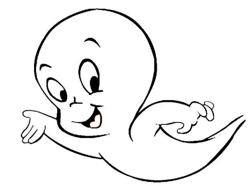


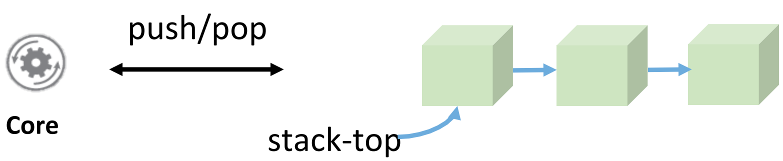
CASPAR: BREAKING SERIALIZATION IN LOCK-FREE MULTICORE SYNCHRONIZATION

Tanmay Gangwani[☥], Adam Morrison[*], Josep Torrellas[☥]

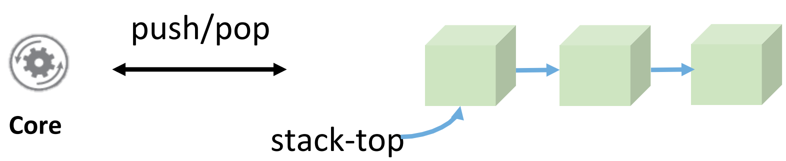
University of Illinois, Urbana-Champaign [☥]

Technion – Israel Institute of Technology[*]

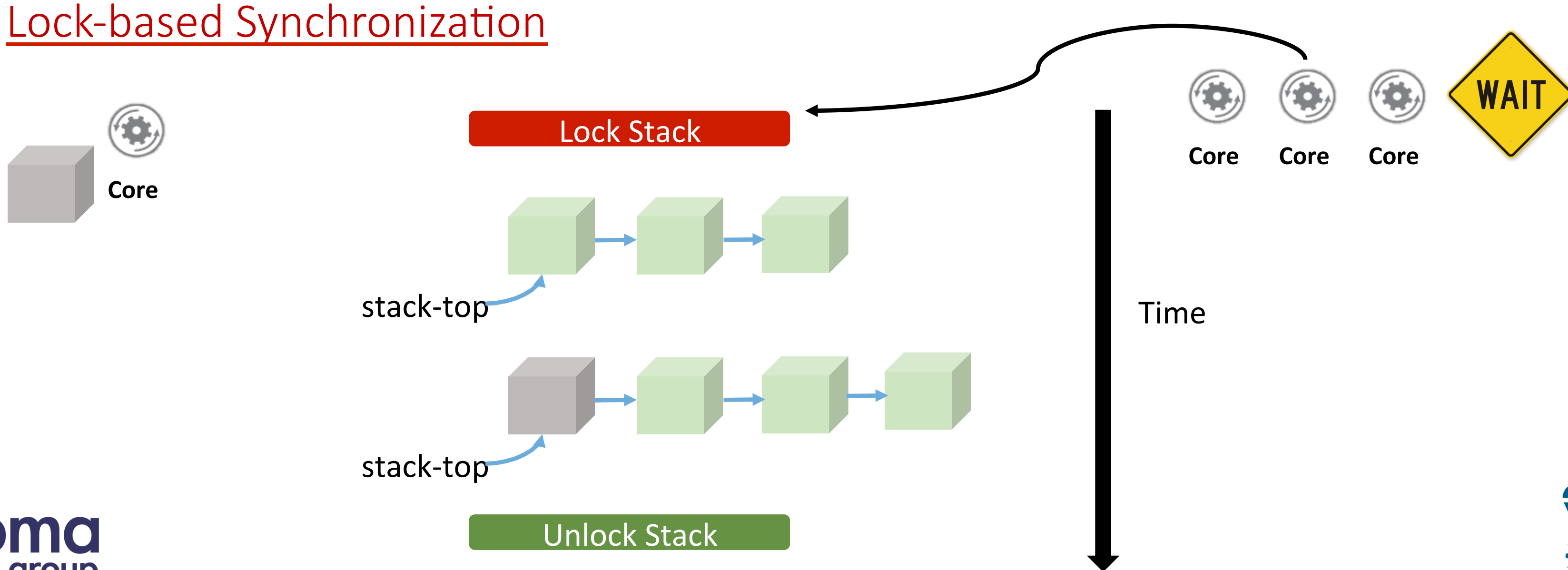
Concurrent Data Structures



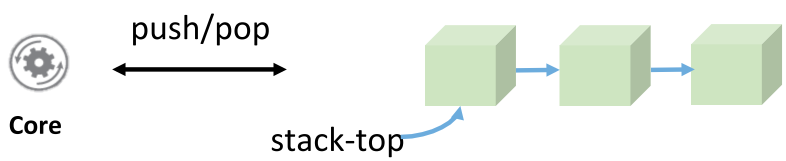
Concurrent Data Structures



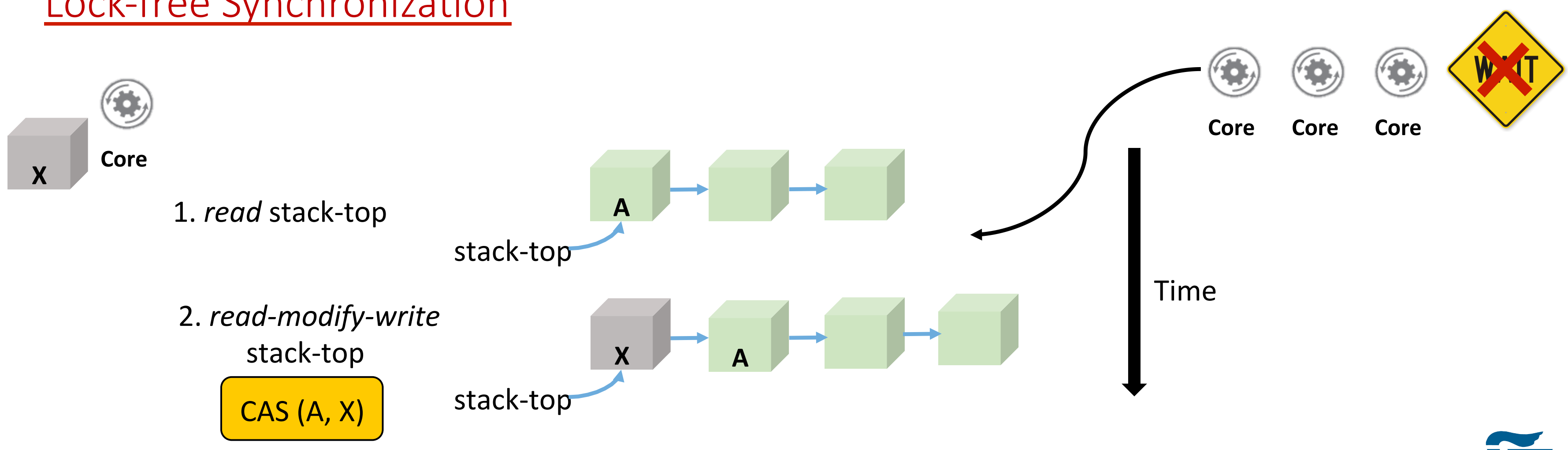
- Lock-based Synchronization



Concurrent Data Structures



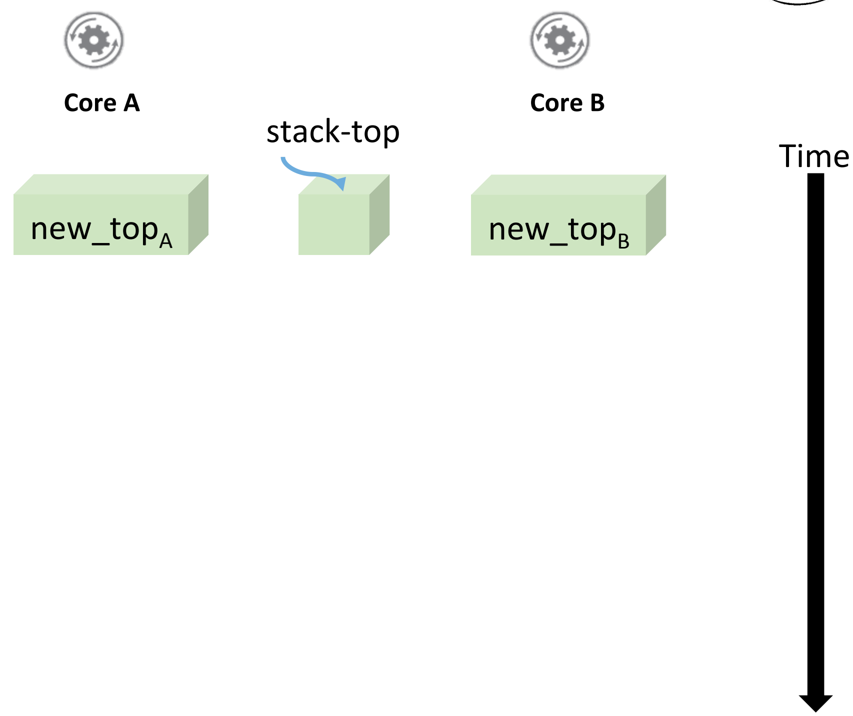
- Lock-free Synchronization



CAS Failures

<u>Adding node to shared stack</u>

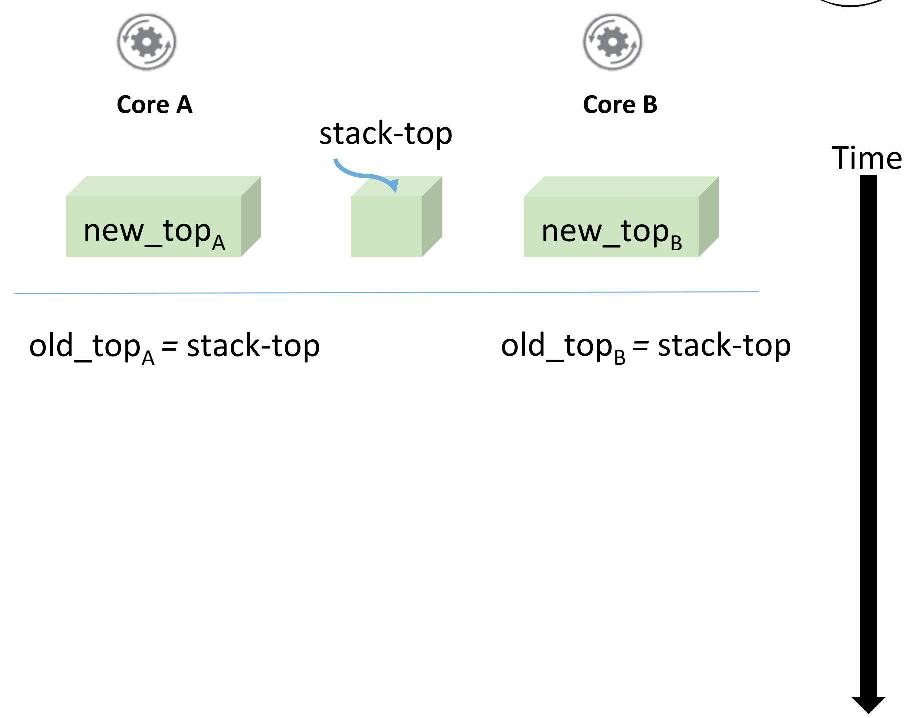
```
void push () {
  Node *new_top = malloc();
  while(true) {
    old_top = stack-top;
    new_top->next = old_top;
    if(CAS(&stack-top, old_top, new_top))
      return;
  }
}
```



CAS Failures

Adding node to shared stack

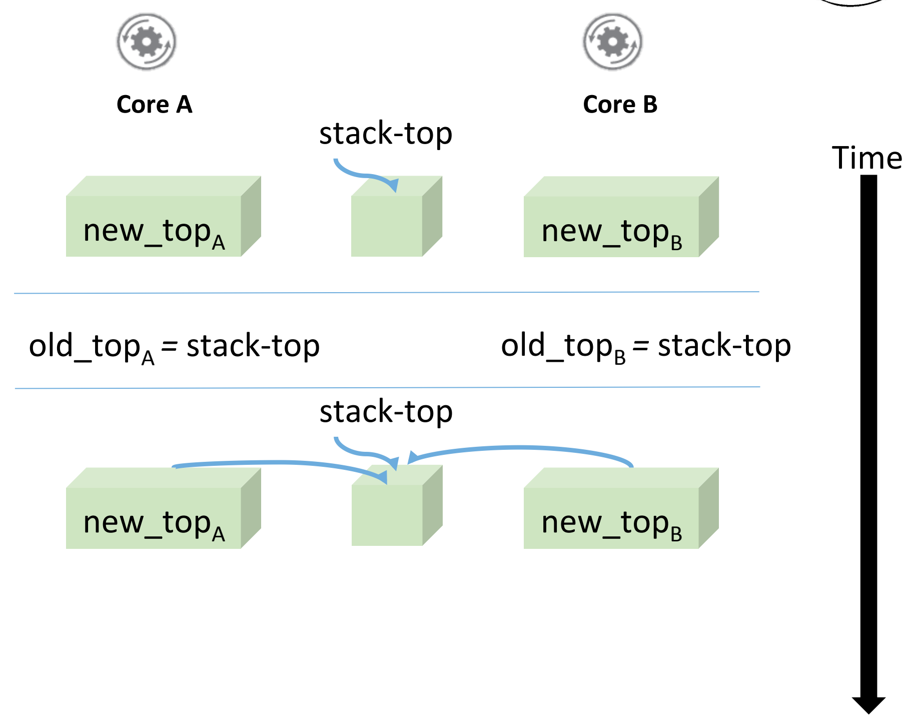
```
void push () {
  Node *new_top = malloc();
  while(true) {
    old_top = stack-top;
    new_top->next = old_top;
    if(CAS(&stack-top, old_top, new_top))
      return;
  }
}
```



