Octet: Capturing and Controlling Cross-Thread Dependences Efficiently

Ohio State
- Michael Bond
- Milind Kulkarni
- Man Cao
- Minjia Zhang
- Meisam Fathi Salmi
- Swarnendu Biswas
- Aritra Sengupta
- Jipeng Huang

Purdue
Parallel programming is mainstream

Shared memory with locks

Challenge: performance & correctness
Need practical runtime support

- Help express parallelism better
- Eliminate concurrency errors
- Diagnose production bugs
- Deal with nondeterminism
Need practical runtime support

- Atomicity checking
- Data race detection
- Record & replay
- Transactional memory
- DRF/SC enforcement
- Deterministic execution
Need **practical** runtime support

- Atomicity checking
- Data race detection
- Record & replay

**Track dependences**

- Transactional memory
- DRF/SC enforcement
- Deterministic execution

**Control dependences**

`o.f = ...` → `... = o.f`
Need **practical** runtime support

- Atomicity checking
- Data race detection
- Record & replay

**Track dependences**

- Transactional memory
- DRF/SC enforcement
- Deterministic execution

**Control dependences**

```
o.f = ...
```

```
... = o.f
```
Need **practical** runtime support

- Atomicity checking
- Data race detection
- Record & replay

*Track dependences*

- Transactional memory
- DRF/SC enforcement
- Deterministic execution

*Control dependences*

Commodity (software-only) approaches slow programs by several times
Need **practical** runtime support

- Atomicity checking
- Data race detection
- Record & replay

**Track dependences**

- Transactional memory
- DRF/SC enforcement
- Deterministic execution

**Control dependences**

Commodity (software-only) approaches slow programs by several times
Need **practical** runtime support

- Atomicity checking
- Data race detection
- Record & replay

Track dependences

- Transactional memory
- DRF/SC enforcement
- Deterministic execution

Control dependences

Any access could **race** → add **synchronization** at every access
Octet

Framework for runtime support
HB edges $\rightarrow$ all dependences
Atomicity of analysis & access

Concurrency control mechanism
Synchronization $\leftrightarrow$ cross-thread dependence
$\rightarrow$ Qualitative performance improvement
Octet

Framework for runtime support
HB edges $\rightarrow$ all dependences
Atomicity of analysis & access

Concurrency control mechanism
Synchronization $\Leftrightarrow$ cross-thread dependence
$\rightarrow$ Qualitative performance improvement

Proofs!
Octet tracks ownership

Each object’s state $\in \{\text{WrEx}_T, \text{RdEx}_T, \text{RdSh}_c\}$
o’s state = $\text{WrEx}_{T_1}$
o’s state = \( \text{WrEx}_{T_1} \)

- T1
  - write check
  - wr o.f
- T2
  - read check

Time
o’s state = $\text{WrEx}_{T_1}$

Diagram:

- T1:
  - write check
  - wr o.f

- T2:
  - read check

Time axis:

- Time arrow pointing downwards
o’s state = $\text{WrEx}_{T_1}$
T1
write check
wr o.f
safe point

T2
read check
Implicit safe point

o’s state = WrEx_{T1}
o’s state = $\text{WrEx}_{T_1}$
o’s state = $RdEx_{T2}$

- T1:
  - write check
  - wr o.f
  - safe point

- T2:
  - read check
o’s state = $\text{RdEx}_{T_2}$
o’s state = \text{RdEx}_{T2}
o’s state = \text{RdEx}_{T2}
o’s state = RdSh_c

T1:
- write check
- wr o.f
- safe point

T2:
- read check
- rd o.f

T3:
- read check
- rd o.f

T4:
o’s state = \text{RdSh}_c

- **T1**: write check
- **T2**: read check
- **T3**: read check
- **T4**: read check

- wr o.f
- safe point
- rd o.f

Path: write check, wr o.f, safe point, rd o.f, read check, read check, read check.
o’s state = $\text{RdSh}_c$
o's state = RdShₐ

Sharing detection
[von Praun & Gross ’01]
Comparison in our paper

Distributed shared memory
Shasta [Scales et al. ’96]

Biased locking
[Kawachiya et al. ’02]
[Russell & Detlefs ’06]
[Hindman & Grossman ’06]
Practical runtime support

- Atomicity checking
- Data race detection
- Record & replay

Track dependences

- Transactional memory
- DRF/SC enforcement
- Deterministic execution

Control dependences

Octet
Framework for runtime support
Concurrency control mechanism
Dependence recorder records happens-before edges.

- T1: write check
- T2: read check
- T3: read check
- T4: read check

Arrows indicate the flow of dependence checks:
- Safe point -> write check (T1)
- Safe point -> read check (T2)
- Read check (T2) -> read check (T3)
- Read check (T3) -> read check (T4)
Implementation in Jikes RVM
Publicly available
http://jikesrvm.org/Research+Archive

Parallel programs
DaCapo Benchmarks 2006 & 2009
SPEC JBB 2000 & 2005

Parallel platform
32 cores (AMD Opteron 6272)
Octet helps enable practical runtime support for reliable, scalable concurrency

Framework for runtime support
HB edges → all dependences
Atomicity of analysis & access

Concurrency control mechanism
Synchronization ⇔ cross-thread dependence
→ Qualitative performance improvement