Akihoko FUNO, Kouji HIRATA, Yoshinobu HIGAMI, **Shin-ya KOBAYASHI**
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING, EHIME UNIVERSITY
3 Bunkyo-Cho Street, Matsuyama, Ehime, 790-8577 Japan

Optimistic Processing Protocol for Multiplexing in External PC Grids

Mr. Akihoko FUNO

He is a master course student of the Department of Electrical and Electronic Engineering and Computer Science, Graduate School of Science and Engineering, Ehime University. His research interests include distributed processing and its performance evaluation. He is a member of IEICE.

e-mail: funo@koblab.cs.ehime-u.ac.jp

Ph.D. Yoshinobu HIGAMI

He is an associate professor at Graduate School of Science and Engineering, Ehime University. In 1998 and 2006, he was also an honorary fellow at University of Wisconsin-Madison, U.S.A. He received the IEICE Best Paper Award in 2005. His research interests include test generation, design for testability and fault diagnosis of logic circuits. He is a senior member of the IEEE and a member of IEICE and IPSJ.

e-mail: higami@cs.ehime-u.ac.jp

Ph.D. Kouji HIRATA

He is an Assistant Professor in the Department of Electrical and Electronic Engineering and Computer Science, Graduate School of Science and Engineering, Ehime University. His research interests include optical networks and theirs performance evaluation. He is a member of IEEE, IPSJ and IEICE.

e-mail: hirat@cs.ehime-u.ac.jp

Ph.D. Shin-ya KOBAYASHI

He is a Professor at Graduate School of Science and Engineering, Ehime University. His research interests include distributed processing, and parallel processing. He is a member of the Information Processing Society of Japan, the Institute of Electric-al Engineers of Japan, IEICE, IEEE, and ACM.

e-mail: kob@cs.ehime-u.ac.jp

Abstract

In external PC grids, it is difficult to protect data from falsifications and analyses because the data is processed by unspecified hosts. In the past, to resolve this problem a concealing method for processing purposes (CMPP) has been proposed. Although CMPP can detect falsifications with high probability, it takes much time to process the whole program due to the majority vote. This paper proposes an optimistic processing protocol for multiplexing. In the proposed protocol, each host starts to execute its segment based on a result of a previous segment from only one host even if the result is not decided by a majority vote. The majority vote is done after results of other hosts arrive. Through simulation experiments, we show that the proposed scheme can improve processing time of programs efficiently.

Keywords: external PC grid, falsification, multiplexing, optimistic processing.

Protokół przetwarzania optymistycznego do multipleksowania w zewnętrznych sieciach komputerów osobistych

Streszczenie

W zewnętrznych sieciach komputerów osobistych trudno jest zabezpieczyć dane przed ich fałszowaniem i analizowaniem, ponieważ dane są przetwarzane przez bliżej nieokreślone serwery. W przeszłości do rozwiązania tego problemu była proponowana metoda ukrywania problemu przy przetwarzaniu (CMPP). Metoda CMPP polega na podzieleniu programu na wiele segmentów, a te segmenty są przetwarzane na różnych serwerach, aby zapobiec wykonaniu niepożądanej analizy programu. Co więcej, aby wykryć fałszerstwa, metoda CMPP zawiera schemat multipleksowania który wykonuje identyczne segmenty na różnych serwerach równolegle i decyduje o wyniku przetwarzania metodą głosowania większościowego. Chociaż CMPP może wykryć fałszerstwa z wysokim prawdopodobieństwem, to jednak głosowanie większościowe wymaga długiego czasu przetworzenia całego programu. W artykule jest zaproponowany protokół optymistycznego przetwarzania do multipleksowania. W zaproponowanym protokole każdy serwer zaczyna wykonywać własny segment w oparciu o wyniki poprzedniego segmentu z jednego z serwerów, nawet jeżeli wynik nie jest potwierdzony przez głosowanie większościowe. Głosowanie większościowe jest wykonywane po nadejściu wyników z innych serwerów. Za pomocą eksperymentów symulacyjnych wykazano, że proponowany schemat może efektywnie skrócić czas przetwarzania programów.

Słowa kluczowe: zewnętrzna sieć komputerów osobistych, fałszowanie, multipleksowanie, przetwarzanie optymistyczne.