CAS Failures

Adding node to shared stack

```
void push () {
  Node *new_top = malloc();
  while(true) {
    old_top = stack-top;
    new_top->next = old_top;
    if(CAS(&stack-top, old_top, new_top))
      return;
  }
}
```

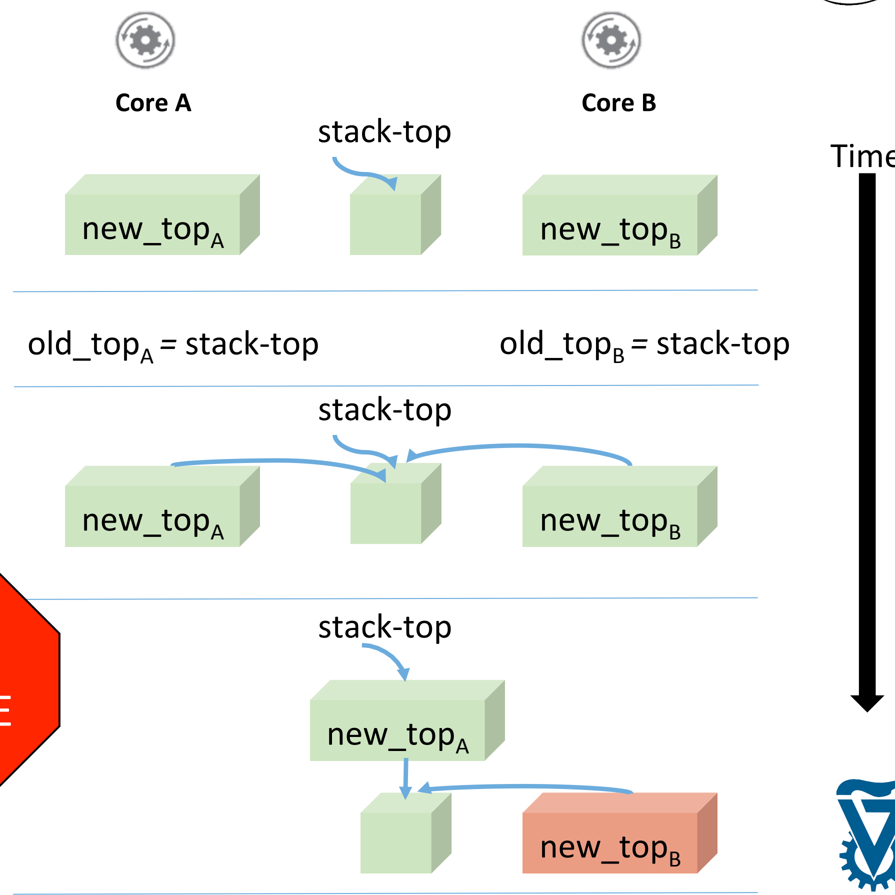


CAS Failures

Adding node to shared stack

```
void push () {
  Node *new_top = malloc();
  while(true) {
    old_top = stack-top;
    new_top->next = old_top;
    if(CAS(&stack-top, old_top, new_top))
      return;
  }
}
```

CAS FAILURE



Load-to-CAS Atomicity

Adding node to shared stack

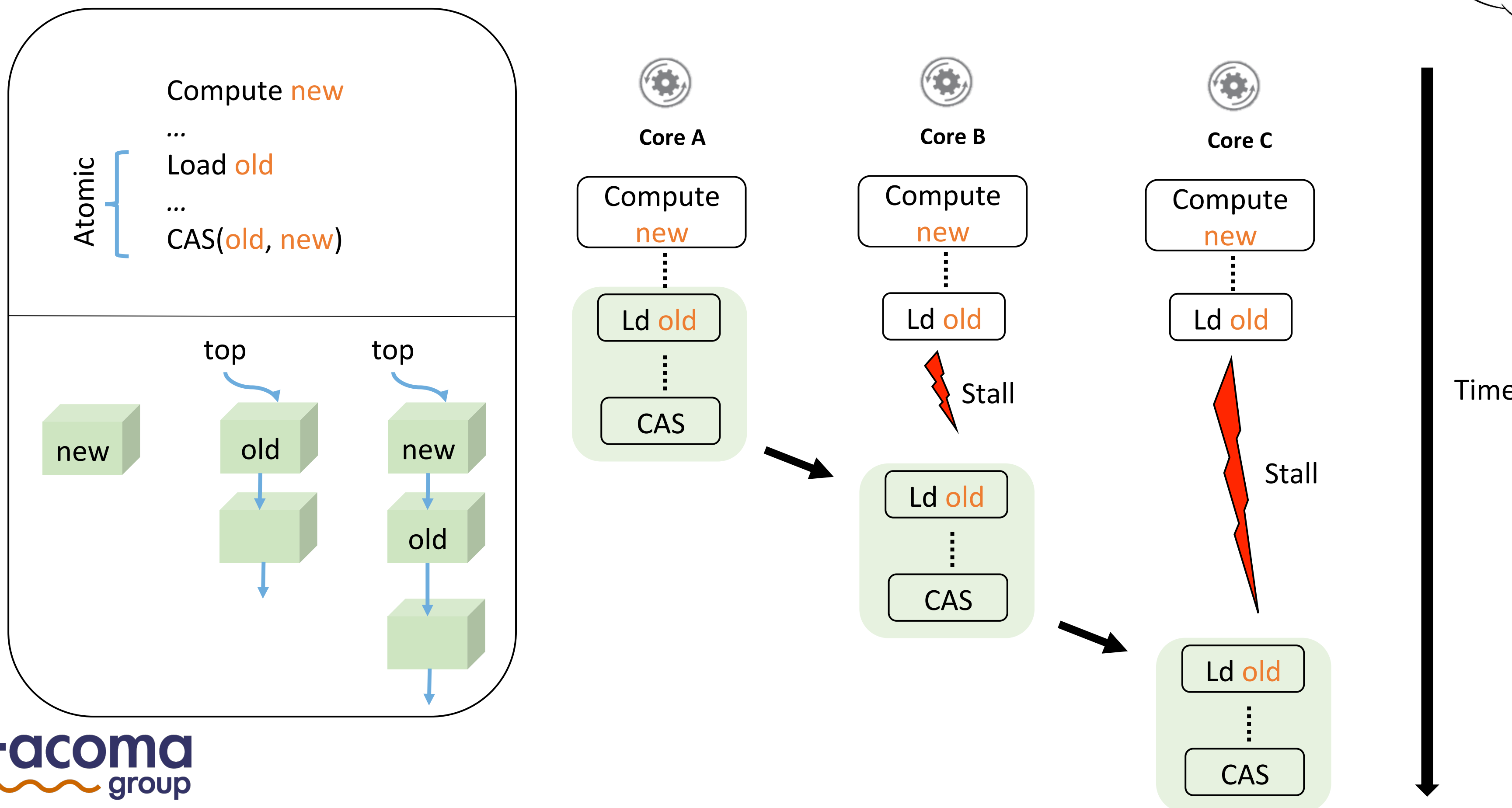
ATOMIC

```
void push () {
  Node *new_top = malloc();
  while(true) {

    old_top = stack-top;
    new_top->next = old_top;
    if(CAS(&stack-top, old_top,
new_top))

      return;
  }
}
```

Cache Line Lock

NO CAS FAILURES

Scourge of Serialization

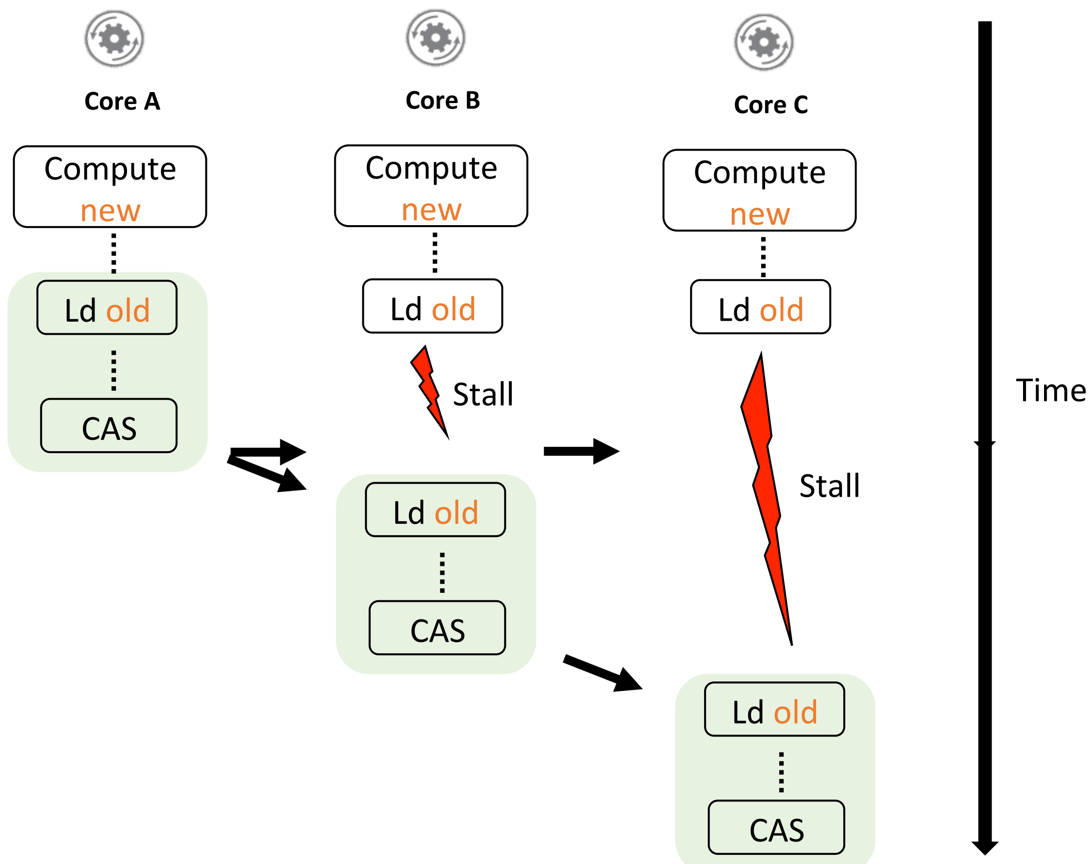


i-acoma group

Contribution

CASPAR reduces stalls in lock-free programs using two new techniques :

✓ Eager Forwarding
✓ Group Validation



CASPAR Fights Serialization

Baseline
(Prior work)

Contributions

Hardware Queueing

Directory chains cores in hardware

+

Contribution I : Eager Forwarding

Directory facilitates passing of speculative values between cores

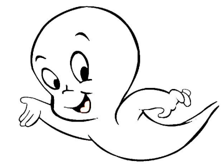
+

Contribution II : Parallel Validation

Directory facilitates group-validation of multiple cores

- ❑ Uses cache line bit for locking
- ❑ Dynamic detection of contended CAS – No source code modifications
- ❑ Deadlock-free

