = M_i$, while the P-T model is used for $P > M_i$. Such selective use of models is called “binning” in transistor modeling [3] and is used to switch models where the dominant physical process is different.

We can also select models according to the data size. As the memory hierarchy is closely related to performance, Ta_i and Tc_i can show disjunct behavior depending on the allocated data size. For example, it is well known that the performance is much degraded when cache-misses frequently occur. Another example is found in Figure 3 (a),

[1] Keep in mind that $P = \sum M_i \geq \forall M_i$.

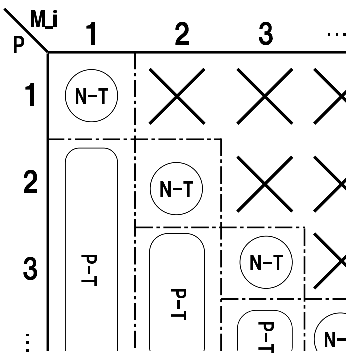

Figure 5. Binning

where the performance of a single Athlon was severely degraded by the shortage of main memory (at $N = 10000$). Since the memory requirement for each node can be predetermined from N and P, it is possible to select an adequate estimation equation according to the required memory size. The model of Ta_i and Tc_i is not necessarily continuous nor differentiable, but it could be a piecewise function.

3.5 Model Composition

In this study, we decided to construct models from the measurement results of a *homogeneous* cluster of that kind. For example, we extract the model parameters for Pentium-II by using eight Pentium-II processors in our heterogeneous cluster, leaving Athlon unused. Though there are many other relevant possibilities in this regard, we leave them for future studies.

N-T models and P-T models have to be built for each processor. It is both difficult and impractical to build separate models for every processor, of which there are many. In such cases, it is far more practical to build some models from those that are actually derived from measurement results. This technique is called *model composition* in this study.

As stated in Section 3.3, at least three sets of measurements must be done on P to build P-T models. If a homogeneous cluster is assumed for parameter extraction, we need three processors of the same kind. If there are not enough processors of that kind, it is impossible to build P-T models from measurements for that kind of processor. This situation often occurs in a heterogeneous cluster.

In this study, we only have one Athlon processor in our heterogeneous cluster, and thus we can not extract the P-T model parameters for Athlon. Therefore, in the evaluation of Section 4, the P-T models of Athlon are composed from the P-T models of Pentium-II that are constructed from measurements.

Table 3. HPL execution time for measurements (Basic model)

Size N	Athlon [sec.]	Pentium-II [sec.]
400	3.9	96.7
600	7.4	130.1
800	10.8	178.8
1200	20.5	305.2
1600	37.4	508.5
2400	97.5	1117.3
3200	197.2	2042.2
4800	566.0	5360.0
6400	1239.5	10950.3
Total	2180.2	20689.1

4 Model Evaluation

4.1 Basic Model

In this section, the estimation models are built and evaluated for a heterogeneous cluster. The models constructed in this section are called the *Basic model* in the following discussion. The specifications of the evaluation platform are listed in Table 1. As each Pentium-II node includes two processors, 8 Pentium-II processors are available in 4 nodes (Node 2–Node 5). All nodes have both 1000base-SX and 100base-TX interfaces, but only the 100base-TX is used in the following measurements.

In the following discussion, P_1 and P_2 designate the respective number of Athlon and Pentium-II processors used for the HPL. M_1 and M_2 represent the number of processes invoked on Athlon and Pentium-II, respectively.

First, measurements for the model construction were made for every combination of parameters, as shown in the “Model Construction” in Table 2. Since an Athlon 1.33 GHz is about 4 times faster than a Pentium-II 400 MHz, the range of M_1 was set to 1, ..., 6. The total number of combinations is 6 configurations of Athlon and 48 of Pentium-II for 9 sets of N; i.e., $(6 + 48) \times 9 = 486$ sets. Table 3 summarizes the HPL execution time for measurements item by item. The total time for measurements was 22869 seconds (about 6 hours).

