Unfair Scheduling Patterns in NUMA Architectures

Naama Ben-David
Ziv Scully
Guy Blelloch

Carnegie Mellon University
NUMA Architectures + Concurrent Programs
NUMA Architectures

cores
L1/L2
L3
memory
NUMA Architectures

Local access faster than remote access
Concurrent Programs

- parallel section
- atomic section
Concurrent Programs

- parallel section
- atomic section
- must avoid data races
Concurrent Programs

must avoid data races

parallel section

atomic section

Lock-Based

lock

unlock
Concurrent Programs

- **Parallel Section**
- **Atomic Section**

- Must avoid data races

Lock-Based

- **Lock**
- **Unlock**
- Wait for unlock...
Concurrent Programs

- **parallel section**
- **atomic section**

must avoid data races

**Lock-Based**
- lock
- unlock

wait for unlock...

**Lock-Free**
- read
- CAS
Concurrent Programs

- Concurrent Programs
  - Parallel section: must avoid data races
  - Atomic section

- Lock-Based
  - Lock
  - Unlock: wait for unlock...

- Lock-Free
  - Read: "start transaction"
  - CAS: "try to commit"
Concurrent Programs

must avoid data races

parallel section

atomic section

If fail

Lock-Based

lock

wait for unlock...

unlock

Lock-Free

read

“start transaction”

if fail

CAS

“try to commit”
node 0  node 1

node 3  node 2
**Question**: where should we allocate shared data?
**Question**: where should we allocate shared data?

**Conventional wisdom**: put data near computation
Question: where should we allocate shared data? Conventional wisdom: put data near computation
**Question:** where should we allocate shared data?

**Conventional wisdom:** put data near computation

**Problem:** conventional wisdom is for **lock-based** algorithms
NUMA Architectures
+
Concurrent Programs
NUMA Architectures + Lock-Free Algorithms

read

if fail

CAS
NUMA Architectures + Lock-Free Algorithms

**Question:** where should we allocate shared data?

Diagram: Diagram showing a process of reading data, followed by a conditional check (if fail) and then a CAS operation.
NUMA Architectures + Lock-Free Algorithms

**Question**: where should we allocate shared data?

**Question**: does NUMA treat remote cores fairly?
NUMA Architectures + Lock-Free Algorithms

Question: where should we allocate shared data?

Question: does NUMA treat remote cores fairly?
Our Contributions
Our Contributions

1. *New tool* revealing NUMA’s effects on *lock-free* algorithms
Our Contributions

Severus

1. *New tool* revealing NUMA’s effects on *lock-free* algorithms
Our Contributions

1. *New tool* revealing NUMA’s effects on **lock-free** algorithms

2. *Case studies* of two machines:
   - AMD Opteron 6278 (Interlagos)
   - Intel Xeon E7-8867 v4 (Broadwell-EX)
Idea: look at *schedule* of memory accesses
Idea: look at schedule of memory accesses
Schedule Matters

ordering
Schedule Matters

ordering

1. read X
   if fail
   CAS X

2. read X
   if fail
   CAS X
Schedule Matters

ordering

read X

if fail

CAS X

R0

C0

R1

C1

if fail

CAS X

read X
Schedule Matters

ordering

Good schedule: \textbf{R0, C0, R1, C1}
Schedule Matters

Good schedule: \(R0, C0, R1, C1\)
Schedule Matters

ordering

Good schedule: R0, C0, R1, C1
Bad schedule: R0, R1, C0, C1
Schedule Matters

ordering

Good schedule: R0, C0, R1, C1
Bad schedule: R0, R1, C0, C1
Idea: look at *schedule* of memory accesses
**Idea:** look at schedule of memory accesses

**Problem:** schedule depends on complex hardware details
Idea: look at schedule of memory accesses

Problem: schedule depends on complex hardware details
  • cache coherence protocol
Idea: look at schedule of memory accesses

Problem: schedule depends on complex hardware details
- cache coherence protocol
- interconnect routing policy
New Tool: Severus
New Tool: **Severus**

**Goal #1**: reveal schedule of memory accesses
New Tool: **Severus**

**Goal #1:** *reveal schedule of memory accesses*

---

**F&I Experiments**
New Tool: Severus

Goal #1: reveal schedule of memory accesses

F&I Experiments

fetch-and-increment (xadd)
New Tool: **Severus**

**Goal #1**: reveal schedule of memory accesses

**Goal #2**: simulate lock-free algorithms

*F&I Experiments*

fetch-and-increment (xadd)
New Tool: **Severus**

**Goal #1:** *reveal schedule of memory accesses*

**Goal #2:** *simulate lock-free algorithms*

**F&I Experiments**

**Read-CAS Experiments**

- *parallel delay*
- *if fail*  
  - *atomic delay*  
  - *CAS*

- *fetch-and-increment (xadd)*
New Tool: **Severus**

**Goal #1:** reveal schedule of memory accesses

**Goal #2:** simulate lock-free algorithms

---

F&I Experiments

- **Today’s talk**: fetch-and-increment (xadd)

