Information Processing Letters 111 (2011) 447–452

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Repeated detection of conjunctive predicates in distributed executions

Ajay D. Kshemkalyani

University of Illinois at Chicago, Chicago, IL 60607, United States

ARTICLE INFO

Article history:
Received 13 September 2010
Received in revised form 28 December 2010
Accepted 23 January 2011
Available online 4 February 2011
Communicated by G. Chockler

Keywords:
Distributed computing
Predicate detection
Intervals
Monitoring
Causality
Global state

ABSTRACT

Given a conjunctive predicate ϕ over a distributed execution, this paper gives an algorithm to detect *all* interval sets, each interval set containing one interval per process, in which the local values satisfy the *Definitely*(ϕ) modality. The time complexity of the algorithm is $O(n^3 p)$, where n is the number of processes and p is the bound on the number of times a local predicate becomes true at any process. The paper also proves that unlike the *Possibly*(ϕ) modality which admits $O(p^n)$ solution interval sets, the *Definitely*(ϕ) modality admits $O(np)$ solution interval sets. The paper also gives an on-line test to determine whether all solution interval sets can be detected in polynomial time under arbitrary fine-grained causality-based modality specifications.

1. Introduction

Predicate detection over a distributed execution is important for various purposes such as monitoring, synchronization and coordination, debugging, and industrial process control. Due to asynchrony in message transmissions and in local executions, different executions of the same distributed program go through different sequences of global states. We often need to make assertions about all states in all possible executions of a distributed program. Therefore, two modalities have been defined under which a predicate ϕ can hold for a distributed execution [4].

- *Possibly*(ϕ): There exists a consistent observation of the execution such that ϕ holds in a global state of the observation.
- *Definitely*(ϕ): For every consistent observation of the execution, there exists a global state of it in which ϕ holds.

An online centralized algorithm to detect *Possibly*(ϕ) and *Definitely*(ϕ) for an arbitrary predicate ϕ was given in [4]. The algorithm works by building a lattice of global states. Although it detects generalized global predicates, the time complexity of the algorithm is e^n, where e is the maximum number of events on any process, and n is the number of processes. To reduce the complexity of the algorithm, researchers focused on special classes of global predicates. Conjunctive global predicates form a popular class for many applications [11], and they can be detected under these modalities in polynomial time. This paper considers only conjunctive predicates.

For *conjunctive* predicates, there are time intervals at each process during which the local predicate is true. A global solution under the *Possibly* or *Definitely* modality identifies $\mathcal{I}$, a set of intervals, containing one interval per process in which the local predicate is true, such that the intervals in $\mathcal{I}$ are related by the modality. During such intervals, actual values of the variables, those in consecutive local states, and those in the corresponding composite global states, do not matter [1,5–8,17]. (Identifying each composite global state in a set of intervals is relevant more for non-conjunctive predicates, for which the algorithm in [4] or more efficient techniques like computation slicing [15,16] can be used.)

E-mail address: ajay@uic.edu.

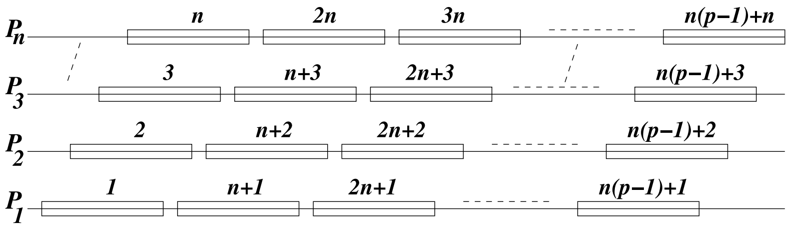

Fig. 1. Example execution using a timing diagram to illustrate the bound on the number of solution interval sets. The message-passing is not shown.

For an execution in which a local predicate becomes true at most p times at a process and n is the number of processes, the best algorithms for detecting *Possibly*(ϕ) [6] and *Definitely*(ϕ) [7] have time complexity $O(n^2p)$ at a central server process. Several distributed algorithms have also been proposed, e.g., [1,5,8,17]. However, all these algorithms detect only the *first interval set* in which ϕ is satisfied under the modality.

