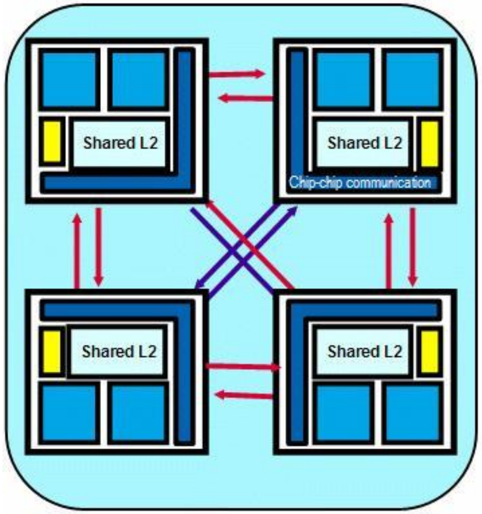
NUMA-Friendly Stack
(using Delegation and Elimination)

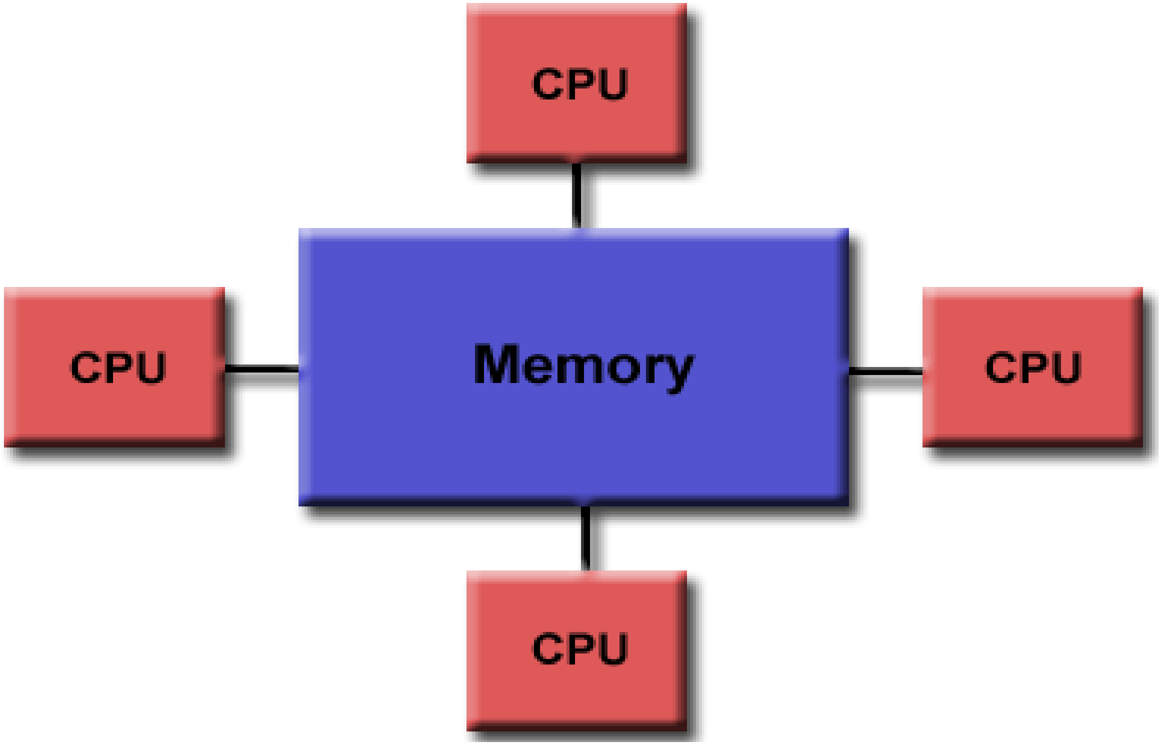
Irina Calciu
Justin Gottschlich
Maurice Herlihy