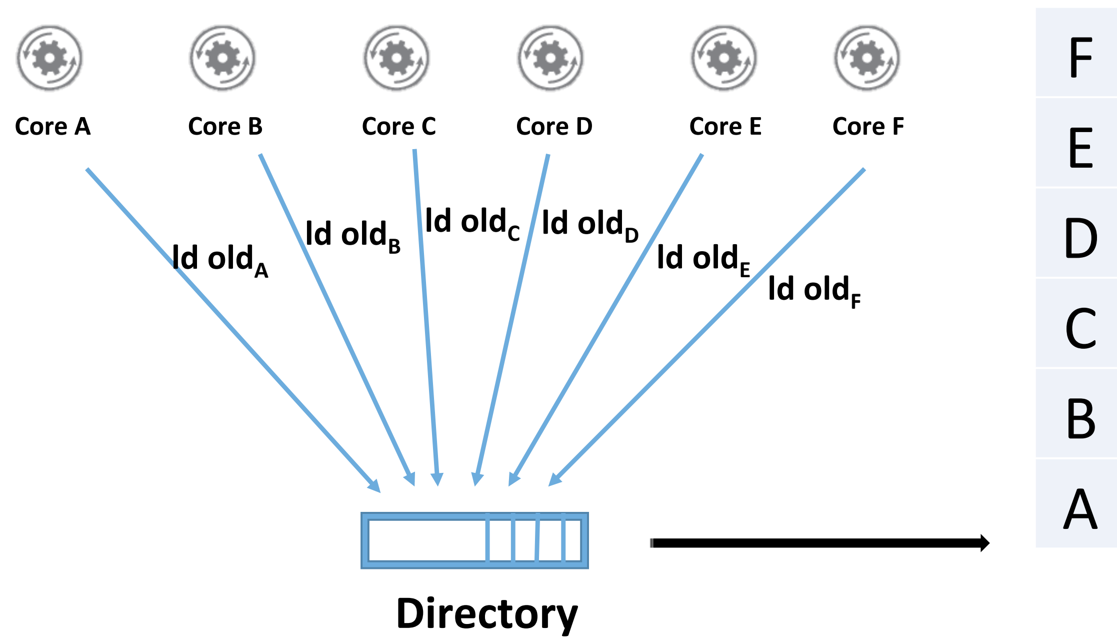
Serialization in Hardware Queueing



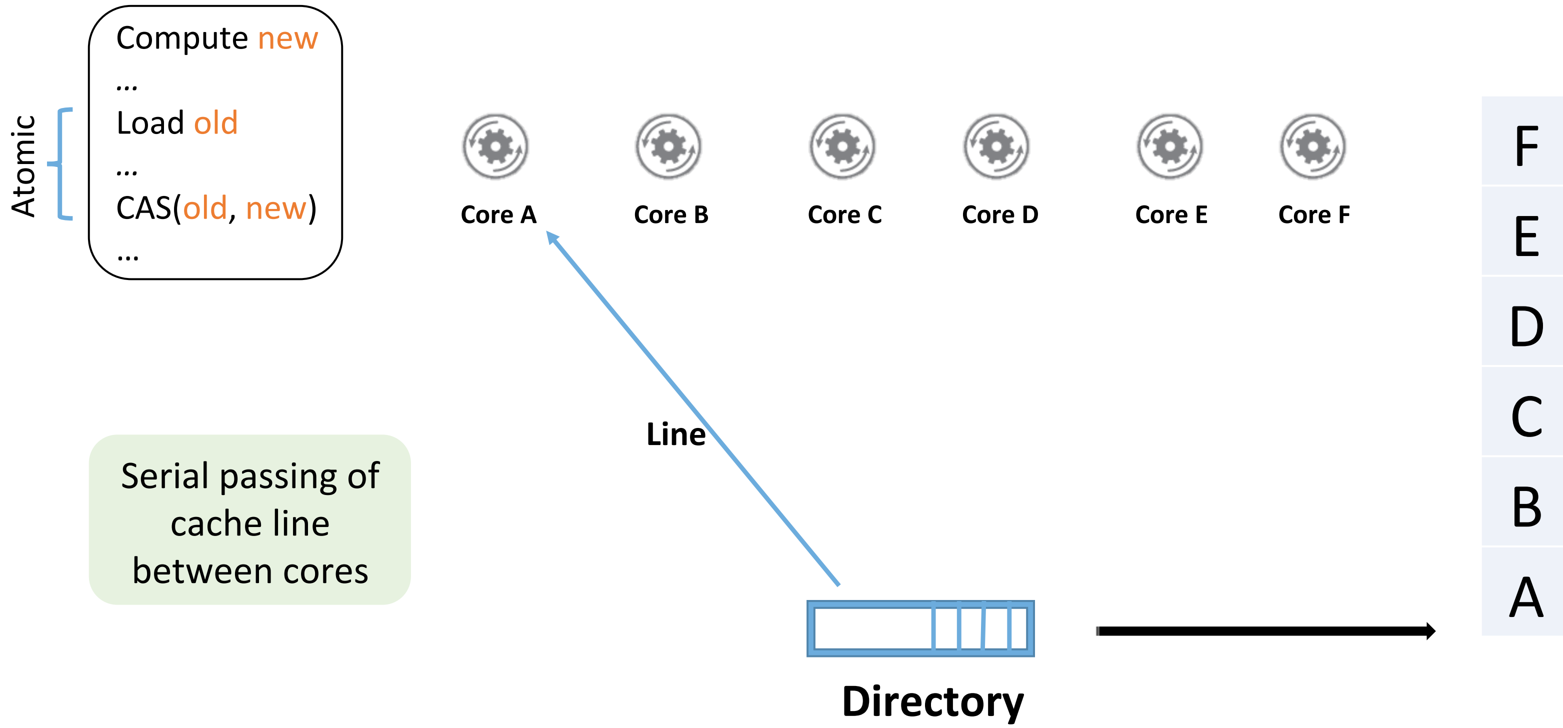
Compute new
…
Load old
…
CAS(old, new)
…

Atomic

Serial passing of cache line between cores



Serialization in Hardware Queueing

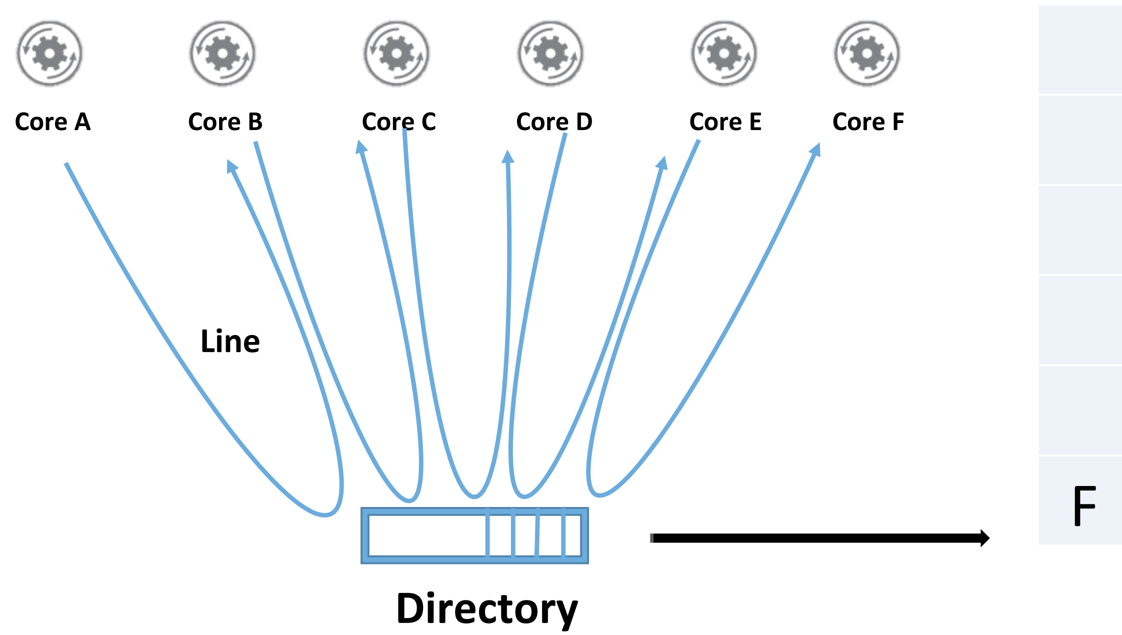


Serial passing of cache line between cores

Serialization in Hardware Queueing

Compute new
…
Load old
…
CAS(old, new)
…

Atomic



Serial passing of cache line between cores

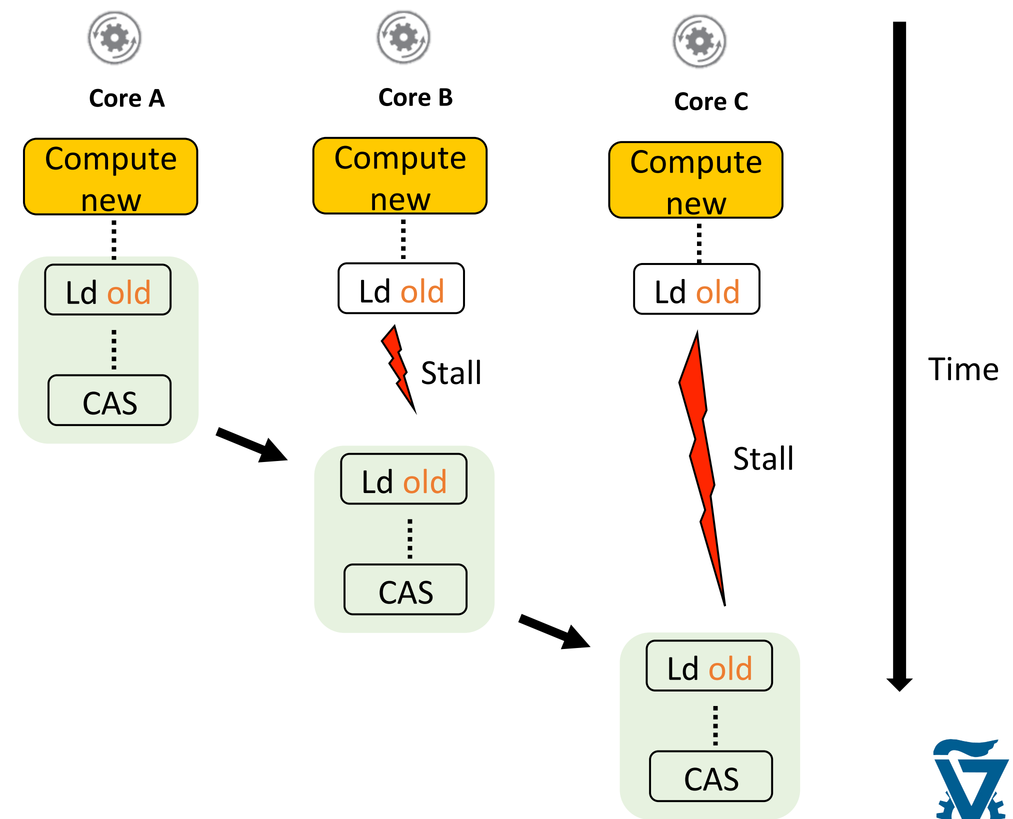
Contribution I : Eager Forwarding - Insight

Adding node to shared stack

```
void push () {
  Node *new_top = malloc();
  while(true) {

    old_top = stack;
    new_top->next = old_top;
    if(CAS(&stack, old_top,
new_top))

      return;
  }
}
```

Atomic

new_top generated early on, independent of old_top



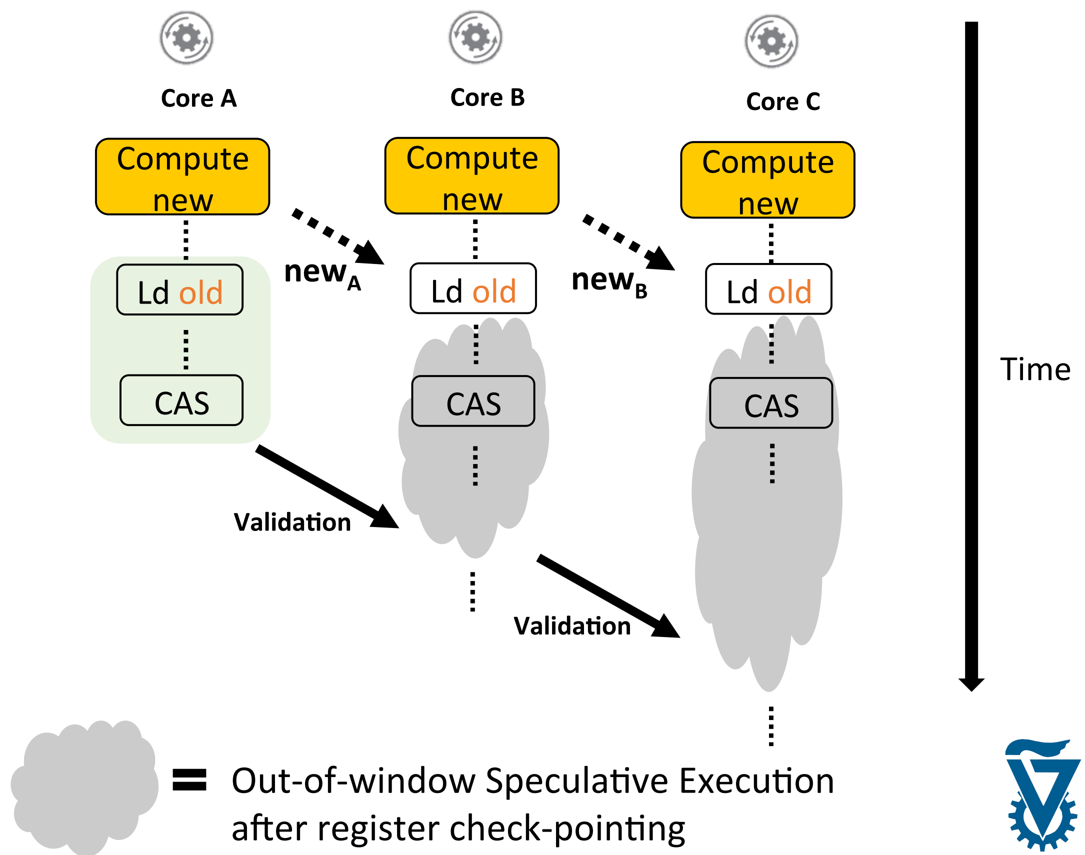
CASPAR – Eager Forwarding

Adding node to shared stack

```
void push () {
  Node *new_top = malloc();
  while(true) {

    old_top = stack;
    new_top->next = old_top;
    if(CAS(&stack, old_top,
new_top))

       return;
  }
}
```