We constructed the models for 54 configurations using the results of these measurements. This step takes as little as 0.69 millisecond on an AthlonXP 2600+ machine with Windows XP. The P-T models of Athlon were composed from the P-T models of Pentium-II, because it is impossible to extract the parameters from a single Athlon as stated in Section 3.5. In the present study, we simply scaled the Ta and Tc of Pentium-II P-T models by constant factors 0.27 and 0.85 to derive the Ta and Tc of Athlon P-T models,

Table 1. HPL execution environment

Node 1	AMD Athlon 1.33 GHz, Main memory 768 MB
Node 2–5	Intel Pentium-II 400 MHz (dual processor), Main memory 768 MB
Network	1000base-SX (NetGear GA-620), 100base-TX (Intel Pro100+)
OS	RedHat Linux7.0J (kernel 2.4.2)
Compiler, options	gcc 2.96, -DHPL_DETAILED_TIMING -fomit-frame-pointer -O3 -funroll-loops -W -Wall
Libraries	MPICH–1.2.5, ATLAS 3.2.1

Table 2. Cluster configuration parameters for Basic model

Model Construction	N = 400, 600, 800, 1200, 1600, 2400, 3200, 4800, 6400, Athlon (P_1: 1, M_1: 1, ..., 6), Pentium-II (P_2: 1, ..., 8, M_2: 1, ..., 6)
Model Evaluation	N = 3200, 4800, 6400, 8000, 9600, Athlon (P_1: 0, 1, M_1: 1, ..., 6), Pentium-II (P_2: 0, ..., 8, M_2: 1)

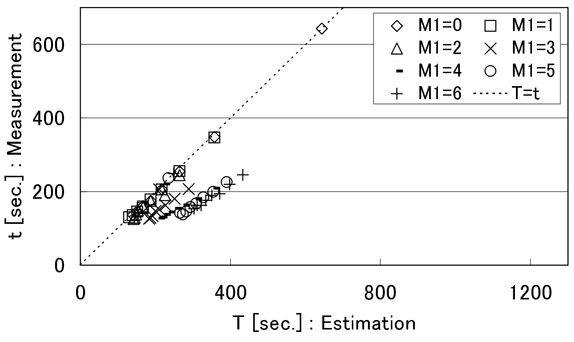

Figure 6. Correlation between original estimations and measurements (N = 6400)

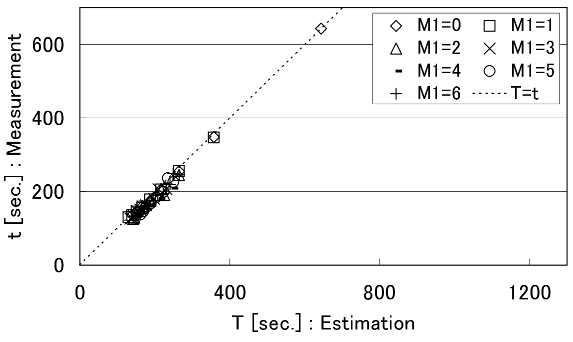

Figure 7. Correlation between estimations and measurements after adjustment (N = 6400)

respectively.

Next, we applied this Basic model and estimated the execution time of 62 possible configurations shown in the "Model Evaluation" in Table 2 to find the optimal configuration for $N = 3200, 4800, 6400$, and 9600. This step takes only 35 milliseconds on an AthlonXP 2600+ processor. Then, we measured the actual execution time of 62 possible configurations to determine the best configuration.

Figure 6 displays the correlation between the estimations and the measurements for various configurations of $N = 6400$. If our models are sufficiently precise, all points should appear on the diagonal (dotted) line. However, as can be readily seen, there are systematic deviations. We examined the reason for this and found that communication models are prone to such errors.

While we are striving to improve our communication models by correcting the source of such errors, there is another way to patch up the problem. As the errors are very regular, we can adjust them by applying linear transformation. This is not the ideal solution, but we adopt it here as a provisional expedient.

For $M_1 \leq 2$, we do not apply any correction to our models, because our models match the measurements very well. For $M_1 \geq 3$, we adjusted the models based on the measurements of $N = 6400$, $P_2 = 8$. Figure 7 displays the correlation after the adjustment of communication models. We apply this adjustment in the following discussions.

We are now ready to verify our scheme against the measurement results summarized in Table 4, where τ is the estimated execution time of the estimated optimal configuration, $\hat{\tau}$ is its actual execution time, and $\hat{T}$ is the execution time of the actual optimal configuration.

