Valor: Low-Overhead Detection of Region Serializability Violations in Software

Swarnendu Biswas, Ohio State University

Collaborators: Minjia Zhang, Michael D. Bond (Ohio State University), Brandon Lucia (Carnegie Mellon University)

C++ and Java Memory Models

Data-race-free programs → Strong semantics

Synchronization-free regions execute atomically

Provides no or weak semantics for racy programs

```
A a = null;
boolean init = false;

Thread 1                          Thread 2

a = new A();                      if (init)
init = true;                        a.method();
```

Can there be a null pointer exception?

"The inability to define reasonable semantics for programs with data races is… a fundamental hole in the foundation of our languages and systems."
– Adve & Boehm, CACM 2010

Data Race Detection

Existing sound and precise techniques are expensive[1]

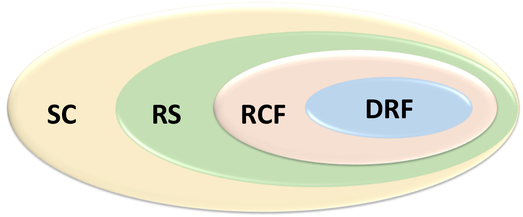
Difficult to use to provide runtime guarantees

1. Flanagan and Freund. FastTrack: Efficient and Precise Dynamic Data Race Detection. PLDI 2009.
2. Lucia et al. Conflict Exceptions: Simplifying Concurrent Language Semantics With Precise Hardware Exceptions for Data-Races. ISCA 2010.
3. Marino et al. DRFx: A Simple and Efficient Memory Model for Concurrent Programming Languages. PLDI 2010.

Region Conflict Detection

Detect all possible SFR serializability violations

An exception corresponds to a true data race



Drawbacks of existing approaches

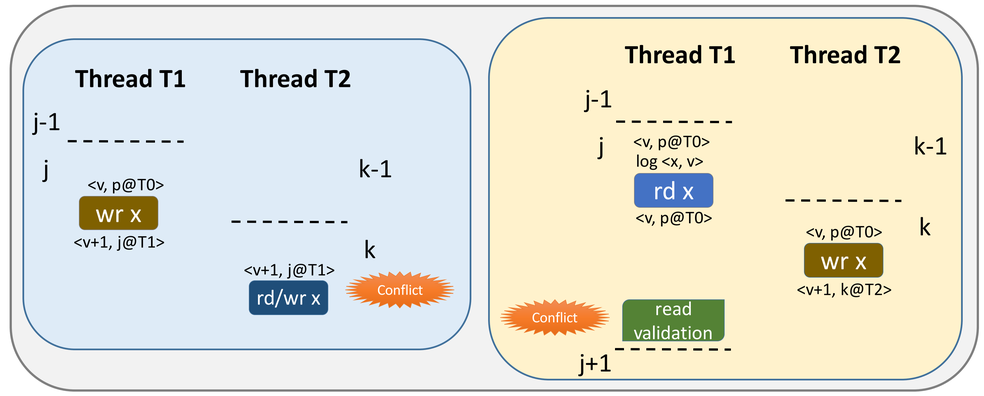
- ✗ Software implementations add high overhead
- ✗ Tracking last readers is expensive
- ✗ Hardware customizations required for performance[2,3]

Valor: Lazy Conflict Detection in Software

Elide tracking last readers, only track last writer

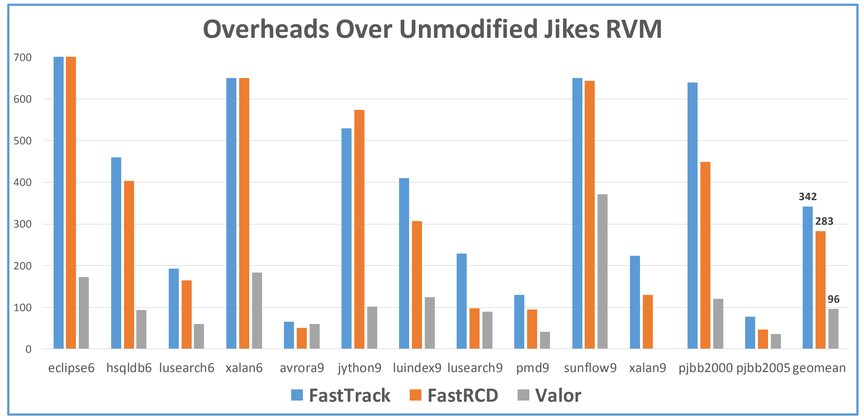
Valor: Validating anti-dependences lazily on release

Precise exceptions for **WAW** and **RAW** conflicts
WAR exceptions delayed till region boundaries



Valor must maintain versions to detect conflicts soundly and precisely

Performance



biswass@cse.ohio-state.edu