Addressing False Causality while Detecting Predicates in Distributed Programs

Ashis Tafardar
ashis@cs.utexas.edu

Vijay K. Garg
garg@ece.utexas.edu

Parallel and Distributed Systems Laboratory
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, 78712

http://maple.ece.utexas.edu
Predicate Detection:

Does a global condition occur in a distributed computation?

Some Applications:

- distributed debugging: global bugs
  Example: Is mutual exclusion violated? \((CRIT_1 \land CRIT_2)\)

- fault-tolerance: global faults
  Example: Has a token been lost? \((\neg TOK_1 \land \neg TOK_2)\)
Goals

- The need for a new model of distributed computations
- Our results in solving predicate detection in the new model
The Interleaved Model

\[ (a, b, c) = (f, f, f) \quad (t, f, f) \quad (f, f, f) \quad (f, t, f) \quad (f, t, f) \quad (f, f, f) \quad (f, f, t) \]

false causality

\[ \text{detect predicate: } (a \land b) \]

computation, state, event
The Happened-Before Model

\[ (b, c) = (f, f) \]

\[ b := t \quad \text{ snd(a)} \]

\[ a := f \quad (t) \quad (f) \quad (f) \]

\[ b := t \quad \text{ rcv(b)} \quad c := t \]

\[ (b, c) = (f, f) \quad (t, f) \quad (f, f) \quad (f, t) \]

consistent global state, happened-before, consistent global state

detect predicates: \( (a \land b), (a \land c) \)
global state, strong causally precedes, consistent global state

detect predicate: \((a \land c)\)
Independent Events

- Multi-threading:

  ```
  create_thread(thread_1);
  c := t;
  wait(thread_1);
  rcv(b);
  ```

- Independent Actions:

  ```
  c := t || rcv(b)
  ```

- Non-blocking receives:

  ```
  x := rcv(b, NON_BLOCK);
  c := t;
  if (¬ x) then
      rcv(b);
  ```
Predicate Detection in the Happened-Before Model

... is difficult (NP-Complete) [Chase, Garg 95]

Intuition: too many global states!

\[ 3^2 = 9 \text{ global states} \]

In general, \( O(m^n) \) global states, where:
- \( m \) is the number of states in a process, and
- \( n \) is the number of processes
Predicate Detection in the Happened-Before Model

Conjunctive Predicates: [Garg, Waldecker 94]

Are two processes critical together? \((CRIT_1 \land CRIT_2)\)
... is difficult even for Conjunctive Predicates (NP-Complete)
Receive-ordered Computations

totally ordered receive states
Receive-ordered Computations

Example: Multi-threaded Server

repeat
    receive a request;
    create a thread to process the request
until done
Linearizing a computation

\[
\begin{align*}
(b, c) &= (f, f) \\
(t, f) \\
(f, c) &= (f, t) \\
(t, f) \\
(c, b) &= (f, f) \\
(t, f) \\
(f, t) \\
(c, b) &= (f, f) \\
(t, f) \\
(f, t)
\end{align*}
\]
Predicate Detection in Strong Causality Model

Key observation:

linearize each process’s computation ensuring that receive states are ordered after all concurrent states

we can now apply predicate detection as before!
Another look at general (not receive-ordered) computations:

There are an exponential number of receive-ordered computations.
But the alternative – interleaved computations – is exponentially worse.
Conclusions

- The need for a new model of distributed computations
  - modeling local independent events
  - detecting more predicates (more bugs!)

- Our results in solving predicate detection in the new model
  - Conjunctive predicate detection is NP-Complete
  - Efficient algorithm for receive-ordered computations
  - Exponential saving for general computations

Also: send-ordered computations