Atomic

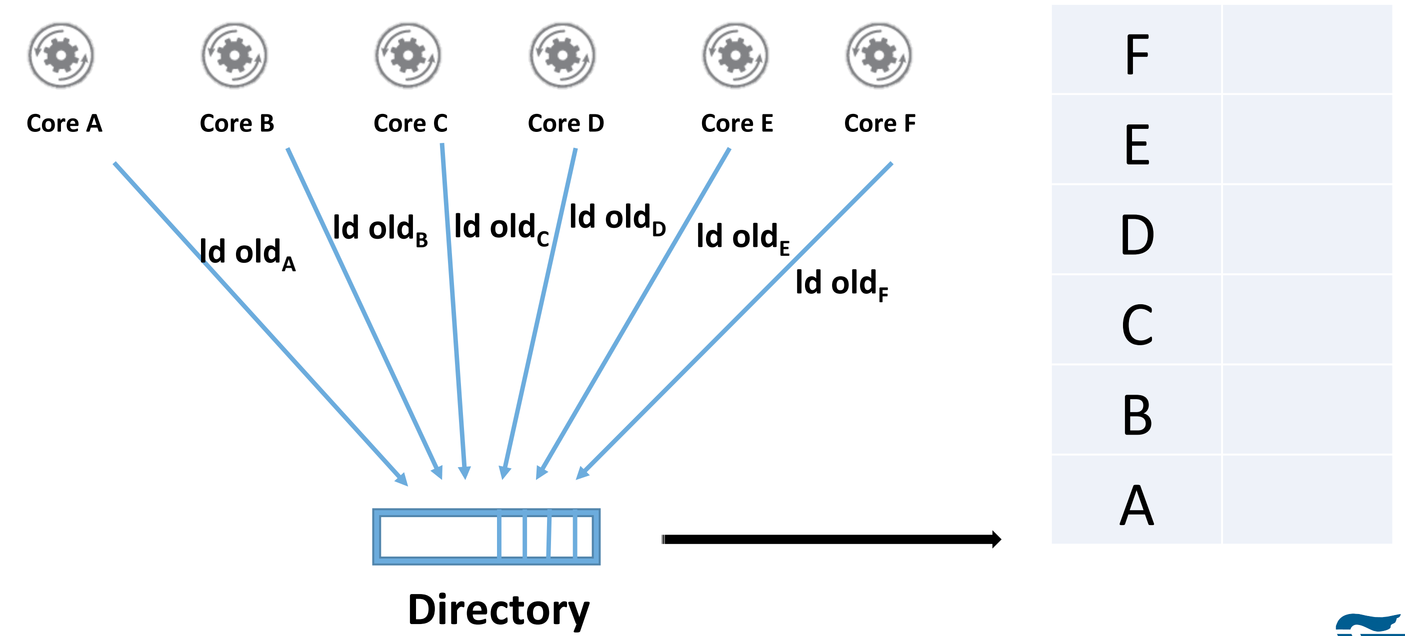
new_top generated early on, independent of old_top



CASPAR – Eager Forwarding

Atomic

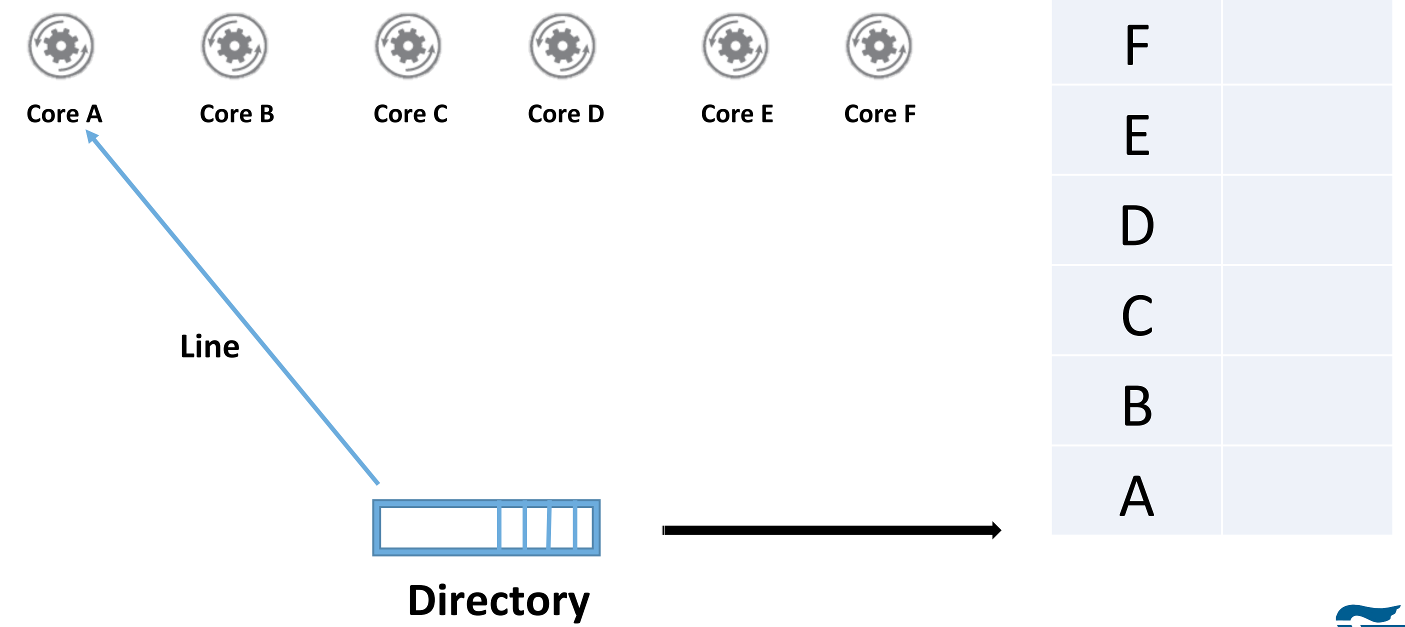
```
Compute new
…
Load old
…
CAS(old, new)
…
```



