Weak Consistency
(TSO as an Example)

Parosh Aziz Abdulla\textsuperscript{1}
Mohamed Faouzi Atig\textsuperscript{1}
Ahmed Bouajjani\textsuperscript{2}
Tuan Phong Ngo\textsuperscript{1}

\textsuperscript{1}Uppsala University
\textsuperscript{2}IRIF, Université Paris Diderot & IUF
Outline

- Weak Consistency
- Total Store Order (TSO)
- Dual TSO
- Verification
- Specification
- Synthesis
Outline

- Weak Consistency
- Total Store Order (TSO)
- Dual TSO
- Verification
- Specification
- Synthesis
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations

P1: w(x,1)
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations

Processes

P1: write
P2: read
P1: read
P2: write

Execution

P1: w(x,1) → P2: r(x,1)
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations

Processes: atomic read/write
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations
- Simple and intuitive
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations
  + Simple and intuitive
  - Too strong

Execution:

P1: \texttt{w(x,1)} \rightarrow \texttt{P2: r(x,1)} \rightarrow \texttt{P2: w(y,1)} \rightarrow \texttt{P1: r(y,1)}
Cloud Computing

- Processes perform local operations
- Operations propagated asynchronously
Cloud Computing
• Processes perform local operations
• Operations propagated asynchronously
Cloud Computing

- Processes perform local operations
- Operations propagated asynchronously

PO: $w(x,1)$
Cloud Computing
- Processes perform local operations
- Operations propagated asynchronously

**Execution**

P0: w(x,1) → P1: w(x,2)
Cloud Computing

- Processes perform local operations
- Operations propagated asynchronously
Cloud Computing

- Processes perform local operations
- Operations propagated asynchronously

Execution

P0: \( w(x,1) \)  
P1: \( w(x,2) \)  
P2: \( r(x,1) \)  
P3: \( r(x,0) \)
Cloud Computing

- Processes perform local operations
- Operations propagated asynchronously

Execution

P0: w(x,1) → P1: w(x,2) → P2: r(x,1) → P3: r(x,0)

P3: r(x,1) → P2: r(x,2) → P3: r(x,2)
Cloud Computing

- Processes perform local operations
- Operations propagated asynchronously
Cloud Computing

- Processes perform local operations
- Operations propagated asynchronously

Execution

P0: w(x,1)
P1: w(x,2)
P2: r(x,1)
P3: r(x,0)
P3: r(x,1)
P2: r(x,2)
P3: r(x,2)
**TSO - Total Store Order**

- Widely used:
  - Used by Sun SPARCv9
  - Formalization of Intel x86

- Memory access optimization:
  - Write operations are slow
  - Introduce *store buffers*
**TSO - Total Store Order**

- Widely used:
  - Used by Sun SPARCv9
  - Formalization of Intel x86
- Memory access optimization:
  - Write operations are slow
  - Introduce **store buffers**
TSO - Classical Semantics

P1: write: x = 1
P1: write: x = 2
P1: read: x = 2
P1: read: y = 0
TSO - Classical Semantics

P1: write: $x = 1$
P1: write: $x = 2$
P1: read: $x = 2$
P1: read: $y = 0$
**TSO - Classical Semantics**

- P1: write: $x = 1$
- P1: write: $x = 2$
- P1: read: $x = 2$
- P1: read: $y = 0$

![Diagram showing the sequence of operations and the buffer state]
TSO - Classical Semantics

P1: write: x = 1
P1: write: x = 2
P1: read: x = 2
P1: read: y = 0
**TSO - Classical Semantics**

- P1: write: \( x = 1 \)
- P1: write: \( x = 2 \)
- P1: read: \( x = 2 \)
- P1: read: \( y = 0 \)

write to buffer
TSO - Classical Semantics

P1: write: \( x = 1 \)
P1: write: \( x = 2 \)
P1: read: \( x = 2 \)
P1: read: \( y = 0 \)
TSO - Classical Semantics

P1: write: x = 1
P1: write: x = 2
P1: read: x = 2
P1: read: y = 0

read from buffer
TSO - Classical Semantics

P1: write: $x = 1$
P1: write: $x = 2$
P1: read: $x = 2$
P1: read: $y = 0$
P1: write: \( x = 1 \)
P1: write: \( x = 2 \)
P1: read: \( x = 2 \)
P1: read: \( y = 0 \)

TSO - Classical Semantics

read from memory
TSO - Classical Semantics

P1: write: x = 1
P2: read: x = 2
P1: read: x = 2
P1: read: y = 0
**TSO - Classical Semantics**

P1: write: \( x = 1 \)

P1: write: \( x = 2 \)

P1: read: \( x = 2 \)

P1: read: \( y = 0 \)

**update memory**
**TSO - Classical Semantics**

**P1:** write: \( x = 1 \)

**P1:** write: \( x = 2 \)

**P1:** read: \( x = 2 \)

**P1:** read: \( y = 0 \)
TSO - Classical Semantics

P1: write: x = 1
P1: write: x = 2
P1: read: x = 2
P1: read: y = 0
TSO - Classical Semantics

P1: write: x = 1
P1: write: x = 2
P1: read: x = 2
P1: read: y = 0

• write to buffer
• read from buffer
• read from memory
• update memory
TSO - Classical Semantics

P1: write: \( x = 1 \)
P1: write: \( x = 2 \)
P1: read: \( x = 2 \)
P1: read: \( y = 0 \)

- write to buffer
- read from buffer
- read from memory
- update memory
**TSO - Classical Semantics**

- **P1:** write: \( x = 1 \)
- **P1:** write: \( x = 2 \)
- **P1:** read: \( x = 2 \)
- **P1:** read: \( y = 0 \)

- write to buffer
- read from buffer
- read from memory
- update memory

**TSO**

- Extra behaviors
- Potentially bad behaviors
Initially: $x = y = 0$

- **P1**
  - write: $x = 1$
  - read: $y = 0$
  - critical section

- **P2**
  - write: $y = 1$
  - read: $x = 0$
  - critical section

Sequential Consistency = Interleaving
Initially: \( x = y = 0 \)

- **P1**
  - write: \( x = 1 \)
  - read: \( y = 0 \)
  - critical section

- **P2**
  - write: \( y = 1 \)
  - read: \( x = 0 \)
  - critical section

Sequential Consistency = Interleaving

At most one process at its CS at any time
Initially: \( x = y = 0 \)

- **P1**
  - write: \( x = 1 \)
  - read: \( y = 0 \)
  - critical section

- **P2**
  - write: \( y = 1 \)
  - read: \( x = 0 \)
  - critical section

**TSO**

\( x = 0 \)