HotPar '13

Trends for Future Architectures



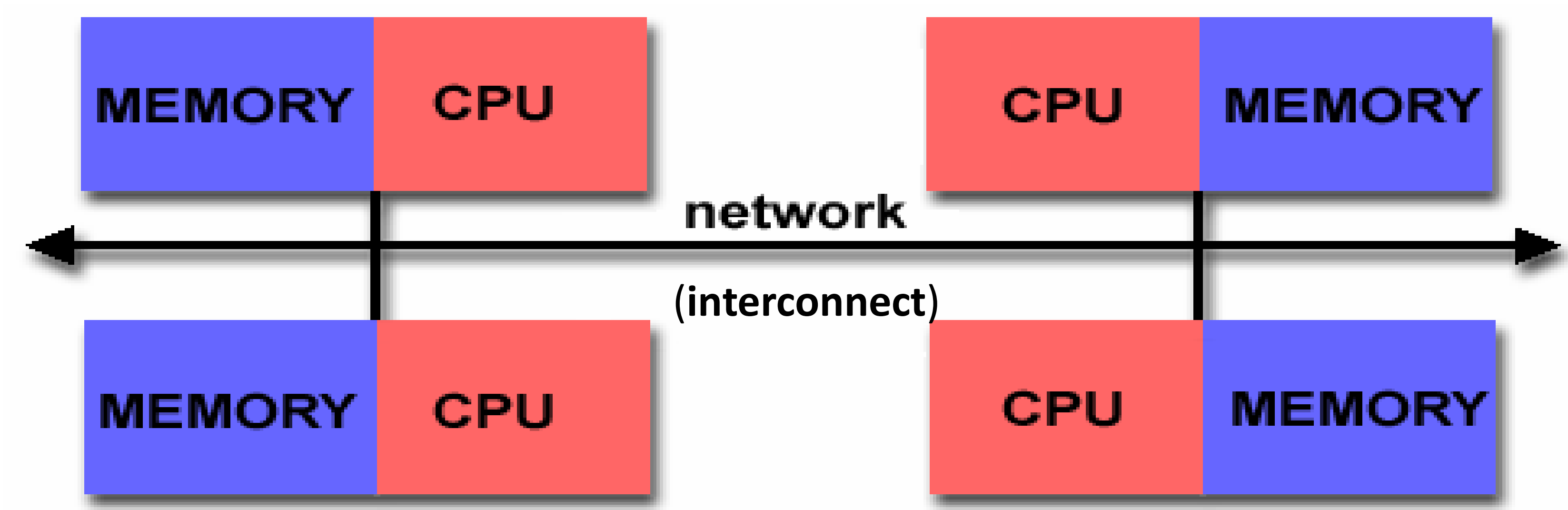
Uniform Memory Access (UMA)



Non-Uniform Memory Access (NUMA)

NUMA NODE (multiple cores, shared Last Level Cache)

NUMA NODE (multiple cores, shared Last Level Cache)



NUMA NODE (multiple cores, shared Last Level Cache)

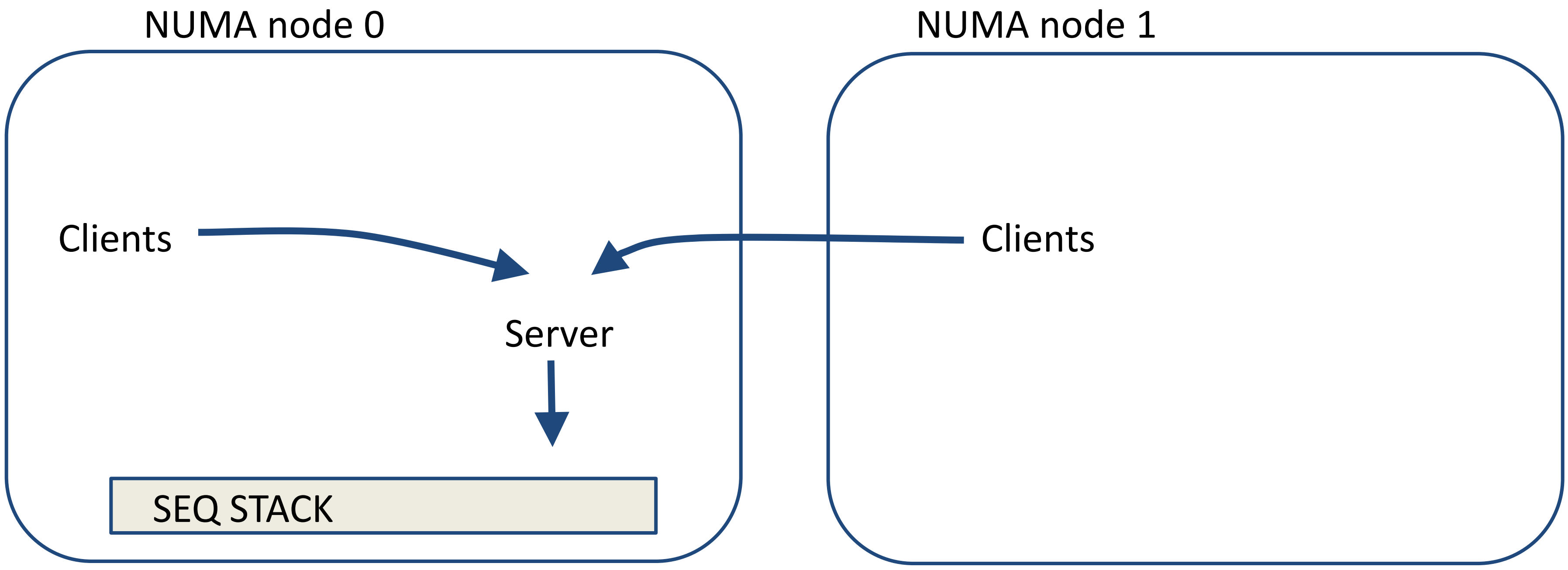
NUMA NODE (multiple cores, shared Last Level Cache)

