Composable Specifications for Structured Shared-Memory Communication

Benjamin P. Wood, Adrian Sampson, Luis Ceze, Dan Grossman

University of Washington
Code-Communication Specifications

Writer Thread

enqueue(...);

Reader Thread

dequeue();
May writes in `enqueue` be read by other threads in `dequeue`?
Code-Communication Specifications

Writer Thread

Reader Thread

What **code** may communicate across threads?

May writes in `enqueue` be read by other threads in `dequeue`?
Code-Communication Specifications

What **code** may communicate across threads?

- **enqueue**
- **dequeue**

May writes in **enqueue** be read by other threads in **dequeue**?
What **code** may communicate across threads?

- `enqueue` → `dequeue`
- `enqueue` → `render`
- `dequeue()`

May writes in `enqueue` be read by other threads in `dequeue`?
Implicitly Shared Memory

```javascript
this.buffer[...] = i;
this.size = this.size + 1;
```
Implicitly Shared Memory

What is shared? What is not?

```javascript
this.buffer[...] = i;
this.size = this.size + 1;
```
Implicitly Shared Memory

```javascript
this.buffer[...] = i;
this.size = this.size + 1;
```

What is shared? What is not?

Thread-private?
Implicitly Shared Memory

Read-only?

```javascript
this.buffer[...] = i;
What is shared? What is not?
this.size = this.size + 1;
```

Thread-private?
Implicitly Shared Memory

this.buffer[...] = i;
What is shared? What is not?
this.size = this.size + 1;

Read-only?

Guarded by lock?

Thread-private?
Implicitly Shared Memory

this.buffer[...] = i;

What is shared? What is not?

this.size = this.size + 1;

Guarded by lock?

Race-free?

Thread-private?

Read-only?
Implicitly Shared Memory

this.buffer[...] = i;
What is shared? What is not?
this.size = this.size + 1;

Read-only?
Race-free?
Atomic?

Guarded by lock?
Thread-private?
These are properties of data or isolation.

- Read-only?
- Race-free?
- Atomic?
- Guarded by lock?
- Thread-private?

this.buffer[...] = i;
What is shared? What is not?
this.size = this.size + 1;
Data- and Isolation-Centric Analyses

**Race detection**
e.g. FastTrack [PLDI’09], Goldilocks [PLDI’07], Effective Static Race Detection [PLDI’06]

**Sharing specifications**
e.g. SharC [PLDI’08], Shoal [PLDI’09], Ownership Policies [POPL’10]

**Atomicity violation detection**
e.g. Velodrome [PLDI’08], A Type and Effect System for Atomicity [PLDI’03]
Data- and Isolation-Centric Analyses

Race detection
e.g. FastTrack [PLDI’09], Goldilocks [PLDI’07], Effective Static Race Detection [PLDI’06]

Are all accesses to location x well-synchronized?

Sharing specifications
e.g. SharC [PLDI’08], Shoal [PLDI’09], Ownership Policies [POPL’10]

Atomicity violation detection
e.g. Velodrome [PLDI’08], A Type and Effect System for Atomicity [PLDI’03]
Data- and Isolation-Centric Analyses

**Race detection**
e.g. FastTrack [PLDI’09], Goldilocks [PLDI’07], Effective Static Race Detection [PLDI’06]

Are all accesses to **location x** well-synchronized?

**Sharing specifications**
e.g. SharC [PLDI’08], Shoal [PLDI’09], Ownership Policies [POPL’10]

Which **locations** may be shared?

**Atomicity violation detection**
e.g. Velodrome [PLDI’08], A Type and Effect System for Atomicity [PLDI’03]
Data- and Isolation-Centric Analyses

**Race detection**
e.g. FastTrack [PLDI’09], Goldilocks [PLDI’07], Effective Static Race Detection [PLDI’06]

Are all accesses to **location x** well-synchronized?

**Sharing specifications**
e.g. SharC [PLDI’08], Shoal [PLDI’09], Ownership Policies [POPL’10]

Which **locations** may be shared?

**Atomicity violation detection**
e.g. Velodrome [PLDI’08], A Type and Effect System for Atomicity [PLDI’03]

Are accesses in this code section **isolated**?
What Shared-Memory Bugs Can We Catch?
What Shared-Memory Bugs Can We Catch?