\( y = 0 \)
Initially: $x = y = 0$

- **P1**
  - write: $x = 1$
  - read: $y = 0$
  - critical section

- **P2**
  - write: $y = 1$
  - read: $x = 0$
  - critical section

TSO

$y = 0$

$y = 0$
Initially: $x = y = 0$

write: $x = 1$

read: $y = 0$

critical section

write: $y = 1$

read: $x = 0$

critical section

P1 P2

x = 0
y = 0

TSO

Dekker Protocol
Initially: $x = y = 0$

P1
- write: $x = 1$
- read: $y = 0$
- critical section

P2
- write: $y = 1$
- read: $x = 0$
- critical section

TSO
- write to buffer

Dekker Protocol
Initially: $x = y = 0$

P1
write: $x = 1$
read: $y = 0$
critical section

P2
write: $y = 1$
read: $x = 0$
critical section

TSO
Initially: $x = y = 0$

P1
- write: $x = 1$
- read: $y = 0$
- critical section

P2
- write: $y = 1$
- read: $x = 0$
- critical section

TSO
Initially: $x = y = 0$

- **P1**
  - Write: $x = 1$
  - Read: $y = 0$

- **P2**
  - Write: $y = 1$
  - Read: $x = 0$

Read from memory

**TSO**
Initially: $x = y = 0$

P1
write: $x = 1$
read: $y = 0$
critical section

P2
write: $y = 1$
read: $x = 0$
critical section

enter CS

TSO
Initially: $x = y = 0$

P1
write: $x = 1$
read: $y = 0$
critical section

P2
write: $y = 1$
read: $x = 0$
critical section

TSO
$x = 0$
y = 0
Initially: $x = y = 0$

write: $x = 1$
read: $y = 0$
critical section

write: $y = 1$
read: $x = 0$
critical section

Dekker Protocol
Initially: $x = y = 0$

write: $x = 1$
read: $y = 0$
critical section

write: $y = 1$
read: $x = 0$
critical section

Dekker Protocol
Initially: \(x = y = 0\)

- **P1**
  - write: \(x = 1\)
  - read: \(y = 0\)
  - critical section

- **P2**
  - write: \(y = 1\)
  - read: \(x = 0\)
  - critical section

**TSO**

\(x = 0\)

\(y = 0\)
Initially: \( x = y = 0 \)

- **P1**
  - write: \( x = 1 \)
  - read: \( y = 0 \)
  - critical section

- **P2**
  - write: \( y = 1 \)
  - read: \( x = 0 \)
  - critical section

TSO

- P1
  - \( x = 1 \)

- P2
  - \( y = 1 \)

read from memory

\( x = 0 \)
\( y = 0 \)
Dekker Protocol

Initially: \( x = y = 0 \)

- \( P_1 \):
  - write: \( x = 1 \)
  - read: \( y = 0 \)
  - critical section

- \( P_2 \):
  - write: \( y = 1 \)
  - read: \( x = 0 \)
  - critical section

TSO

\( x = 0 \)
\( y = 0 \)

enter CS
Initially: $x = y = 0$

- **P1**: write: $x = 1$
  - read: $y = 0$
  - critical section

- **P2**: write: $y = 1$
  - read: $x = 0$
  - critical section

2 processes in **CS** at the same time
Initially: $x = y = 0$

P1
write: $x = 1$
read: $y = 0$
critical section

P2
write: $y = 1$
read: $x = 0$
critical section

Dekker Protocol
Initially: $x = y = 0$

P1

write: $x = 1$
read: $y = 0$
critical section

P2

write: $y = 1$
read: $x = 0$
critical section

Dekker Protocol

P1

write: $x = 1$
read: $y = 0$
critical section

P2

write: $y = 1$
read: $x = 0$
critical section

TSO
Initially: $x = y = 0$

write: $x = 1$

read: $y = 0$

critical section

write: $y = 1$

critical section

"read overtaking write"
Dekker Protocol

Initially: $x = y = 0$

- **P1**
  - write: $x = 1$
  - read: $y = 0$
  - critical section

- **P2**
  - write: $y = 1$
  - read: $x = 0$
  - critical section

"read overtaking write"

TSO

"read overtaking write"
Weakly Consistent Systems

- Cloud
- Weak memories
- Weak cache protocols
- etc

+ Efficiency
- Non-intuitive behaviours
Weakly Consistent Systems

- Cloud
- Weak memories
- Weak cache protocols
- etc

+ Efficiency
- Non-intuitive behaviours

- Semantics
- Correctness analysis: simulation, testing, verification, synthesis
- Methods and tools: decidability, complexity, algorithms
- Specifications
Verification under TSO is Difficult

while (1)
write: x=1

PO

x = 0
y = 0
Verification under TSO is Difficult

while (1)
write: x=1

P0: write: x = 1
P0: write: x = 1
...
P0: write: x = 1
...

x = 0
y = 0
Verification under TSO is Difficult

```
while (1)
  write: x=1
  P0: write:  x = 1
  P0: write:  x = 1
  P0: write:  x = 1
  ...
```

```
x = 0
y = 0
```
Verification under TSO is Difficult

while (1)
  write: x=1

P0: write: x = 1
P0: write: x = 1
  ...
P0: write: x = 1
  ...

P0

x=1

x = 0
y = 0
Verification under TSO is Difficult
Verification under TSO is Difficult

while (1) write: x=1

P0: write: x = 1
P0: write: x = 1
P0: write: x = 1
...
Verification under TSO is Difficult

while (1)
  write: x=1

PO: write: x = 1
PO: write: x = 1
  ...
PO: write: x = 1
  ...

x = 0
y = 0

unbounded buffer
Verification under TSO is Difficult

while (1)
write: x=1

P0: write: x = 1
P0: write: x = 1
... 
P0: write: x = 1
... 

unbounded buffer

infinite state space
Outline

• Weak Consistency
• Total Store Order (TSO)
  • Dual TSO
• Verification
• Specification
• Synthesis
Dual TSO

- store buffer \rightarrow load buffer
- write immediately updates memory
- buffers contain expected reads
- messages: self, other

Processes

P1
- x,1,self

P2
- y,2,other

Load buffer
- \( x = 0 \)
- \( y = 0 \)

FIFO buffer

Shared variables

- \( x = 0 \)
- \( y = 0 \)
Dual TSO

- P1: write: x = 1
- P1: read: x = 1
- P1: read: y = 0
Dual TSO

P1: write: \( x = 1 \)
P1: read: \( x = 1 \)
P1: read: \( y = 0 \)
Dual TSO

P1: write: $x = 1$
P1: read: $x = 1$
P1: read: $y = 0$

update memory
Dual TSO

P1: write: $x = 1$
P1: read: $x = 1$
P1: read: $y = 0$

update memory

x=1,self

y=0
Dual TSO

P1: write: \( x = 1 \)
P1: read: \( x = 1 \)
P1: read: \( y = 0 \)

propagate to yourself

update memory

P1 \( x=1, \text{self} \)

P2

x=1

y=0
Dual TSO

P1: write: $x = 1$
P1: read: $x = 1$
P1: read: $y = 0$
Dual TSO

P1: write: \( x = 1 \)
P1: read: \( x = 1 \)
P1: read: \( y = 0 \)

propagate from memory
**Dual TSO**

- **P1**: write: $x = 1$
- **P1**: read: $x = 1$
- **P1**: read: $y = 0$

Propagate from memory
**Dual TSO**

P1: write: \( x = 1 \)

P1: read: \( x = 1 \)