1. Introduction

Grid computing integrates geographically distributed computing resources, such as CPUs and storages, through communication networks. A external PC grid, one of grid computing technologies, can achieve high performance computing by using computing resources of many unspecified computers in offices and homes through the Internet. Fig. 1 shows an example of a structure of external PC grids. When a user has a task to process, the user submits the task to a master node. The master node assigns the task to processing hosts which consists of unspecified computers connected to the Internet. After processing of the task, a result is sent to the user.

External PC grids have serious issues regarding security because unspecified computers process tasks. If a processed task is falsified by a malicious host, a user receives an incorrect result. Thus we have to protect tasks from such falsifications. Furthermore, contents of processed tasks may be analyzed by a malicious host, so that we cannot handle sensitive tasks in external PC grids. A protection of the tasks against falsifications and analyses is one of the most important issues in external PC grids [1, 2, 3].

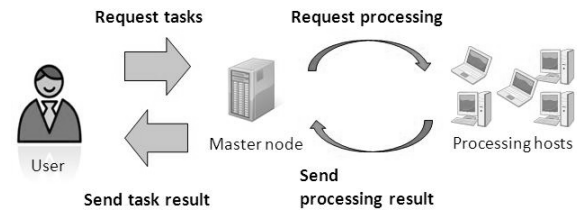

Fig. 1. A structure of external PC grids
Rys. 1. Struktura zewnętrznej sieci komputerów osobistych

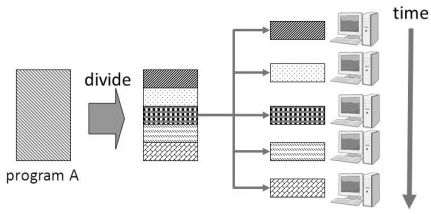

Fig. 2. Program segmentation
Rys. 2. Segmentacja programu

In order to protect tasks against falsifications and analyses, a concealing method for processing purpose (CMPP) has been proposed in the past [4, 5]. In CMPP, a program which processes a corresponding task is divided into multiple small segments and those segments are executed sequentially on different hosts as shown in Fig. 2. Each host executes a segment and a result of the segment is used as input data of a following host. By repeating such procedure, CMPP obtains a result of the whole program. Hosts hardly analyze the whole content of the program because they receive only one of those segments. Furthermore, to detect falsifications CMPP provides a multiplexing scheme which executes the identical segments at different hosts in parallel and decides a result of the execution by means of a majority vote. Although CMPP can detect falsifications with high probability, it takes much time to execute the whole program due to the majority vote. In order to resolve this problem, this paper proposes an optimistic processing protocol for multiplexing. In the proposed protocol, each host starts to execute its segment based on a result of a previous segment from only one host even if the result is not decided by a majority vote. The majority vote is done after results of other hosts arrive. Therefore, the proposed scheme is expected to improve processing time of programs efficiently.

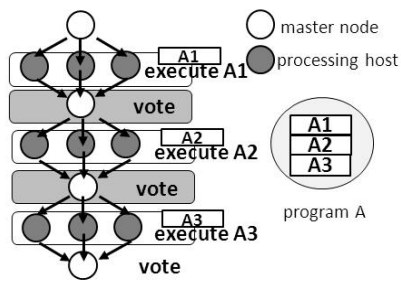

Fig. 3. Multiplexing in CMPP
Rys. 3. Multipleksowanie przy użyciu metody ukrywania problemu (CMPP)

The rest of this paper is organized as follows. In Section 2, we describe the multiplexing scheme and its shortcoming. Section 3 describes our proposed scheme. In Section 4, performance of the proposed scheme is discussed with the results of experiments. Finally, we conclude the paper in Section 5.

2. Multiplexing in CMPP

In CMPP, a program is divided into multiple segments and those segments are multiplexed as shown in Fig. 3. The identical segments are executed by N (N>1) different hosts in parallel. After executing segments, the hosts send results to a master node. The master node adopts one of results by a majority vote. As soon as the master node detects that the maximum number M of hosts which return the identical result exceeds threshold T (>N/2), the result is adopted and is sent to following hosts as their input data. In this case, the master node does not wait until all results are returned. If M is less than T after all results are returned, the segment is executed at different hosts and the master node performs a majority vote using a new result again. By repeating such voting, the proposed scheme obtains a correct result of the program.