**Data-centric:**
- illegal sharing
- data races

**Isolation-centric:**
- atomicity violations
What Shared-Memory Bugs Can We Catch?

- **Data-centric:**
  - illegal sharing
  - data races

- **Isolation-centric:**
  - atomicity violations

- **Code-centric:**
  - illegal communication
Outline

A Code-Centric View of Shared-Memory

**Code-Communication Specification Language**

- Making Specifications Modular and Concise
- Specification Language Evaluation

**Dynamic Specification Checker**

- Making Communication Checking Fast Enough
- Performance Evaluation
Specification Constructs
Specification Constructs

Module

A set of related methods
(often aligned with data abstractions)
## Specification Constructs

**Module**

A set of related methods  
(often aligned with data abstractions)

**Module Specification**

Which pairs of methods may communicate
## Specification Constructs

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module</strong></td>
<td>A set of related methods (often aligned with data abstractions)</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td><strong>Specification</strong></td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td><strong>Interface</strong></td>
</tr>
<tr>
<td></td>
<td>Which pairs of methods may communicate</td>
</tr>
<tr>
<td></td>
<td>Which communication is encapsulated or visible to callers outside the module</td>
</tr>
</tbody>
</table>
## Specification Constructs

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module</strong></td>
<td>A set of related methods (often aligned with data abstractions)</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>Which pairs of methods may communicate</td>
</tr>
<tr>
<td><strong>Specification</strong></td>
<td>Which communication is encapsulated or visible to callers outside the module</td>
</tr>
<tr>
<td><strong>Interface</strong></td>
<td>Which communication is encapsulated or visible to callers outside the module</td>
</tr>
<tr>
<td><strong>Inlining</strong></td>
<td>Assigns communication to the caller</td>
</tr>
</tbody>
</table>
Inter-Thread Communication

Writer Thread

buffer[3] = ...;
Inter-Thread Communication

Writer Thread

Reader Thread

buffer[3] = ...; return buffer[3];

communication
Inter-Thread Communication

Writer Thread  Reader Thread

in enqueue(...):

buffer[3] = ...; return buffer[3];

communication
Inter-Thread Communication

Writer Thread

Reader Thread

in enqueue(...):

buffer[3] = ...;

return buffer[3];

Code communication is **directed**.
Inter-Thread Communication

Writer Thread

in produce(...):

Reader Thread

in consume(...):

in enqueue(...):

in dequeue(...):

buffer[3] = ...;  return buffer[3];

communication
Inter-Thread Communication

Writer Thread

in produce(...):

buffer[3] = ...;

Reader Thread

in consume(...):

return buffer[3];

in enqueue(...):

in dequeue(...):

Code communication is layered.
package buffer;
public class BoundedBuffer {
    Item[] buffer = new Item[10];
    int size = 0;

    public synchronized void enqueue(Item i) {
        while (size == buffer.length) wait();
        buffer[size] = i;
        size++;
        notifyAll();
    }

    public synchronized Item dequeue() {
        while (size == 0) wait();
        size--;
        notifyAll();
        return buffer[size--];
    }
}

package pipeline;
import buffer.BoundedBuffer;
class Pipeline {
    BoundedBuffer pipe;

    // Producer threads
    void produce() {
        ... pipe.enqueue(...); ...
    }

    // Consumer threads
    void consume() {
        ... = pipe.dequeue(); ...
    }
}
package buffer;
public class BoundedBuffer {
    Item[] buffer = new Item[10];
    int size = 0;

    public synchronized void enqueue(Item i) {
        while (size == buffer.length) wait();
        buffer[size] = i;
        size++;
        notifyAll();
    }

    public synchronized Item dequeue() {
        while (size == 0) wait();
        size--;
        notifyAll();
        return buffer[size];
    }
}

package pipeline;
import buffer.BoundedBuffer;
class Pipeline {
    BoundedBuffer pipe;

    // Producer threads
    void produce() {
        ... pipe.enqueue(...); ...
    }

    // Consumer threads
    void consume() {
        ... = pipe.dequeue(); ...
    }
}

