行政國家科學委員會專題研究計劃成果報告

連結式全域述句偵測中無用事件之排除

Removing Useless Events in Detecting Conjunctive Global Predicates

計劃編號 :NSC 87-2213-E-009-009

執行期限 :86 年 8 月 1 日至 87 年 7 月 31 日

主持人 : 黃廷祿　　國立交通大學資訊工程學系

E-mail address: tlhuang@csie.nctu.edu.tw

一、中英文摘要

針對分散式程式進行除錯及測試時，一項很重要的關鍵技術是偵測全域述句(global predicates)的真偽值。因為一般形式的全域述句必須處理呈組合爆炸(combinatorial explosion)成長的可能狀態，故傳統上多以偵測表達能力較弱但可處理(tractable)的連結式(conjunctive)全域述句。這種做法將全域述句在邏輯上層次上分解成多個區域述句，每個區域述句皆可由單一的程序來個別偵測。各個程序會將引發區域述句成立的事件(稱之為述句事件 – predicate events)統一送至一偵測程序(checker process)，由偵測程序來決定這些個別的區域事件是否可形成一致的全域事件，進而決定此全域述句是否成立。我們提出的方法針對傳統連結式全域述句的偵測方法，提出可以利用偵測程式的空閒時間(idle time)，預先找出並移除事件佇列中無用(useless)事件的改進方法。這個方法可以減低事件佇列的空間需求並加快述句偵測的處理時間。以往的研究者所做的結果無法找出並移除所有無用的事件而我們所提出的方法卻可以。

關鍵詞 :分散式程式、分散式測試、分散式除錯、全域述句、一致全域事件、一致事件集合。

Abstract

One important task is testing and debugging a distributed program is to detect truth values of global predicates. Detecting a general-form predicate involves handling global states with combinatorial explosion. Therefore, many researchers focused on the detection of conjunctive-form global predicates which have a weaker expressivability but the problem itself becomes tractable. This approach decomposes logically a global predicate into local predicates, each of which can be detected independently by a single process. Each process sends its predicate events, the events that cause its local predicate to be true, to a dedicate process called the checker process. The checker process stores events in event queues and determines whether a collected set of events can from a consistent event set or not and therefore determines whether the global predicate is satisfied. Our proposed method provides an improvement to the conventional approach. It takes advantage of idle time to find and remove in advance useless events pending in event queues. This method can reduce the space requirement of event queues and increase the processing speed of the checker process. Previous work in this area does not find and remove all useless events while our proposed method does.

Keywords: distributed programs, distributed testing, distributed debugging, global

predicates, consistent global events, consistent event sets.

二、緣由與目的

One important task in testing and debugging a distributed program is to answer whether a given execution run of this program fulfills a particular property. Such a property is often specified as a global predicate – a Boolean expression whose value depends on the states of multiple processes and, perhaps, communication channels. Detecting global predicates involves identifying consistent global states [1], on which predicates are to be evaluated. In general, the number of consistent global states is exponential in the number of processes [2]. Therefore, an exhaustive search for all system states, which is necessary for detecting a general-form predicate, will suffer from the combinatorial explosion problem. Many researchers avoid this problem by placing restriction on the types of predicates. In particular, they have considered global predicates that can be logically decomposed into sub-expressions, each of which is locally detected by a single process [4,5,6,12]. Such sub-expressions are called local predicates. Only local states upon which local predicates are satisfied have to be examined to see if any combination of them can form a consistent global state. The number of states is therefore reduced. Each process sends its events that cause its local predicate to be true to a dedicate process called the checker process. The checker process stores events in event queues and determines whether a collected set of events can form a consistent event set and therefore determines whether the global predicate is satisfied.

The problem of finding all removable events was not previously solved [6]. Suppose that k event queues are non-empty, each of which has n events. A straightforward approach might involve looking at all n^k possible sets consisting of one event from each of the k event queues [6]. Chiou and Korfhage [6] proposed an algorithm for finding removable events that has an $O(k^2n^2)$ time complexity. Their method, however, cannot identify all useless events. Although the necessary and sufficient conditions for useless events have been formulated [9,10,14], as to the author's knowledge, no practical algorithm has been proposed. Previous work in this area cannot find and remove all useless events. We plan to find all useless events.