Cache coherency maintained between caches on different NUMA nodes

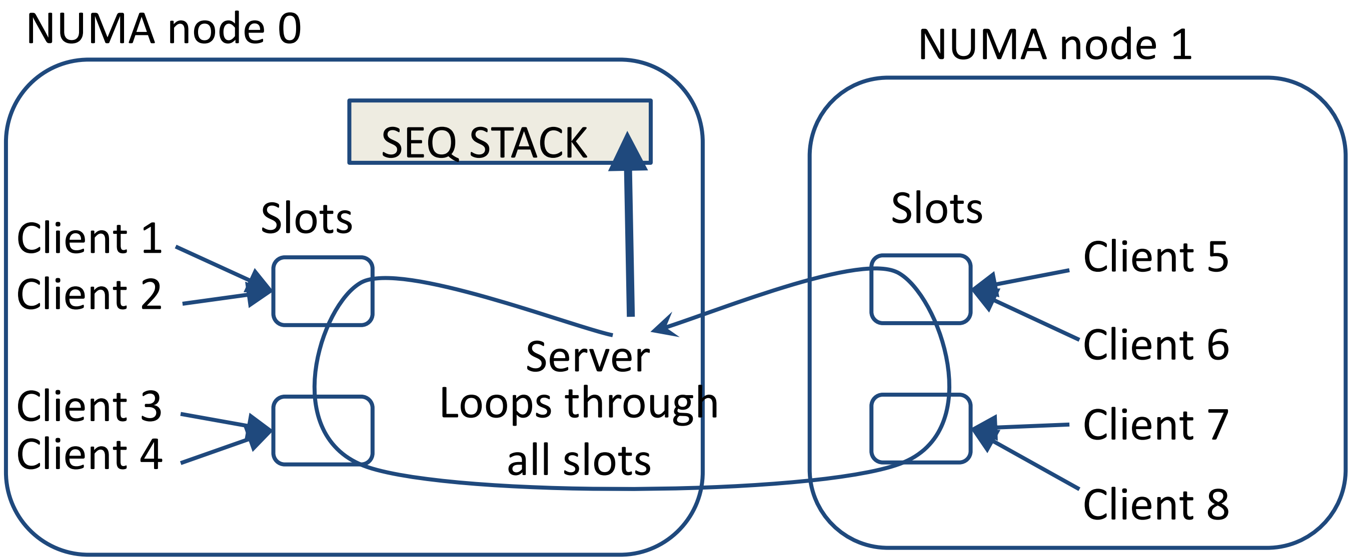
Overview

- Motivation
- Algorithms
- Results
- Conclusions

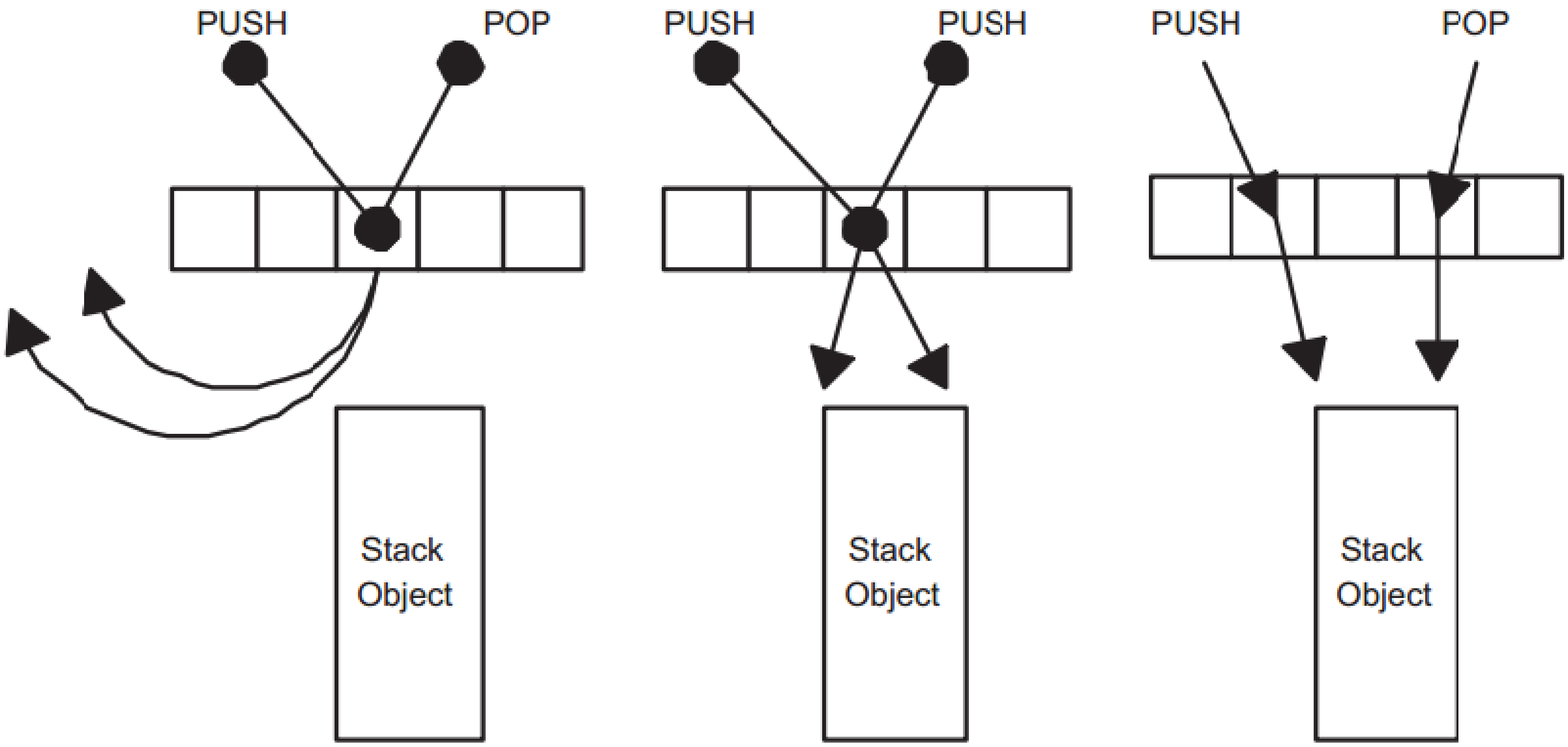
