Optimizing the Configuration of a Heterogeneous Cluster with Multiprocessing and Execution-Time Estimation [⋆]

Yoshinori Kishimoto [a,1], Shuichi Ichikawa [a,b,*]

[a] *The Department of Knowledge-based Information Engineering, Toyohashi University of Technology, 1-1 Hibarigaoka, Tempaku, Toyohashi, Aichi 441-8580, Japan.*

[b] *The Intelligent Sensing System Research Center, Toyohashi University of Technology, 1-1 Hibarigaoka, Tempaku, Toyohashi, Aichi 441-8580, Japan.*

Received October 31, 2004; revised March 15, 2005; accepted April 15, 2005

Abstract

Although heterogeneous clusters are flexible and cost-effective, they entail intrinsic difficulties in optimization. Whereas it is simple to invoke multiple processes on fast processing elements (PEs) to alleviate load imbalance, the optimal process allocation is not obvious. Communication time is another problem. Though it is sometimes better to exclude slow PEs to avoid performance degradation, 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. The derived models are then used to estimate the optimal PE configuration and process allocation. We implemented various models for HPL (High Performance Linpack benchmark) on a heterogeneous cluster, and estimated the optimal configurations for various problem sizes. In the case of a heterogeneous cluster of Athlon and Pentium-II, the execution time of the estimated optimal configuration was 0%–7.4% longer than that of the actual optimal configuration. In a heterogeneous cluster of 3 kinds of processors that includes dual-processors, the excess time was 13.6%–31.5%.

Key words:
Heterogeneous cluster, Static load balancing, Optimization, High Performance Linpack, Automatic configuration

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 wide range of processing elements (PEs) from fast to slow. However, 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 available PEs, since superfluous communications can prolong the total execution time. 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 for a heterogeneous cluster.

Many applications for parallel computers or homogeneous clusters are written to distribute workloads equally among PEs. Although it is desirable to rewrite the application for heterogeneous clusters, it requires much time and effort to adapt it to 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

⋆ A preliminary version of this work was presented at the 13th Heterogeneous Computing Workshop (HCW 2004), Santa Fe, New Mexico, USA, April 2004.

∗ Corresponding author. Tel./Fax.: +81-532-44-6897.
Email address: `ichikawa@tutkie.tut.ac.jp` (Shuichi Ichikawa).
URL: `http://meta.tutkie.tut.ac.jp/~ichikawa/index-e.html` (Shuichi Ichikawa).

[1] The author is presently affiliated with the Asahi Kasei Information Systems Co., Ltd.

clusters. Although we examine HPL (High Performance Linpack benchmark) [1] as a sample application in this study, our approach is not limited to HPL alone but is expected to be widely applicable to many other applications.

Section 2 introduces some related studies, and then briefly summarizes the background of this study. In Section 3, the execution time is modeled from the measurement results of various configurations. The derived models are used to estimate the optimal PE configuration and process allocation. The evaluation results are found in Section 4. Section 5 concludes the study.

2 Background and related works

Though a block cyclic distribution is very popular in balancing the load 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 [2] presented a “heterogeneous block cyclic distribution” for the Cholesky factorization of square dense matrices. Beaumont et al. [3] reported a “2D heterogeneous grid allocation” for the heterogeneous cluster ScaLAPACK [4], while Sasou et al. [5] modified the HPL source code for heterogeneous clusters to dispatch multiple panels to fast PEs in LU decomposition.

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

Sasou et al. [5] stated that the performance of the multiprocessing approach is rather poor compared to their HPL implementation specialized for heterogeneous clusters. However, their conclusion is not yet definitive. They evaluated performance by fixing the configuration of a heterogeneous cluster; i.e., using a predetermined set of PEs with predetermined numbers of processes on each PE. The performance of multiprocessing might be improved by selecting the best configuration of PEs and processes.

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 perfor-

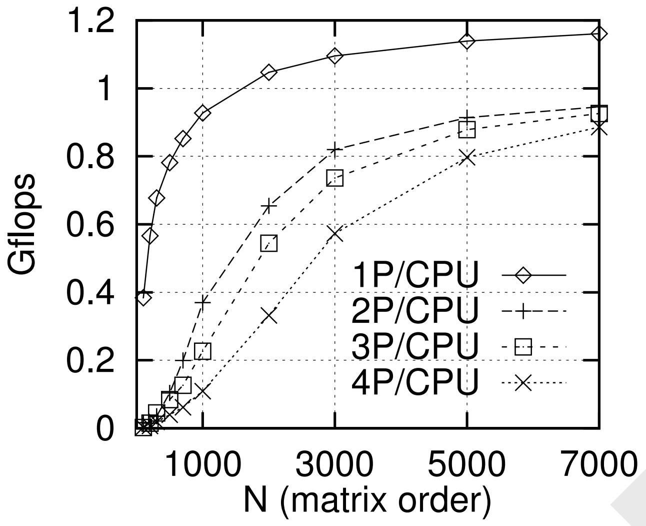

Fig. 1. Performance overhead of multiprocessing on an Athlon processor.

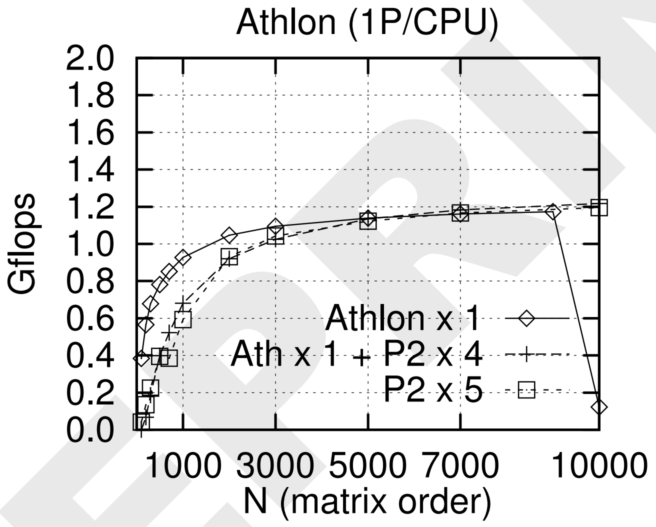

Fig. 2. HPL performance of a heterogeneous cluster of various configurations; load imbalance.

mance reported by HPL. In the figure, “nP/CPU” denotes that n processes were simultaneously executed on that processor. Although the performance decreases as n increases, the loss is not too great for practical applications.