CASPAR – Eager Forwarding

Atomic

```
Compute new
…
Load old
…
CAS(old, new)
…
```

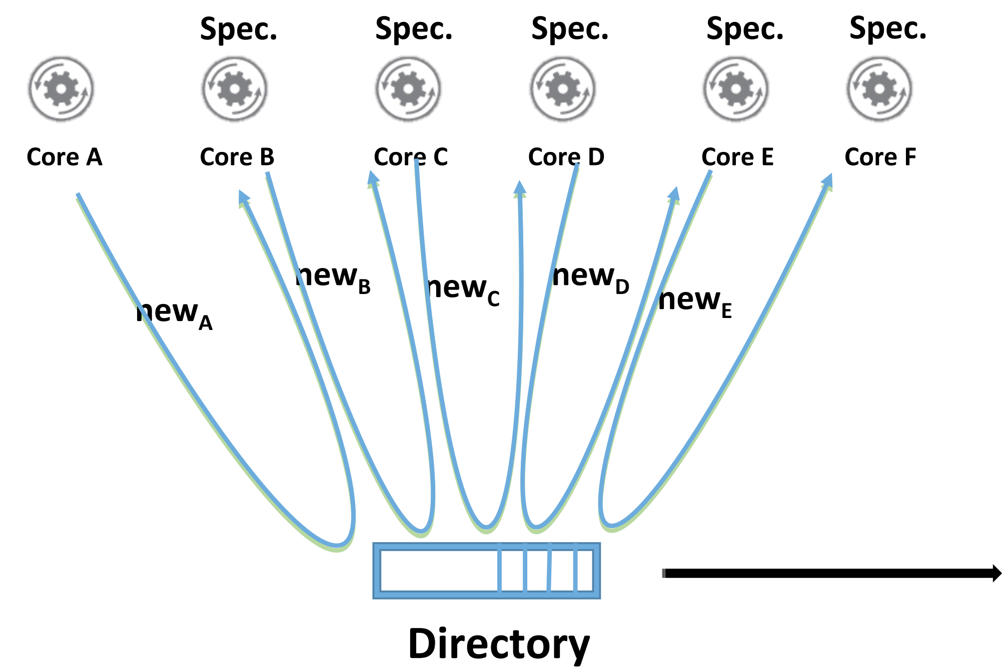


CASPAR – Eager Forwarding

Atomic

Compute new
…
Load old
…
CAS(old, new)
…

Parallel passing of speculative values between cores



F	new_F
E	new_E
D	new_D
C	new_C
B	new_B
A	new_A

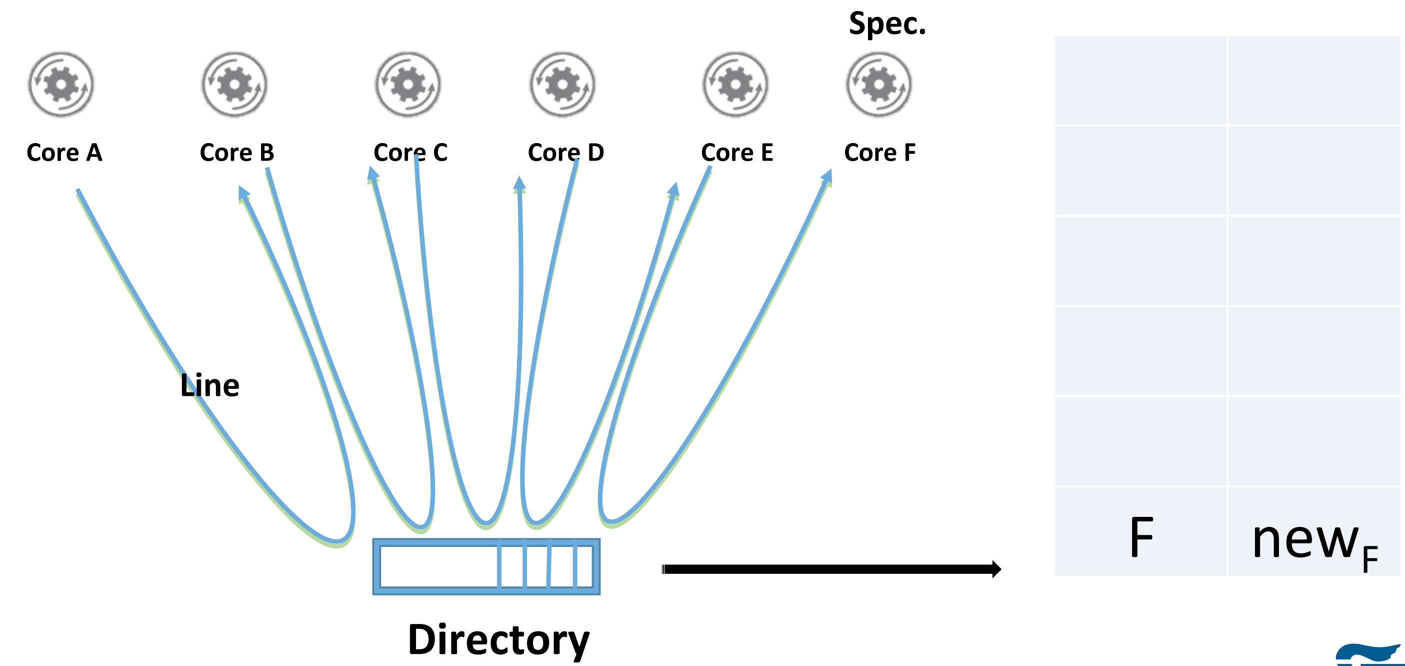
i-acoma group

CASPAR – Eager Forwarding

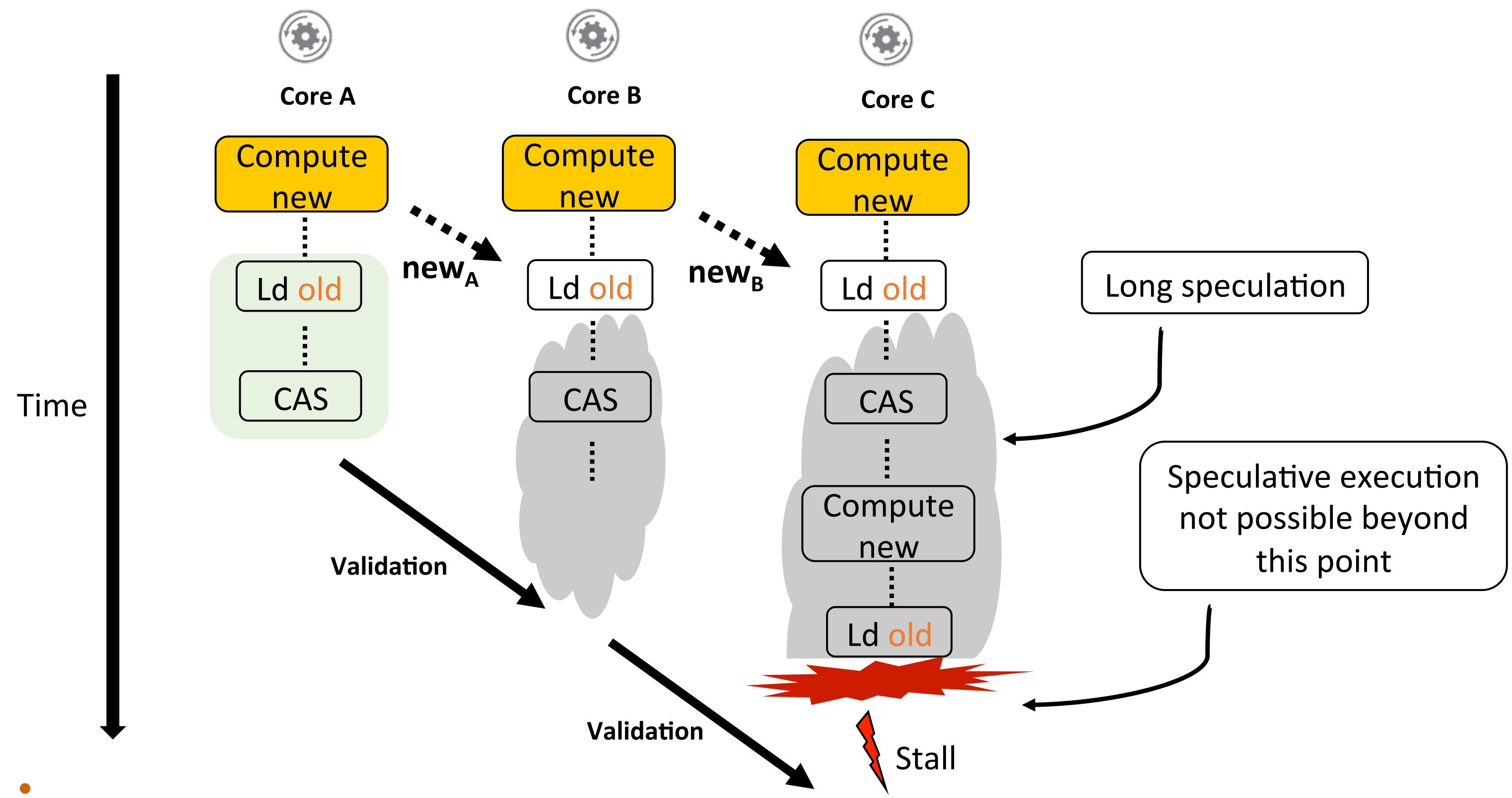
Compute new
…
Load old
…
CAS(old, new)
…

Atomic

Serial Validation

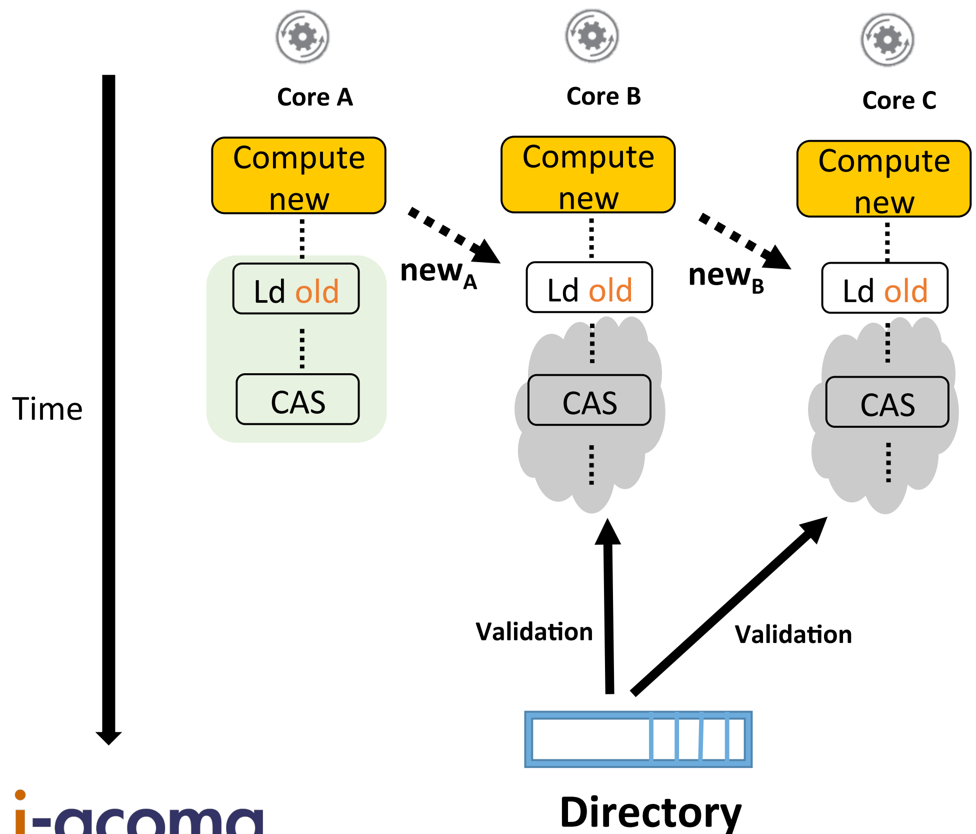


Limitations of Eager Forwarding



i-acoma group

Contribution II : Parallel Validation



TWO-PHASE COMMIT (2PC) PROTOCOL TO PERFORM PARALLEL VALIDATION

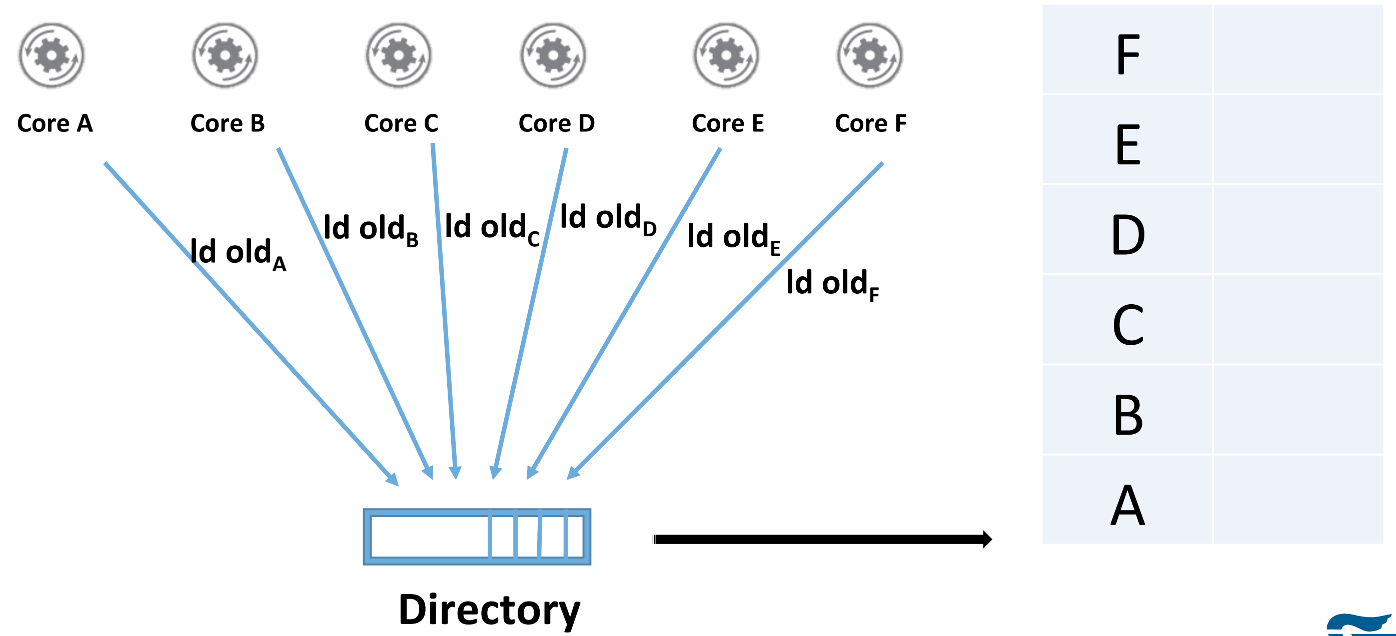
Idea: Validate in the directory without ever sending the full line to the core

CASPAR – Parallel Validation

```
Compute new
…
Load old
…
CAS(old, new)
…
```

Atomic

Parallel passing of speculative values between cores

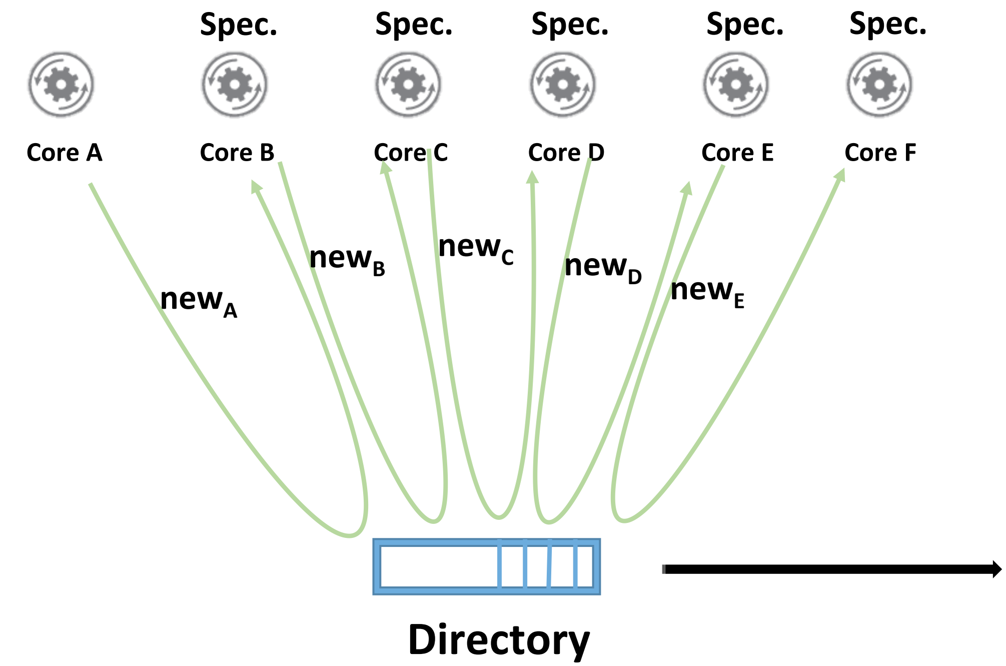


CASPAR – Parallel Validation

Atomic

Compute new
…
Load old
…
CAS(old, new)
…

Parallel passing of speculative values between cores



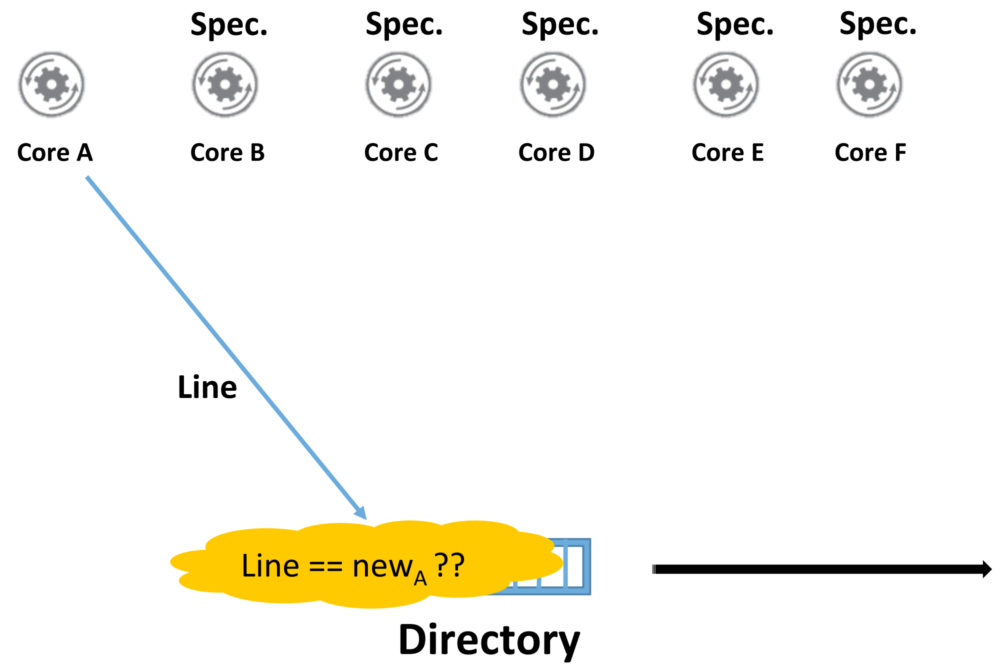
F	new_F
E	new_E
D	new_D
C	new_C
B	new_B
A	new_A