Module Specification
Communication Modules

package pipeline;
import buffer.BoundedBuffer;

class Pipeline {
    BoundedBuffer pipe;

    // Producer threads
    void produce() {
        ... pipe.enqueue(...); ...
    }

    // Consumer threads
    void consume() {
        ... = pipe.dequeue(); ...
    }
}

package buffer;
public class BoundedBuffer {
    Item[] buffer = new Item[10];
    int size = 0;

    public synchronized void enqueue(Item i) {
        while (size == buffer.length) wait();
        buffer[...] = i;
        size++;
        notifyAll();
    }

    public synchronized Item dequeue() {
        while (size == 0) wait();
        size--;
        notifyAll();
        return buffer[...];
    }
}
package buffer;
public class BoundedBuffer {
    Item[] buffer;
    int size = 0;

    public synchronized void enqueue(Item i) {
        while (size == buffer.length) wait();
        buffer[size] = i;
        size++;
        notifyAll();
    }

    public synchronized Item dequeue() {
        while (size == 0) wait();
        size--;
        notifyAll();
        return buffer[size];
    }
}

package pipeline;
import buffer.BoundedBuffer;
class Pipeline {
    BoundedBuffer pipe;

    // Producer threads
    void produce() {
        ... pipe.enqueue(...); ... 
    }

    // Consumer threads
    void consume() {
        ... = pipe.dequeue(); ...
    }
}
package buffer;
public class BoundedBuffer {
  Item[] buffer = new Item[10];
  int size = 0;

  public synchronized void enqueue(Item i) {
    while (size == buffer.length)
      wait();
    buffer[size] = i;
    size++;
    notifyAll();
  }

  public synchronized Item dequeue() {
    while (size == 0)
      wait();
    size--;
    notifyAll();
    return buffer[size];
  }
}

package pipeline;
import buffer.BoundedBuffer;
class Pipeline {
  BoundedBuffer pipe;

  // Producer threads
  void produce() {
    ... pipe.enqueue(...);
  }

  // Consumer threads
  void consume() {
    ... = pipe.dequeue();
  }
}

Module Specification
package pipeline;
import buffer.BoundedBuffer;

class Pipeline {
  BoundedBuffer pipe;

  // Producer threads
  void produce() {
    ... pipe.enqueue(...); ...
  }

  // Consumer threads
  void consume() {
    ... = pipe.dequeue(); ...
  }
}

package buffer;
public class BoundedBuffer {
  Item[] buffer = new Item[10];
  int size = 0;

  public synchronized void enqueue(Item i) {
    while (size == buffer.length) wait();
    buffer[...] = i;
    size++;
    notifyAll();
  }

  public synchronized Item dequeue() {
    while (size == 0) wait();
    size--;
    notifyAll();
    return buffer[...];
  }
}
Checking Communication Specifications

Writer Thread

in produce(...):

in enqueue(...):

buffer[3] = ...;

Reader Thread

in consume(...):

in dequeue(...):

return buffer[3];
Checking Communication Specifications

**Writer Thread**

```plaintext
in produce(...):
buffer[3] = ...;
```

**Reader Thread**

```plaintext
in consume(...):
return buffer[3];
```

**in enqueue(...):**

**in dequeue(...):**
Checking Communication Specifications

Writer Thread

in produce(...):

in enqueue(...):

buffer[3] = ...;

Reader Thread

in consume(...):

in dequeue(...):

return buffer[3];
Checking Communication Specifications

Writer Thread

in produce(...):

in enqueue(...):

buffer[3] = ...;

Reader Thread

in consume(...):

in dequeue(...):
Checking Communication Specifications

Writer Thread

in produce(...):

Buffer[3] = ...;

Reader Thread

in consume(...):

return Buffer[3];

in enqueue(...):

in dequeue(...):
Checking Communication Specifications

Writer Thread ☑ Reader Thread

in produce(...): ➔ in consume(...):

in enqueue(...): ➔ in dequeue(...):

buffer[3] = ...; return buffer[3];
Communication Module Interfaces

Writer Thread  |  Reader Thread

in consume(...):

in deqeuue(...):

size--;  

... = size;

in produce(...):

in enqueue(...):
Communication Module Interfaces

Writer Thread

in consume(...):

size--;

