Verifying Weakly Consistent Systems (TSO as an Example)

Parosh Aziz Abdulla
Mohamed Faouzi Atig
Ahmed Bouajjani
Tuan Phong Ngo

1 Uppsala University
2 IRIF, Université Paris Diderot & IUF
Outline

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

• **Weak Consistency**
  • Total Store Order (TSO)
  • Dual TSO
  • Verification
  • Monitors
  • 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

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

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

Processes

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

Execution

P1: \text{w}(x,1) \rightarrow P2: \text{r}(x,1) \rightarrow P2: \text{w}(y,1)
Sequential Consistency (SC)

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

Processes

P1: write
P2: write

P1: read
P2: read

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

Execution

Memory
Sequential Consistency (SC)

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

**P1**: \( w(x,1) \)  
**P2**: \( r(x,1) \)  
**P2**: \( w(y,1) \)  
**P1**: \( r(y,1) \)
Sequential Consistency (SC)

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

\[ P1: \text{write: } x = 1 \]
\[ P1: \text{write: } x = 2 \]
\[ P1: \text{read: } x = 2 \]
\[ P1: \text{read: } y = 0 \]
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
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
TSO - Classical Semantics

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

read from 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

update memory

P1

P2

x=2

x=1

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

update memory

P1

P2

x=2

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: \( y = 0 \)

**Extra behaviors**
- 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: $x = y = 0$

write: $x = 1$

read: $y = 0$

critical section

write: $y = 1$

read: $x = 0$

critical section
Initially: $x = y = 0$

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

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

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

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

write to buffer
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: \( y = 0 \)
  - critical section

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

**TSO**

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

read from memory
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**
- **P2**

**enter CS**
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**

$x = 0$
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
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
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
Dekker Protocol

Initially: $x = y = 0$

write: $x = 1$

read: $y = 0$

P1

critical section

P2

write: $y = 1$

read: $x = 0$

critical section

P1

$x = 1$

P2

$y = 1$

tso

$x = 0$

$y = 0$

read from memory
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**

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

**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

TSO
Dekker Protocol

Initially: $x = y = 0$

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

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

P1

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

P2

TSO

x=1

y=1
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"

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

“read overtaking write”

Dekker Protocol
Weakly Consistent Systems

- Cloud
- Weak memories
- Weak cache protocols
- Languages: C11

+ Efficiency
- Non-intuitive behaviours
Weakly Consistent Systems

- Cloud
- Weak memories
- Weak cache protocols
- Languages: C11

+ Efficiency
- Non-intuitive behaviours

- Semantics
- Correctness analysis: simulation, testing, verification, synthesis
- Methods and tools: decidability, complexity, algorithms
- Monitoring
Outline

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

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

Additionally:

- `fence` instruction
- flushes the buffer
- TSO

Potential Bad Behaviour - Dekker
Potential Bad Behaviour - Dekker

Initially: $x = y = 0$

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

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

PO

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

P1

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

P0

P1

TSO

fence instruction

flushes the buffer

prevents re-ordering

Potential Bad Behaviour - Dekker
Verification and Correction

specification

reachability analysis

program

reachable?

insert fences

execution analysis

preventable?

program correct

no

program incorrect

no
Verification and Correction

specification

reachability analysis

program

reachable?

reachability analysis

insert fences

yes

execution analysis

preventable?

no

program correct

no

program incorrect
Verification and Correction

reachability analysis

reachability?

program correct

program incorrect

specification

insert fences

no

yes

no

program correct

program incorrect
Verification and Correction

specification

reachability analysis

program

reachability analysis

reachable?

no

program correct

yes

execution analysis

preventable?

no

program incorrect

yes

insert fences

39
Verification and Correction

specification

program

reachability analysis

reachable?

yes

no

program correct

program incorrect

execution analysis

preventable?

yes

no

insert fences

yes

no
Verification and Correction

1. **Specification**
2. **Program**
3. **Reachability Analysis**
   - **Reachable?**
     - **Yes**
     - **No**
6. **Execution Analysis**
   - **Preventable?**
     - **Yes**
     - **No**
9. **Program Correct**
10. **Program Incorrect**
Verification and Correction

specification → program

reachability analysis

program → reachable?