i-acoma group

CASPAR – Parallel Validation

Compute new
…
Atomic: Load old
…
CAS(old, new)
…

Parallel passing of speculative values between cores



F	new_F
E	new_E
D	new_D
C	new_C
B	new_B
A	new_A

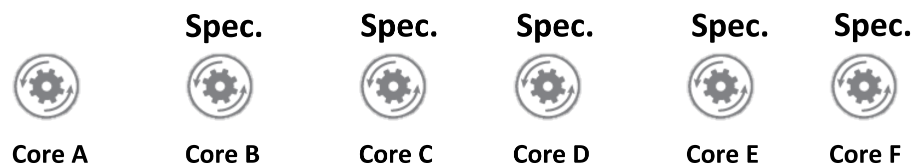
CASPAR – Parallel Validation

Atomic

```
Compute new
…
Load old
…
CAS(old, new)
…
```

Parallel passing of speculative values between cores

Parallel Validation



F	new_F
E	new_E
D	new_D
C	new_C
B	new_B

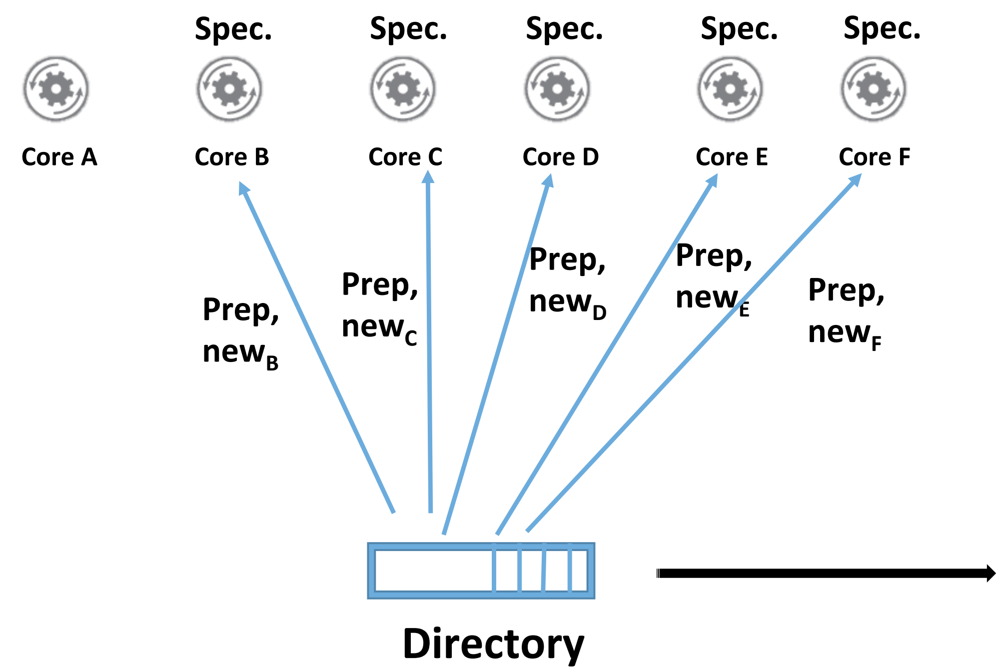
CASPAR – Parallel Validation

Compute new
…
Load old
…
CAS(old, new)
…

Atomic

Parallel passing of speculative values between cores

Parallel Validation



F	new_F
E	new_E
D	new_D
C	new_C
B	new_B

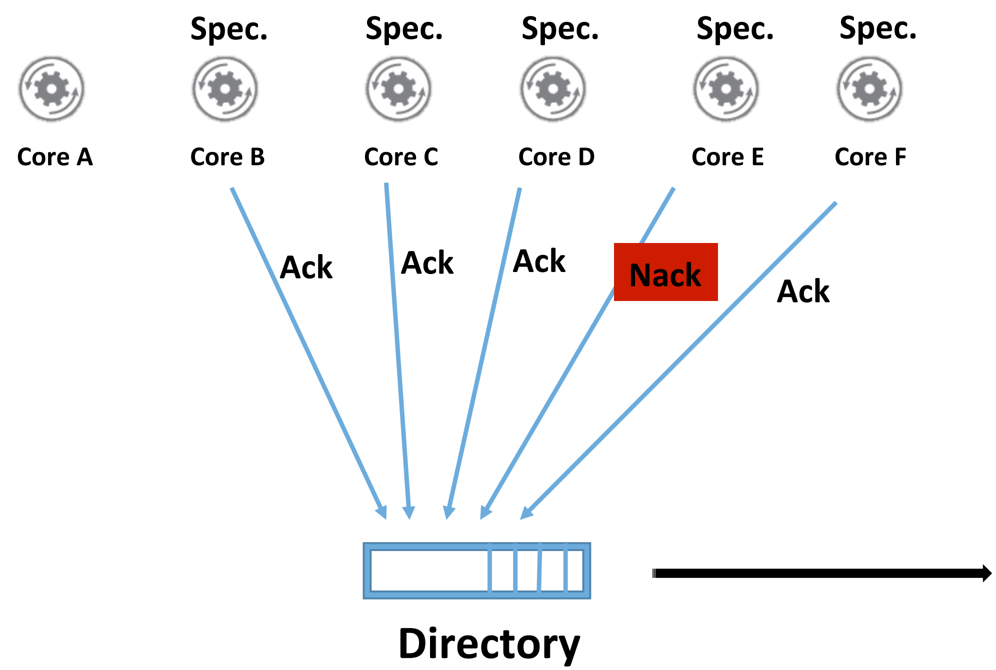
CASPAR – Parallel Validation

```
Compute new
…
Load old
…
CAS(old, new)
…
```