The performance of HPL is defined by the following calculation. The arithmetic operations required for HPL of size N is $2N^3/3 + 3N^2/2$ flops, where $2N^3/3 - N^2/2$ is for LU factorization and $2N^2$ is to solve the main problem. HPL measures the maximum walltime of all PEs involved, and the overall performance is calculated by dividing $2N^3/3 + 3N^2/2$ flops by the measured maximum walltime. Therefore, HPL performance is not the sum of the performance of each PEs, but it represents the overall performance of the system.

Figure 2 and 3 summarize the HPL performances of various subsets of a hetero-

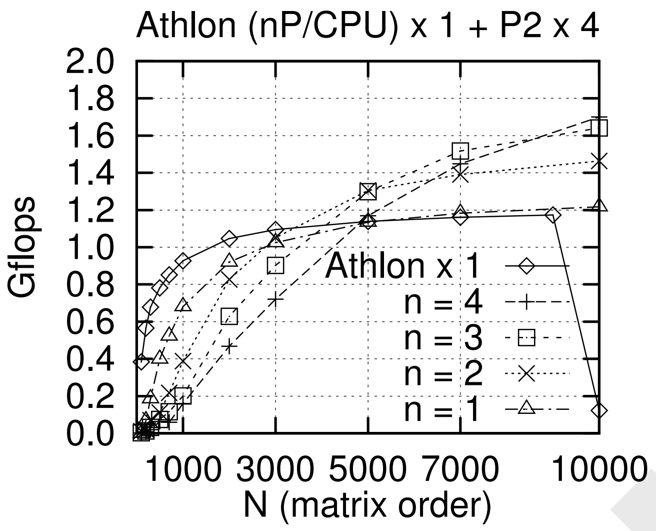

Fig. 3. HPL performance of a heterogeneous cluster of various configurations; multiprocessing.

geneous cluster, which consists of an Athlon 1.33 GHz node, four Pentium-II 400 MHz dual-processor nodes, and a 1000base-SX network. This cluster is a subset of the heterogeneous cluster shown in Table 1.

In Figure 2, “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 when $N \geq 10000$ for the shortage of main memory. “P2 x 5” denotes the performance of five Pentium-II processors, which is almost the same as that 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 when $N = 10000$ 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 that of “P2 x 5”. Since an Athlon 1.33 GHz is about 5 times faster than a Pentium-II 400 MHz, 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 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” denotes that two processes are simultaneously 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 contrast. Since the workload of HPL is equally distributed to each process, an Athlon can undertake more computational workloads by invoking more processes. If the appropriate number of processes are invoked on an Athlon, the load imbalance could be resolved and no wait would occur for synchronization.

From Figure 3, it is readily seen that the load imbalance can be alleviated 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, when $N < 5000$, "n = 4" demonstrates a lower performance than "n = 1", owing to the multiprocessing overhead. For superior performance, the choice should be "Athlon x 1" when $N \leq 3000$, "n = 2" when $3000 < N \leq 5000$, "n = 3" when $5000 < N \leq 8000$, and "n = 4" when $8000 < N \leq 10000$, judging from Figure 3.

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 task 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 based on 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 the 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 [6], 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 factors such as the miscellaneous overhead caused by cache misses, or communication buffer management.

We make the following assumptions here to simplify our models and evaluations:

- Ignore the overlap of computation and communication, and assume the total execution time to be the sum of the computation time and the communication time.
- Invoke the same number of processes on equivalent PEs (PEs of the same specification).
- Ignore the network topology, and assume that the network is homogeneous.
- Assume that the communication time is independent of the receiver. This assumption liberates us from having to measure the communication time of every possible combination of sender/receiver.

Such simplification may 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 proved to be unacceptably large, we would have to rebuild our models based on other assumptions.

3.2 N-T model

Let N be the size of HPL. G_i is a group of PEs comprised of equivalent PEs in a heterogeneous cluster. P_i is the number of PEs actually used for the job in G_i ($0 \leq P_i \leq |G_i|$). M_i is the number of processes on each PE in G_i, if $P_i \neq 0$; M_i is zero, if $P_i = 0$. P is the total number of processes in the cluster; i.e., $P = \sum_i P_i M_i$. Our goal is to build models that estimate the execution time T_i of G_i from N, P, and M_i. The total execution time T is estimated by $T = \max_i T_i$.

T_i consists of the computation time Ta_i and the communication time Tc_i. We assume $T_i = Ta_i + Tc_i$ here, as stated in Section 3.1. Clearly, T_i depends on the process grid. Although, in this study, we examine only the case of a 1-by-P process grid (one-dimensional block cyclic distribution), our scheme is universally applicable to any other process grid.

To estimate Ta_i and Tc_i, we have to examine the execution time item by item. Figure 4 illustrates 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* denotes 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 HPL_DETAILED_TIMING in compiling HPL. To measure *bcast*, we had to add some lines to the original source code.