Delegation



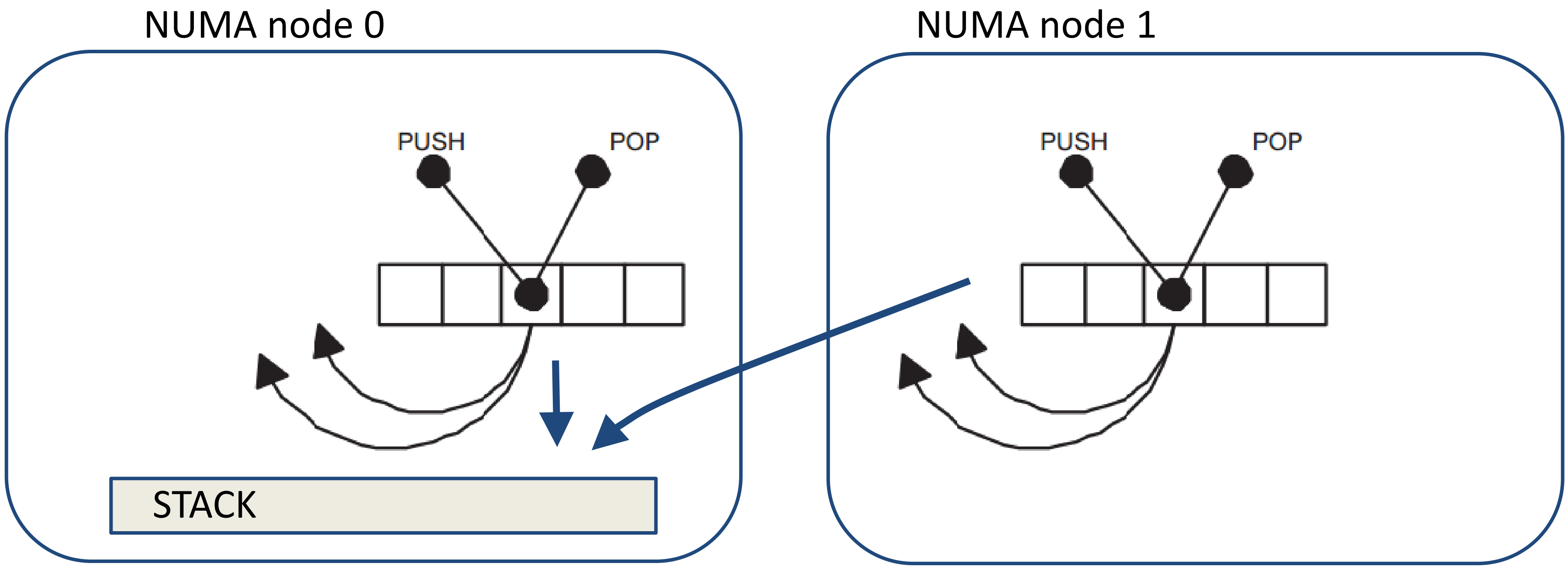
Delegation


NUMA node 0
SEQ STACK
Slots
Client 1
Client 2
Client 3
Client 4
Server
Loops through
all slots
NUMA node 1
Slots
Client 5
Client 6
Client 7
Client 8