For the case of $N = 3200$, the estimated optimal configuration was the actual optimal configuration, which uses only one Athlon node with one process on it. The estima-

Table 4. Errors in estimated best configurations after adjustment vs. measured best configurations (Basic model)

Size	Estimated best configuration			Actual best configuration		Error	
N	P_1, M_1, P_2, M_2	τ	$\hat{\tau}$	P_1, M_1, P_2, M_2	$\hat{T}$	$(\tau - \hat{T})/\hat{T}$	$(\hat{\tau} - \hat{T})/\hat{T}$
3200	1,1,0,0	20.0	20.4	1,1,0,0	20.4	-0.019	0.000
4800	1,1,8,1	65.2	64.0	1,1,8,1	64.0	0.019	0.000
6400	1,1,8,1	129.8	129.7	1,2,8,1	125.2	0.037	0.036
8000	1,3,8,1	219.9	222.9	1,3,8,1	222.9	-0.013	0.000
9600	1,3,8,1	338.9	341.1	1,4,8,1	340.9	-0.006	0.001

tion error in the execution time was about 2%. For the case of $N = 4800$, as well, the estimated optimal configuration was actually optimal, using one Athlon and eight Pentium-IIs and invoking one process for each processor. The error in estimation was about 2%. For $N = 6400$, the estimated configurations were suboptimal, and the error was about 4%. As the model was constructed from $N = 400$–6400, these results indicate the range of model accuracy.

For the case of $N = 9600$, although the estimation was suboptimal, the error was less than 1%. In this case, the estimation was extrapolated from the model of $N = 400$–6400, but showed a good fit. This is because (1) the communication time is negligible compared to the computation time in a large problem, and (2) the computation time is very precisely estimated.

4.2 NL model

As stated in Section 3.2 and 3.3, we have to measure at least four sets of N and three sets of P to extract parameters for N-T and P-T models. The Basic models in Section 4.1 were constructed using 9 sets of N and 8 sets of P_2, which are more than necessary. Although the models are expected to be more precise by using more measurements, the measurements for the Basic models take more than 6 hours even for our simple heterogeneous cluster. Thus, a reduction of the measurement time for model construction is a priority.

In this section, we reduce measurements for model construction to determine what happens. Table 5 summarizes the cluster configuration parameters for this section. Here, we examine only four sets of N and four sets of P_2. As the range of N is relatively large, the derived models are referred to as *NL models* in what follows.

The total number of combinations is 6 configurations of Athlon and 24 of Pentium-II for 4 sets of N; i.e., $(6 + 24) \times 4 = 120$ sets. Table 6 lists the measurement times of various N. The total measurement time for NL models was 12235 seconds (about 3 hours), most of which is consumed by Pentium-II. The measurement time would be much shorter if we had three or more Athlon processors and we could use fast Athlons to compose Pentium-II P-T models. The model

Table 5. Cluster configuration parameters of NL model

Model Construction	N = 1600, 3200, 4800, 6400, Athlon (P_1: 1, M_1: 1, ..., 6), Pentium-II (P_2: 1, 2, 4, 8, M_2: 1, ..., 6)
Model Evaluation	N = 1600, 3200, 4800, 6400, 8000, 9600, Athlon (P_1: 0, 1, M_1: 1, ..., 6), Pentium-II (P_2: 0, ..., 8, M_2: 1)

Table 6. HPL execution time for measurements (NL and NS models)

Size N	Athlon [sec.]	Pentium-II [sec.]
400	3.9	38.1
600	7.4	56.7
800	10.8	79.0
1200	20.5	140.3
1600	37.4	241.7
2400	97.5	554.1
3200	197.2	1059.2
4800	566.0	2901.4
6400	1239.5	6093.0

construction time for 30 configurations is 0.52 millisecond, which is negligible compared to the measurement time.

Figure 8 and 9 display the correlation between the estimations and the measurements for various configurations of $N = 1600$ and 6400, respectively. As seen in these figures, systematic deviations exist even in NL models as in the Basic model. Figure 10 and 11 display the correlation after linear transformation.

Table 7 lists the errors of the estimated best configurations with NL models. We estimated the execution time of 62 possible configurations shown in the "Model Evaluation" in Table 5 to find the optimal configuration for $N = 1600, 3200, 4800, 6400$, and 9600 with NL models, which took 26.4 milliseconds on an AthlonXP 2600+ pro-

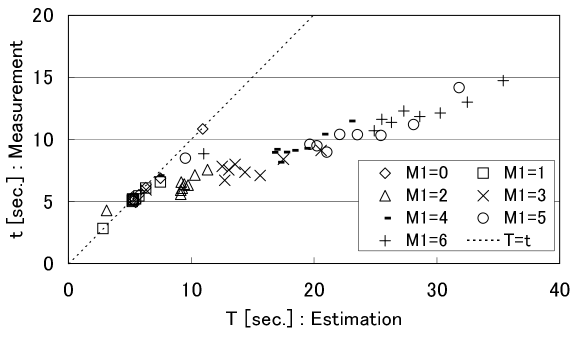

Figure 8. Correlation between original estimations and measurements (NL model, N =1600)