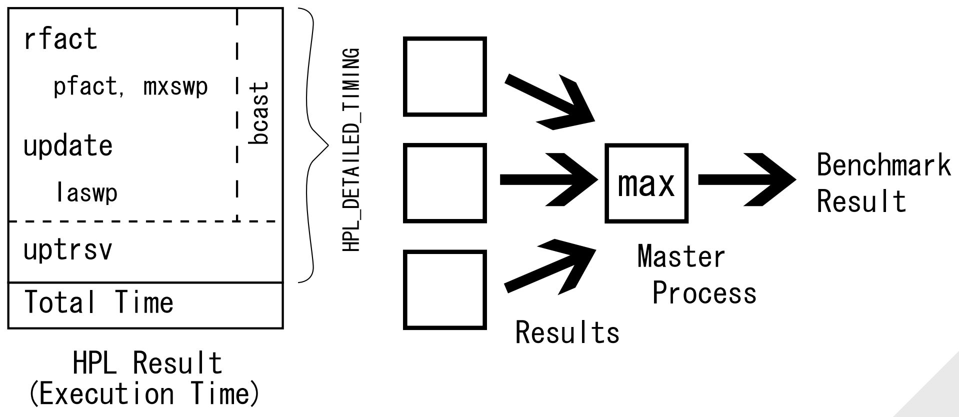

Fig. 4. Detailed timing measurement.

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

$$Ta_i = (rfact - mxswp) + (update - laswp) + uptrsv \tag{1}$$
$$Tc_i = mxswp + laswp + bcast \tag{2}$$

The order of computation is determined from the algorithm, and is summarized by the following equations [5]:

$$rfact = \frac{3}{2} \cdot N^2 + O(N) \tag{3}$$
$$update = \frac{2N^3}{3P} + \frac{P+1}{P} \cdot O(N^2) + O(N) \tag{4}$$
$$uptrsv = \frac{1}{P} \cdot O(N^2) \tag{5}$$

In the same way, the order of communication is estimated and summarized as follows.

$$mxswp = O(1) \tag{6}$$
$$laswp = \frac{1}{P} \cdot O(N^2) \tag{7}$$
$$bcast = (P-1) \cdot O(N^2) \tag{8}$$

Thus, Ta_i, Tc_i, and T_i are estimated by the following equations.

$$Ta_i = \frac{1}{P} \cdot O(N^3) + O(N^2) \tag{9}$$

$$Tc_i = P \cdot O(N^2) + \frac{1}{P} \cdot O(N^2) + O(N^2) \tag{10}$$

$$T_i = \frac{1}{P} \cdot O(N^3) + P \cdot O(N^2) + O(N^2) \tag{11}$$

As seen in above equations, the order of Ta_i is the same as that of *update*. As a matter of fact, *update* is the dominant factor of Ta_i; According to the actual measurement results, *update* is approximately 100 times greater than *uptrsv* and $rfact$ when $N = 9600$.

In our previous study [7], we built the Ta_i and Tc_i models separately from detailed measurement results. However, the derived models showed relatively major errors in estimating execution time. We found that deviations remain in the Tc_i model, and that simple linear transformations can compensate for them [7].

Such estimation errors may be caused by the simplifications stated in Section 3.1. The models may be improved by (1) building more elaborate models with more parameters, (2) using more models according to individual cases, and (3) specializing in the behavior of a specific application. However, we do not favor any of these approaches. We already have too many parameters to handle in heterogeneous clusters, and do not want any more complicated models. We also prefer general models rather than application-specific models.

In this study, we adopt a simple model that approximates T as a whole. Since T is very easy to measure, this scheme would be generally applicable to any applications. This simply means that it is *applicable* to any applications, but does not necessarily mean that the derived models would show a good fit to reality. However, we concentrate here on examining the feasibility of this simple scheme, while leaving the improvement and enhancement of models for future studies.

From the equation (11), the following equation is derived to approximate T_i for a given set of P and M_i.

$$T_i(N)|_{P,M_i} = k_0N^3 + k_1N^2 + k_2N + k_3 \tag{12}$$

The constant factors k_0–k_3 are determined from the measurement results by the least-squares method. This model is designated as *N-T model* in the following discussions. As this equation is a linear function of k_0–k_3, the coefficients k_0–k_3 can be extracted by using the `gsl_multifit_linear()` function of GSL (GNU Scientific Library) [8]. In order to extract four coefficients k_0–k_3, we have to measure $T_i(N)$ of (at least) four different sizes (Ns) for each configuration.

3.3 P-T model

Each N-T model was constructed to be specific to the configuration (P and M_i). Though N-T models can interpolate or extrapolate T from the size N, they are incapable of estimating T for other configurations. In other words, we have to build a separate N-T model for each possible configuration (Fig. 5 (left)).

Since it is not practical to manage many N-T models for every combination of parameters, we integrate N-T models of the same M_i into a new model. Since this model includes P as a variable, it is called a *P-T model*. P-T models are represented by the following equation, considering the equation (11).

$$T_i(N,P)|_{M_i} = k_4 P \cdot T_i(N)|_{P,M_i} + \frac{k_5}{P} \cdot T_i(N)|_{P,M_i} + k_6 \tag{13}$$

Here, the constant factors k_4–k_6 have to be extracted from the corresponding N-T models by the least-squares method. As these equations are linear functions of k_4–k_6, their coefficients can also be extracted by using the `gsl_multifit_linear()` function [8]. To extract these coefficients, we have to measure (at least) three different Ps for each P-T model. With P-T models, we can interpolate or extrapolate T from P, thus substantially reducing the model construction time.

In this study, we decided to construct models from the measurement results of each G_i; i.e., a *homogeneous* sub-cluster is used to construct a model. For example, we extract the model parameters for Pentium-II by using eight Pentium-II processors in our heterogeneous cluster, leaving Pentium-III and Athlon unused. This greatly simplifies the model construction, since it drastically reduces the number of combinations for measurements. Though there are many other relevant possibilities in this regard, we leave them for future studies.

It might be also possible to extract constant factors of the equation (11) directly from the measurement results. Such possibility is also left for future studies.

3.4 Binning

Both N-T and P-T models are selectively used according to the circumstances. Figure 5 (right) summarizes the selection of models. The "X" in Figure 5 denotes that there are no such cases (keep in mind that $P = \sum_i P_i M_i \geq \forall M_i$ holds).