Multiplexing can detect falsifications with high probability. However, it increases processing time of programs due to a majority vote. A external PC grid comprises many unspecified computers, so that performances of processing hosts are not homogeneous. Specifically, there exist both of high-performance computers and low-performance computers. Because a result for a segment is not adopted until M exceeds threshold T, processing time of the segment depends on low-performance computers. As a result, processing time in CMPP is large. To resolve this problem, the proposed scheme provides optimistic processing.

3. Optimistic processing protocol

The proposed scheme consists of a optimistic processing phase and a voting phase. In the optimistic processing phase, a segment is processed without a majority vote. After the optimistic processing phase, a majority vote is done in the voting phase. In what follows, we explain the details of those phases.

3.1. Optimistic processing phase

In the optimistic processing phase, a master node adopts a first result as a temporary result when the master node receives the first result from one of processing hosts. Then the master node sends the temporary result to next processing hosts without a majority vote. By doing so, processing time of a segment depends on the highest performance host in a set of hosts assigned the same segment. The next processing hosts execute their segments with the temporary result. This procedure is repeated. As a result, the proposed scheme improves processing time of programs efficiently. Fig. 4 shows an example of the proposed scheme, where the number of hosts assigned the same segment is three and a program is divided into three segments, A1, A2, and A3. When a master node receives a first result of A1 from one of processing hosts, the master node sends the result to next processing hosts assigned A2 as input data. The hosts assigned A2 then execute A2. After receiving a first result of A2 from these hosts, the master node sends the result to processing hosts assigned A3 as input data and A3 is executed. Finally, the master node receives a result of A3.

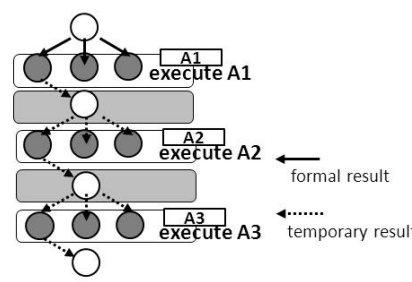

Fig. 4. Optimistic processing phase
Rys. 4. Faza przetwarzania optymistycznego

3.2. Voting phase

As mentioned earlier, in the optimistic processing phase a master node adopts a first result as a temporary result. The temporary result may be falsified by a malicious host and it is not checked in the optimistic processing phase. If the temporary result is incorrect, a result of the whole program is also incorrect. Therefore, the proposed scheme verifies that the temporary result is correct in the voting phase. Whenever the master node receives a result from a processing host assigned the same segment, the master node compares the result to the temporary result. Then the master node performs the following procedure based on the comparison.

1. If the number of hosts whose results are identical to the temporary result exceeds T, the master node adopts the

temporary result as a formal result. Then the master node sends a message, which advertises that the temporary result is correct, to successive processing hosts. The processing hosts continue to execute their segments.

2. If the number of hosts which return the identical result which is different from the temporary result exceeds T, the master node decides that the temporary result is not correct and adopts the new result as a formal result. Then the master node sends messages which abort optimistic processes to all hosts executing the optimistic processes as shown in Fig. 5. Furthermore, the master node sends the new result to next processing hosts. The processing hosts start new processes with the new result.
3. If the maximum number M of hosts which return the identical result does not exceed threshold T after all results are returned, the master node does not adopt any results and a corresponding segment is executed again at different processing hosts as follows. The master node sends abort messages to all hosts executing the optimistic processes. Then the master node selects other computers as new processing hosts and sends segments to these new hosts. The hosts execute the segments and send new results to the master node. Note that we restrict the maximum iteration count R of executions of each segment. If an iteration count exceeds R, a corresponding program is aborted.

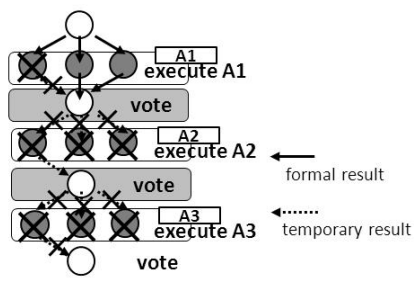

Fig. 5. Multiplexing in CMPP
Rys. 5. Multipleksowanie przy użyciu metody ukrywania problemu (CMPP)

In external PC grids, very low-performance computers may exist. Furthermore, computers may withdraw from external PC grids during processing. To address these cases, the proposed scheme uses a time-out control. If there are hosts which do not return their results during a given time interval, the proposed scheme aborts processing at these hosts. Then the proposed scheme performs a majority vote with holding results.