P1: read: \( y = 0 \)

\( x=1 \), self

\( y=0 \), other

**propagate from memory**
Dual TSO

P1: write: $x = 1$
P1: read: $x = 1$
P1: read: $y = 0$
Dual TSO

P1: write: x = 1
P1: read: x = 1
P1: read: y = 0

read own
write
Dual TSO

P1: write: \( x = 1 \)
P1: read: \( x = 1 \)
P1: read: \( y = 0 \)
Dual TSO

P1: write: $x = 1$
P1: read: $x = 1$
P1: read: $y = 0$

P2

remove oldest write

$x = 1$, self
$y = 0$, other

$x = 1$
$y = 0$
Dual TSO

P1: write: x = 1
P1: read: x = 1
P1: read: y = 0

remove oldest write
Dual TSO

P1: write: x = 1
P1: read: x = 1
P1: read: y = 0

P2

y=0,other

x=1

y=0
Dual TSO

P1: write: x = 1
P1: read: x = 1
P1: read: y = 0

read oldest
write

P2

y=0, other

x=1

y=0
Dual TSO

P1: write: x = 1
P1: read: x = 1
P1: read: y = 0

read oldest write

P1

y=0, other

x=1

y=0

P2
Dual TSO

P1: write: \( x = 1 \)
P1: read: \( x = 1 \)
P1: read: \( y = 0 \)

- write + self-propagation
- propagate from memory
- read own-writes
- read oldest write
- remove oldest write
**Dual TSO**

- **P1**: write: $x = 1$
- **P1**: read: $x = 1$
- **P1**: read: $y = 0$

- write + self-propagation
- propagate from memory
- read own-writes
- read oldest write
- remove oldest write
**Dual TSO**

- **P1**: write: $x = 1$
- **P1**: read: $x = 1$
- **P1**: read: $y = 0$

- write + self-propagation
- propagate from memory
- read own-writes
- read oldest write
- remove oldest write

**TSO ≡ Dual-TSO**
Dual TSO

P1: write: \( x = 1 \)
P1: read: \( x = 1 \)
P1: read: \( y = 0 \)

- write + self-propagation
- propagate from memory
- read own-writes
- read oldest write
- remove oldest write

TSO \equiv\text{ Dual-TSO}
Classical TSO
\textbf{P1:} w(x, 2)

\textbf{Classical TSO}
Classical TSO

P1: w(x,2)
Classical TSO
Classical TSO
Classical TSO
Classical TSO
P1: \( w(x, 2) \)  
P1: \( r(y, 0) \)  
P2: \( w(y, 1) \)  
P2: \( w(x, 1) \)  

Classical TSO
Classical TSO
P1: \( w(x,2) \) → P1: \( r(y,0) \) → P2: \( w(y,1) \) → P2: \( w(x,1) \)

**Classical TSO**
P1: \(w(x,2)\)  P1: \(r(y,0)\)  P2: \(w(y,1)\)  P2: \(w(x,1)\)

Classical TSO
P1: w(x,2) \rightarrow P1: r(y,0) \rightarrow P2: w(y,1) \rightarrow P2: w(x,1)
Classical TSO
Classical TSO
P1: \( w(x,2) \)

P2: \( r(x,2) \)

P1: \( r(y,0) \)

P2: \( w(y,1) \)

P2: \( w(x,1) \)

Dual TSO

Classical TSO
P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO

Dual TSO
P1: \( w(x,2) \) → P1: \( r(y,0) \) → P2: \( w(y,1) \) → P2: \( w(x,1) \) → P2: \( r(x,2) \)

Classical TSO

P1: \( w(x,2) \) → P1: \( r(y,0) \) → P2: \( w(y,1) \) → P2: \( w(x,1) \) → P2: \( r(x,2) \)

Dual TSO

P2: \( w(y,1) \) → P2: \( w(x,1) \)
Dual TSO

Classical TSO
P2: w(y,1)

Dual TSO

P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO
Dual TSO

Classical TSO
P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO

Dual TSO

P1: w(x,2) → P2: w(y,1) → P1: w(x,2)

x=2

y=0,other

y=1,self

x=1,self

y=1

P2: w(x,1)

P2: w(y,1)
P1: \( w(x,2) \) ➔ P2: \( w(y,1) \) ➔ P1: \( w(x,2) \)

P2: \( w(y,1) \) ➔ P2: \( w(x,1) \) ➔ P1: \( w(x,2) \)

Dual TSO

Classical TSO
Dual TSO

Classical TSO
Dual TSO

Classical TSO
Dual TSO

Classical TSO
Dual TSO

Classical TSO
Dual TSO

Classical TSO
P2: \( w(y,1) \) → P2: \( w(x,1) \) → P1: \( w(x,2) \) → P2: \( r(x,2) \) → P1: \( r(y,0) \)

P1: \( w(x,2) \) → P1: \( r(y,0) \) → P2: \( w(y,1) \) → P2: \( w(x,1) \) → P2: \( r(x,2) \)

Dual TSO

Classical TSO
Dual TSO

Classical TSO
Dual TSO

Classical TSO
Dual TSO

Classical TSO
Outline

- Weak Consistency
- Total Store Order (TSO)
- Dual TSO
- Verification
- Specification
- Synthesis
Dual TSO - Monotonicity

partition of load buffer

Old

New
Dual TSO - Monotonicity

partition of load buffer

newest self message on y
partition of load buffer

Dual TSO - Monotonicity

Old

newest self message on x

newest self message on y

New
Dual TSO - Monotonicity

Partition of load buffer

x=2, self
y=1, self
x=1, other
y=0, self
x=0, other

Newest self message on x
Newest self message on y
Dual TSO - Monotonicity

Ordering on Buffers

x=2,self y=1,self x=1,other y=0,self x=0,other

x=2,self y=1,self y=0,self x=0,other
Dual TSO - Monotonicity

Ordering on Buffers
Dual TSO - Monotonicity

Ordering on Buffers

- x=2, self
- y=1, self
- x=1, other
- y=0, self
- x=0, other
- x=2, self
- y=1, self
- y=0, self
- x=0, other
Dual TSO - Monotonicity

Ordering on Buffers

\(x=2, \text{self}\) \(\preceq\) \(y=1, \text{self}\) \(\preceq\) \(x=1, \text{other}\) \(\preceq\) \(y=0, \text{self}\) \(\preceq\) \(x=0, \text{other}\)

\(\text{subword}\)

\(x=2, \text{self}\) \(\preceq\) \(y=1, \text{self}\) \(\preceq\) \(y=0, \text{self}\) \(\preceq\) \(x=0, \text{other}\)
Dual TSO - Monotonicity

Ordering on Buffers

\[ ab \subseteq xaybzy \]
Dual TSO - Monotonicity

Ordering on Buffers

\[ ab \sqsubseteq xaybhz \]
Dual TSO - Monotonicity

Ordering on Configurations

- identical process states
- identical memory state
- sub-word relation on buffers
Dual TSO - Monotonicity

Ordering on Configurations

- identical process states
- identical memory state
- sub-word relation on buffers