---

Read-CAS Experiments

- parallel delay
- CAS
- if fail
- atomic delay
- read

See paper
F&I Experiments

All cores F&I same target location
F&I Experiments

All cores F&I same **target** location

allocated on N0
F&I Experiments

All cores F&I same target location

N0

allocated on N0

N1

N2

N3
F&I Experiments

All cores F&I same target location

Schedule

thread ID  value
F&I Experiments

All cores F&I same target location

Schedule

<table>
<thead>
<tr>
<th>thread ID</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
F&I Experiments

All cores F&I same target location

Schedule

<table>
<thead>
<tr>
<th>thread ID</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
F&I Experiments

All cores F&I same **target** location

```
Schedule
thread ID   value
5          0→1
```
F&I Experiments

All cores F&I same target location

Schedule

<table>
<thead>
<tr>
<th>thread ID</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0→1</td>
</tr>
<tr>
<td>8</td>
<td>1→2</td>
</tr>
<tr>
<td>0</td>
<td>2→3</td>
</tr>
<tr>
<td>8</td>
<td>3→4</td>
</tr>
<tr>
<td>5</td>
<td>4→5</td>
</tr>
</tbody>
</table>
F&I Experiments

All cores F&I same **target** location

<table>
<thead>
<tr>
<th>Schedule</th>
<th>thread ID</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>0→1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1→2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2→3</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3→4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4→5</td>
</tr>
</tbody>
</table>
F&I Experiments

All cores F&I same target location

Local Logs

<table>
<thead>
<tr>
<th>0</th>
<th>2, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0, 4, ...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1, 3, ...</td>
</tr>
</tbody>
</table>

Schedule

<table>
<thead>
<tr>
<th>thread ID</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0→1</td>
</tr>
<tr>
<td>8</td>
<td>1→2</td>
</tr>
<tr>
<td>0</td>
<td>2→3</td>
</tr>
<tr>
<td>8</td>
<td>3→4</td>
</tr>
<tr>
<td>5</td>
<td>4→5</td>
</tr>
</tbody>
</table>
F&I Experiments

All cores F&I same target location

Local Logs

<table>
<thead>
<tr>
<th>0</th>
<th>2, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0, 4, ...</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1, 3, ...</td>
</tr>
</tbody>
</table>

Schedule

<table>
<thead>
<tr>
<th>thread ID</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0→1</td>
</tr>
<tr>
<td>8</td>
<td>1→2</td>
</tr>
<tr>
<td>0</td>
<td>2→3</td>
</tr>
<tr>
<td>8</td>
<td>3→4</td>
</tr>
<tr>
<td>5</td>
<td>4→5</td>
</tr>
</tbody>
</table>

offline reconstruction
AMD Interlagos
F&I Experiments
Interlagos Setup

AMD Opteron 6278
8 nodes
Interlagos Setup

AMD Opteron 6278
8 nodes
4 modules/node
2 cores/module
(shared L1)
Interlagos Setup

**Setup**: all cores F&I **target**, which is allocated on N0
**Setup**: all cores F&I target, which is allocated on N0

**Question**: which nodes’ cores do most F&I/sec?
**Setup**: all cores F&I **target**, which is allocated on N0

**Question**: which nodes’ cores do most F&I/sec?

A. distance 0 (N0)
B. distance 1 (N1, N2, N4, N6)
C. distance 2 (N3, N5, N7)
D. all equal
E. something else
**Setup:** all cores F&I **target**, which is allocated on N0

**Question:** which nodes’ cores do most F&I/sec?

A. distance 0 (N0)
B. distance 1 (N1, N2, N4, N6)
C. distance 2 (N3, N5, N7)
D. all equal
E. something else
**Interlagos F&I Throughput**

Setup: all nodes running, target on N0
Interlagos F&I Throughput

Setup: all nodes running, target on N0
Setup: all nodes running, target on N0
Setup: all nodes running, target on N0
Interlagos F&I Schedule

Setup: all nodes running, target on N0
Module visit: consecutive F&Is by cores in same module
Module visits: consecutive F&Is by cores in same module
Module Visits

Module visit: consecutive F&Is by cores in same module

time interval with target in L1
Module Visits

Module visit: consecutive F&Is by cores in same module
Visit length: number of F&Is in a visit
Module Visits