三、結果與討論

In our proposed method, the checker process performs two routines. The first routine examines if the event set comprising all current head events in each queue is consistent. The technique used in this routine is to repeatedly find two causally related head events and remove the one that happened before the other [4]. When no head event is removable and no event queue is empty, a consistent global predicate is identified.

Events usually do not arrive at a constant rate, so it is possible that some event queue grows lengthily while others drain. If any event queue is empty, the first routine must wait for all absent events before it can make a determination. The second routine identifies and removes all events currently pending in the queues that are unable to be consistent with other events (including those not arrived yet), even under the condition that some event queue

is empty. The main purpose of this routine is to remove useless events in advance to reduce the space requirement of the event queues and avoid further process of useless events by the first routine. This removal does not impose additional overhead on the checker process: while the first routine cannot progress, the checker process can perform this routine rather than being idle. In our proposed method, we focus on the design of the second routine.

The second routine consists of two modules. The first module collects precedence relation between event intervals. The second module removes useless predicate events by examining the collected information, and discards collected information that is obsolete. These two modules are not assumed to be executed in parallel: only one module can be activate at any instant of time. We do not prescribe any particular scheduling policy of them.

四、成果自評

Our proposed method exploits the result in [10], which established the theory of event intervals, and treats the problem of finding useless events as an on-line computation of an adjacency matrix representing an event interval graph. We avoid unlimitedly expansion of the matrix by cutting down obsolete rows and columns, saving both memory space and execution time. The validity proof of our method is provided. The simulation result indicates that the cutting technique effectively reduces the matrix size.

五、參考文獻

[1] K.M. Chandy and L. Lamport, "Distributed snapshots: Determining global states of distributed systems," *ACM Trans. Comput. Syst.*, vol. 3, pp. 63-75, Feb. 1985.

[2] R. Copper and K. Marzullo, "Consistent detection of global predicates," in *Proceedings of the ACM/ONR Workshop on parallel and Distributed Debugging, ACM SIGPLAN Notices*, vol.26, pp. 167-174, Dec. 1991.

[3] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Comm. ACM*, vol. 21, pp. 538-565, July 1978.

[4] V.K. Garg and B. Waldecker, "Detection of weak unstable predicates in distributed programs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, pp.299-307, Mar. 1994.

[5] H.-K. Chiou and W. Korfhage, " Efficient global event predicates detection," in *Proceedings of the 14th International Conference on Distributed Systems*, pp. 642-649, July 1994.

[6] K.-K Chiou and W. Korfhage, " Enhancing distributed event predicate detection algorithms," *IEEE Trans. Parallel Distrib. Syst.*, vol.7, pp. 673-676, July 1996.

[7] F. Mattern, "Virtual time and global states of distributed systems," in *Proceedings of the International Workshop on Parallel and Distributed Algorithms* (M.C. et al., ed.), (North-Holland), pp. 215-226, Elsevier Science, 1989.

[8] J. Fidge, "Timestamps in message-passing systems that preserve the partial ordering," in *Proceedings of the 11th Australian Computer Science Conference*, pp. 56-66, Feb. 1988.

[9] R.H.B. Netzer and J. Xu, "Necessary and sufficient conditions for consistent global snapshots," *IEEE Trans. Parallel Distrib. Syst.*, vol. 6, pp. 165-169, Feb. 1995.

[10] R. Baldoni, J.-M. Helary, and M. Raynal, "About state recording in asynchronous computations," in *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing*, 1996.

[11] V.K. Garg and C.M. Chase, "Distributed algorithms for detecting conjunctive predicates," in *Proceedings of the 15th International Conference on Distributed Computing Systems*, May 30 – June 2, 1995.

[12] S. Venkatesan and B. Dathan, "Testing and debugging distributed programs using global predicates," *IEEE Trans. Software Engrg.*, vol. 21, no. 2, pp.163-177, Feb. 1995.

[13] Y.-M. Wang, "Consistent gobal checkpoints that contains a given set of local checkpoints," *IEEE Trans. Comput.*, vol. 46, no. 4, pp. 456-468, Apr. 1997.

[14] D. Manivannan, R.H.B. Netzer, and M. Singhal, " Finding consistent global checkpoints in a distributed computation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, no. 6, pp.623-627, June 1997.