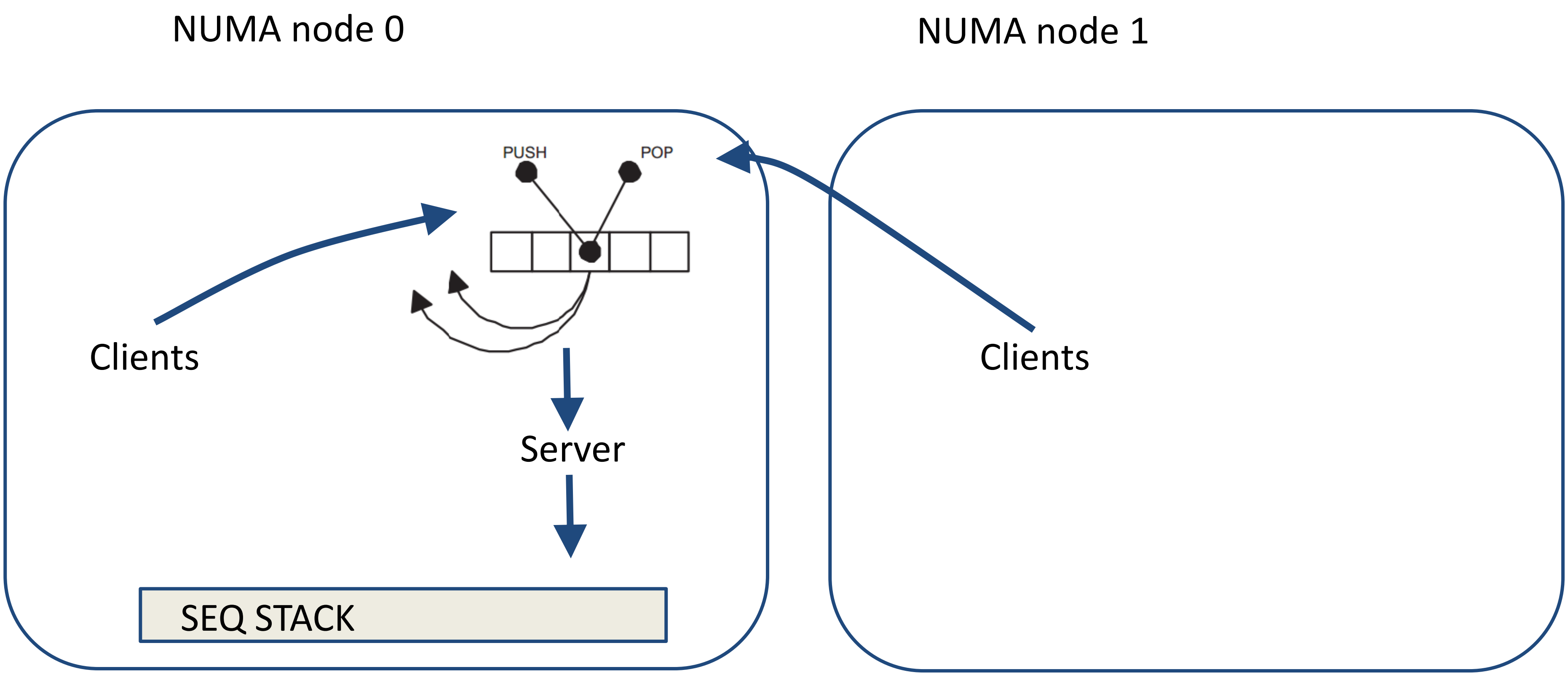
Elimination, Rendezvous



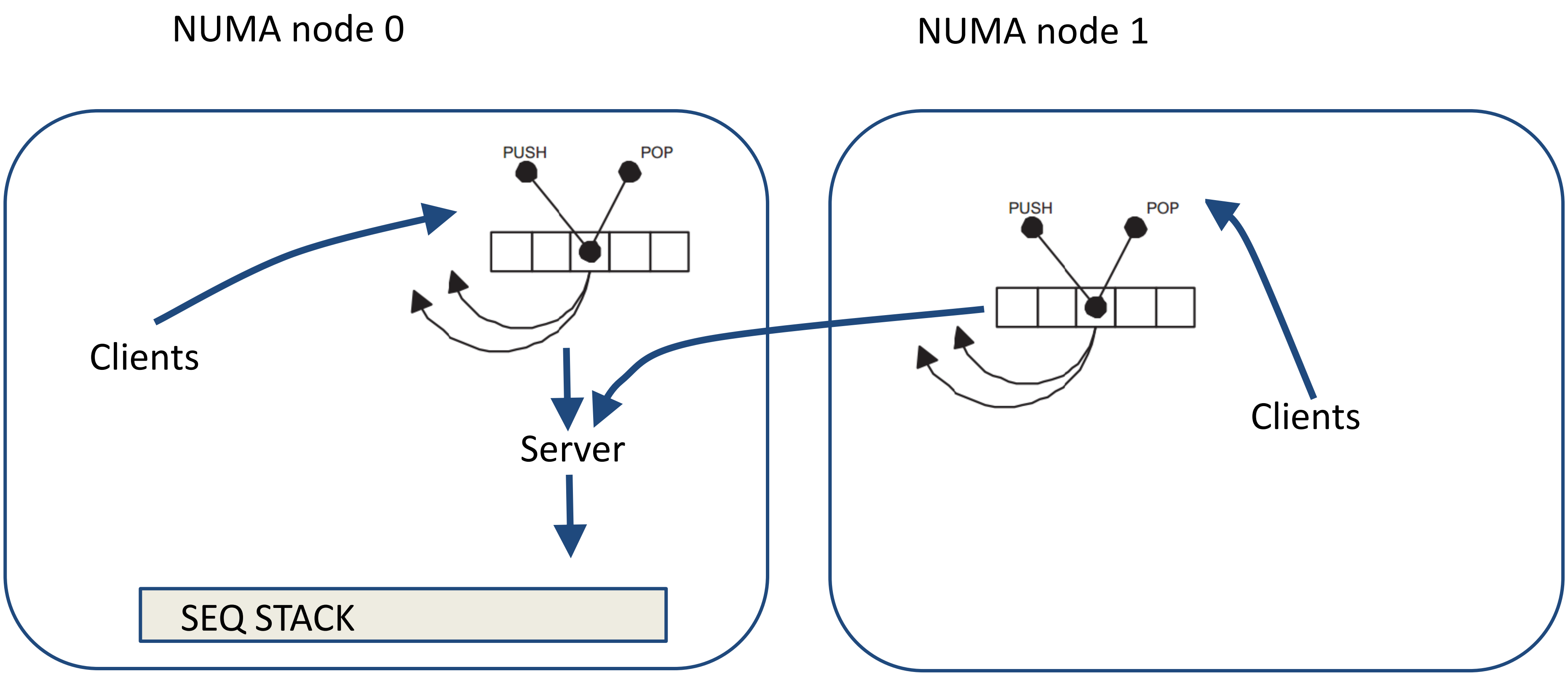
Local Rendezvous



