Drinking from Both Glasses: Combining Pessimistic and Optimistic Tracking of Cross-Thread Dependences

MAN CAO
MINJIA ZHANG
ARITRA SENGUPTA
MICHAEL D. BOND

PPoPP 2016
Dynamic Analyses for Parallel Programs

- Error detection
  - Data Race Detector
  - Atomicity Violation Detector

- Programming model
  - Transactional Memory
  - Enforcement of Strong Memory Model

- Debugging
  - Record & Replay
  - Deterministic Execution
Dynamic Analyses for Parallel Programs

- Error detection
  - Data Race Detector
  - Atomicity Violation Detector
- Programming model
  - Transactional Memory
  - Enforcement of Strong Memory Model
- Debugging
  - Record & Replay
  - Deterministic Execution

Bad performance!
Dynamic Analyses for Parallel Programs

- Error detection
  - Data Race Detector
  - Atomicity Violation Detector
- Programming model
  - Transactional Memory
  - Enforcement of Strong Memory Model
- Debugging
  - Record & Replay
  - Deterministic Execution

Bad performance!

Difficulties?
Cross-thread dependences

T1

\[ \text{o.f} = \ldots \]

T2

\[ \ldots = \text{o.f} \]
Cross-thread dependences

Tracking cross-thread dependences
Cross-thread dependences

Tracking cross-thread dependences

- Detecting
- Controlling
Outline

◦ Dynamic Analyses and Cross-thread Dependences
◦ Pessimistic Tracking
◦ Optimistic Tracking
◦ Our approach
  ◦ Hybrid Tracking
◦ Evaluation
Pessimistic Tracking

Per-object metadata: o.state
last writer/reader thread
Pessimistic Tracking

Per-object metadata: o.state
last writer-reader thread
At each object access:

rd/wr o.f
Pessimistic Tracking

Per-object metadata: o.state
last writer/reader thread
At each object access:

- Check o.state
- Analysis-specific work
- rd/wr o.f
- Update o.state
Pessimistic Tracking

Per-object metadata: o.state
last writer/reader thread
At each object access:

- Check o.state
- Analysis-specific work
- rd/wr o.f
- Update o.state
Pessimistic Tracking

Per-object metadata: o.state
last writer/reader thread
At each object access:

1. Lock o.state
2. Check o.state
3. Analysis-specific work
4. rd/wr o.f
5. Update o.state
6. Unlock o.state
Pessimistic Tracking

Per-object metadata: o.state
last writer/reader thread
At each object access:

- Lock o.state
- Check o.state
- Analysis-specific work
- rd/wr o.f
- Update o.state
- Unlock o.state

NOT program locks
Pessimistic Tracking

- Lock o.state
- ...
- wr o.f
- Unlock o.state

- Lock o.state
- ...
- rd o.f
- Unlock o.state
Pessimistic Tracking

Lock o.state
...
wr o.f
Unlock o.state

Synchronization and write metadata

Lock o.state
...
rd o.f
Unlock o.state
Performance of Pessimistic Tracking Alone

Overhead (%)

- Eclipse6: 29,000
- hsqlb6: 1,300
- lusearch6: 340%
- xalan6: 29,000
- avrora9: 1,300
- jython9: 340%
- luindex9: 29,000
- lusearch9: 1,300
- pmd9: 340%
- sunflow9: 29,000
- xalan9: 1,300
- jbb2000: 340%
- jbb2005: 29,000
- geomean: 1,300
Outline

- Dynamic Analyses and Cross-thread Dependences
- Pessimistic Tracking
- Optimistic Tracking
- Our approach
  - Hybrid Tracking
- Evaluation
Optimistic Tracking

- Biased reader-writer lock for o.state
Optimistic Tracking

- Biased reader-writer lock for o.state
- Avoid synchronization for non-conflicting accesses
Optimistic Tracking

- Biased reader-writer lock for o.state
- Avoid synchronization for non-conflicting accesses
- Heavyweight coordination for conflicting accesses
Optimistic Tracking

o.state: WrExT1

write check
wr o.f
Optimistic Tracking

T1
- o.state: WrExT1
  - write check
  - wr o.f

T2
- read check
- o.state: WrExT1
Optimistic Tracking

T1

write check
wr o.f

T2

read check
Analysis-specific work

o.state: WrExT1

Request
Optimistic Tracking

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

T2
- read check
- Analysis-specific work
- change state
- rd o.f

o.state: WrExT1

Request → Response

o.state ← RdExT2
Performance of Optimistic Tracking Alone

[Bar chart showing overhead (%) for various benchmarks with pessimistic and optimistic tracking.]

- **Pessimistic Tracking**
- **Optimistic Tracking**

28%
Performance of Optimistic Tracking Alone

- Overhead (%)
  - Pessimistic Tracking
  - Optimistic Tracking

Performance of Optimistic Tracking Alone: 28%
Cost of Different Tracking

<table>
<thead>
<tr>
<th>Pessimistic</th>
<th>Optimistic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Same state</td>
</tr>
<tr>
<td>150</td>
<td>47</td>
</tr>
</tbody>
</table>

- In CPU cycles
- Averaged across all programs
Optimistic tracking performs best if there are few conflicting accesses.
Pessimistic tracking is cheaper for conflicting accesses.
Drink from both glasses?