4. Performance Evaluation

4.1. Model

To evaluate the performance of the proposed scheme, we conduct simulation experiments. We assume that processing hosts belong to the following groups A and B.

A: Processing hosts which return correct results.
B: Processing hosts which falsify results.

We also assume that hosts belonging to B return the identical result when they execute the identical segment. We define p_1 and p_2 as ratios of the numbers of processing hosts belonging to A and B, respectively, where $p_1+p_2=1$. We assume that processing performance x of hosts follows a normal distribution with mean μ=2285.5 and variance σ^2=77447.7. For simplicity, we assume that the size D of each segment is fixed to 1000. We define t as the processing time of segments:

$$t = \frac{D}{x}.$$

Threshold T is set to be $\lceil N/2 \rceil$, where N denotes the number of processing hosts assigned the same segment. The time-out interval of each segment execution is set to be 0.7. We do not use the maximum iteration count.

Under those assumptions, we examine the performance of the proposed scheme as follows. First, we sample $N \times S$ hosts from an infinite population, where S denotes the number of different segments of a program. We then assign a segment to each host and each host processes the segment according to the proposed scheme. Finally, a master node receives a result of the whole program. We repeat this procedure and evaluate the performance of the proposed scheme. For each scenario, we collect 1000 independent samples from simulation experiments.

4.2. Results

Fig. 6 shows the average processing time of programs in the proposed scheme as a function of the number N of processing hosts assigned the same segment, where p_1=0.9; p_2=0.1, and S=10. For the sake of comparison, we plot the results of CMPP. Note that the case of N=1 means that multiplexing is not applied to CMPP. As we can see from Fig 6, the proposed scheme improves the average processing time efficiently. Furthermore, we observe that the average processing time in the proposed scheme decreases with the increase of N. The proposed scheme adopts a first result as a temporary result. Therefore, the processing time of each segment in the proposed scheme depends on the highest performance host in a set of hosts assigned the same segment. As a result, the proposed scheme can execute a program faster when N is high.

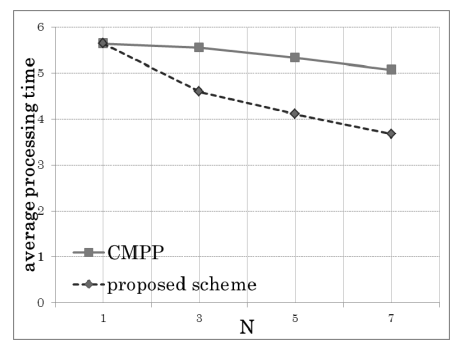

Fig. 6. Average processing time
Rys. 6. Średni czas przetwarzania

Next, we evaluate the performance of the proposed scheme against the ratio of malicious hosts. Fig. 7 shows the average processing time in the proposed scheme as a function of the ratio p_2 of malicious hosts which return incorrect results, where N=3 and S=10. We observe that the average processing time in both the proposed scheme and CMPP increases with p_2 when p_2<0.5. On the other hand, when p_2>0.5, the average processing time decreases with the increase of p_2. This is because of a majority vote. The majority vote is completed as soon as the maximum number M of hosts which return the identical result exceeds threshold T. Therefore, the increase of incorrect results delays the completion of the majority vote. Furthermore, in experiments, we assume that malicious hosts return the identical incorrect result for a segment. As a result, even if results are falsified, processing time decreases with the increase of p_2 , when p_2>0.5.

Finally, we estimate the probability P that the proposed scheme returns an incorrect result of the whole program. Note that the probability that CMPP returns an incorrect result is equal to P. The probability P_i that a result of the i-th segment is incorrect is estimated to be:

$$P_i = \sum_{k=0}^{T-1} \binom{N}{k} p_1^k p_2^{N-k}.$$

Therefore, P is estimated to be:

$$P = 1 - (1 - P_i)^S .$$

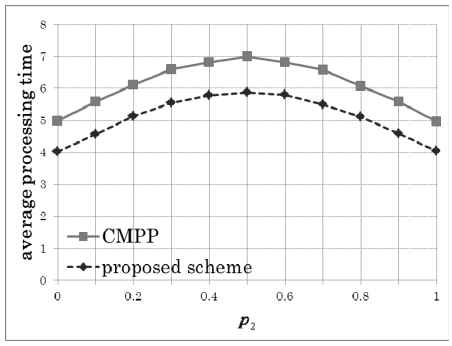

Fig. 7. Average processing time
Rys. 7. Średni czas przetwarzania

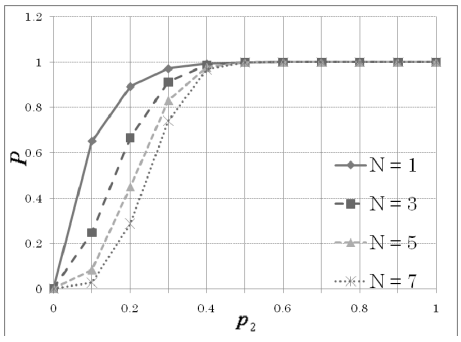

Fig. 8. Probability that the proposed scheme returns an incorrect result of the whole program
Rys. 8. Prawdopodobieństwo zwrócenia nieprawidłowego wyniku całego programu

Fig. 8 shows the probability P as a function of p_2. We observe that P decreases with the increase of N. Moreover, as shown in Fig. 6, the average processing time in the proposed scheme decreases with the increase of N. Therefore, we should use the proposed scheme with large N.

5. Conclusion

This paper proposed an optimistic processing protocol for multiplexing in CMPP. The proposed scheme starts to process a next segment based on a result from only one host even if a result of processing is not decided by a majority vote. Through simulation experiments, we showed that the proposed scheme can improve processing time of programs.

6. References

[1] Cohen F. B. Operating system protection through program evolution. Computers and Security. 12, 6, 1993.
[2] Sander T., Tschudin C. F. Towards mobile cryptography. in Proc. IEEE Symposium on Security and Privacy. pp. 215-224, 1998.
[3] Barak B., Goldreich O., Impagliazzo R., Rudich S., Sahai A., Vadhan S., and Yang K. On the (im)possibility of obfuscating programs. In Proc. the 21st Annual International Cryptology Conference on Advances in Cryptology. pp. 1-18, 2001.
[4] Kobayashi S., Morigaki S., Nelson E., Kashiwagi K., Higami Y., Hukuda M:. Code migration concealment by interleaving dummy segments. In Proc. IEEE Pacific Rim Conference on Communications. Computers and Signal Processing. pp. 269-272, 2005.
[5] Sugimoto K., Hirata K., Higami Y., Kobayashi S.: Multiplexing Scheme with Distributed Processing in External Grids. In Proc. Advanced Computer Systems, 2009.

otrzymano / received: 21.08.2010
przyjęto do druku / accepted: 01.10.2010 *artykuł recenzowany*

INFORMACJE

Nowa inicjatywa PAK

Na stronie internetowej Wydawnictwa PAK został utworzony dział: **Niepewność wyników pomiarów** w którym są zamieszczane aktualne informacje dotyczące problemów teoretycznych i praktycznych związanych z szacowaniem niepewności wyników pomiarów. W dziale znajdują się:

- aktualne informacje o publikacjach dotyczących niepewności wyników,
- informacje o przedsięwzięciach naukowo–technicznych i edukacyjnych, o tematyce związanej z niepewnością,
- dokumenty dotyczące niepewności,
- pytania do ekspertów (FAQs).

Zapraszamy:

- autorów opublikowanych prac dotyczących niepewności o nadsyłanie tekstów do zamieszczenia w tym dziale,
- organizatorów przedsięwzięć naukowo – technicznych lub edukacyjnych do nadsyłania informacji o imprezach planowanych lub odbytych,
- zainteresowanych zagadnieniami szczegółowymi do nadsyłania pytań do ekspertów.

Materiały mogą mieć formę plików lub linków do źródeł. Warunkiem zamieszczenia w tym dziale strony internetowej PAK materiałów lub linków jest przysłanie do redakcji PAK pocztą zwykłą zgody właściciela praw autorskich na takie rozpowszechnienie. Zamieszczanie i pobieranie materiałów i informacji w tym dziale strony internetowej jest bezpłatne. Redakcja PAK będzie nadzorować zawartość działu, ale za szczegółowe treści merytoryczne odpowiadają autorzy nadsyłanych materiałów.

Tadeusz SKUBIS
Redaktor naczelny Wydawnictwa PAK