reachability analysis → execution analysis

reachable? yes → preventable?

yes → program correct

no → program incorrect

execute fences

no → preventable?

no → program incorrect

yes → preventable?

no → program incorrect

yes → program correct
Verification and Correction

reachability analysis

specification

insert fences

reachable?

reachability analysis

program

execution analysis

preventable?

program correct

no reordering = bug not due to memory model

program incorrect

no reordering

specification

program

yes
Verification and Correction

specification

program

reachability analysis

reachable?

yes

no

program correct

execution analysis

preventable?

yes

no

program incorrect

insert fences

yes

no
Verification and correction

specification

reachability analysis

program

reachability analysis

reachable?

execution analysis

preventable?

program correct

program incorrect
Verification and Correction

- **Specification**
  - Program
  - Reachability Analysis
  - Reachable?
    - Yes
    - No
      - Program Correct
      - Program Incorrect

- Insert Fences
  - Preventable?
    - Yes
    - No
Verification and Correction

1. Specification
2. Program
3. Reachability Analysis
   - Reachable?
     - Yes: Insert fences
     - No: Try again
4. Execution Analysis
   - Preventable?
     - Yes: Program incorrect
     - No: Program correct

Verification and Correction

- Specification
- Reachability analysis
- Reachable?
- Program correct
- Program incorrect
- Insert fences
- Execution analysis
- Preventable?
- Optimal = Smallest set of fences needed for correctness
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