We address the problem of identifying *all* solution interval sets $\mathcal{I}$ in a distributed execution that satisfy the *Definitely* modality, not just the first solution set. This problem arises in sensing applications where the monitor program has to raise an alarm each time a predicate becomes true under a certain modality. For example, (i) `reset thermostat to 27deg each time "motion detected" ∧ "temp > 30deg" becomes true`; (ii) `lock the office_door each time "lights off" ∧ "no motion detected" becomes true`; and (iii) `raise alarm each time "stock_S > 85" ∧ "commodity_C ⩽ 20" becomes true`. This problem cannot be solved by simply re-executing the algorithms [6,7] to detect the modality (*Possibly* or *Definitely*, respectively). To appreciate this, consider an example execution, such as that in Fig. 1, in which there is no message communication, or messages might be sent after each interval but asynchronously reach other processes at the end of the execution. In this case, it is necessary for each interval to be considered as a possible candidate for inclusion in a global solution set $\mathcal{I}$. It is not hard to observe that there are p^n "interval sets" in the state–interval lattice. Under the *Possibly* modality, all of these interval sets are solutions to our problem – hence enumerating them will cost $\Omega(p^n)$ time. The current algorithm for detecting the first solution that satisfies *Possibly* (running in $O(n^2p)$) is clearly inadequate.

We note that the algorithm for detecting *Definitely* is very similar to that for *Possibly* and both cost $O(n^2p)$ to detect the *first* solution set. Although we cannot polynomially detect all solution sets for *Possibly*, this paper proposes an algorithm that detects *every* solution set that satisfies *Definitely* in $O(n^3p)$ time. We also prove that there are only $O(np)$ solutions (interval sets) that can satisfy the predicate under the *Definitely* modality, unlike the case for the *Possibly* modality which admits up to $O(p^n)$ solution sets.

2. Model and background

We assume an asynchronous distributed system in which n processes communicate by reliable message passing. Messages may be delivered out of order on the channels. A poset event structure model $(E, \rightarrow)$, where $\rightarrow$ is an irreflexive partial ordering representing the causality relation [12] on the event set E, is used as the model for a distributed system execution. Three kinds of events are considered: send, receive, and internal events. E is partitioned into local executions at each process. Let N denote the set of all processes. Each E_i is a totally ordered set of events executed by process P_i. We assume vector clocks are available [13,14]. Each process maintains a vector clock V of size $n = |N|$ integers, by using the following rules. (1) Before an internal event at process P_i, the process P_i executes $V_i[i] = V_i[i] + 1$. (2) Before a send event at process P_i, the process P_i executes $V_i[i] = V_i[i] + 1$. It then sends the message timestamped by V_i. (3) When process P_j receives a message with timestamp T from process P_i, P_j executes $(\forall k \in [1, \ldots, n])$ $V_j[k] = \max(V_j[k], T[k])$; $V_j[j] = V_j[j] + 1$ before delivering the message. The timestamp of an event is the value of the vector clock when the event occurs.

A *conjunctive predicate* $\phi = \bigwedge_i \phi_i$, where ϕ_i is a predicate defined on variables local to process P_i. Let us define durations of interest at each process as the durations in which the local predicate is true. Such an interval at process P_i is identified by the (totally ordered) subset of adjacent events of E_i for which the predicate is true. We use $V_i^-(X)$ and $V_i^+(X)$ to denote the vector timestamp for interval X at process P_i at the start and the end of X, respectively.

We assume that intervals X and Y occur at P_i and P_j, respectively, and are denoted as X_i and Y_j, respectively. We also assume that there are a maximum of p intervals at any process. For any two intervals X and X' that occur at the same process, if X ends before X' begins, we say that X' is a *successor* of X and denote it as $X' = succ(X)$.

For intervals X and Y, we define: $X \hookrightarrow Y$ iff $\exists x \in X, \exists y \in Y, x \rightarrow y$. The relation $\hookrightarrow$ is used by the algorithm to detect *Definitely*(ϕ). In terms of vector timestamps, $X_i \hookrightarrow Y_j$ iff $V_i^-(X_i)[i] \leqslant V_j^+(Y_j)[i]$.

The following two results [7,9] are used in the context of detecting *Definitely*(ϕ).

Theorem 1. *Let $\phi_{i,j} = \phi_i \wedge \phi_j$. Definitely($\phi_{i,j}$) holds if and only if $X_i \hookrightarrow Y_j$ and $Y_j \hookrightarrow X_i$.*

Theorem 1 holds when the local predicate is false in the initial state and final state. To uphold the theorem when ϕ_i is true in these states, one can engineer as follows. When ϕ_i is true in the initial state, P_i broadcasts a control message that is received by all in their initial states, inducing