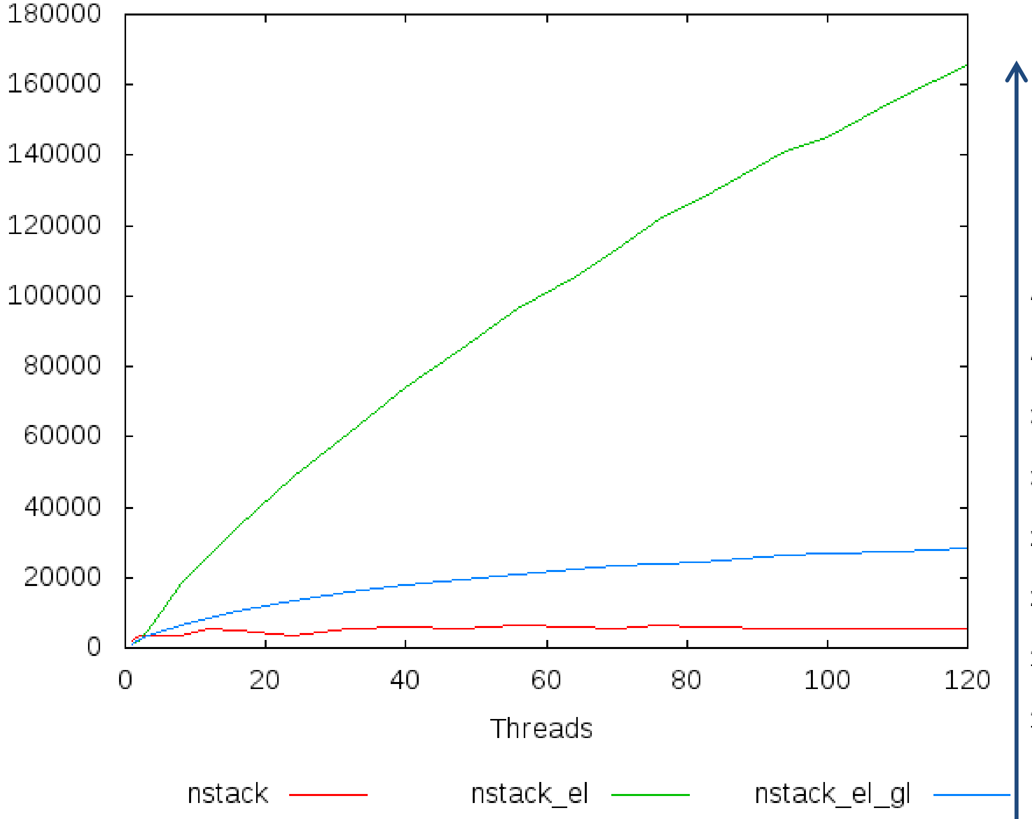
Delegation + Elimination



Delegation + LOCAL Elimination