P1

P2

x = 1, self

x, l, other

x = 1

y = 0
Dual TSO - Monotonicity

Ordering on Configurations

- identical process states
- identical memory state
- sub-word relation on buffers
Dual TSO - Monotonicity

Ordering on Configurations

- identical process states
- identical memory state
- sub-word relation on buffers
Dual TSO - Monotonicity

Ordering on Configurations

\[ c_1 \rightarrow c_2 \rightarrow c_3 \]

Monotonicity
Dual TSO - Monotonicity

Ordering on Configurations

\[ c_1 \rightarrow c_2 \]
\[ |\_| \]
\[ c_3 \rightarrow c_4 \]
\[
\text{Monotonicity}
\]
Dual TSO - Monotonicity

• finite-state programs running on TSO:
  • reachability analysis terminates
  • reachability decidable
Experimental Results

Tool: Memorax

https://github.com/memorax/memorax
## Experimental Results

### Tool: Memorax

<table>
<thead>
<tr>
<th>Program</th>
<th>#P</th>
<th>Safe under SC</th>
<th>Safe under TSO</th>
<th>Time (secs)</th>
<th># generated configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>5</td>
<td>yes</td>
<td>no</td>
<td>0.3</td>
<td>10641</td>
</tr>
<tr>
<td>LB</td>
<td>3</td>
<td>yes</td>
<td>no</td>
<td>0.0</td>
<td>2048</td>
</tr>
<tr>
<td>WRC</td>
<td>4</td>
<td>yes</td>
<td>yes</td>
<td>0.0</td>
<td>1507</td>
</tr>
<tr>
<td>ISA2</td>
<td>3</td>
<td>yes</td>
<td>yes</td>
<td>0.0</td>
<td>509</td>
</tr>
<tr>
<td>RWC</td>
<td>5</td>
<td>yes</td>
<td>no</td>
<td>0.1</td>
<td>4277</td>
</tr>
<tr>
<td>W+RWC</td>
<td>4</td>
<td>yes</td>
<td>no</td>
<td>0.0</td>
<td>1713</td>
</tr>
<tr>
<td>IRIW</td>
<td>4</td>
<td>yes</td>
<td>yes</td>
<td>0.0</td>
<td>520</td>
</tr>
<tr>
<td>MP</td>
<td>4</td>
<td>yes</td>
<td>yes</td>
<td>0.0</td>
<td>883</td>
</tr>
<tr>
<td>Simple Dekker</td>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>0.0</td>
<td>98</td>
</tr>
<tr>
<td>Dekker</td>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>0.1</td>
<td>5053</td>
</tr>
<tr>
<td>Peterson</td>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>0.1</td>
<td>5442</td>
</tr>
<tr>
<td>Repeated Peterson</td>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>0.2</td>
<td>7632</td>
</tr>
<tr>
<td>Bakery</td>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>2.6</td>
<td>82050</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>0.2</td>
<td>8324</td>
</tr>
<tr>
<td>Szymanski</td>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>0.6</td>
<td>29018</td>
</tr>
<tr>
<td>Ticket Spin Lock</td>
<td>3</td>
<td>yes</td>
<td>yes</td>
<td>0.9</td>
<td>18963</td>
</tr>
<tr>
<td>Lamport’s Fast Mutex</td>
<td>3</td>
<td>yes</td>
<td>no</td>
<td>17.7</td>
<td>292543</td>
</tr>
<tr>
<td>Burns</td>
<td>4</td>
<td>yes</td>
<td>no</td>
<td>124.3</td>
<td>2762578</td>
</tr>
<tr>
<td>NBW-W-WR</td>
<td>2</td>
<td>yes</td>
<td>yes</td>
<td>0.0</td>
<td>222</td>
</tr>
<tr>
<td>Sense Reversing Barrier</td>
<td>2</td>
<td>yes</td>
<td>yes</td>
<td>0.1</td>
<td>1704</td>
</tr>
</tbody>
</table>

**standard benchmarks:**
litmus tests and mutual exclusion
Experimental Results

Tool: Memorax

Parameterized verification

time (secs)

# generated configurations

<table>
<thead>
<tr>
<th>Program</th>
<th>#T</th>
<th>#C</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>0.0</td>
<td>147</td>
</tr>
<tr>
<td>LB</td>
<td>0.6</td>
<td>1028</td>
</tr>
<tr>
<td>MP</td>
<td>0.0</td>
<td>149</td>
</tr>
<tr>
<td>WRC</td>
<td>0.8</td>
<td>618</td>
</tr>
<tr>
<td>ISA2</td>
<td>4.3</td>
<td>1539</td>
</tr>
<tr>
<td>RWC</td>
<td>0.2</td>
<td>293</td>
</tr>
<tr>
<td>W+RWC</td>
<td>1.5</td>
<td>828</td>
</tr>
<tr>
<td>IRIW</td>
<td>4.6</td>
<td>648</td>
</tr>
</tbody>
</table>
Outline

- Weak Consistency
- Total Store Order (TSO)
- Dual TSO
- Verification
- **Specification**
- Synthesis
Cache
Coherence
Protocol

? \models SC
? \models sc
Cache Coherence Protocol

? TSO

monitors
TSO-CC: Consistency directed cache coherence for TSO

Marco Elver
University of Edinburgh
marco.elver@ed.ac.uk

Vijay Nagarajan
University of Edinburgh
vijay.nagarajan@ed.ac.uk

Racer: TSO Consistency via Race Detection

Alberto Ros
Department of Computer Engineering
Universidad de Murcia, Spain
aros@ditec.um.es

Stefanos Kaxiras
Department of Information Technology
Uppsala Universitet, Sweden
stefanos.kaxiras@it.uu.se
TSO-Counter-Examples
$P_1: w(x,1)$

TSO-Counter-Examples
P1: w(x,1) → P2: r(x,1)

TSO-Counter-Examples
P1: \( w(x,1) \) → P2: \( r(x,1) \) → P3: \( w(x,2) \)
P1: $w(x,1)$ → P2: $r(x,1)$ → P3: $w(x,2)$ → P4: $r(x,2)$

TSO-Counter-Examples
P1: w(x,1) → P2: r(x,1) → P3: w(x,2) → P4: r(x,2) → P5: r(x,1)
TSO-Counter-Examples

TSO ≡ 12 counter-examples
Outline

- Weak Consistency
- Total Store Order (TSO)
- Dual TSO
- Verification
- Specification
- Synthesis
Potential Bad Behaviour - Dekker

Initially: \( x = y = 0 \)

- **PO**
  - write: \( x = 1 \)
  - mfence
  - read: \( y = 0 \)
  - critical section

- **P1**
  - write: \( y = 1 \)
  - mfence
  - read: \( x = 0 \)
  - critical section

**TSO**

- **P0**
  - x = 1

- **P1**
  - y = 1

- **x = 0**
- **y = 0**

fence instruction
Potential Bad Behaviour - Dekker

Initially: \( x = y = 0 \)

- P₀
  - write: \( x = 1 \)
  - mfence
  - read: \( y = 0 \)
  - critical section