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

x = 0
y = 0
Verification under TSO is Difficult

while (1)
write: x=1

PO: write: x = 1
... 
PO: 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
...
```
Verification under TSO is Difficult

while (1)
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
... 

unbounded buffer

x = 0
y = 0
Verification under TSO is Difficult

while (1)
  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
- Monitors
- Synthesis
Dual TSO

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

P1

P2

x,1,self ➔ x=0

y,2,other ➔ y=0
Dual TSO

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

Processes

FIFO buffer

Load buffer

Shared variables

P1

P2

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

P1

P2

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,self

P2

y=0

x=1
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

**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

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: \( y = 0 \)
**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

remove oldest 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
Dual TSO

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

read oldest
write

x=1
y=0
y=0,other
**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

\( y = 0, \text{other} \)
**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: $y = 0$

- **TSO ≡ Dual-TSO**

  - 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

reachability
Classical TSO

P1

P2

x=0

y=0
P1: $w(x, 2)$

Classical TSO
P1: \( w(x,2) \)

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

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

Dual TSO

Classical TSO
P1: \(w(x,2)\)

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

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

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

Dual TSO

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

x=0

y=0, other

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)
Dual TSO

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)

Dual TSO

Classical TSO

P2: w(y,1)

P1: w(y,1)
**Dual TSO**

**Classical TSO**

- **P1**: \( w(x,2) \rightarrow r(y,0) \rightarrow w(y,1) \rightarrow w(x,1) \rightarrow r(x,2) \)
- **P2**: \( w(y,1) \rightarrow w(x,1) \)

**Variables**:
- \( x = 0 \)
- \( y = 0, \text{other} \)
- \( y = 1, \text{self} \)
- \( y = 1 \)
\( x = 1 \)
\( y = 1 \)
\( y = 0, \text{other} \)

**P1**
- \( w(x,2) \)
- \( r(y,0) \)

**P2**
- \( w(y,1) \)
- \( w(x,1) \)
- \( r(x,2) \)

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

**Classical TSO**
- \( P2: w(y,1) \)
- \( P2: w(x,1) \)
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)

P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)
\begin{itemize}
  \item P1: \texttt{w(x,2)}
  \item P1: \texttt{r(y,0)}
  \item P2: \texttt{w(y,1)}
  \item P2: \texttt{w(x,1)}
  \item P2: \texttt{r(x,2)}
\end{itemize}

\textbf{Classical TSO}

\begin{itemize}
  \item P2: \texttt{w(x,1)}
  \item P1: \texttt{w(x,2)}
  \item P1: \texttt{r(y,0)}
  \item P2: \texttt{w(y,1)}
  \item P2: \texttt{w(x,1)}
  \item P2: \texttt{r(x,2)}
\end{itemize}

\textbf{Dual TSO}

\begin{itemize}
  \item P2: \texttt{w(y,1)}
  \item P2: \texttt{w(x,1)}
  \item P2: \texttt{w(y,1)}
  \item P2: \texttt{w(x,1)}
  \item P2: \texttt{r(x,2)}
\end{itemize}
Dual TSO

P2: w(y,1) → P2: w(x,1) → P1: w(x,2)

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
Dual TSO

Classical TSO
Dual TSO

Classical TSO
Dual TSO

Classical TSO
Outline

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

partition of load buffer

Old

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

New
Dual TSO - Monotonicity

Partition of load buffer

Newest self message on y

Old

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

New
Dual TSO - Monotonicity

partition of load buffer

newest self message on x

newest self message on y

Old

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

New
Dual TSO - Monotonicity

partition of load buffer

newest self message on x

newest self message on y

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

Old  New
Dual TSO - Monotonicity

Ordering on Buffers

Buffers:
- x=2, self
- y=1, self
- x=1, other
- y=0, self
- x=0, other

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

Ordering on Buffers

\[
\begin{align*}
(x=2, \text{self}) & \quad (y=1, \text{self}) & \quad (x=1, \text{other}) & \quad (y=0, \text{self}) & \quad (x=0, \text{other}) \\
& \leq & \leq & \leq & \leq
\end{align*}
\]
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

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

Ordering on Buffers

\[
\begin{array}{cccc}
x=2, \text{self} & y=1, \text{self} & x=1, \text{other} & y=0, \text{self} & x=0, \text{other} \\
\end{array}
\]

\[
\begin{array}{cc}
\text{subword} & \text{subword} \\
\end{array}
\]
Dual TSO - Monotonicity

Ordering on Buffers

ab \sqsubseteq xaybz
Dual TSO - Monotonicity

Ordering on Buffers

\[ ab \subseteq xaybzc \]
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

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

P1

P2

x = 1, self

x, 1, 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

\[ c_1 \xrightarrow{\text{Monotonicity}} c_2 \]

\[ c_3 \]
Dual TSO - Monotonicity

Ordering on Configurations

\[ c_1 \xrightarrow{\text{Monotonicity}} c_2 \]
\[ c_3 \xrightarrow{\text{Monotonicity}} c_4 \]
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>#T</th>
<th>#C</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>yes</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

<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>

Time (secs) and # generated configurations
Outline

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

? \Rightarrow SC
? \models \text{sc}
Cache Coherence Protocol

monitors

TSO
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
arios@ditec.um.es
Stefanos Kaxiras
Department of Information Technology
Uppsala Universitet, Sweden
stefanos.kaxiras@it.uu.se

monitors
Pl: \( w(x,1) \)
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: \( r(x,2) \)
P1: w(x,1) → P2: r(x,1) → P3: w(x,2) → P4: r(x,2) → P5: r(x,1)

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

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

TSO \equiv \text{12 counter-examples}
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

<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 Memorax

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

https://www.it.uu.se/katalog/tuang296/dual-tso
Experimental Results

Dual-TSO vs Memorax

- Running time
- Memory consumption

<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>

https://www.it.uu.se/katalog/tuang296/dual-tso
Experimental Results

### Dual-TSO vs Memorax

- Running time
- Memory consumption

#### Standard Benchmarks:

- Litmus tests and mutual exclusion algorithms

<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>
## Experimental Results

<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>

**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>

Dual-TSO vs Memorax

- Running time
- Memory consumption

Generated configurations
Dual-TSO vs Memorax

- Running time
- Memory consumption

Dual-TSO is faster and uses less memory in most of examples

Experimental Results

<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>
Experimental Results
Parameterised Cases

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

Experimental Results
Parameterised Cases
Experimental Results
Parameterised Cases

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

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