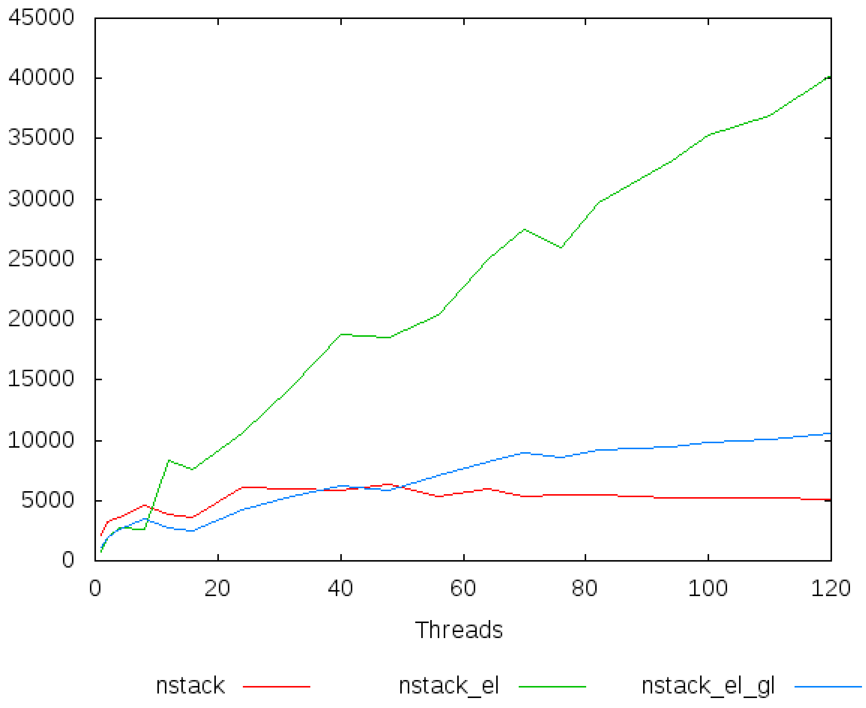
Effect of Elimination



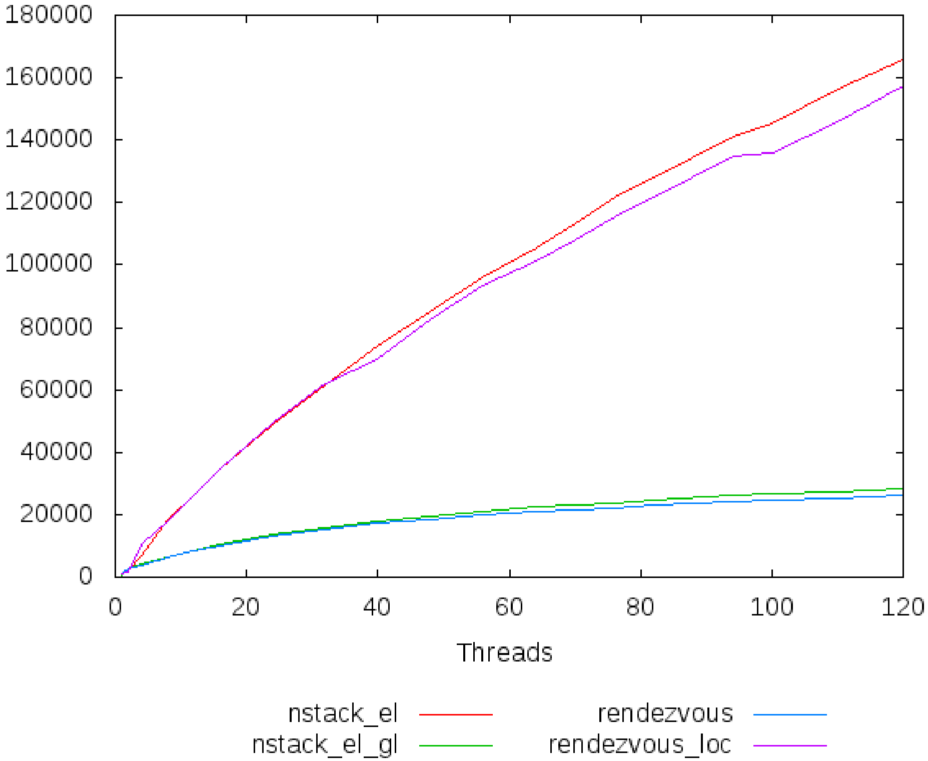
50% push 50% pop

Throughput (Better)

