Weak Consistency
(TSO as an Example)

Parosh Aziz Abdulla¹
Mohamed Faouzi Atig¹
Ahmed Bouajjani²
Tuan Phong Ngo¹

¹Uppsala University
²IRIF, 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
Sequential Consistency (SC)

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

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

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

Processes:

- P1
  - write
  - read

- P2
  - write
  - read

Execution:

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
Sequential Consistency (SC)

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

+ Simple and intuitive
- Too strong
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

\[ P0: w(x,1) \]
Cloud Computing

- Processes perform local operations
- Operations propagated asynchronously

P0: \( w(x,1) \)

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

Execution
**Cloud Computing**

- Processes perform local operations
- Operations propagated asynchronously

**Execution**

- **P0:** `w(x,1)`
- **P1:** `w(x,2)`
- **P2:** `r(x,1)`
**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

![Diagram showing cloud computing operations]

**Execution**

- P0: \textcolor{red}{w(x,1)}
- P1: \textcolor{blue}{w(x,2)}
- P2: \textcolor{green}{r(x,1)}
- P3: \textcolor{black}{r(x,0)}
- P3: \textcolor{black}{r(x,1)}
- P2: \textcolor{green}{r(x,2)}
- P3: \textcolor{black}{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
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: $y = 0$

![Diagram showing the execution order of P1 and P2 with variables x and y]
**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: \( y = 0 \)

Diagram:

- **P1:** \( x = 2 \) → **P2:** \( x = 2 \) → **update memory:**
  - **P1:** \( x = 1 \)
  - **P2:** \( y = 0 \)
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: $y = 0$

- Write to buffer
- Read from buffer
- Read from memory
- Update memory

TSO

- Extra behaviors
- Potentially bad behaviors
Dekker Protocol

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**

$y = 0$

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

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

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

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

- **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

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
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: $y = 0$

**P2**

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

Critical section

**TSO**

$read: x = 0$

$read: y = 0$

$read from memory$
Initially: \( x = y = 0 \)

- **P1**
  - Write: \( x = 1 \)
  - Read: \( y = 0 \)
  - Critical section

- **P2**
  - Write: \( y = 1 \)
  - Read: \( x = 0 \)
  - Critical section
Initially: $x = y = 0$

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

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

TSO
Initially: $x = y = 0$

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
- write to buffer

Dekker Protocol
Initially: $x = y = 0$

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

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

P1 enters the critical section:

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

P2 enters the critical section:

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

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

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

**TSO**

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

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

write: $x = 1$

read: $y = 0$

critical section

write: $y = 1$

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
Initially: $x = y = 0$

- P1: write: $x = 1$
- P1: read: $y = 0$
- P2: read: $x = 0$
- P2: write: $y = 1$

Dekker Protocol

“read overtaking write”
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

“read overtaking write”
Weakly Consistent Systems

- Microprocessors:
  - TSO, POWER, ARM, ...
- Weak Cache Protocols:
  - TSO-CC, Racer, SISD, ...
- Programming Languages:
  - C11, Java, ...
- Distributed Data Stores:
  - Amazon, Facebook, Google, ...

+ Efficiency
- Non-intuitive behaviours
Weakly Consistent Systems

- Microprocessors:
  - TSO, POWER, ARM, ...
- Weak Cache Protocols:
  - TSO-CC, Racer, SISD, ...
- Programming Languages:
  - C11, Java, ...
- Distributed Data Stores:
  - Amazon, Facebook, Google, ...

+ Efficiency
- Non-intuitive behaviours

- Semantics

Correctness analysis: simulation, testing, verification, synthesis

Methods and tools: decidability, complexity, algorithms

Specifications
Weakly Consistent Systems

- **Microprocessors:**
  - TSO, POWER, ARM, ...

- **Weak Cache Protocols:**
  - TSO-CC, Racer, SISD, ...

- **Programming Languages:**
  - C11, Java, ...

- **Distributed Data Stores:**
  - Amazon, Facebook, Google, ...

+ 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

P0

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
...
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
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
    ...

unbounded buffer

x = 0
y = 0
Verification under TSO is Difficult

```plaintext
while (1)
  write: x=1

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

infinite state space
```
Outline

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

- store buffer \xrightarrow{\text{load buffer}}
- write \textit{immediately updates} memory
- buffers contain expected reads
- messages: self, other

```
\begin{align*}
\text{P1} & \quad x,1,\text{self} \\
\text{P2} & \quad y,2,\text{other}
\end{align*}
```

\begin{align*}
\text{FIFO buffer} & \quad x=0 \\
\text{load buffer} & \quad y=0
\end{align*}

\text{shared variables}
Dual TSO

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

P2
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

P1

P2

x=1,self

y=0
**Dual TSO**

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

**Diagram:**
- **P1:**
  - Propagate to yourself
  - Update memory
- **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: \( y = 0 \)
Dual TSO

