Predictive Analysis for Detecting Serializability Errors Through Trace Segmentation

Arnab Sinha, Sharad Malik, Princeton University

GSRC Annual Symposium

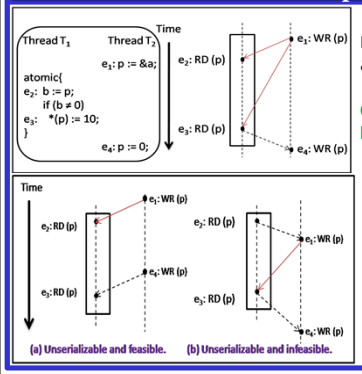
Sep 28-Oct 1, 2010

Resilient Systems, Task 5.5.1

Motivation

- ❑ More than 69% of concurrency bugs are atomicity violations.
  - ❑ detected by checking for conflict-serializability.
- ❑ Checking a single trace (trace monitoring) is efficient but limited.
- ❑ Checking all possible traces through formal verification is intractable.
- ❑ Checking alternate interleavings of a given trace (Predictive Analysis) is a compromise between formal verification and monitoring.
- ❑ Still intractable due to the large number of interleavings. *Need to prune number of interleavings considered.*
- ❑ *Need to ensure feasible interleavings to avoid false positives.*

An Example



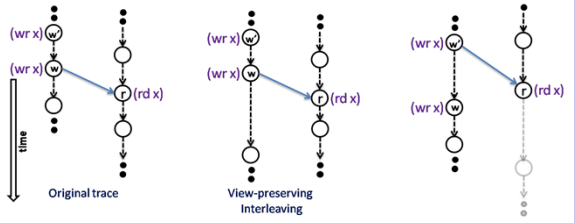
Read-couples: (e_1,e_2) &(e_1,e_3).

Original program trace might be bug-free.

(a) *Feasible* since view preserved until e_2.

(b) *Infeasible* since view is not preserved until e_2.

Almost View Preserving (AVP) Interleavings

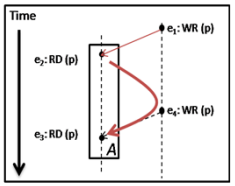


An event *v* is *skipped* in an interleaving ρ, if v_r must happen before *v* s.t. v_r is the read in a broken read-couple in ρ.

Acknowledgements

The authors acknowledge the support of the Gigascale Systems Research Center. They also thank their collaborators: Dr. Chao Wang and Dr. Aarti Gupta.

An Error Path

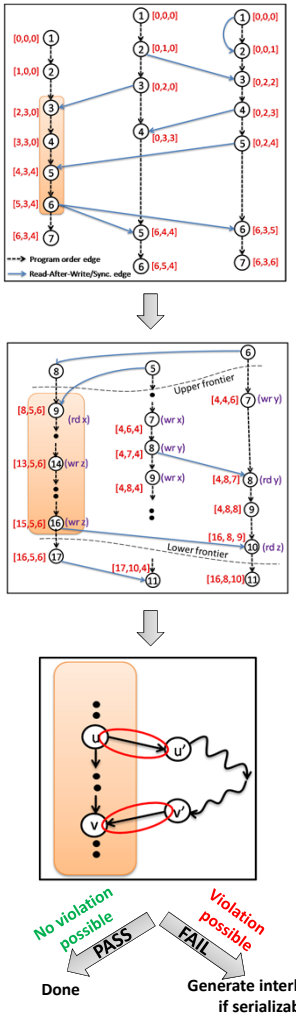


An error path P(ρ') in a feasible interleaving ρ' is a simple path of alternating (RD/WR) vertices and conflict edges in ρ', s.t.

- ❑ it originates and terminates in *A*,
- ❑ visits at least one vertex outside *A*.

Atomicity violation for *A* ⟺ Error path P(ρ').

The Methodology



Partial-order Graph:

- ❑ Vertices: events (read/write, synch. events, atomic begin & end)
- ❑ Edges: program order, read-after-writes, wait-notify, fork-first event, terminate-join.

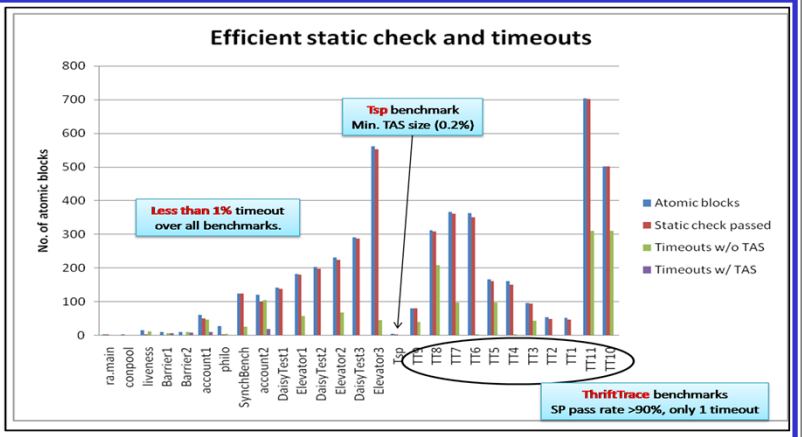
Vector:

- ❑ Vector (*k*-tuple) denotes the events of all threads which must happen before that vertex in any execution.
- ❑ Events above (below) the upper (lower) frontier must happen before (after) the atomic block in all AVP interleavings.
- ❑ Frontiers bound the *Trace Atomicity Segments* (TAS).
- ❑ Key Result: *No error-path for the atomic block can contain events outside the TAS. This limits the set of interleavings considered.*

Observation:
There must exist at least 2 events within A which conflict with other access(es) outside A for an error path.

Static Check:
Check *A* if it is devoid of at least two conflicting access.

Results



Observations:

- ❑ *The static check is very effective*. Avg. pass rate is **78.2%**.
- ❑ In SynchBench, **100%** of atomic blocks pass.
- ❑ Less than **1%** of checks timeout.
- ❑ Avg. TAS size is only **26.2%** (best case Tsp **0.2%**).
- ❑ TAS needed for non-terminating/ streaming applications.
- ❑ TAS + DPOR works well for both tightly & loosely coupled threads.

Related Work

- ❑ Semantic Analysis (*performance bottleneck!*)
  - ❑ partial interpretation of transactions [Rosu et al. TR UIUC]
  - ❑ Trace and source code analysis [Wang et al. TACAS'10]
- ❑ Complexity Study (*bogus errors reported too!*)
  - ❑ Predicted errors are not guaranteed to show up. [Farzan and Madhusudan CAV'09, TACAS'09]
- ❑ Runtime monitoring (*performance bottleneck!*)
  - ❑ Reports real bugs; performance bottleneck [Rosu et al. CAV'07, Sinha et al. IPDPS'10]

Contributions

Several major contributions in predictive analysis for atomicity checking:

- ❑ Almost View Preserving (AVP) interleavings as a useful set of feasible interleavings.
- ❑ Sufficiency of Trace Atomicity Segment (TAS) for considering all error paths in all AVP interleavings.
- ❑ Practical utility of the static check (enabled by the TAS).
- ❑ Validation through detailed experiments.