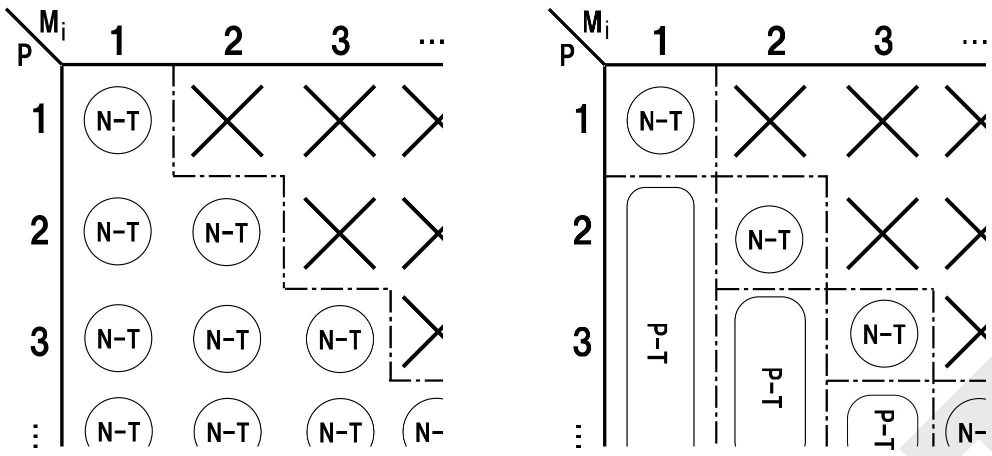

Fig. 5. N-T models (left) and binning (right).

When HPL is executed on a single PE (i.e., $P = \exists M_i$), *no* inter-PE communication emerges. This case is distinct from the execution with multiple processors (i.e., $P > \forall M_i$). It is illogical and imprecise to treat 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 [6], 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, T_i can show disjunct behavior depending on the allocated data size. For example, it is well known that performance is significantly degraded when cache-misses frequently occur. Another example is shown in Figure 2, where the performance of a single Athlon is 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 T_i is not necessarily continuous nor differentiable, but it could be a piecewise function.

3.5 Model composition

N-T and P-T models have to be built for each group G_i. On the other hand, it is both difficult and impractical to build models for every G_i, of which there are so many. In such cases, it is far more practical to build some of the 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 P measurements must be taken to build P-T models. If a homogeneous sub-cluster is used for parameter ex-

Table 1
HPL execution environment.

Node 1	AMD Athlon 1.33 GHz, Main memory 768 MB
Nodes 2–3	Intel Pentium-III 866 MHz (dual-processor), Main memory 768 MB
Nodes 4–7	Intel Pentium-II 400 MHz (dual-processor), Main memory 768 MB
Network	1000base-SX (GA-620), 100base-TX (Intel Pro100+)
OS	RedHat Linux7.0J (kernel 2.4.2)
Compiler	gcc 2.96, -DHPL_DETAILED_TIMING -fomit-frame-pointer -O3 -funroll-loops -W -Wall
Libraries	MPICH–1.2.5, ATLAS 3.2.1

traction, we need three or more processors of the same kind. If there are not enough processors in a group, it is impossible to build P-T models from measurements for that group. Such a situation often occurs in a heterogeneous cluster.

In this study, since we only have one Athlon processor in our heterogeneous cluster, we cannot 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 constructed from measurements.

4 Evaluation

In this section, the estimation models are built and evaluated for a heterogeneous cluster. The specifications of the evaluation platform are listed in Table 1. As each Pentium-II node includes two processors, a total of 8 Pentium-II processors are available in 4 nodes (Node 4–Node 7). Likewise, 4 Pentium-III processors are available in Node 2 and Node 3. All nodes have both 1000base-SX and 100base-TX interfaces, but only the 100base-TX is used in the following measurements to clearly demonstrate the effect of communication time.

4.1 Heterogeneous cluster of Athlon and Pentium-II

Here we examine the fundamental properties of our models using two groups of PEs (Athlon and Pentium-II). 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 each Athlon and Pentium-II, respectively.

Table 2
Errors in estimated best configurations vs. measured best configurations; using only N-T models.

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	3.40	3.48	1, 1, 0, 0	3.48	-0.022	0.000
3200	1, 1, 0, 0	26.13	25.93	1, 1, 0, 0	25.93	0.008	0.000
4800	1, 2, 8, 1	71.15	71.29	1, 1, 8, 1	71.27	-0.002	0.000
6400	1, 4, 8, 1	138.70	138.72	1, 4, 8, 1	138.72	-0.000	0.000
8000	1, 4, 8, 1	231.90	249.06	1, 4, 8, 1	249.06	-0.069	0.000
9600	1, 5, 7, 1	356.38	408.91	1, 4, 8, 1	386.55	-0.078	0.058

4.1.1 Evaluation of N-T models

First of all, we have to examine the accuracy of N-T models. Since our models are designed based on the assumptions in Section 3.1, they must be verified by measurements.

Each N-T model was constructed from the measurements of 9 sizes ($N = 400, 600, 800, 1200, 1600, 2400, 3200, 4800,$ and 6400). Configurations were as follows; $0 \leq P_1 \leq 1$, $1 \leq M_1 \leq 6$, $0 \leq P_2 \leq 8$, and $M_2 = 1$. Since an Athlon 1.33 GHz is about 4 times faster than a Pentium-II 400 MHz, M_1 was set to be $1 \leq M_1 \leq 6$, while M_2 was fixed to 1. There are 54 (6×9) configurations for $P_1 = 1$, and 8 for $P_1 = 0$, for a total of 62 configurations.

The total measurement time was 41043.7 seconds (about 11 hours) for 9 sizes of 62 configurations. We extracted parameters from these measurements and constructed N-T models for all 62 configurations. This step takes no more than a millisecond on an AthlonXP 2600+ processor. We also measured the actual execution time of 62 possible configurations for various Ns to determine the best configuration.

The derived N-T models were then applied to estimate the execution time of 62 configurations to determine the optimal configuration for $N = 1600, 3200, 4800, 6400, 8000,$ and 9600. The estimation and measurement results are summarized in Table 2, where τ is the estimated execution time of the estimated optimal configuration, while $\hat{\tau}$ is its actual execution time. $\hat{T}$ is the execution time of the actual optimal configuration. As readily seen, optimal or sub-optimal configurations were determined using our N-T models. The estimation errors of execution time were less than 8%, and the excess time of the estimated optimal configurations was less than 6%.