Goal

◦ **Optimistic** tracking for most non-conflicting accesses
◦ **Pessimistic** tracking for most conflicting accesses
Outline

- Dynamic Analyses and Cross-thread Dependences
- Pessimistic Tracking
- Optimistic Tracking
- Our approach
  - Hybrid Tracking
- Evaluation
Outline

- Dynamic Analyses and Cross-thread Dependences
- Pessimistic Tracking
- Optimistic Tracking
- Our approach
  - Hybrid Tracking
- Evaluation

Hybrid State Model
Adaptive Policy
Outline

- Dynamic Analyses and Cross-thread Dependences
- Pessimistic Tracking
- Optimistic Tracking
- Our approach
  - Hybrid Tracking
- Evaluation

Hybrid State Model
Adaptive Policy

Challenging!
Outline

- Dynamic Analyses and Cross-thread Dependences
- Pessimistic Tracking
- Optimistic Tracking
- Our approach
  - Hybrid Tracking
- Evaluation

Hybrid State Model
Deferred Unlocking
Adaptive Policy
Pessimistic-Optimistic Mismatch

Pessimistic Tracking

- Lock o.state
- ...
- ...
- rd/wr o.f
- Unlock o.state

Optimistic Tracking

- write check
- wr o.f
- read check
- ...
- change state
- rd o.f
Pessimistic-Optimistic Mismatch (#1)

Pessimistic Tracking

- Lock o.state
- ...
- ...
- rd/wr o.f
- Unlock o.state

Optimistic Tracking

- write check
- wr o.f
- read check
- ...
- change state
- safe point
- rd o.f
- No unlock!
Pessimistic-Optimistic Mismatch (#1)

Pessimistic Tracking

- Lock o.state
- ...
- ...
- rd/wr o.f
- Unlock o.state

Optimistic Tracking

- write check
- wr o.f
- read check
- ...
- change state
- safe point
- rd o.f

Conditional unlock?

No unlock!
Pessimistic-Optimistic Mismatch (#2)

Pessimistic Tracking
- Lock o.state
- ...
- rd/wr o.f
- Unlock o.state

Optimistic Tracking
- write check
- wr o.f
- read check
- ...
- change state
- rd o.f
- safe point
Pessimistic-Optimistic Mismatch (#2)

Pessimistic Tracking

- Lock o.state
- ...
- rd/wr o.f
- Unlock o.state

Atomic

Optimistic Tracking

- write check
- wr o.f
- safe point
- read check
- ...
- change state
- rd o.f

Atomic

Atomicity granularity may affect specific analysis
Pessimistic-Optimistic Mismatch (#2)

Pessimistic Tracking

- Lock o.state
- ...
- rd/wr o.f
- Unlock o.state

Optimistic Tracking

- write check
- wr o.f
- read check
- ...
- change state
- safe point
- rd o.f

Atomicity granularity may affect specific analysis
Conditional unlock
Key Insights

- **Coarsening** atomicity granularity for pessimistic tracking
Key Insights

• **Coarsening** atomicity granularity for pessimistic tracking

• **Program synchronization** may hint at cross-thread dependences
Addressing Pessimistic-Optimistic Mismatch

Defer unlocking of pessimistic state
  ◦ Till program synchronization release operation (PSRO)
Addressing Pessimistic-Optimistic Mismatch

Defer unlocking of pessimistic state

- Till program synchronization release operation (PSRO)
- Reader-writer locking
Addressing Pessimistic-Optimistic Mismatch

Defer unlocking of pessimistic state

- Till program synchronization release operation (PSRO)
- Reader-writer locking
- Fall back to coordination on contention when locking a state
Deferred Unlocking Example 1

synchronized (m) {
    Lock o.state
    wr o.f
    ...  
    Unlock all states
}

synchronized (m) {
    Lock o.state
    rd o.f
    ...
Deferred Unlocking Example 2

`synchronized (m) {
    Lock o.state
    wr o.f
    ...
    safe point
    Unlock all states
    ...
}

Lock o.state
rd o.f
Hybrid State Model

- **Pessimistic Unlocked**
  - Any access (uncontended)
- **Pessimistic Locked**
  - Conflicting access (contended)
  - Non-conflicting access (uncontended & possibly reentrant)
- **Optimistic**
  - Non-conflicting access
  - Conflicting access

Key:
- **Load or CAS**
- **Coordination**
- **Adaptive policy**
Hybrid State Model

- **Pessimistic Unlocked**: Any access (uncontended)
- **Pessimistic Locked**: Conflicting access (contended)
- **Optimistic**: Non-conflicting access