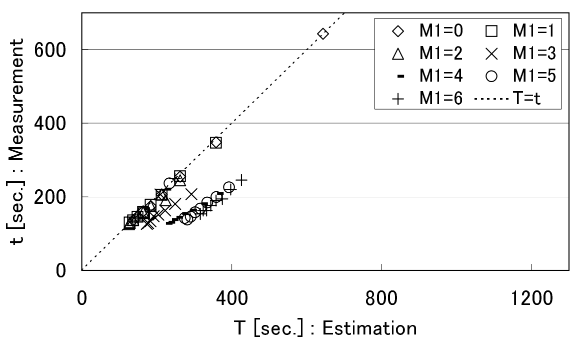

Figure 9. Correlation between original estimations and measurements (NL model, N =6400)

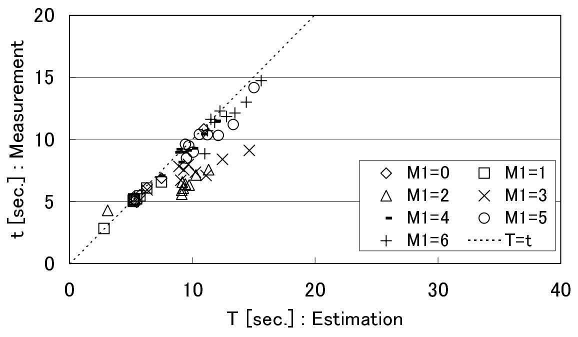

Figure 10. Correlation between estimations and measurements after adjustment (NL model, N = 1600)

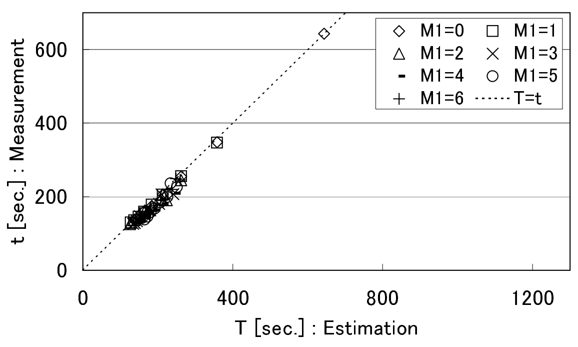

Figure 11. Correlation between estimations and measurements after adjustment (NL model, N = 6400)

cessor. The estimated best configurations were not far from the actual best configurations, and the errors of NL models were modest (0.0%–4.3%).

4.3 NS model

NL models show a good fit with reality, but they still require much time for measurements. Since the measurements for a large N take a long time, we attempt to construct models from small N to reduce measurement time in this section. These models are referred to as *NS models*. Table 8 lists the configuration parameters of NS models, where the range of N is small ($N = 400, 800, 1200,$ and 1600). Other than that, all configuration parameters are the same as for NL models. The total number of combinations is 120, the same as for NL models. The total measurement time is reduced to 571.7 seconds (about 10 minutes), which can be readily seen in Table 6.

A serious problem of NS models is the estimation error for large N. Figure 12 and 14 display the correlations between the estimations and the measurements for various configurations of $N = 1600$ and 6400, respectively. Figure 13 and 15 display the correlations after linear transformation.

Since the measurements for $N = 1600$ are used for model construction, NS models show a tolerable fit with reality in Figure 13. However, when extrapolated for $N = 6400$, such a simple linear transformation can no longer compensate for the deviations. Figure 15 shows the distinct residue of deviations even after linear transformation. Although we might find better ways of compensation than a simple linear transformation, it would be an ad hoc treatment. We prefer to pinpoint the reasons for the systematic deviations in order to correct them.