Reader Thread

in produce(...):

in dequeue(...):

... = size;

in enqueue(...):
Communication Module Interfaces

Writer Thread

in consume(...):

size--;  

in dequeue(...):

... = size;

Reader Thread

in produce(...):

−→

in enqueue(...):

−→

−→
package pipeline;
import buffer.BoundedBuffer;
class Pipeline {
    BoundedBuffer pipe;

    // Producer threads
    void produce() {
        ...; pipe.enqueue(...); ...
    }

    // Consumer threads
    void consume() {
        ... = pipe.dequeue(); ...
    }
}

package buffer;
public class BoundedBuffer {
    Item[] buffer = new Item[10];
    int size = 0;

    public synchronized void enqueue(Item i) {
        while (size == buffer.length) wait();
        buffer[...] = i;
        size++; ...
        notifyAll();
    }

    public synchronized Item dequeue() {
        while (size == 0) wait();
        size--; ...
        notifyAll();
        return buffer[...];
    }
}
package pipeline;
import buffer.BoundedBuffer;
class Pipeline {
    BoundedBuffer pipe;

    // Producer threads
    void produce() {
        ...; pipe.enqueue(...); ...
    }

    // Consumer threads
    void consume() {
        ... = pipe.dequeue(); ...
    }
}

package buffer;
public class BoundedBuffer {
    Item[] buffer = new Item[10];
    int size = 0;

    public synchronized void enqueue(Item i) {
        while (size == buffer.length) wait();
        buffer[...] = i;
        size++; ...
        notifyAll();
    }

    public synchronized Item dequeue() {
        while (size == 0) wait();
        size--; ...
        notifyAll();
        return buffer[...];
    }
}
Communication Module Interfaces

Writer Thread

in consume(...):

size--; ...

Reader Thread

in produce(...):

in dequeue(...):

in enqueue(...):

... = size;
Communication Module Interfaces

Writer Thread

in consume(...):

size--;

Reader Thread

in produce(...):

... = size;

in dequeue(...):

in enqueue(...):
Communication Module Interfaces

Writer Thread

in consume(...):

Reader Thread

in produce(...):

in dequeue(...): → in enqueue(...):

size--; ...

... = size;
Communication Module Interfaces

Writer Thread

Reader Thread

in consume(...):

in produce(...):

encapsulated

in dequeue(...):

in enqueue(...):

size--;

... = size;
Communication Inlining

Writer Thread

in enqueue(...):

in arrayCopy(...):

buffer[3] = ...;

Reader Thread

in dequeue(...):

return buffer[3];
Communication Inlining

Writer Thread

\[\text{in enqueue(...):}\]

\[\text{in arrayCopy(...):}\]

\[\text{buffer[3] = \ldots;}\]

Reader Thread

\[\text{in dequeue(...):}\]

\[\text{return buffer[3];}\]

arrayCopy communicates only for its caller.
Communication Inlining

Writer Thread

in enqueue(...):

@Inline arrayCopy(...):

buffer[3] = ...;

Reader Thread

in dequeue(...): return buffer[3];

arrayCopy communicates only for its caller.
Communication Inlining

Writer Thread

Reader Thread

in enqueue(...):

buffer[3] = ...;

in dequeue(...):

return buffer[3];

arrayCopy communicates only for its caller.
Communication Inlining

Writer Thread ✔ Reader Thread

in enqueue(...):  in dequeue(...):

buffer[3] = ...; return buffer[3];

arrayCopy communicates only for its caller.
Specification Constructs
Specification Constructs

**Module**
A set of related methods
(often aligned with data abstractions)
Specification Constructs

Module
- A set of related methods
  (often aligned with data abstractions)

Module Specification
- Which pairs of methods may communicate
Specification Constructs

**Module**
- A set of related methods
  - (often aligned with data abstractions)

**Module Specification**
- Which pairs of methods may communicate

**Module Interface**
- Which communication is encapsulated or visible to callers outside the module
## Specification Constructs