Atomic

Parallel passing of speculative values between cores

Parallel Validation



F	new_F
E	new_E
D	new_D
C	new_C
B	new_B

i-acoma group

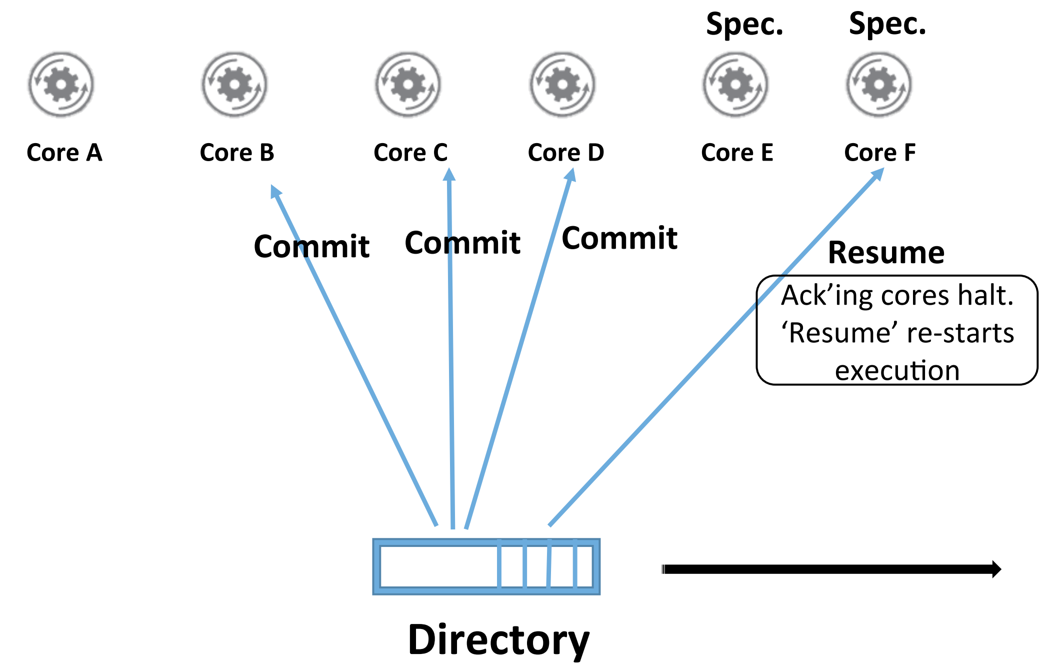
CASPAR – Parallel Validation

Compute new
…
Load old
…
CAS(old, new)
…

Atomic

Parallel passing of speculative values between cores

Parallel Validation



F	new_F
E	new_E

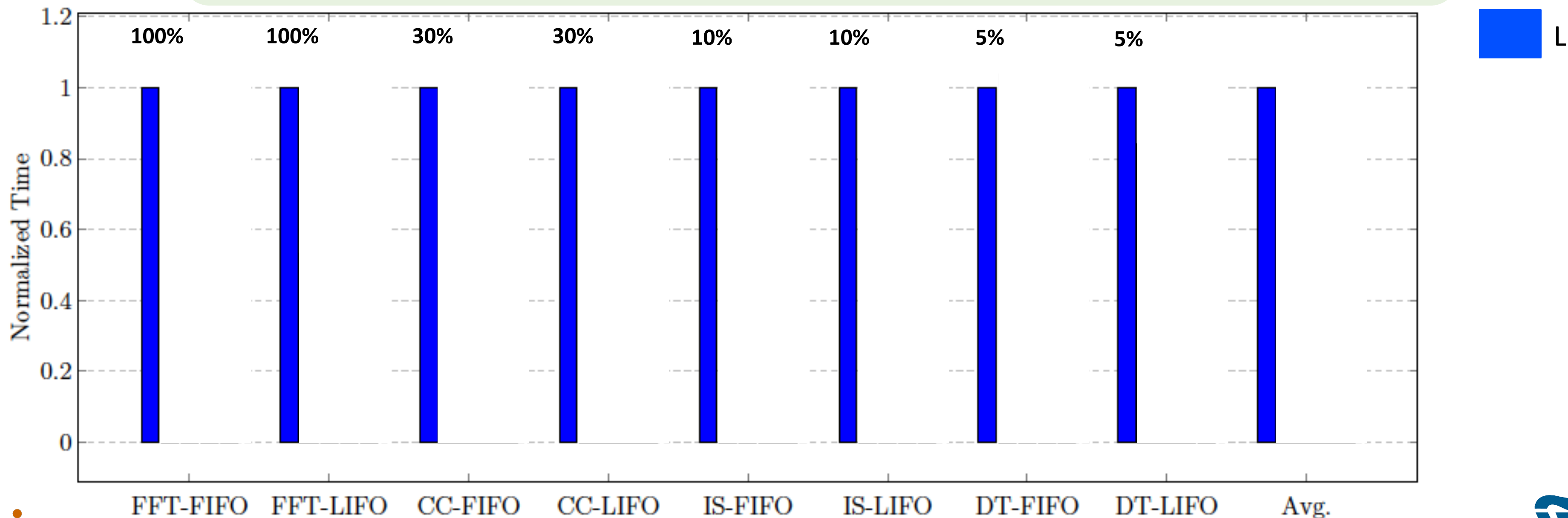
i-acoma group

Outcome

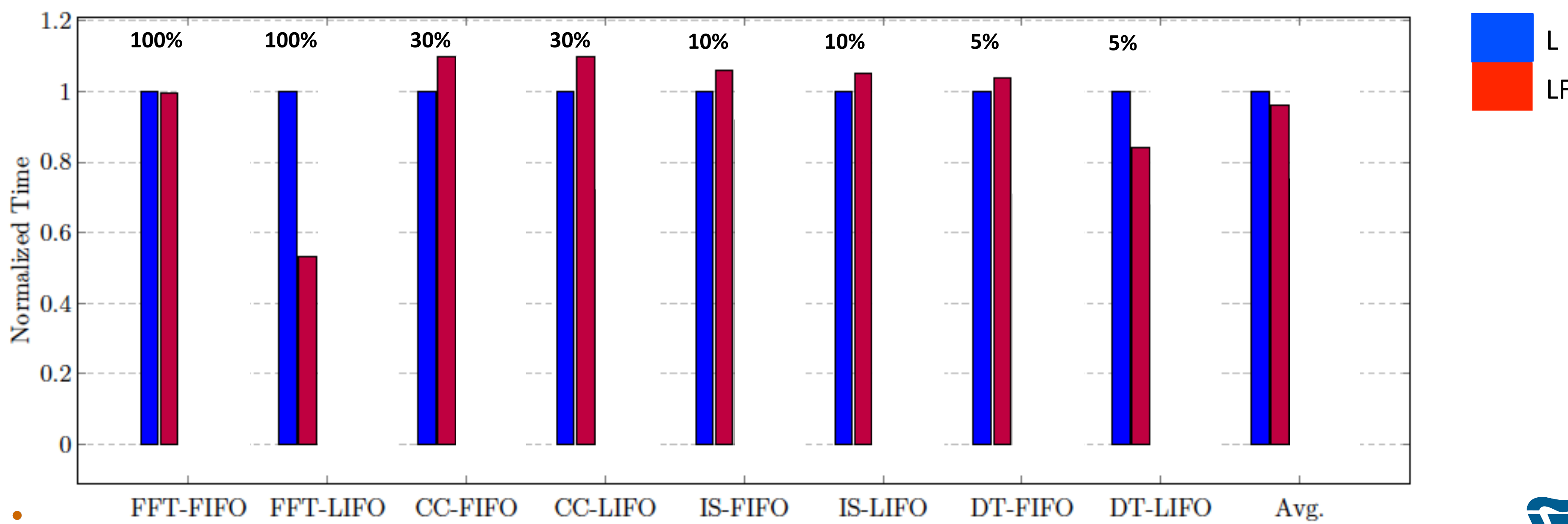
✓**Fully Parallel Data Transfer**
✓**Fully Parallel De-queue**

Evaluation

- Sniper simulator, 64 cores, OOO
- **Kernels** : FIFO, LIFO, LIFO-push-only, Mandelbrot, Larson
- **Applications** : Galois graph analytics suite (3x2), Barcelona OpenMP Tasks Suite (1x2)

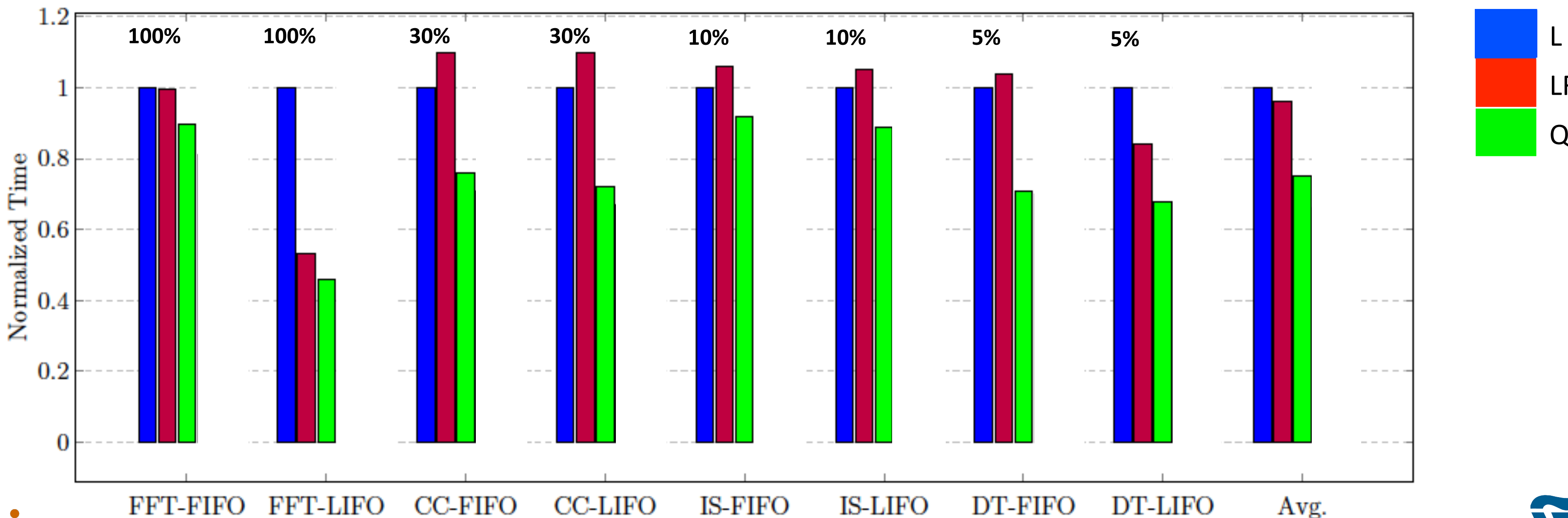


Evaluation



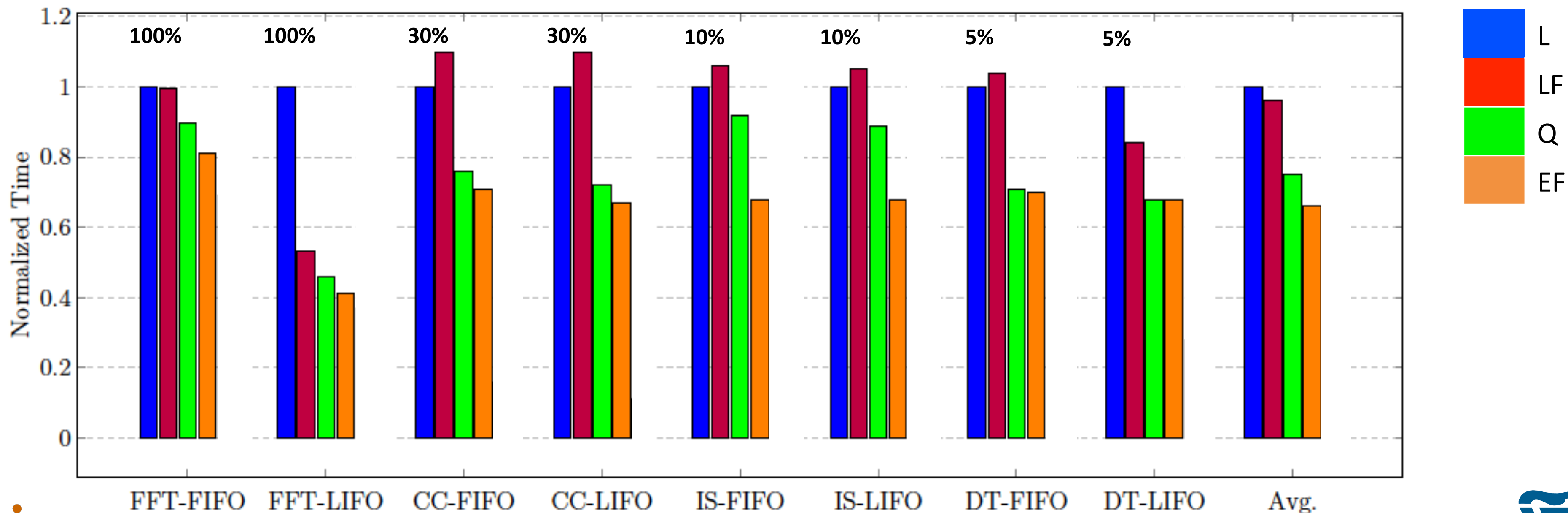
Normalized Time
1.2
1
0.8
0.6
0.4
0.2
0
100%
100%
30%
30%
10%
10%
5%
5%
FFT-FIFO
FFT-LIFO
CC-FIFO
CC-LIFO
IS-FIFO
IS-LIFO
DT-FIFO
DT-LIFO
Avg.
L
LF

Evaluation



100%
100%
30%
30%
10%
10%
5%
5%
1.2
1
0.8
0.6
0.4
0.2
0
Normalized Time
FFT-FIFO
FFT-LIFO
CC-FIFO
CC-LIFO
IS-FIFO
IS-LIFO
DT-FIFO
DT-LIFO
Avg.
L
LF
Q

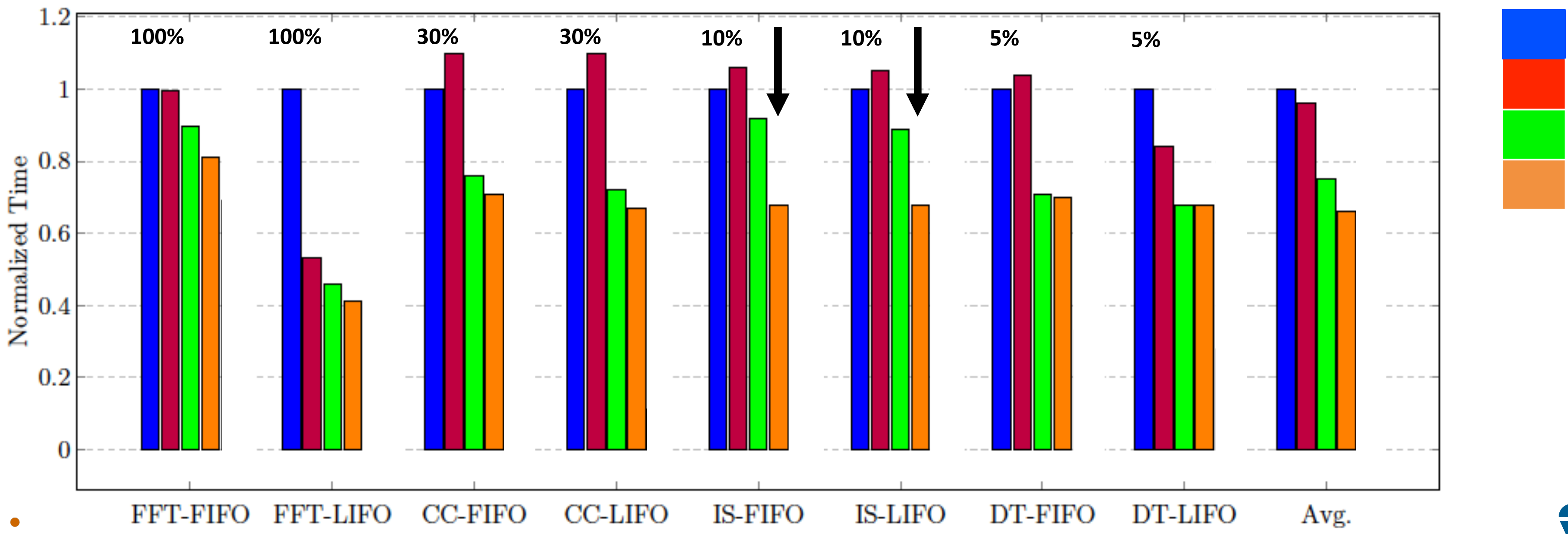
Evaluation



1.2
1
0.8
0.6
0.4
0.2
0
Normalized Time
100%
100%
30%
30%
10%
10%
5%
5%
FFT-FIFO
FFT-LIFO
CC-FIFO
CC-LIFO
IS-FIFO
IS-LIFO
DT-FIFO
DT-LIFO
Avg.
L
LF
Q
EF