- P₁
  - write: \( y = 1 \)
  - mfence
  - read: \( x = 0 \)
  - critical section

Potential Bad Behaviour - Dekker

- fence instruction
  - flushes the buffer
Potential Bad Behaviour - Dekker

Initially: $x = y = 0$

- **PO**
  - write: $x = 1$
  - mfence
  - read: $y = 0$
  - critical section

- **P1**
  - write: $y = 1$
  - mfence
  - read: $x = 0$
  - critical section

**TSO**

- **PO**
  - $x = 1$

- **P1**
  - $y = 1$

- $x = 0$
- $y = 0$

**fence instruction**
- flushes the buffer
- prevents re-orchirng
Potential Bad Behaviour - Dekker

Initially: $x = y = 0$

- **P0**
  - write: $x = 1$
  - mfence
  - read: $y = 0$
  - critical section

- **P1**
  - write: $y = 1$
  - mfence
  - read: $x = 0$
  - critical section

**TSO**

- $x = 0$
- $y = 0$
Potential Bad Behaviour - Dekker

Initially: \( x = y = 0 \)

- **P0**
  - write: \( x = 1 \)
  - mfence
  - read: \( y = 0 \)
  - critical section

- **P1**
  - write: \( y = 1 \)
  - mfence
  - read: \( x = 0 \)
  - critical section

[Diagram of processes P0 and P1 with TSO indicating potential bad behaviour]
Potentially Bad Behaviour - Dekker

Initially: \( x = y = 0 \)

- **P0**
  - write: \( x = 1 \)
  - mfence
  - read: \( y = 0 \)
  - critical section

- **P1**
  - write: \( y = 1 \)
  - mfence
  - read: \( x = 0 \)
  - critical section

---

Potential Bad Behaviour - Dekker
Potential Bad Behaviour - Dekker

Initially: \( x = y = 0 \)

- **P0**
  - write: \( x = 1 \)
  - mfence
  - read: \( y = 0 \)
  - critical section

- **P1**
  - write: \( y = 1 \)
  - mfence
  - read: \( x = 0 \)
  - critical section

---

execute

fence

---

**TSO**

**P0**

**P1**

x = 1
y = 0

y = 1
Initially: $x = y = 0$

- **P0**
  - Write: $x = 1$
  - MFENCE
  - Read: $y = 0$
  - Critical section

- **P1**
  - Write: $y = 1$
  - MFENCE
  - Read: $x = 0$
  - Critical section

Potential Bad Behaviour - Dekker
Potential Bad Behaviour - Dekker

Initially: \( x = y = 0 \)

- **P0**
  - write: \( x = 1 \)
  - mfence
  - read: \( y = 0 \)
  - critical section

- **P1**
  - write: \( y = 1 \)
  - mfence
  - read: \( x = 0 \)
  - critical section

Potential Bad Behaviour - Dekker
Initially: $x = y = 0$

**P0**
- write: $x = 1$
- mfence
- read: $y = 0$
- critical section

**P1**
- write: $y = 1$
- mfence
- read: $x = 0$
- critical section

Potential Bad Behaviour - Dekker
Potential Bad Behaviour - Dekker

Initially: \( x = y = 0 \)

- **P0**
  - write: \( x = 1 \)
  - mfence
  - read: \( y = 0 \)
  - critical section

- **P1**
  - write: \( y = 1 \)
  - mfence
  - read: \( x = 0 \)
  - critical section

execute fence

- **TSO**
Initially: $x = y = 0$

$P_0$
- write: $x = 1$
- mfence
- read: $y = 0$
- critical section

$P_1$
- write: $y = 1$
- mfence
- read: $x = 0$
- critical section

Potential Bad Behaviour - Dekker
Initially: \( x = y = 0 \)

- **P0**
  - write: \( x = 1 \)
  - mfence
  - read: \( y = 0 \)
  - critical section

- **P1**
  - write: \( y = 1 \)
  - mfence
  - read: \( x = 0 \)
  - critical section

**Potential Bad Behaviour - Dekker**
Potential Bad Behaviour - Dekker

Initially: $x = y = 0$

- **PO**
  - write: $x = 1$
  - mfence
  - read: $y = 0$
  - critical section

- **P1**
  - write: $y = 1$
  - mfence
  - read: $x = 0$
  - critical section

TSO

**Potential Bad Behaviour** - Dekker
Potential Bad Behaviour - Dekker

Initially: $x = y = 0$

write: $x = 1$

read: $y = 0$

critical section

write: $y = 1$

critical section

At most one process executes its CS at any time
Verification and Correction

specification → program

reachability analysis → reachable?

reachability analysis

execution analysis

preventable?

program correct

program incorrect
Verification and Correction

specification → program

reachability analysis

reachable? yes → execution analysis

preventable? yes → insert fences

program correct

program incorrect
Verification and Correction

specification

program

reachability analysis

reachable?

no

program correct

yes

insert fences

execution analysis

preventable?

no

program incorrect

yes
Verification and Correction

specification

reachability analysis

reachable analysis

program

reachability?

yes

no

program correct

program incorrect

insert fences

execution analysis

preventable?

yes

no
Verification and Correction

Reachability analysis

Reachable?

Execution analysis

Preventable?

Program correct

Program incorrect

Insert fences

Yes

No
Verification and Correction

specification

reachability analysis

program

reachable?

execution analysis

preventable?

program correct

program incorrect

insert fences

yes

no

yes

no

yes
Verification and Correction

1. Specification
2. Program
3. Reachability analysis
   - Reachable?
     - Yes: Execution analysis
     - No: Program correct
4. Insert fences
   - Yes
5. Preventable?
   - No: Program incorrect
Verification and Correction

1. Specify the program
2. Perform reachability analysis
3. Determine if the program is reachable?
   - Yes: Insert fences
   - No: Continue
4. Execute analysis
5. Determine if the program is preventable?
   - Yes: Program correct
   - No: Program may be incorrect
6. Check if there is no reordering
   - Yes: Bug not due to memory model
   - No: Program incorrect
Verification and Correction

reachability analysis → reachable? → execution analysis → preventable?

specification → program

no → program correct

yes, insert fences

no → program incorrect
Verification and Correction

- Reachability analysis
  - Reachable?
    - Yes: Execution analysis
      - Preventable?
        - Yes: Insert fences
        - No: Program correct
    - No: Program correct
  - Yes: Program incorrect
Verification and Correction

reachability analysis

reachable?

execution analysis

preventable?

program correct

program incorrect

insert fences

yes

no
Verification and Correction

specification → program → reachability analysis

reachability analysis → reachable?

reachable? yes → execution analysis

execution analysis → preventable?

preventable? no → program incorrect

preventable? yes → insert fences

insert fences yes → yes

yes → program correct

no → try again
Verification and Correction

specification

reachability analysis

program

reachability analysis

reachable?

yes

execution analysis

preventable?

no

no

program correct

program incorrect

optimality = smallest set of fences needed for correctness
Conclusion

• Weak Consistency
• Total Store Order (TSO)
• Dual TSO

Current Work

• Weak Cache Verification
• Other memory models, e.g., POWER, ARM, C11
• Stateless Model Checking
• Monitor Design
# Experimental Results