Figure 6 and 7 display the correlation between the estimations and the measurements of N-T models for various configurations. If the models are accurate, all points should appear on the diagonal (dotted) line. Figure 6 shows the correlations of the four sizes ($N = 1600, 3200, 4800, 6400$) used for model con-

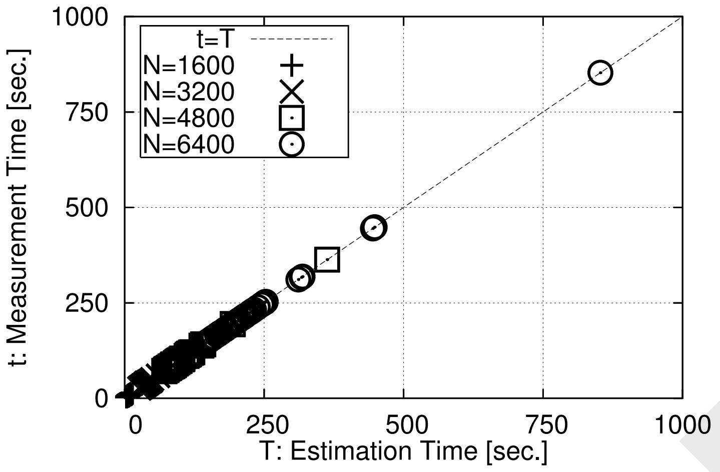

Fig. 6. Correlation between estimations and measurements of N-T models for $N = 1600, 3200, 4800, 6400$.

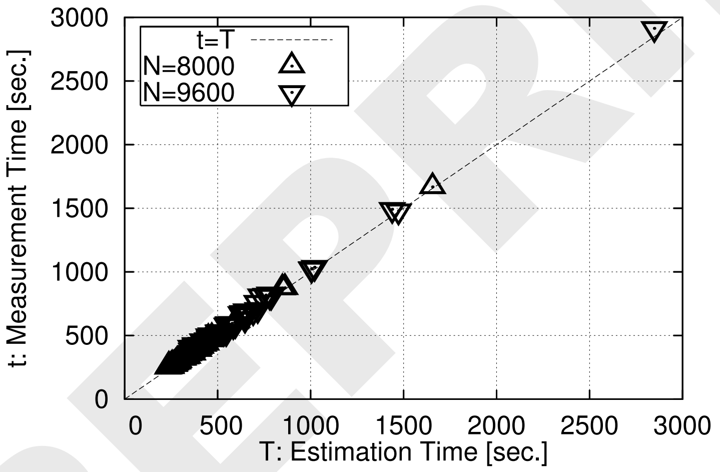

Fig. 7. Correlation between estimations and measurements of N-T models for $N = 8000, 9600$.

struction, and Figure 7 shows the correlations of larger sizes ($N = 8000, 9600$) extrapolated from the model. As readily seen, our N-T models are sufficiently accurate for both cases.

4.1.2 Evaluation of P-T models with binning

Now, since our N-T models are regarded as reliable, we proceed to the evaluation of P-T models. As shown in Figure 5 (right), N-T models are still used with P-T models. Models were constructed for each G_i as stated in Section 3.5, and the measurements for model construction were made for every combina-

Table 3
Cluster configuration parameters for Athlon and Pentium-II.

	Size	Athlon		Pentium-II		Number of
	N	P_1	M_1	P_2	M_2	configurations
Model Construction	400, ..., 6400	1	1, ..., 6	1, ..., 8	1, ..., 6	54
Model Evaluation	400, ..., 9600	0, 1	1, ..., 6	0, ..., 8	1	62

Table 4
HPL execution time for measurements of Athlon and Pentium-II.

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

tion of parameters in the "Model Construction" of Table 3. The total number of combination is 6 configurations of Athlon and 48 of Pentium-II for 9 sizes; i.e., $(6 + 48) \times 9 = 486$ sets. Table 4 summarizes the HPL execution time for measurements item by item. The total time for measurements was 22869 seconds (about 6 hours). The models constructed here are designated as *N9 models* in the following discussion.

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, since it is impossible to extract the parameters from a single Athlon (Section 3.5). In the present study, we simply scaled Pentium-II P-T models by a constant factor of 0.307 to derive Athlon P-T models. This factor was determined by comparing the Athlon N-T model of $P_1 = M_1 = 1$ with the Pentium-II N-T model of $P_2 = M_2 = 1$.

With N9 models, we estimated the execution time of 62 possible configurations shown in the "Model Evaluation" in Table 3 to determine the optimal configuration for $N = 1600, ..., 9600$. This step takes only 35 milliseconds on an AthlonXP 2600+ processor. We also measured the actual execution time of these 62 possible configurations to determine the actual best configuration.

The evaluation results of N9 models are summarized in Table 5. It is readily seen that optimal or sub-optimal configurations were found with N9 models. Though the error is relatively large for $N = 1600$, the execution time of this case is short, with an excess of only 1.46 seconds. In other cases, the excess

Table 5
Errors in estimated best configurations vs. measured best configurations (N9 models).