Table 9 summarizes the errors of the estimated best configurations with NS models. For $N = 1600$, the error is small because the measurements of $N = 1600$ were used to construct NS models. However, for $N \geq 3200$, the estimated best configurations were far from the actual best configurations, and the errors in execution time were 28%–

Table 7. Errors in estimated best configurations after adjustment vs. measured best configurations (NL model)

Size	Estimated best configuration			Actual best configuration		Error	
N	P_1, M_1, P_2, M_2	τ	$\hat{\tau}$	P_1, M_1, P_2, M_2	$\hat{T}$	$(\tau - \hat{T})/\hat{T}$	$(\hat{\tau} - \hat{T})/\hat{T}$
1600	1,1,0,0	2.83	2.82	1,1,0,0	2.82	0.004	0.000
3200	1,1,0,0	20.8	20.4	1,1,0,0	20.4	0.020	0.000
4800	1,1,8,1	66.8	64.0	1,1,8,1	64.0	0.044	0.000
6400	1,2,7,1	127.3	129.3	1,2,8,1	125.2	0.017	0.033
8000	1,2,8,1	198.9	223.1	1,3,8,1	222.9	-0.108	0.001
9600	1,2,8,1	289.9	355.4	1,4,8,1	340.9	-0.150	0.043

Table 8. Cluster configuration parameters of NS model

Model Construction	N = 400, 800, 1200, 1600, Athlon (P_1: 1, M_1: 1, ..., 6), Pentium-II (P_2: 1, 2, 4, 8, M_2: 1, ..., 6)
Model Evaluation	N = 1600, 3200, 4800, 6400, 8000, 9600, Athlon (P_1: 0, 1, M_1: 1, ..., 6), Pentium-II (P_2: 0, ..., 8, M_2: 1)

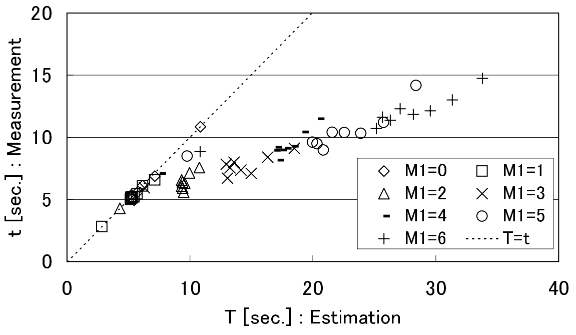

Figure 12. Correlation between original estimations and measurements (NS model, N = 1600)

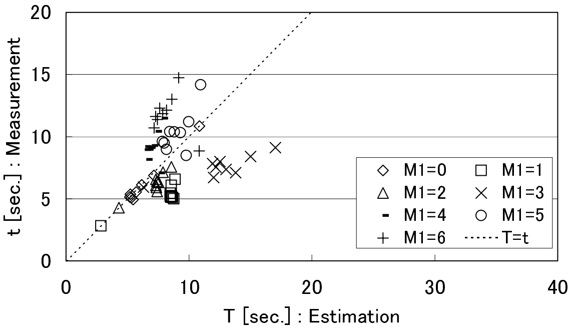

Figure 13. Correlation between estimations and measurements after adjustment (NS model, N = 1600)

82%.

Although NS models take little measurement time for construction, the error is large particularly for large N. On the contrary, NL models can show relatively large errors for small N ($N < 1600$), since they are constructed from the measurements of $1600 \leq N \leq 6400$. Practically, NL models are better than NS models, because precise estimation models are more desired for large N than for small N. The errors in small N are practically not serious. As the execution time of such small problems as $N < 1600$ is less than a couple of seconds, even 100% error means a negligible increase in execution time.

5 Conclusion

The results of this study are still preliminary and improvements are underway. More extensive studies are required for various cluster configurations. Our aim remains (1) to make the estimation model more elegant and unified, (2) to reduce the model construction time, and (3) to reduce the errors in estimation.

In the present study, every possible configuration was examined to find the estimated optimal configuration. For larger clusters, it is essential to find a way to reduce the search space. Approximation algorithms (i.e., heuristics) are also worth considering.

This study examined one specific application (HPL), but other parallel applications should be also examined. All these tasks must be left to future studies.

Table 9. Errors in estimated best configurations after adjustment vs. measured best configurations (NS model)

Size	Estimated best configuration			Actual best configuration		Error	
N	P_1, M_1, P_2, M_2	τ	$\hat{\tau}$	P_1, M_1, P_2, M_2	$\hat{T}$	$(\tau - \hat{T})/\hat{T}$	$(\hat{\tau} - \hat{T})/\hat{T}$
1600	1,1,0,0	2.84	2.82	1,1,0,0	2.82	0.006	0.000
3200	1,2,0,0	14.2	26.1	1,1,0,0	20.4	-0.304	0.276
4800	1,2,0,0	20.6	82.3	1,1,8,1	64.0	-0.678	0.286
6400	1,2,0,0	13.7	190.8	1,2,8,1	125.2	-0.890	0.523
8000	1,2,0,0	15.0	371.1	1,3,8,1	222.9	-0.933	0.665
9600	1,2,0,0	19.9	619.7	1,4,8,1	340.9	-0.942	0.818