Dual-TSO vs Memorax

- **Running time**
- **Memory consumption**

### Dual-TSO vs Memorax

<table>
<thead>
<tr>
<th>Program</th>
<th>#P</th>
<th>Dual-TSO</th>
<th>Memorax</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#T</td>
<td>#C</td>
</tr>
<tr>
<td>SB</td>
<td>5</td>
<td>0.3</td>
<td>10641</td>
</tr>
<tr>
<td>LB</td>
<td>3</td>
<td>0.0</td>
<td>2048</td>
</tr>
<tr>
<td>WRC</td>
<td>4</td>
<td>0.0</td>
<td>1507</td>
</tr>
<tr>
<td>ISA2</td>
<td>3</td>
<td>0.0</td>
<td>509</td>
</tr>
<tr>
<td>RWC</td>
<td>5</td>
<td>0.1</td>
<td>4277</td>
</tr>
<tr>
<td>W+RWC</td>
<td>4</td>
<td>0.0</td>
<td>1713</td>
</tr>
<tr>
<td>IRIW</td>
<td>4</td>
<td>0.0</td>
<td>520</td>
</tr>
<tr>
<td>Nbw_w_wr</td>
<td>2</td>
<td>0.0</td>
<td>222</td>
</tr>
<tr>
<td>Sense_rev_bar</td>
<td>2</td>
<td>0.1</td>
<td>1704</td>
</tr>
<tr>
<td>Dekker</td>
<td>2</td>
<td>0.1</td>
<td>5053</td>
</tr>
<tr>
<td>Dekker_simple</td>
<td>2</td>
<td>0.0</td>
<td>98</td>
</tr>
<tr>
<td>Peterson</td>
<td>2</td>
<td>0.1</td>
<td>5442</td>
</tr>
<tr>
<td>Peterson_loop</td>
<td>2</td>
<td>0.2</td>
<td>7632</td>
</tr>
<tr>
<td>Szymanski</td>
<td>2</td>
<td>0.6</td>
<td>29018</td>
</tr>
<tr>
<td>MP</td>
<td>4</td>
<td>0.0</td>
<td>883</td>
</tr>
<tr>
<td>Ticket_spin_lock</td>
<td>3</td>
<td>0.9</td>
<td>18963</td>
</tr>
<tr>
<td>Bakery</td>
<td>2</td>
<td>2.6</td>
<td>82050</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>2</td>
<td>0.2</td>
<td>8324</td>
</tr>
<tr>
<td>Lamport_fast</td>
<td>3</td>
<td>17.7</td>
<td>292543</td>
</tr>
<tr>
<td>Burns</td>
<td>4</td>
<td>124.3</td>
<td>2762578</td>
</tr>
</tbody>
</table>

[https://www.it.uu.se/katalog/tuang296/dual-tso](https://www.it.uu.se/katalog/tuang296/dual-tso)
Experimental Results

Single buffer approach (exact method [TACAS12+13])

Dual-TSO vs Memorax

- Running time
- Memory consumption

| Program       | #P | Dual-TSO | | Memorax |
|---------------|----|----------|----------|
|               |    | #T       | #C       | #T       | #C       |
| SB            | 5  | 0.3      | 10641    | 559.7    | 10515914 |
| LB            | 3  | 0.0      | 2048     | 71.4     | 1499475  |
| WRC           | 4  | 0.0      | 1507     | 63.3     | 1398393  |
| ISA2          | 3  | 0.0      | 509      | 21.1     | 226519   |
| RWC           | 5  | 0.1      | 4277     | 61.5     | 1196988  |
| W+RWC         | 4  | 0.0      | 1713     | 83.6     | 1389009  |
| IRIW          | 4  | 0.0      | 520      | 34.4     | 358057   |
| Nbw_w_wr      | 2  | 0.0      | 222      | 10.7     | 200844   |
| Sense_rev_bar | 2  | 0.1      | 1704     | 0.8      | 20577    |
| Dekker        | 2  | 0.1      | 5053     | 1.1      | 19788    |
| Dekker_simple | 2  | 0.0      | 98       | 0.0      | 595      |
| Peterson      | 2  | 0.1      | 5442     | 5.2      | 90301    |
| Peterson_loop | 2  | 0.2      | 7632     | 5.6      | 100082   |
| Szymanski     | 2  | 0.6      | 29018    | 1.0      | 26003    |
| MP            | 4  | 0.0      | 883      | TO       |          |
| Ticket_spin_lock | 3  | 0.9    | 18963    | TO       |          |
| Bakery        | 2  | 2.6      | 82050    | TO       |          |
| Dijkstra      | 2  | 0.2      | 8324     | TO       |          |
| Lamport_fast  | 3  | 17.7     | 292543   | TO       |          |
| Burns         | 4  | 124.3    | 2762578  | TO       |          |

https://www.it.uu.se/katalog/tuang296/dual-tso
### Dual-TSO vs Memorax

- **Running time**
- **Memory consumption**

#### Experimental Results

<table>
<thead>
<tr>
<th>Program</th>
<th>#P</th>
<th>Dual-TSO</th>
<th></th>
<th>Memorax</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#T</td>
<td>#C</td>
<td>#T</td>
<td>#C</td>
</tr>
<tr>
<td>SB</td>
<td>5</td>
<td>0.3</td>
<td>10641</td>
<td>559.7</td>
<td>10515914</td>
</tr>
<tr>
<td>LB</td>
<td>3</td>
<td>0.0</td>
<td>2048</td>
<td>71.4</td>
<td>1499475</td>
</tr>
<tr>
<td>WRC</td>
<td>4</td>
<td>0.0</td>
<td>1507</td>
<td>63.3</td>
<td>1398393</td>
</tr>
<tr>
<td>ISA2</td>
<td>3</td>
<td>0.0</td>
<td>509</td>
<td>21.1</td>
<td>226519</td>
</tr>
<tr>
<td>RWC</td>
<td>5</td>
<td>0.1</td>
<td>4277</td>
<td>61.5</td>
<td>1196988</td>
</tr>
<tr>
<td>W+RWC</td>
<td>4</td>
<td>0.0</td>
<td>1713</td>
<td>83.6</td>
<td>1389009</td>
</tr>
<tr>
<td>IRIW</td>
<td>4</td>
<td>0.0</td>
<td>520</td>
<td>34.4</td>
<td>358057</td>
</tr>
<tr>
<td>Nbw_w_wr</td>
<td>2</td>
<td>0.0</td>
<td>222</td>
<td>10.7</td>
<td>200844</td>
</tr>
<tr>
<td>Sense_rev_bar</td>
<td>2</td>
<td>0.1</td>
<td>1704</td>
<td>0.8</td>
<td>20577</td>
</tr>
<tr>
<td>Dekker</td>
<td>2</td>
<td>0.1</td>
<td>5053</td>
<td>1.1</td>
<td>19788</td>
</tr>
<tr>
<td>Dekker_simple</td>
<td>2</td>
<td>0.0</td>
<td>98</td>
<td>0.0</td>
<td>595</td>
</tr>
<tr>
<td>Peterson</td>
<td>2</td>
<td>0.1</td>
<td>5442</td>
<td>5.2</td>
<td>90301</td>
</tr>
<tr>
<td>Peterson_loop</td>
<td>2</td>
<td>0.2</td>
<td>7632</td>
<td>5.6</td>
<td>100082</td>
</tr>
<tr>
<td>Szymanski</td>
<td>2</td>
<td>0.6</td>
<td>29018</td>
<td>1.0</td>
<td>26003</td>
</tr>
<tr>
<td>MP</td>
<td>4</td>
<td>0.0</td>
<td>883</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Ticket_spin_lock</td>
<td>3</td>
<td>0.9</td>
<td>18963</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Bakery</td>
<td>2</td>
<td>2.6</td>
<td>82050</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Dijkstra</td>
<td>2</td>
<td>0.2</td>
<td>8324</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Lamport_fast</td>
<td>3</td>
<td>17.7</td>
<td>292543</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Burns</td>
<td>4</td>
<td>124.3</td>
<td>2762578</td>
<td>TO</td>
<td></td>
</tr>
</tbody>
</table>