**Transitions**:
- **Load or CAS**:- Coordination
- **Adaptive policy**

**States**:
- **Pessimistic Unlocked**
- **Pessimistic Locked**
- **Optimistic**

**Actions**:
- PSRO & responding safe point
- Non-conflicting access
- Conflicting access
Outline

- Dynamic Analyses and Cross-thread Dependences
- Pessimistic Tracking
- Optimistic Tracking
- Our approach
  - Hybrid Tracking
- Evaluation

Hybrid State Model
Deferred Unlocking
Adaptive Policy
Adaptive Policy

Decide *when* to transition *between* pessimistic and optimistic states
Adaptive Policy

Decide *when* to transition between pessimistic and optimistic states

- Cost—benefit model
Adaptive Policy

Decide *when* to transition *between* pessimistic and optimistic states

- Cost—benefit model
  - Boil down to *counting* state transitions
Adaptive Policy

Decide **when** to transition **between** pessimistic and optimistic states

- **Cost**—benefit model
  - Boil down to **counting** state transitions

- **Online** profiling
  - Per-object
  - Simple yet effective
Application of Hybrid Tracking

Two dynamic analyses
- Hybrid dependence \textit{recorder} and replayer (detect)
- Hybrid region serializability (RS) \textit{enforcer} (control)
Application of Hybrid Tracking

Two dynamic analyses
  ◦ Hybrid dependence **recorder** and replayer (detect)
  ◦ Hybrid region serializability (RS) **enforcer** (control)

Deferred unlocking helps overcome key challenges!
Outline

- Dynamic Analyses and Cross-thread Dependences
- Pessimistic Tracking
- Optimistic Tracking
- Our approach
  - Hybrid Tracking
- Evaluation
Implementation

Jikes RVM 3.1.3
Implementation

Jikes RVM 3.1.3

Pessimistic tracking, optimistic tracking
- [Octet, Bond et al. OOPSLA’13]

Optimistic recorder and replayer
- [Replay, Bond et al. PPPJ’15]

Optimistic RS enforcer
- [EnfoRSer, Sengupta et al, ASPLOS’15]
Implementation

Jikes RVM 3.1.3

Pessimistic tracking, optimistic tracking
- [Octet, Bond et al. OOPSLA’13]

Optimistic recorder and replayer
- [Replay, Bond et al. PPPJ’15]

Optimistic RS enforcer
- [EnfoRSer, Sengupta et al, ASPLOS’15]

Hybrid tracking, hybrid **recorder** and replayer, hybrid RS enforcer
- publicly available
Performance of Tracking

Overhead (%)

- Pessimistic Tracking
- Optimistic Tracking
- Hybrid Tracking

22%
Performance of Tracking

- Overhead (%)
- Pessimistic Tracking
- Optimistic Tracking
- Hybrid Tracking

<table>
<thead>
<tr>
<th>Software</th>
<th>Pessimistic Tracking</th>
<th>Optimistic Tracking</th>
<th>Hybrid Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>eclipse6</td>
<td>140</td>
<td>290</td>
<td>500</td>
</tr>
<tr>
<td>hsqldb6</td>
<td>290</td>
<td>290</td>
<td>240</td>
</tr>
<tr>
<td>lusearch6</td>
<td>290</td>
<td>290</td>
<td>240</td>
</tr>
<tr>
<td>xalan6</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>avrora9</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>jython9</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>luindex9</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>lusearch9</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>pmd9</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>sunflow9</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>xalan9</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>jbb2000</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>jbb2005</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
<tr>
<td>geomean</td>
<td>29,000</td>
<td>1,300</td>
<td>120</td>
</tr>
</tbody>
</table>
Performance of Tracking

Most programs are low-conflict

22%
Performance of Recorders and RS enforcers

![Chart showing the performance of optimist and hybrid recorders and enforcers.](chart.png)
Additional Materials

Complete State Transition Table

Run-time Characteristics

Stress Tests

Please check the paper
Related work

Analyses that use pessimistic tracking
- [FastTrack, Flanagan & Freund, PLDI’09]
- [Velodrome, Flanagan et al., PLDI’08]
- [Chimera, Lee et al., PLDI’12]
- [Lightweight Transactions, Harris & Fraser, OOPSLA’03]
- [DMP, Devietti et al., ASPLOS’09]

Analyses that use optimistic tracking
- [Shasta, Scales et al. ASPLOS’96]
- [Object Race Detection, von Praun & Gross, OOPSLA’01]
- [DoubleChecker, Biswas et al. PLDI’14]
- [LarkTM, Zhang et al, PPoPP’15]

Adaptive Mechanisms
- [Adaptive Locks, Usui et al. PACT’09]
- [Strong Atomicity TM, Abadi et al. PPoPP’09]
- [Adaptive Lock Elision, Dice et al, SPAA’14]
- [Concurrency Control, Ziv et al, PLDI’15]
Contributions

Hybrid tracking combines pessimistic tracking and optimistic tracking effectively and efficiently

Hybrid tracking achieves better overall performance

- Never significantly degrades performance
- Sometimes improves performance substantially
- Suitable for workload of diverse communication patterns