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

x=1,self

x=1

y=0,other

propagate from memory
Dual TSO

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

P2: \( x = 1, \text{self} \)
P2: \( y = 0, \text{other} \)

propagate from memory
Dual TSO

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

Dual TSO

P1

**P1**

x=1,self
y=0,other

P2

read own
write

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

P2

x=1\_\text{self} \quad y=0\_\text{other}

remove oldest write

P1

x=1

P2

y=0
Dual TSO

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

P2

remove oldest write

y=0, other

P1

x=1

y=0

P2
Dual TSO

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

x = 1
y = 0
y = 0, other
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

P1
y=0,other

P2

x=1
y=0
**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

\[ x = 1 \]
\[ y = 0 \]
**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 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}

reachability
Classical TSO
$P_1: \text{w}(x,2)$

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

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

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

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

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

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
Classical 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

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

Dual TSO

Classical TSO
y = 0
x = 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

x = 0
y = 0
y = 0, other
Dual TSO

Classical TSO
Dual TSO

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

P2: w(y,1)

Dual TSO

Classical TSO
**Dual TSO**

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

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

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

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

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

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

Dual TSO

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

\[
\begin{align*}
P1: \ w(y,1) \quad & \quad P2: \ w(x,1) \\
P2: \ w(x,1) \quad & \quad P1: \ w(x,2)
\end{align*}
\]

Dual TSO

\[
\begin{align*}
P2: \ w(y,1) \quad & \quad P2: \ w(x,1) \\
P1: \ w(x,2) \quad & \quad P1: \ r(y,0) \\
P2: \ w(y,1) \quad & \quad P2: \ w(x,1) \\
P2: \ r(x,2)
\end{align*}
\]

Classical TSO
Dual TSO

Classical TSO
**Dual TSO**

**Classical TSO**

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

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) → 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)

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

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

Dual TSO

Classical TSO
Outline

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

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

Partition of load buffer

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

Partition of load buffer

Newest self message on x

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

#### Ordering on Buffers

<table>
<thead>
<tr>
<th>x=2, self</th>
<th>y=1, self</th>
<th>x=1, other</th>
<th>y=0, self</th>
<th>x=0, other</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=2, self</td>
<td>y=1, self</td>
<td>x=1, other</td>
<td>y=0, self</td>
<td>x=0, other</td>
</tr>
</tbody>
</table>
Dual TSO - Monotonicity

Ordering on Buffers
Dual TSO - Monotonicity

Ordering on Buffers
Dual TSO - Monotonicity

Ordering on Buffers

\[ x=2, \text{self} \quad y=1, \text{self} \quad x=1, \text{other} \quad y=0, \text{self} \quad x=0, \text{other} \]

\[ x=2, \text{self} \quad y=1, \text{self} \quad x=1, \text{other} \quad y=0, \text{self} \quad x=0, \text{other} \]

\[ \text{subword} \]

\[ \text{subword} \]
Dual TSO - Monotonicity

Ordering on Buffers

ab \sqsubseteq xaybzz
Dual TSO - Monotonicity

Ordering on Buffers

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

$ab \sqsubseteq xaybz$

subword

subword
Dual TSO - Monotonicity

Ordering on Configurations

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

\[ x = 1 \]
\[ y = 0 \]
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

\[ x = 1 \]
\[ y = 0 \]
Dual TSO - Monotonicity

Ordering on Configurations

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

Ordering on Configurations

\[ c_1 \rightarrow c_2 \]

Monotonicity
Dual TSO - Monotonicity

Ordering on Configurations

\[ v_1, c_1, c_2, v_3, c_4 \]
**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</th>
<th>#T</th>
<th>#C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>5</td>
<td>yes</td>
<td>0.3</td>
<td>10641</td>
</tr>
<tr>
<td>LB</td>
<td>3</td>
<td>yes</td>
<td>0.0</td>
<td>2048</td>
</tr>
<tr>
<td>WRC</td>
<td>4</td>
<td>yes</td>
<td>0.0</td>
<td>1507</td>
</tr>
<tr>
<td>ISA2</td>
<td>3</td>
<td>yes</td>
<td>0.0</td>
<td>509</td>
</tr>
<tr>
<td>RWC</td>
<td>5</td>
<td>no</td>
<td>0.1</td>
<td>4277</td>
</tr>
<tr>
<td>W+RWC</td>
<td>4</td>
<td>no</td>
<td>0.0</td>
<td>1713</td>
</tr>
<tr>
<td>IRIIW</td>
<td>4</td>
<td>yes</td>
<td>0.0</td>
<td>520</td>
</tr>
<tr>
<td>MP</td>
<td>4</td>
<td>yes</td>
<td>0.0</td>
<td>883</td>
</tr>
<tr>
<td>Simple Dekker</td>
<td>2</td>
<td>no</td>
<td>0.0</td>
<td>98</td>
</tr>
<tr>
<td>Dekker</td>
<td>2</td>
<td>no</td>
<td>0.1</td>
<td>5053</td>
</tr>
<tr>
<td>Peterson</td>
<td>2</td>
<td>no</td>
<td>0.1</td>
<td>5442</td>
</tr>
<tr>
<td>Repeated Peterson</td>
<td>2</td>
<td>yes</td>
<td>0.2</td>
<td>7632</td>
</tr>
<tr>
<td>Bakery</td>
<td>2</td>
<td>no</td>
<td>2.6</td>
<td>82050</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>2</td>
<td>no</td>
<td>0.2</td>
<td>8324</td>
</tr>
<tr>
<td>Szymanski</td>
<td>2</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>0.9</td>
<td>18963</td>
</tr>
<tr>
<td>Lamport’s Fast Mutex</td>
<td>3</td>
<td>no</td>
<td>17.7</td>
<td>292543</td>
</tr>
<tr>
<td>Burns</td>
<td>4</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>0.0</td>
<td>222</td>
</tr>
<tr>
<td>Sense Reversing Barrier</td>
<td>2</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