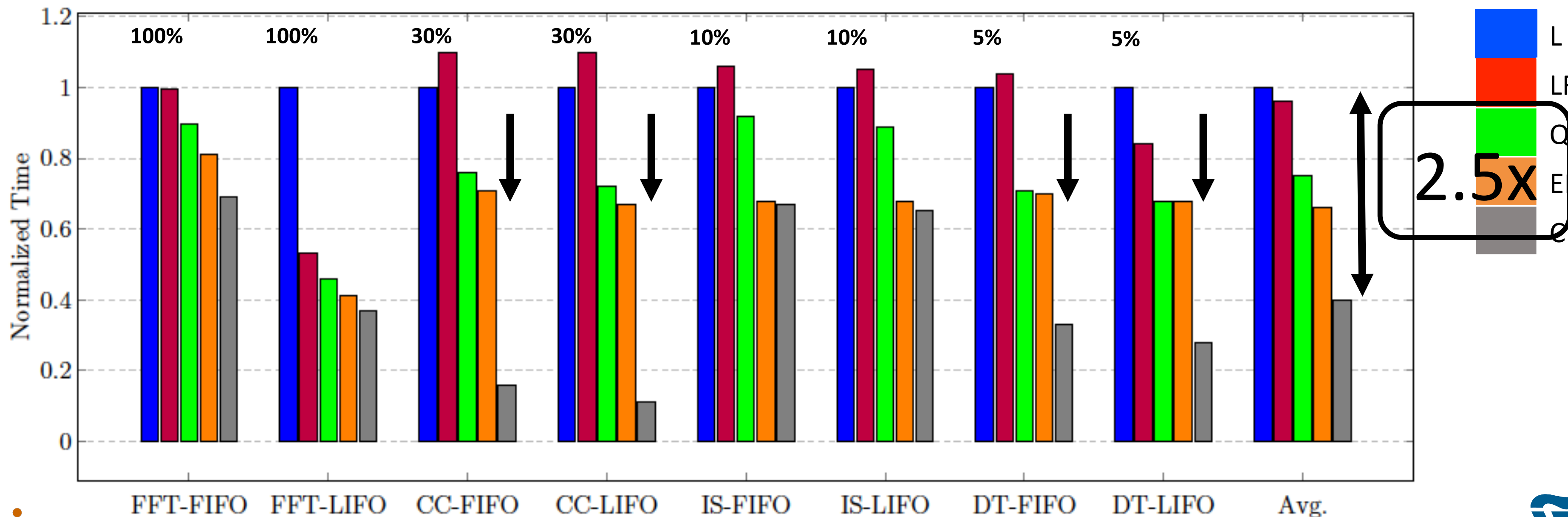
Evaluation

Independent Set (IS) gains from Eager Forwarding since early data transfer exposes work that could be done speculatively and in parallel



Evaluation

Connected Components (CC) and Delaunay Triangulation (DT) gain from Parallel Validation as it removes stalls in EF using group commit



Also in the paper

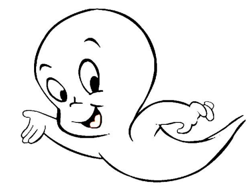
- ✓ Information on proposed hardware changes
- ✓ Handling memory consistency issues
- ✓ Detailed working and correctness of Eager Forwarding and 2PC protocols
- ✓ Scalability plots, cycle-level breakdown for applications
- ✓ Supporting other primitives (e.g. LL/SC)

Conclusions

- CASPAR enables true parallelism in codes using lock-free synchronization
  - Eager Forwarding : Parallel transfer of data
  - Parallel Validation : Parallel de-queue

- No source code modifications

- **Future Work**
  - Can be used to break conflict serialization in TM designs



CASPAR: BREAKING SERIALIZATION IN LOCK-FREE MULTICORE SYNCHRONIZATION

Tanmay Gangwani[†], Adam Morrison[*], Josep Torrellas[†]

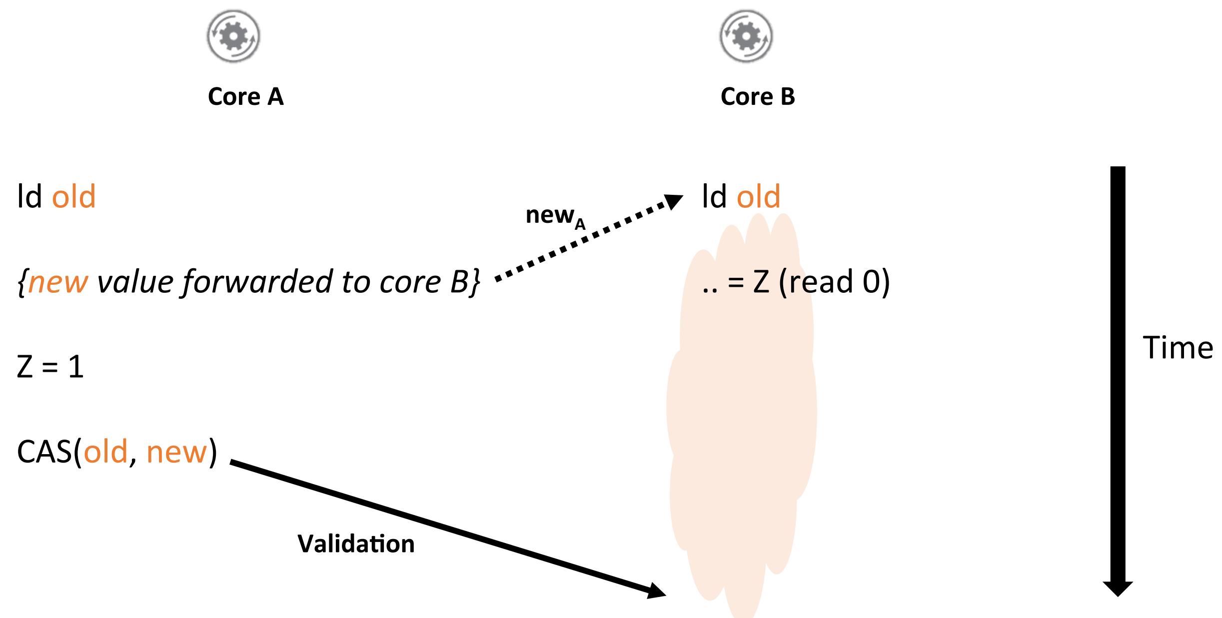
University of Illinois, Urbana-Champaign[†]

Technion – Israel Institute of Technology[*]

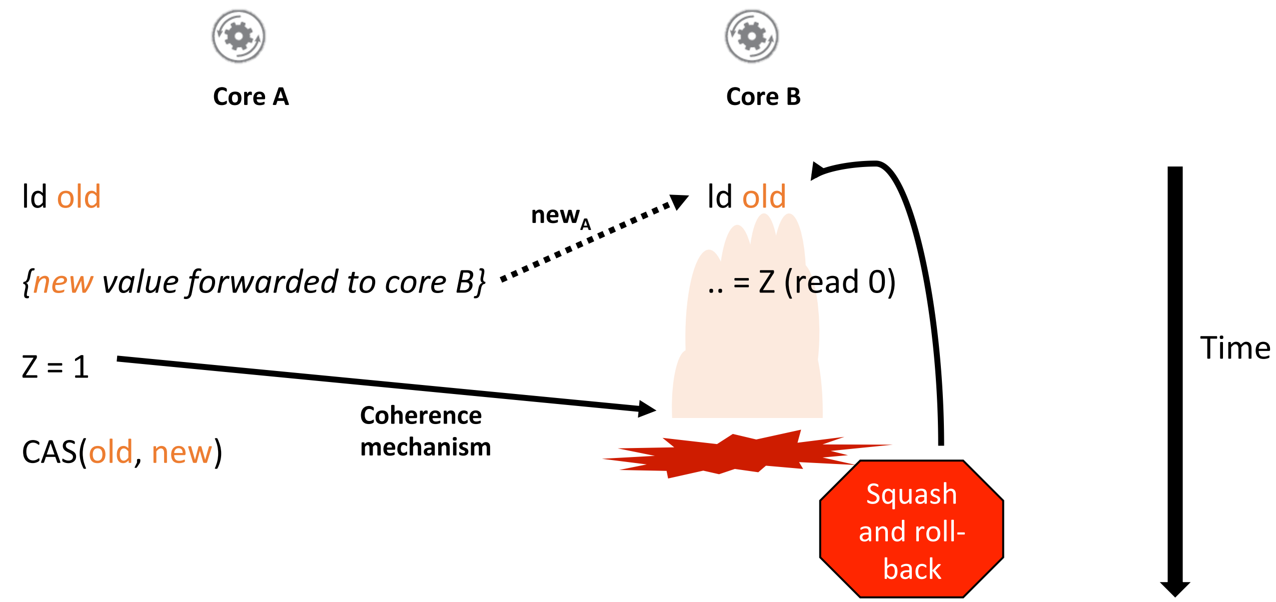
CASPAR

BACKUP

Maintaining Consistency

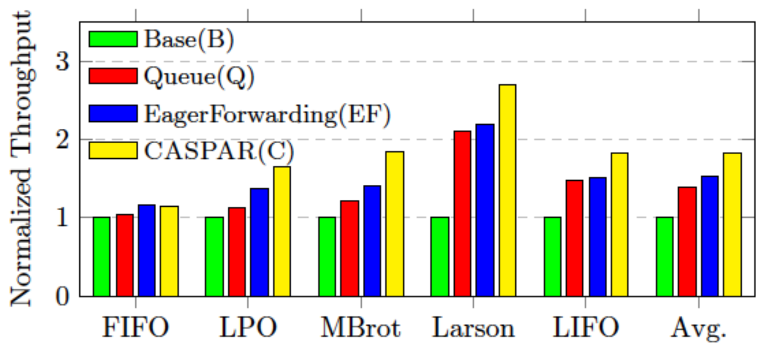


Maintaining Consistency



Forwarding causes no consistency violations

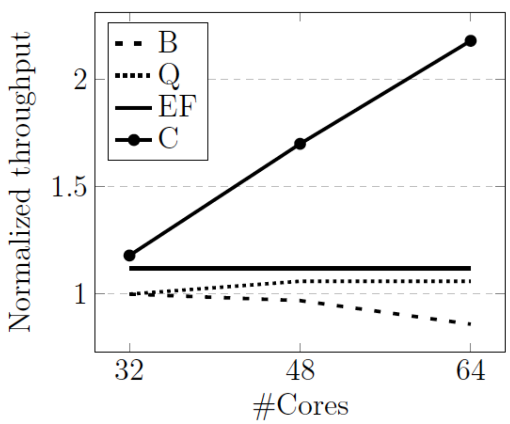
Evaluation (kernels)



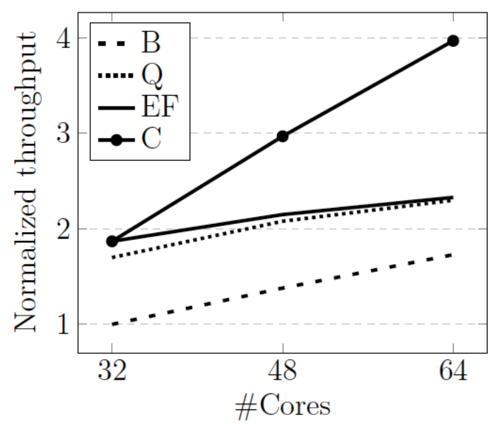
<u>CAS throughput improvements</u>

- EF over Base : 53%
- C over Base : 83%
- EF over Queue : 10%
- C over Queue : 32%

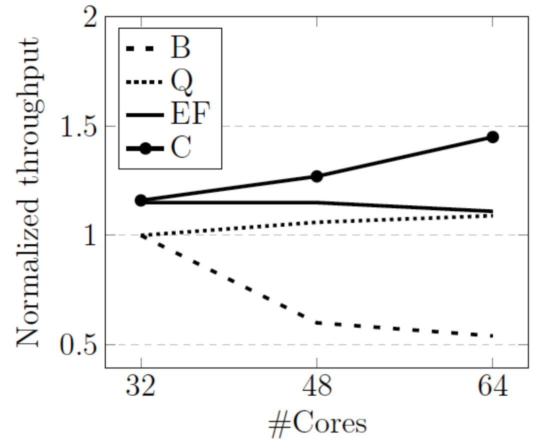
Evaluation (scalability)



LIFO-push-only



Mandelbrot



LIFO

CASPAR greatly improves scalability with number of cores

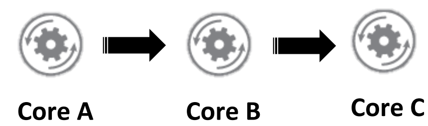
i-acoma group

CASPAR in Entirety

Effective Serialization

Module I : Hardware Queueing

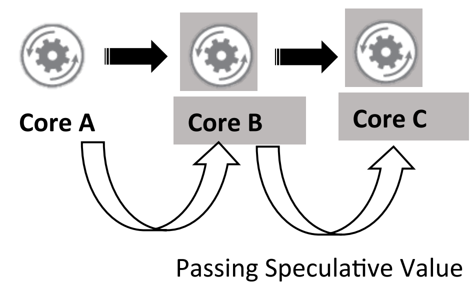
Directory chains cores in hardware



Breaking Serialization

Module II : Eager Forwarding

Directory facilitates passing of speculative values between cores



Module III : Two-Phase Group Commit

Directory facilitates collective-validation of multiple cores