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, 2, 0, 0	0.48	4.28	1, 1, 0, 0	2.82	-0.828	0.518
3200	1, 1, 0, 0	20.04	20.42	1, 1, 0, 0	20.42	-0.018	0.000
4800	1, 4, 8, 1	57.67	68.73	1, 1, 8, 1	64.00	-0.099	0.074
6400	1, 4, 8, 1	113.19	128.04	1, 2, 8, 1	125.24	-0.096	0.022
8000	1, 4, 8, 1	195.33	226.25	1, 3, 8, 1	222.86	-0.124	0.015
9600	1, 4, 8, 1	309.25	340.86	1, 4, 8, 1	340.86	-0.093	0.000

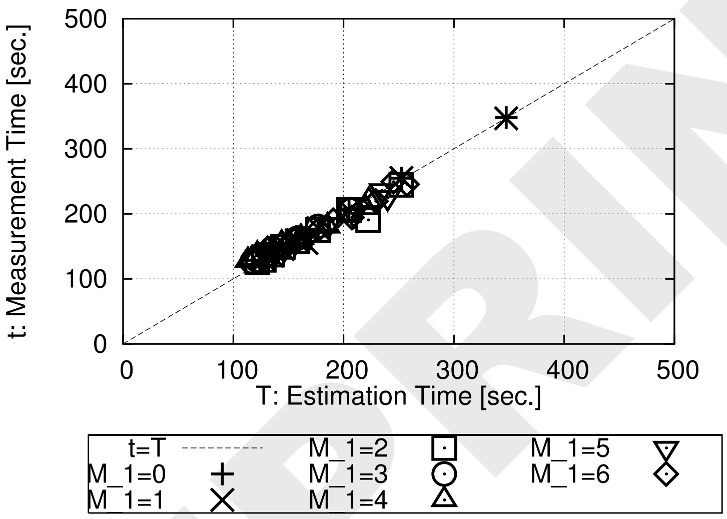

Fig. 8. Correlation between estimations and measurements of N9 models ($N = 6400$).

time of the estimated optimal configuration was less than 7.4%, while the error in estimated optimal execution time was less than 12.4%.

Figure 8 displays the correlation between the estimations and measurements for $N = 6400$. Although the error in the estimated optimal execution time was 9.6% in this case, a positive correlation is evident. We conclude that N9 models are accurate enough to give satisfactory approximations for practical use.

4.1.3 Reduction of measurement time

As stated in Sections 3.2 and 3.3, we have to measure at least four Ns and three Ps to extract parameters for the N-T and P-T models. N9 models in Section 4.1.2 were constructed using measurements of nine Ns and eight P_2s, which are more than necessary. Although the models are expected to be ren-

Table 6
Errors in estimated best configurations vs. measured best configurations (N5 models).

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	2.82	2.82	1, 1, 0, 0	2.82	-0.001	0.000
3200	1, 1, 0, 0	20.81	20.42	1, 1, 0, 0	20.42	0.019	0.000
4800	1, 4, 8, 1	58.89	68.73	1, 1, 8, 1	64.00	-0.080	0.074
6400	1, 4, 8, 1	113.29	128.04	1, 2, 8, 1	125.24	-0.095	0.022
8000	1, 4, 8, 1	190.43	226.25	1, 3, 8, 1	222.86	-0.146	0.015
9600	1, 4, 8, 1	293.52	340.86	1, 4, 8, 1	340.86	-0.139	0.000

dered more accurate by using more measurements, the measurement time of N9 models is more than 6 hours even for our simple heterogeneous cluster. Thus, a further reduction in the measurement time for model construction is a priority.

In this section, we reduce the measurements for N and determine the result. Cluster configuration parameters are the same as those in Table 3, except that only 5 sizes of HPL ($N = 400, 800, 1600, 3200$, and 6400) are measured. These models are designated as *N5 models* in the following discussion. We can also reduce measurements for P, but that will be evaluated later in Section 4.2. In this section, N5 models are constructed using eight P_2s as N9 models.

The consequent measurement time is 15265 seconds (about 4.2 hours), which is about two thirds that of N9 models. As seen in Table 4, most of the measurement time is consumed by Pentium-II. Since our heterogeneous cluster consists of fewer fast processors and more slow processors (see Table 1), P-T models of Athlon are composed using the measurements of slow Pentium-IIs, significantly prolonging the measurement time. If we had three or more of the faster Athlon processors, we could use them to compose Pentium-II P-T models, thus considerably reducing the total measurement time.

The evaluation results of N5 models are summarized in Table 6. Optimal or sub-optimal configurations were determined using N5 models. Although the errors of N5 models are greater than those of N9 models, the excess time of the estimated optimal configuration is less than 7.4%. The error in estimated optimal execution time was less than 14.6%. In conclusion, N5 models prove to be almost as accurate as N9 models, while reducing the measurement time to 66.7% of the latter.

Although N5 models show a good fit with reality, they still require considerable time for measurements. Since the measurements for a large N take a long time, we then attempt to reduce the measurement time by constructing models from small Ns. The configuration parameters are the same as those in N5 models, except that 5 small sizes of HPL ($N = 400, 600, 800, 1200$, and 1600)

Table 7
Errors in estimated best configurations vs. measured best configurations (N5S models).

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	0.007	0.000
3200	1, 4, 8, 1	18.25	32.83	1, 1, 0, 0	20.42	-0.106	0.608
4800	1, 5, 8, 1	28.24	79.24	1, 1, 8, 1	64.00	-0.559	0.238
6400	1, 5, 8, 1	26.64	142.05	1, 2, 8, 1	125.24	-0.787	0.134
8000	1, 5, 8, 1	3.82	245.21	1, 3, 8, 1	222.86	-0.983	0.100
9600	1, 5, 8, 1	-49.66	374.49	1, 4, 8, 1	340.86	-1.146	0.099

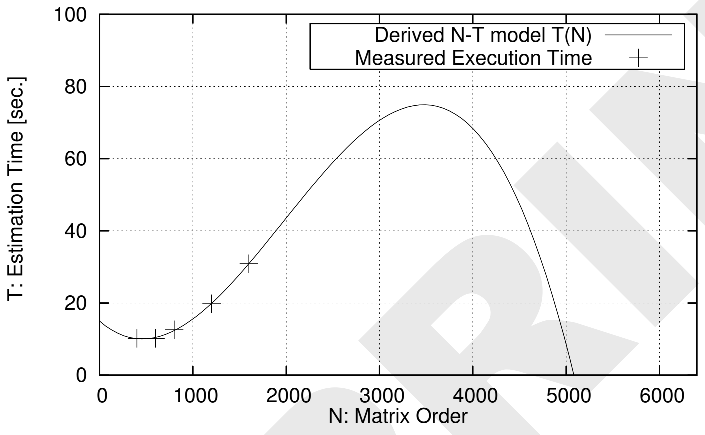

Fig. 9. Failure of model construction ($P_2 = 8$, $M_2 = 5$).

are measured. The models thus derived are referred to as *N5S models* in the following discussion. The total measurement time of N5S models is 1299.3 seconds (21.5 minutes), which is as little as 5.7% of that in N9 models.