<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
Cache Coherence Protocol

? \[ \equiv \] 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

?? monitors
TSO-Counter-Examples
P1: \( w(x,1) \)  \( \rightarrow \)  P2: \( r(x,1) \)  \( \rightarrow \)  P3: \( w(x,2) \)

TSO-Counter-Examples
TSO-Counter-Examples

P1: w(x,1) → P2: r(x,1) → P3: w(x,2) → P4: r(x,2)
TSO ⋄ 12 counter-examples
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

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

**TSO**

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

- fence instruction
- flushes the buffer
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
- prevents re-ordering
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

94
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

TSO

$y = 1$

$P0 \rightarrow y = 1$

$P1 \rightarrow x = 1$

$y = 0$

$P0 \rightarrow P1$

$P1 \rightarrow P0$
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

execute
fence

$y = 1$

$P0$ -> $P1$ -> $x = 1$

$P0$ -> $y = 0$

TSO

$97$
Potential Bad Behaviour - Dekker

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
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 = 1, y = 1)
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

$100$

$P0$

$P1$

$x = 1$

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

write: $x = 1$

read: $y = 0$

critical section

write: $y = 1$

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

$T S O$

$P0$

$P1$

$x = 1$

$y = 1$
Potential Bad Behaviour - Dekker

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

$P_0 \to TSO \to P_1 \to TSO$

$P_0$

$P_1$

$P_1 \to TSO \to P_0 \to TSO$

$x = 1$

$y = 1$
Potential Bad Behaviour - Dekker

Initially: $x = y = 0$

- $P_0$ writes: $x = 1$
- `mfence`
- $P_0$ reads: $y = 0$
- `critical section`

- $P_1$ writes: $y = 1$
- `mfence`
- $P_1$ reads: $x = 0$
- `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

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

1. **specification**
2. **program**
3. **reachability analysis**
   - **reachability?**
     - yes
     - no
       - **program correct**
4. **insert fences**
5. **execution analysis**
   - **preventable?**
     - yes
     - no
       - **program incorrect**
     - **no**
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

program

reachability analysis

reachable? yes

execution analysis

preventable? yes

program correct

insert fences

no

program incorrect

no

yes

107
Verification and Correction

specification → program

reachability analysis → reachable?

reachable? → execution analysis

preventable? → yes

program correct

no

insert fences

Program incorrect
Verification and Correction

specification

reachability analysis

program

reachable?

reachability analysis

execution analysis

preventable?

program correct

program incorrect

insert fences

yes

no
Verification and Correction

specification

program

reachability analysis

reachable?

reachability analysis

execution analysis

preventable?

program correct

program incorrect

insert fences

yes

no

yes

no

yes

no

specification

program
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

1. **specification**
2. **program**
3. **reachability analysis**
4. **reachable?**
5. **execution analysis**
6. **preventable?**
7. **insert fences**
8. **yes**
9. **no reordering**
10. **program correct**
11. **no**
12. **program incorrect**
13. **bug not due to memory model**
Verification and Correction

specification

program

reachability analysis

reachable?

no

program correct

yes

reachability analysis

execution analysis

preventable?

no

program incorrect

yes

insert fences
Verification and Correction

reachability analysis

specification

program

reachable? yes

reachability analysis

no

program correct

execution analysis

preventable?

no

program incorrect

yes

find reordering and prevent it

insert fences

specification

program
Verification and Correction

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

113
Verification and Correction

reachability analysis

specification

program

reachable?

reachability analysis

try again

program correct

execution analysis

preventable?

program incorrect

insert fences

yes

no

yes

no

113
Verification and Correction

reachability analysis

specification

program

reachability analysis

reachable?

program correct

try again

optimal = 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**

## Experimental Results Table

<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](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

<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
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

*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>
</tr>
</thead>
<tbody>
<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

## 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>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

Dual-TSO vs Memorax

- Running time
- Memory consumption

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

<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

<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>
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>
</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>

Graphs showing the trend of increasing processes for different programs.
Experimental Results
Parameterised Cases

Dual-TSO is more scalable
Experimental Results
Parameterised Cases

Dual-TSO is more efficient and scalable

<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>