Module visit: consecutive F&Is by cores in same module

Visit length: number of F&Is in a visit

Visit distance: number of other visits between two visits to the same module
Interlagos F&I Visit Distances

Module on N7 (distance 2)

Setup: all nodes running, target on N0
Setup: all nodes running, target on N0
Interlagos F&I Visit Distances

**Module on N0 (distance 0)**

Setup: all nodes running, target on N0
**Interlagos F&I Visit Distances**

**Setup:** all nodes running, target on N0

- **Module on N0 (distance 0)**
  - Round-robin order!
  - ... with skips on N0

- **Module on N4 (distance 1)**

- **Module on N7 (distance 2)**
## Interlagos F&I Unfairness

<table>
<thead>
<tr>
<th>distance</th>
<th>skip probability</th>
<th>mean visit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Interlagos F&I Unfairness

<table>
<thead>
<tr>
<th>distance</th>
<th>skip probability</th>
<th>mean visit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\approx .40$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$\approx .03$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\approx .03$</td>
<td></td>
</tr>
</tbody>
</table>
## Interlagos F&I Unfairness

<table>
<thead>
<tr>
<th>distance</th>
<th>skip probability</th>
<th>mean visit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( \approx 0.40 )</td>
<td>( \approx 1.1 )</td>
</tr>
<tr>
<td>1</td>
<td>( \approx 0.03 )</td>
<td>( \approx 1.1 )</td>
</tr>
<tr>
<td>2</td>
<td>( \approx 0.03 )</td>
<td>( \approx 1.6 )</td>
</tr>
</tbody>
</table>
# Interlagos F&I Unfairness

<table>
<thead>
<tr>
<th>distance</th>
<th>skip probability</th>
<th>mean visit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\approx 0.40$</td>
<td>$\approx 1.1$</td>
</tr>
<tr>
<td>1</td>
<td>$\approx 0.03$</td>
<td>$\approx 1.1$</td>
</tr>
<tr>
<td>2</td>
<td>$\approx 0.03$</td>
<td>$\approx 1.6$</td>
</tr>
</tbody>
</table>

Think of skips as length-0 visits
### Interlagos F&I Unfairness

<table>
<thead>
<tr>
<th>distance</th>
<th>skip probability</th>
<th>mean visit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\approx .40$</td>
<td>$\approx 1.1$</td>
</tr>
<tr>
<td>1</td>
<td>$\approx .03$</td>
<td>$\approx 1.1$</td>
</tr>
<tr>
<td>2</td>
<td>$\approx .03$</td>
<td>$\approx 1.6$</td>
</tr>
</tbody>
</table>

Think of skips as length-0 visits

Larger distance $\Rightarrow$ longer module visit
Potential Explanation

Target’s cache coherence messages go to/from N0
Potential Explanation

Target’s cache coherence messages go to/from N0
Potential Explanation

Target’s cache coherence messages go to/from N0
... even though target always in some module’s L1!
Interlagos F&I One-Node Throughput

Setup: one node running at a time, target on N0
Interlagos F&I One-Node Throughput

Setup: one node running at a time, target on N0

Same effects occur with just one node
Intel Broadwell-EX

F&I Experiments

(preview)
Broadwell-EX F&I Schedule

Setup: all nodes running, target on N0
**Setup:** all nodes running, target on N0
Setup: all nodes running, target on N0

Major differences between machines
Question: how does NUMA affect memory access schedules?
Summary

Question: how does NUMA affect memory access schedules?

Contributions:
1. New tool, Severus
2. Case studies on two machines
Summary

Question: how does NUMA affect memory access schedules?

Contributions:
1. New tool, Severus
2. Case studies on two machines

Findings:
• NUMA can be unfair to local cores
• Schedule is decipherable!
Summary

Question: how does NUMA affect memory access schedules?

Contributions:
1. New tool, Severus
2. Case studies on two machines

Findings:
• NUMA can be unfair to local cores
• Schedule is decipherable!

https://github.com/cmuparlay/severus
AMD Interlagos
Read-CAS Experiments
Interlagos Read-CAS Throughput

Setup: all nodes running, target on N0
Interlagos Read-CAS Throughput

Setup: all nodes running, target on N0
Interlagos Read-CAS Throughput

Unfairness at all atomic delays

Setup: all nodes running, target on N0
Interlagos Read-CAS Many-Target Throughput

Setup: all nodes running, targets on each node
Interlagos Read-CAS Many-Target Throughput

Setup: all nodes running, targets on each node
Interlagos Read-CAS Many-Target Throughput

**Setup:** all nodes running, *targets on each node*

Remote beats local at all atomic delays