The evaluation results of N5S models are summarized in Table 7. It is obvious that the behavior of τ is irregular. For $4800 \leq N \leq 9600$, τ decreases as N increases with the same configuration, which should not occur. The actual execution time $\hat{\tau}$ increases as N increases, which can be seen in Table 7.

This anomaly of τ is caused by a failure of parameter extraction. Figure 9 illustrates the measurement results and the derived N-T model of $P_2 = 8$ and $M_2 = 5$. This model yields a negative estimation time for $N > 5000$, which makes no sense. Although it is possible to extract four coefficients from five measurements, the derived model would be inaccurate, making such empirical models invalid for extensive extrapolations. We cannot build accurate models without measuring a sufficient number of executions of the appropriate size.

It may seem problematic that the estimated optimal configurations of N5S models are not far from the actual optimal configurations. This is due to the

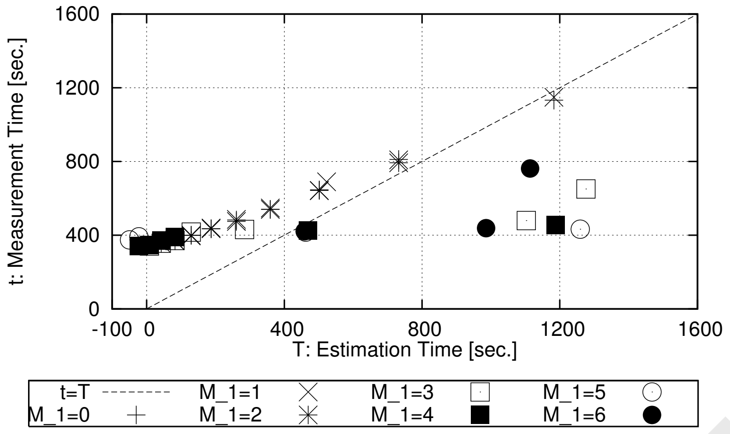

Fig. 10. Correlation between estimations and measurements of N5S models ($N = 9600$).

Table 8
Cluster configuration parameters for Athlon, Pentium-III, and Pentium-II.

	Size	Athlon		Pentium-III		Pentium-II		Num. of
	N	P_1	M_1	P_2	M_2	P_3	M_3	config.
Model Const.	400, ..., 6400	1	1, ..., 6	1, ..., 4	1, 2, 3	1, ..., 8	1, ..., 6	66
Model Eval.	1600, ..., 9600	0, 1	1, ..., 6	0, ..., 4	1, 2, 3	0, ..., 8	1	818

positive correlation between estimations and measurements. Figure 10 displays the correlation of $N = 9600$. Although the error in estimation time is large, the configurations of the short estimation time actually show a correspondingly short execution time. Therefore, the estimated optimal configuration is not far from the actual optimal configuration, even if it is not actually optimal.

4.2 Heterogeneous cluster of Athlon, Pentium-III, and Pentium-II

In the previous section, we evaluated our scheme using a simple heterogeneous cluster with 2 kinds of processors. Here we examine a larger heterogeneous cluster, which consists of 13 processors (3 groups), shown in Table 1. In this section, P_1, P_2, and P_3 designate the respective number of Athlon, Pentium-III, and Pentium-II processors used for the HPL. M_1, M_2, and M_3 represent the number of processes invoked on each Athlon, Pentium-III, and Pentium-II, respectively. The models are constructed from the measurements of 9 sizes of N, which are the same as those in N9 models (Section 4.1.2).

Measurements for the model construction were made for every combination of parameters in the "Model Construction" of Table 8. Models were constructed for each group, giving a total of 66 combinations, i.e., 6 for Athlon, 12 for Pentium-III, and 48 for Pentium-II. The total measurement time is 28069

Table 9
HPL execution time for measurements of Athlon, Pentium-III, and Pentium-II.

Size N	Athlon [sec.]	Pentium-III [sec.]	Pentium-II [sec.]
400	3.9	9.3	96.7
600	7.4	16.0	130.1
800	10.8	22.6	178.8
1200	20.5	51.0	305.2
1600	37.4	104.7	508.5
2400	97.5	228.7	1117.3
3200	197.2	464.0	2042.2
4800	566.0	1350.7	5360.0
6400	1239.5	2953.0	10950.3
Total	2180.2	5199.9	20689.1

Table 10
Errors in estimated best configurations vs. measured best configurations, where Pentium-III P-T models were constructed from measurement results.

Size	Estimated best configuration			Actual best configuration		Error	
N	$P_1, M_1, ..., P_3, M_3$	τ	$\hat{\tau}$	$P_1, M_1, ..., P_3, M_3$	$\hat{T}$	$(\tau - \hat{T})/\hat{T}$	$(\hat{\tau} - \hat{T})/\hat{T}$
1600	1, 1, 0, 0, 0, 0	3.38	3.47	1, 1, 2, 2, 0, 0	3.33	0.014	0.042
3200	1, 4, 0, 0, 8, 1	24.64	33.49	1, 1, 2, 2, 0, 0	18.33	0.344	0.827
4800	1, 4, 0, 0, 8, 1	63.15	74.79	1, 2, 2, 2, 0, 0	51.61	0.224	0.449
6400	1, 5, 4, 3, 8, 1	108.98	129.54	1, 2, 2, 2, 0, 0	104.65	0.041	0.238
8000	1, 4, 4, 3, 8, 1	202.36	204.01	1, 2, 4, 2, 8, 1	188.60	0.073	0.082
9600	1, 5, 4, 3, 0, 0	314.81	385.58	1, 3, 4, 2, 8, 1	292.77	0.075	0.317

seconds (7.8 hours), as shown in Table 9.