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module</strong></td>
<td>A set of related methods (often aligned with data abstractions)</td>
</tr>
<tr>
<td><strong>Module Specification</strong></td>
<td>Which pairs of methods may communicate</td>
</tr>
<tr>
<td><strong>Module Interface</strong></td>
<td>Which communication is encapsulated or visible to callers outside the module</td>
</tr>
<tr>
<td><strong>Inlining</strong></td>
<td>Assigns communication to the caller</td>
</tr>
</tbody>
</table>
Evaluation: Specification Size

DaCapo

Java Grande
# Evaluation: Specification Size

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Avrora</strong></td>
<td>70,000</td>
</tr>
<tr>
<td><strong>Batik</strong></td>
<td>190,000</td>
</tr>
<tr>
<td><strong>Xalan</strong></td>
<td>180,000</td>
</tr>
<tr>
<td><strong>Crypt</strong></td>
<td>300</td>
</tr>
<tr>
<td><strong>LUFact</strong></td>
<td>500</td>
</tr>
<tr>
<td><strong>MolDyn</strong></td>
<td>500</td>
</tr>
<tr>
<td><strong>MonteCarlo</strong></td>
<td>1,200</td>
</tr>
<tr>
<td><strong>RayTracer</strong></td>
<td>700</td>
</tr>
<tr>
<td><strong>Series</strong></td>
<td>200</td>
</tr>
<tr>
<td><strong>SOR</strong></td>
<td>200</td>
</tr>
<tr>
<td><strong>Sparsematmult</strong></td>
<td>200</td>
</tr>
</tbody>
</table>
## Evaluation: Specification Size

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>LOC</th>
<th>Total Annotations</th>
<th>Ann. / KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avrora</td>
<td>70,000</td>
<td>175</td>
<td>2.5</td>
</tr>
<tr>
<td>Batik</td>
<td>190,000</td>
<td>16</td>
<td>0.01</td>
</tr>
<tr>
<td>Xalan</td>
<td>180,000</td>
<td>90</td>
<td>0.5</td>
</tr>
<tr>
<td>Crypt</td>
<td>300</td>
<td>16</td>
<td>53</td>
</tr>
<tr>
<td>LUFact</td>
<td>500</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>MolDyn</td>
<td>500</td>
<td>39</td>
<td>78</td>
</tr>
<tr>
<td>MonteCarlo</td>
<td>1,200</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>RayTracer</td>
<td>700</td>
<td>37</td>
<td>53</td>
</tr>
<tr>
<td>Series</td>
<td>200</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>SOR</td>
<td>200</td>
<td>14</td>
<td>70</td>
</tr>
<tr>
<td>Sparsematmult</td>
<td>200</td>
<td>9</td>
<td>45</td>
</tr>
</tbody>
</table>
## Evaluation: Specification Size

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>LOC</th>
<th>Total Annotations</th>
<th>Ann. / KLOC</th>
<th>Methods</th>
<th>Methods Annotated</th>
<th>% Methods Annotated</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DaCapo</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avrora</td>
<td>70,000</td>
<td>175</td>
<td>2.5</td>
<td>9,775</td>
<td>85</td>
<td>0.9%</td>
</tr>
<tr>
<td>Batik</td>
<td>190,000</td>
<td>16</td>
<td>0.01</td>
<td>15,547</td>
<td>8</td>
<td>0.05%</td>
</tr>
<tr>
<td>Xalan</td>
<td>180,000</td>
<td>90</td>
<td>0.5</td>
<td>7,854</td>
<td>42</td>
<td>0.5%</td>
</tr>
<tr>
<td><strong>Java Grande</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crypt</td>
<td>300</td>
<td>16</td>
<td>53</td>
<td>17</td>
<td>5</td>
<td>29%</td>
</tr>
<tr>
<td>LUFact</td>
<td>500</td>
<td>15</td>
<td>30</td>
<td>29</td>
<td>6</td>
<td>21%</td>
</tr>
<tr>
<td>MolDyn</td>
<td>500</td>
<td>39</td>
<td>78</td>
<td>27</td>
<td>16</td>
<td>59%</td>
</tr>
<tr>
<td>MonteCarlo</td>
<td>1,200</td>
<td>19</td>
<td>16</td>
<td>172</td>
<td>11</td>
<td>6%</td>
</tr>
<tr>
<td>RayTracer</td>
<td>700</td>
<td>37</td>
<td>53</td>
<td>77</td>
<td>15</td>
<td>19%</td>
</tr>
<tr>
<td>Series</td>
<td>200</td>
<td>10</td>
<td>50</td>
<td>15</td>
<td>6</td>
<td>40%</td>
</tr>
<tr>
<td>SOR</td>
<td>200</td>
<td>14</td>
<td>70</td>
<td>13</td>
<td>5</td>
<td>38%</td>
</tr>
<tr>
<td>Sparsematmult</td>
<td>200</td>
<td>9</td>
<td>45</td>
<td>12</td>
<td>4</td>
<td>33%</td>
</tr>
</tbody>
</table>
Specification Expressiveness
Specification Expressiveness