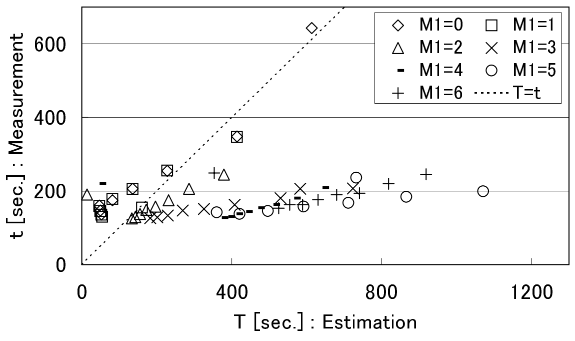

Figure 14. Correlation between original estimations and measurements (NS model, N = 6400)

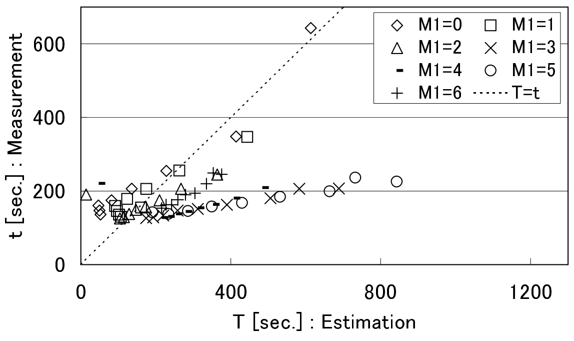

Figure 15. Correlation between estimations and measurements after adjustment (NS model, N = 6400)

Acknowledgments

This work was partially supported by a Grant-in-Aid for Scientific Research from the Japan Society for the Promotion of Science (JSPS), as well as a grant from Hori Information Science Promotion Foundation. This work was also supported by the 21st Century COE Program "Intelligent Human Sensing" from the Ministry of Education, Culture, Sports, Science and Technology.

References

[1] O. Beaumont, V. Boudet, A. Petitet, F. Rastello, and Y. Robert. A proposal for a heterogeneous cluster ScaLAPACK (dense linear solvers). *IEEE Transaction on Computers*, 50(10):1052–1070, Oct. 2001.

[2] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[3] Y. Cheng and C. Hu. *MOSFET Modeling and BSIM3 User's Guide*. Kluwer Academic, 1999.

[4] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi. *GNU Scientific Library Reference Manual*, August 2003. (Edition 1.4 for GSL version 1.4).

[5] W. Gropp and E. Lusk. MPICH – a portable implementation of MPI. http://www-unix.mcs.anl.gov/mpi/mpich/.

[6] G. Helmer. NetPIPE – a network protocol independent performance evaluator. http://www.scl.ameslab.gov/netpipe/.

[7] A. Kalinov and A. Lastovetsky. Heterogeneous distribution of computations while solving linear algebra problems on networks of heterogeneous computers. In P. Sloot, M. Bubak, A. Hoekstra, and B. Hertzberger, editors, *Proc. HPCN Europe 1999*, pages 191–200. Springer, 1999.

[8] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. HPL – a portable implementation of the high-performance Linpack benchmark for distributed-memory computers. http://www.netlib.org/benchmark/hpl/.

[9] T. Sasou, S. Matsuoka, and O. Tatebe. The optimization of the LINPACK benchmark for heterogeneous clusters. *IPSJ SIG Notes 2001–HPC–86*, pages 49–54, 2001. (In Japanese).

Biography

Yoshinori Kishimoto received his B.E. degree in 2001 from the Department of Knowledge-based Information

Engineering of Toyohashi University of Technology. Presently, he is studying for his master's degree at that institution. He is also a student member of the IPSJ.

Shuichi Ichikawa received his D.S. degree in Information Science from the University of Tokyo in 1991. He has been affiliated with Mitsubishi Electric Corporation (1991–1994), Nagoya University (1994–1996), and Toyohashi University of Technology (since 1997). Currently, he is an associate professor of the Department of Knowledge-based Information Engineering of Toyohashi University of Technology. His research interests include parallel and distributed processing, high-performance computing, and custom computing machinery. He is a member of ACM, IEEE, IPSJ, and IEICE.