Though Athlon P-T models are built from Pentium-II P-T models as stated in Section 4.1.2, Pentium-III P-T models can be constructed from the measurement results of Pentium-III. Since we have four Pentium-III processors in the cluster, it is possible to construct Pentium-III P-T models from three N-T models of $P_2 = 2, 3$, and 4. The configuration $P_2 = 1$ is not usable for P-T models, since $P_i > 1$ must hold when P-T models are used (cf. Section 3.4).

Table 10 summarizes the evaluation results of these models. As readily seen, the errors are significant, and the estimated optimal configurations are far from satisfactory. Thus, we are concerned that Pentium-III P-T models may not be accurate enough. We can construct a P-T model from three N-T models, but that is the minimum number needed to extract three coefficients.

Hence, we attempted another evaluation, in which Pentium-III P-T models are built from Pentium-II P-T models. Since Pentium-II P-T models are constructed with 8 processors, they are expected to be more accurate. Pentium-III P-T models were composed by simply scaling Pentium-II P-T models by a constant factor of 0.637. The evaluation results are shown in Table 11.

Table 11
Errors in estimated best configurations vs. measured best configurations, where scaled Pentium-II P-T models were substituted for Pentium-III P-T models.

Size	Estimated best configuration			Actual best configuration		Error	
N	$P_1, M_1, ..., P_3, M_3$	τ	$\hat{\tau}$	$P_1, M_1, ..., P_3, M_3$	$\hat{T}$	$(\tau - \hat{T})/\hat{T}$	$(\hat{\tau} - \hat{T})/\hat{T}$
1600	1, 1, 4, 1, 0, 0	3.25	4.38	1, 1, 2, 2, 0, 0	3.33	-0.023	0.315
3200	1, 2, 4, 1, 0, 0	17.11	22.98	1, 1, 2, 2, 0, 0	18.33	-0.067	0.254
4800	1, 2, 4, 1, 0, 0	47.58	59.23	1, 2, 2, 2, 0, 0	51.61	-0.078	0.148
6400	1, 2, 4, 1, 0, 0	100.48	118.90	1, 2, 2, 2, 0, 0	104.65	-0.040	0.136
8000	1, 4, 4, 1, 8, 1	179.30	218.27	1, 2, 4, 2, 8, 1	188.60	-0.049	0.157
9600	1, 4, 4, 1, 8, 1	270.37	343.51	1, 3, 4, 2, 8, 1	292.77	-0.076	0.173

The estimated optimal configurations now seem rational and sub-optimal. The errors in estimated optimal execution time are less than 8%, which is good enough for practical use. The excess time of estimated optimal configurations ranged from 13.6% to 31.5%, which is tolerable but not very satisfactory.

Let us examine the cases when $N = 4800$ and 6400, in which the estimated optimal configuration was $(P_1, M_1, P_2, M_2, P_3, M_3) = (1, 2, 4, 1, 0, 0)$ for both cases, while the actual optimal configuration was $(1, 2, 2, 2, 0, 0)$. Here, we have to keep in mind that each of our Pentium-III nodes is a dual-processor system. We used two processors of two separate nodes (one processor for each node) in the measurements of $P_2 = 2$. Since two processes on a dual-processor node are simultaneously dispatched to two individual processors, $(P_2, M_2) = (4, 1)$ is basically equivalent to $(2, 2)$. From this point of view, our models have succeeded in determining the optimal configuration for $N = 4800$ and 6400.

However, the fact is that the measured execution time of $(1, 2, 4, 1, 0, 0)$ is about 14% slower than that of $(1, 2, 2, 2, 0, 0)$, which directly leads to the excess in the estimated optimal execution time. We infer that this is due to the difference in data allocation (or process allocation). If all nodes were single-processor nodes, $(1, 2, 2, 2, 0, 0)$ would have been much slower than $(1, 2, 4, 1, 0, 0)$.

In constructing models for this present study, we did not pay any special attention to dual-processors or multi-processors. Therefore, it is very reasonable to conclude that our scheme could not handle such special cases properly but chose the case $(P_2, M_2) = (4, 1)$ rather than $(P_2, M_2) = (2, 2)$. The improvements for proper handling of the dual or multi-processors are left for future studies.

5 Conclusion

The results of this study are still preliminary, and many improvements are anticipated. Moreover, more extensive studies are required for various cluster configurations. Our aim, however, 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.

One of the major concerns of this approach might be the scalability. The estimated best configuration was not very satisfactory in Section 4.2, because of the estimation error of dual-processor nodes. This is not the problem of scalability, but the problem of modeling. We have to examine a larger heterogeneous cluster with more precise models to investigate the scalability of our method.

In the present study, all possible configurations were examined to determine the estimated optimal configuration. This took no more than a second in our experiments. However, for larger and more heterogeneous clusters, it is essential to find a way to reduce the search space. Approximation algorithms (e.g., heuristics) and a branch-and-bound method may be worth considering for this purpose.

Although our method is expected to be applicable to other parallel applications, this study was confined to one specific application (HPL). It is essential to examine other applications to show the advantages and limitations of this method. All such tasks must be left to future studies.

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 the Hori Information Science Promotion Foundation. Further support was also provided by the 21st Century COE Program "Intelligent Human Sensing" from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

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

[2] A. Kalinov, A. Lastovetsky, Heterogeneous distribution of computations while solving linear algebra problems on networks of heterogeneous computers, in: P. Sloot, M. Bubak, A. Hoekstra, B. Hertzberger (Eds.), Proc. HPCN Europe 1999, Springer, 1999, pp. 191–200.

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

[4] 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, R. C. Whaley, ScaLAPACK Users' Guide, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

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

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

[7] Y. Kishimoto, S. Ichikawa, An execution-time estimation model for heterogeneous clusters, in: Proc. 18th International Parallel and Distributed Symposium (IPDPS'04), IEEE Computer Society, 2004, (CD-ROM).

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