**Strengths:**

✓ Concise and intuitive
✓ Encapsulation useful in many benchmarks
✓ Sensitive to error
Specification Expressiveness

**Strengths:**
- ✓ Concise and intuitive
- ✓ Encapsulation useful in many benchmarks
- ✓ Sensitive to error

**Limitations / Future Work:**
- • Improve support for non-layered communication
- • Integrate data-centric properties to reduce specification size
Specification Expressiveness

**Strengths:**
- ✓ Concise and intuitive
- ✓ Encapsulation useful in many benchmarks
- ✓ Sensitive to error

**Limitations / Future Work:**
- • Improve support for non-layered communication
- • Integrate data-centric properties to reduce specification size

**Also in the Paper:**
- • Java annotation syntax
- • Formal semantics
Outline

A Code-Centric View of Shared-Memory

Code-Communication Specification Language

- Making Specifications Modular and Concise
- Specification Language Evaluation

Dynamic Specification Checker

- Making Communication Checking Fast Enough
- Performance Evaluation
Fundamental Instrumentation Costs

class C {
    int x;
    State x__lastWriter;
}
class C {
    int x;
    State x__lastWriter;

    write Store current thread and call stack as last writer.
Fundamental Instrumentation Costs

class C {
    int x;
    State x__lastWriter;
}

**write**  Store current thread and call stack as last writer.

**read**   **Check** if communication is allowed from last writer to current reader.
## Optimizing Read Checks

<table>
<thead>
<tr>
<th>Check</th>
<th>Action</th>
<th>Mem. Ops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same thread?</td>
<td>✔</td>
<td>1</td>
</tr>
<tr>
<td>Full check passes?</td>
<td>✔</td>
<td>&gt;30</td>
</tr>
<tr>
<td>Else illegal.</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Throw exception.</td>
<td></td>
</tr>
</tbody>
</table>
## Optimizing Read Checks

<table>
<thead>
<tr>
<th>Check</th>
<th>Action</th>
<th>Mem. Ops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same thread?</td>
<td>✔️</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Check</th>
<th>Action</th>
<th>Mem. Ops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full check passes?</td>
<td>✔️</td>
<td>&gt;30</td>
</tr>
<tr>
<td>Else illegal.</td>
<td>✗</td>
<td>Throw exception.</td>
</tr>
</tbody>
</table>
## Optimizing Read Checks

<table>
<thead>
<tr>
<th>Check</th>
<th>Action</th>
<th>Mem. Ops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same thread?</td>
<td>✔</td>
<td>1</td>
</tr>
<tr>
<td>Full check passes?</td>
<td>✔ Add pair to global memo table.</td>
<td>&gt;30</td>
</tr>
<tr>
<td>Else illegal.</td>
<td>✘ Throw exception.</td>
<td></td>
</tr>
</tbody>
</table>
Optimizing Read Checks

<table>
<thead>
<tr>
<th>Check</th>
<th>Action</th>
<th>Mem. Ops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same thread?</td>
<td>✔</td>
<td>1</td>
</tr>
</tbody>
</table>

| Stack pair in global memo table?   | ✔                                           | 12        |
|                                    | Full check passes?                         | Add pair to global memo table. | >30 |
|                                    | Else illegal.                              | Throw exception.               |     |