*standard benchmarks: litmus tests and mutual exclusion algorithms*
## Experimental Results

### Dual-TSO vs Memorax

- **Running time**
- **Memory consumption**

<table>
<thead>
<tr>
<th>Program</th>
<th>#P</th>
<th>Dual-TSO</th>
<th>Memorax</th>
<th>Running time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>5</td>
<td>0.3</td>
<td>10641</td>
<td>559.7</td>
</tr>
<tr>
<td>LB</td>
<td>3</td>
<td>0.0</td>
<td>2048</td>
<td>71.4</td>
</tr>
<tr>
<td>WRC</td>
<td>4</td>
<td>0.0</td>
<td>1507</td>
<td>63.3</td>
</tr>
<tr>
<td>ISA2</td>
<td>3</td>
<td>0.0</td>
<td>509</td>
<td>21.1</td>
</tr>
<tr>
<td>RWC</td>
<td>5</td>
<td>0.1</td>
<td>4277</td>
<td>61.5</td>
</tr>
<tr>
<td>W+RWC</td>
<td>4</td>
<td>0.0</td>
<td>1713</td>
<td>83.6</td>
</tr>
<tr>
<td>IRIW</td>
<td>4</td>
<td>0.0</td>
<td>520</td>
<td>34.4</td>
</tr>
<tr>
<td>Nbw_w_wr</td>
<td>2</td>
<td>0.0</td>
<td>222</td>
<td>10.7</td>
</tr>
<tr>
<td>Sense_rev_bar</td>
<td>2</td>
<td>0.1</td>
<td>1704</td>
<td>0.8</td>
</tr>
<tr>
<td>Dekker</td>
<td>2</td>
<td>0.1</td>
<td>5053</td>
<td>1.1</td>
</tr>
<tr>
<td>Dekker_simple</td>
<td>2</td>
<td>0.0</td>
<td>98</td>
<td>0.0</td>
</tr>
<tr>
<td>Peterson</td>
<td>2</td>
<td>0.1</td>
<td>5442</td>
<td>5.2</td>
</tr>
<tr>
<td>Peterson_loop</td>
<td>2</td>
<td>0.2</td>
<td>7632</td>
<td>5.6</td>
</tr>
<tr>
<td>Szymanski</td>
<td>2</td>
<td>0.6</td>
<td>29018</td>
<td>1.0</td>
</tr>
<tr>
<td>MP</td>
<td>4</td>
<td>0.0</td>
<td>883</td>
<td>TO</td>
</tr>
<tr>
<td>Ticket_spin_lock</td>
<td>3</td>
<td>0.9</td>
<td>18963</td>
<td>TO</td>
</tr>
<tr>
<td>Bakery</td>
<td>2</td>
<td>2.6</td>
<td>82050</td>
<td>TO</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>2</td>
<td>0.2</td>
<td>8324</td>
<td>TO</td>
</tr>
<tr>
<td>Lamport_fast</td>
<td>3</td>
<td>17.7</td>
<td>292543</td>
<td>TO</td>
</tr>
<tr>
<td>Burns</td>
<td>4</td>
<td>124.3</td>
<td>2762578</td>
<td>TO</td>
</tr>
</tbody>
</table>
## Experimental Results

### Dual-TSO vs Memorax

- **Running time**
- **Memory consumption**

<table>
<thead>
<tr>
<th>Program</th>
<th>#P</th>
<th>Dual-TSO</th>
<th>Memorax</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#T</td>
<td>#C</td>
</tr>
<tr>
<td>SB</td>
<td>5</td>
<td>10641</td>
<td>559.7</td>
</tr>
<tr>
<td>LB</td>
<td>3</td>
<td>2048</td>
<td>71.4</td>
</tr>
<tr>
<td>WRC</td>
<td>4</td>
<td>1507</td>
<td>63.3</td>
</tr>
<tr>
<td>ISA2</td>
<td>3</td>
<td>509</td>
<td>21.1</td>
</tr>
<tr>
<td>RWC</td>
<td>5</td>
<td>4277</td>
<td>61.5</td>
</tr>
<tr>
<td>W+RWC</td>
<td>4</td>
<td>1713</td>
<td>83.6</td>
</tr>
<tr>
<td>IRIW</td>
<td>4</td>
<td>520</td>
<td>34.4</td>
</tr>
<tr>
<td>Nbw_w_wr</td>
<td>2</td>
<td>222</td>
<td>10.7</td>
</tr>
<tr>
<td>Sense_rev_bar</td>
<td>2</td>
<td>1704</td>
<td>0.8</td>
</tr>
<tr>
<td>Dekker</td>
<td>2</td>
<td>5053</td>
<td>1.1</td>
</tr>
<tr>
<td>Dekker_simple</td>
<td>2</td>
<td>98</td>
<td>0.0</td>
</tr>
<tr>
<td>Peterson</td>
<td>2</td>
<td>5442</td>
<td>5.2</td>
</tr>
<tr>
<td>Peterson_loop</td>
<td>2</td>
<td>7632</td>
<td>5.6</td>
</tr>
<tr>
<td>Szymanski</td>
<td>2</td>
<td>29018</td>
<td>1.0</td>
</tr>
<tr>
<td>MP</td>
<td>4</td>
<td>883</td>
<td>TO</td>
</tr>
<tr>
<td>Ticket_spin_lock</td>
<td>3</td>
<td>18963</td>
<td>TO</td>
</tr>
<tr>
<td>Bakery</td>
<td>2</td>
<td>82050</td>
<td>TO</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>2</td>
<td>8324</td>
<td>TO</td>
</tr>
<tr>
<td>Lamport_fast</td>
<td>3</td>
<td>292543</td>
<td>TO</td>
</tr>
<tr>
<td>Burns</td>
<td>4</td>
<td>2762578</td>
<td>TO</td>
</tr>
</tbody>
</table>