25
## Optimizing Read Checks

<table>
<thead>
<tr>
<th>Check</th>
<th>Action</th>
<th>Mem. Ops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same thread?</td>
<td>✔</td>
<td>1</td>
</tr>
<tr>
<td>Stack pair in global memo table?</td>
<td>✔  Add writer stack ID to reader stack's cache.</td>
<td>12</td>
</tr>
<tr>
<td>Full check passes?</td>
<td>✔  Add pair to global memo table.</td>
<td>&gt;30</td>
</tr>
<tr>
<td>Else illegal.</td>
<td>✘  Throw exception.</td>
<td></td>
</tr>
</tbody>
</table>
## Optimizing Read Checks

<table>
<thead>
<tr>
<th>Check</th>
<th>Action</th>
<th>Mem. Ops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same thread?</td>
<td>✔</td>
<td>1</td>
</tr>
<tr>
<td>Writer stack ID in reader stack’s cache?</td>
<td>✔</td>
<td>4</td>
</tr>
<tr>
<td>Stack pair in global memo table?</td>
<td>✔</td>
<td>12</td>
</tr>
<tr>
<td>Full check passes?</td>
<td>✔</td>
<td>&gt;30</td>
</tr>
<tr>
<td>Else illegal.</td>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>

- **Check**: Determines if the operation is valid based on various conditions.
- **Action**: Specifies the action to take if the check passes.
- **Mem. Ops.**: The number of memory operations required for each check.

- **Same thread?**: If true, proceed.
- **Writer stack ID in reader stack’s cache?**: If true, add writer stack ID to reader stack’s cache.
- **Stack pair in global memo table?**: If true, add pair to global memo table.
- **Full check passes?**: If true, proceed.
- **Else illegal.**: If false, throw exception.
## Optimizing Read Checks

<table>
<thead>
<tr>
<th>Check</th>
<th>Action</th>
<th>Mem. Ops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same thread?</td>
<td>✔</td>
<td>1</td>
</tr>
<tr>
<td>Writer stack ID in reader stack’s cache?</td>
<td>✔</td>
<td>4</td>
</tr>
<tr>
<td>Stack pair in global memo table?</td>
<td>✔ Add writer stack ID to reader stack’s cache.</td>
<td>12</td>
</tr>
<tr>
<td>Full check passes?</td>
<td>✔ Add pair to global memo table.</td>
<td>&gt;30</td>
</tr>
<tr>
<td>Else illegal.</td>
<td>✗ Throw exception.</td>
<td></td>
</tr>
</tbody>
</table>
## Experimental Configuration

<table>
<thead>
<tr>
<th></th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Benchmarks</strong></td>
<td>8 Java Grande, large inputs, 8 threads</td>
</tr>
<tr>
<td></td>
<td>3 DaCapo 9.12, default inputs, 8 threads</td>
</tr>
<tr>
<td><strong>Machine</strong></td>
<td>8-core 2.8GHz Intel Xeon, 10GB RAM</td>
</tr>
<tr>
<td></td>
<td>Ubuntu 8.10</td>
</tr>
<tr>
<td><strong>JVM</strong></td>
<td>HotSpot 64-bit client VM 1.6.0</td>
</tr>
<tr>
<td></td>
<td>max heap size 8GB</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>Average over 10 runs</td>
</tr>
<tr>
<td></td>
<td>separate performance and profiling</td>
</tr>
</tbody>
</table>
Execution Profile
Execution Profile

> 99.99999% of reads checked on fast paths
Execution Profile

> 99.99999% of reads checked on fast paths

up to 6 billion communicating reads
Execution Profile

> 99.99999% of reads checked on fast paths

up to 6 billion communicating reads

≤ 697 full stack checks
Communicating Read Operations

% of reads that communicate

- Avrora
- Batik
- Xalan
- Crypt
- LUFact
- MolDyn
- MonteCarlo
- RayTracer
- Series
- SOR
- SparseMatmult
Communicating Read Operations

% of reads that communicate

Avrora  Batik  Xalan  Crypt  LUFact  MolDyn  MonteCarlo  RayTracer  Series  SOR  SparseMatmult
Time Overhead

Running time vs. uninstrumented
Time Overhead

![Bar Chart](image)

- Avrora
- Batik
- Xalan
- Crypt
- LUFact
- MolDyn
- MonteCarlo
- RayTracer
- Series
- SOR
- SparseMatmult

Running time vs. uninstrumented
Space Overhead

Peak memory vs. uninstrumented

Avrora, Batik, Xalan, Crypt, LUFact, MolDyn, MonteCarlo, RayTracer, Series, SOR, SparseMatmult
Space Overhead

Peak memory vs. uninstrumented

- Avrora
- Batik
- Xalan
- Crypt
- LUFact
- MolDyn
- MonteCarlo
- RayTracer
- Series
- SOR
- SparseMatmult

Crypt shows a space overhead of 34.3x compared to uninstrumented.
Space Overhead

Peak memory vs. uninstrumented

Avrora 8x
Batik 6x
Xalan 11x
Crypt 34.3x
LUFact 2x
MolDyn 4x
MonteCarlo 5x
RayTracer 0.5x
Series 10x
SOR 1x
SparseMatmult 2x
Summary

1. A Code-Centric View of Shared-Memory

2. Code-Communication Specification Language
   • Concise and modular specifications
   • Fit communication patterns in real programs

3. Dynamic Specification Checker
   • Aggressive optimization of communication checks
   • Debugging-level performance
Summary

1. A Code-Centric View of Shared-Memory

2. Code-Communication Specification Language
   - Concise and modular specifications
   - Fit communication patterns in real programs

3. Dynamic Specification Checker
   - Aggressive optimization of communication checks
   - Debugging-level performance

Download: www.cs.washington.edu/homes/bpw/
This slide intentionally not left blank
Thread 1

```
work() {
    sync(map) {
        line = 0;
        map.put(line, x);
    }
    ... 
}
```

Thread 2

```
work() {
    ... 
}
```

Thread 3

```
consume() {
    sync(map) {
        if (line == 768) {
            line = line + 2;
            map.put(line, x);
        }
    }
}
```

Communication Specifications, ✔️ Race Detection, Sharing Specs, Atomicity Checker

```
int line; Map map;
```
Communication Specifications, ✔
Race Detection, Sharing Specs, Atomicity Checker

Thread 1

```java
work() {
    sync(map) {
        line = 0;
        map.put(line, x);
    }
}
```

Thread 2

```java
work() { ...
    consume() {
        ...
    }
}
```

Thread 3

```java
consume() {
    ...
}
```

int line; Map map;
int line; Map map;

Communication Specifications, ✔
Race Detection, Sharing Specs, Atomicity Checker

Thread 1

work() {
    sync(map) {
        line = 0;
        map.put(line, x);
    } ...
}

Thread 2

work() { ...
    sync(map) {
        line = line + 2;
        map.put(line, x);
    } ...
}

Thread 3

consume() {
    sync(map) {
        if (line == 768)
            ...
    }

```java
int line; Map map;

Thread 1
work() {
    sync(map) {
        line = 0;
        map.put(line, x);
    }
}

Thread 2
work() {
    ... sync(map) {
        line = line + 2;
        map.put(line, x);
    }
}

Thread 3
consume() {
    sync(map) {
        if (line == 768)
            line = 0;
        map.put(line, x);
    }
}

Communication Specifications,
Race Detection, Sharing Specs, Atomicity Checker

line: lock was insufficient
should not be thread-local or read-only

int line; Map map;
```
Thread 1

```java
work() {
    sync(map) {
        line = 0;
        map.put(0, x);
    }
    ... // end work()
}
```

Thread 2

```java
work() { ... }

correct version
```

Thread 3

```java
consume() {
    sync(map) {
        line = line + 2;
        map.put(1, x);
    }
}
```

Communication Specifications, ✔
Race Detection, Sharing Specs, Atomicity Checker

Correct version is not intended to be atomic.

```java
int line; Map map;
```
This slide intentionally not left blank
Callbacks

Writer Thread

Reader Thread

in Simulator.run(...):
in EventList.fireAll(...):
in Simulator.run(...):
in Action.create(...):
in Action.fire(...):

buffer[3] = ...; return buffer[3];
Callbacks

Writer Thread

in Simulator.run(...):

in Action.create(...):

buffer[3] = ...;

return buffer[3];

Reader Thread

in EventList.fireAll(...):

in Action.fire(...):
This slide intentionally not left blank