*Generated configurations*
Experimental Results

<table>
<thead>
<tr>
<th>Program</th>
<th>#P</th>
<th>Dual-TSO #T</th>
<th>Dual-TSO #C</th>
<th>Memorax #T</th>
<th>Memorax #C</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>5</td>
<td>0.3</td>
<td>10641</td>
<td>559.7</td>
<td>10515914</td>
</tr>
<tr>
<td>LB</td>
<td>3</td>
<td>0.0</td>
<td>2048</td>
<td>71.4</td>
<td>1499475</td>
</tr>
<tr>
<td>WRC</td>
<td>4</td>
<td>0.0</td>
<td>1507</td>
<td>63.3</td>
<td>1398393</td>
</tr>
<tr>
<td>ISA2</td>
<td>3</td>
<td>0.0</td>
<td>509</td>
<td>21.1</td>
<td>226519</td>
</tr>
<tr>
<td>RWC</td>
<td>5</td>
<td>0.1</td>
<td>4277</td>
<td>61.5</td>
<td>1196988</td>
</tr>
<tr>
<td>W+RWC</td>
<td>4</td>
<td>0.0</td>
<td>1713</td>
<td>83.6</td>
<td>1389009</td>
</tr>
<tr>
<td>IRIW</td>
<td>4</td>
<td>0.0</td>
<td>520</td>
<td>34.4</td>
<td>358057</td>
</tr>
<tr>
<td>Nbw_w_wr</td>
<td>2</td>
<td>0.0</td>
<td>222</td>
<td>10.7</td>
<td>200844</td>
</tr>
<tr>
<td>Sense_rev_bar</td>
<td>2</td>
<td>0.1</td>
<td>1704</td>
<td>0.8</td>
<td>20577</td>
</tr>
<tr>
<td>Dekker</td>
<td>2</td>
<td>0.1</td>
<td>5053</td>
<td>1.1</td>
<td>19788</td>
</tr>
<tr>
<td>Dekker_simple</td>
<td>2</td>
<td>0.0</td>
<td>98</td>
<td>0.0</td>
<td>595</td>
</tr>
<tr>
<td>Peterson</td>
<td>2</td>
<td>0.1</td>
<td>5442</td>
<td>5.2</td>
<td>90301</td>
</tr>
<tr>
<td>Peterson_loop</td>
<td>2</td>
<td>0.2</td>
<td>7632</td>
<td>5.6</td>
<td>100082</td>
</tr>
<tr>
<td>Szymanski</td>
<td>2</td>
<td>0.6</td>
<td>29018</td>
<td>1.0</td>
<td>26003</td>
</tr>
<tr>
<td>MP</td>
<td>4</td>
<td>0.0</td>
<td>883</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Ticket_spin_lock</td>
<td>3</td>
<td>0.9</td>
<td>18963</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Bakery</td>
<td>2</td>
<td>2.6</td>
<td>82050</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Dijkstra</td>
<td>2</td>
<td>0.2</td>
<td>8324</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Lamport_fast</td>
<td>3</td>
<td>17.7</td>
<td>292543</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Burns</td>
<td>4</td>
<td>124.3</td>
<td>2762578</td>
<td>TO</td>
<td></td>
</tr>
</tbody>
</table>

**Dual-TSO vs Memoriax**

- Running time
- Memory consumption

**Dual-TSO is faster and uses less memory in most of examples**
Experimental Results
Parameterised Cases

<table>
<thead>
<tr>
<th>Program</th>
<th>Dual-TSO #T</th>
<th>Dual-TSO #C</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>0.0</td>
<td>147</td>
</tr>
<tr>
<td>LB</td>
<td>0.6</td>
<td>1028</td>
</tr>
<tr>
<td>MP</td>
<td>0.0</td>
<td>149</td>
</tr>
<tr>
<td>WRC</td>
<td>0.8</td>
<td>618</td>
</tr>
<tr>
<td>ISA2</td>
<td>4.3</td>
<td>1539</td>
</tr>
<tr>
<td>RWC</td>
<td>0.2</td>
<td>293</td>
</tr>
<tr>
<td>W+RWC</td>
<td>1.5</td>
<td>828</td>
</tr>
<tr>
<td>IRIW</td>
<td>4.6</td>
<td>648</td>
</tr>
</tbody>
</table>
Experimental Results
Parameterised Cases

unbounded number of processes

<table>
<thead>
<tr>
<th>Program</th>
<th>Dual-TSO #T</th>
<th>Dual-TSO #C</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>0.0</td>
<td>147</td>
</tr>
<tr>
<td>LB</td>
<td>0.6</td>
<td>1028</td>
</tr>
<tr>
<td>MP</td>
<td>0.0</td>
<td>149</td>
</tr>
<tr>
<td>WRC</td>
<td>0.8</td>
<td>618</td>
</tr>
<tr>
<td>ISA2</td>
<td>4.3</td>
<td>1539</td>
</tr>
<tr>
<td>RWC</td>
<td>0.2</td>
<td>293</td>
</tr>
<tr>
<td>W+RWC</td>
<td>1.5</td>
<td>828</td>
</tr>
<tr>
<td>IRIW</td>
<td>4.6</td>
<td>648</td>
</tr>
</tbody>
</table>
Experimental Results
Parameterised Cases

increasing the number of processes

<table>
<thead>
<tr>
<th>Program</th>
<th>Dual-TSO #T</th>
<th>Dual-TSO #C</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>0.0</td>
<td>147</td>
</tr>
<tr>
<td>LB</td>
<td>0.6</td>
<td>1028</td>
</tr>
<tr>
<td>MP</td>
<td>0.0</td>
<td>149</td>
</tr>
<tr>
<td>WRC</td>
<td>0.8</td>
<td>618</td>
</tr>
<tr>
<td>ISA2</td>
<td>4.3</td>
<td>1539</td>
</tr>
<tr>
<td>RWC</td>
<td>0.2</td>
<td>293</td>
</tr>
<tr>
<td>W+RWC</td>
<td>1.5</td>
<td>828</td>
</tr>
<tr>
<td>IRIW</td>
<td>4.6</td>
<td>648</td>
</tr>
</tbody>
</table>
Experimental Results
Parameterised Cases

Dual-TSO is more scalable
Dual-TSO is more **efficient** and **scalable**

**Experimental Results**

**Parameterised Cases**

<table>
<thead>
<tr>
<th>Program</th>
<th>Dual-TSO #T</th>
<th>Dual-TSO #C</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>0.0</td>
<td>147</td>
</tr>
<tr>
<td>LB</td>
<td>0.8</td>
<td>618</td>
</tr>
<tr>
<td>ISA2</td>
<td>4.3</td>
<td>1539</td>
</tr>
<tr>
<td>MP</td>
<td>0.0</td>
<td>149</td>
</tr>
<tr>
<td>WRC</td>
<td>0.8</td>
<td>618</td>
</tr>
<tr>
<td>RWC</td>
<td>0.2</td>
<td>293</td>
</tr>
<tr>
<td>W+RWC</td>
<td>1.5</td>
<td>828</td>
</tr>
<tr>
<td>IRIW</td>
<td>4.6</td>
<td>648</td>
</tr>
</tbody>
</table>