90% push 10% pop



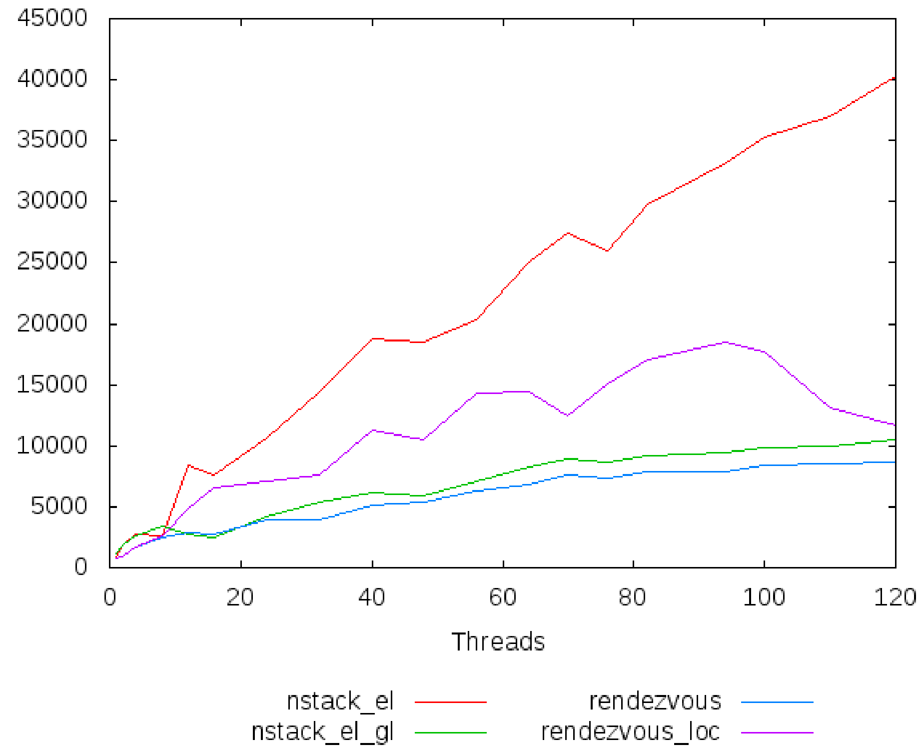
Effect of Delegation



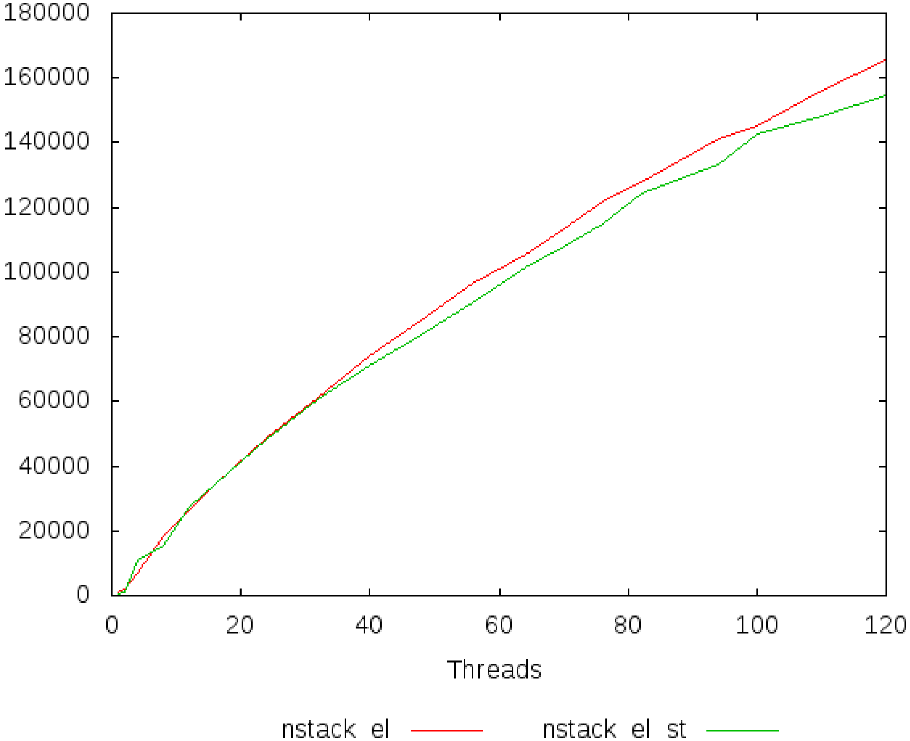
50% push 50% pop

Throughput (Better)

90% push 10% pop



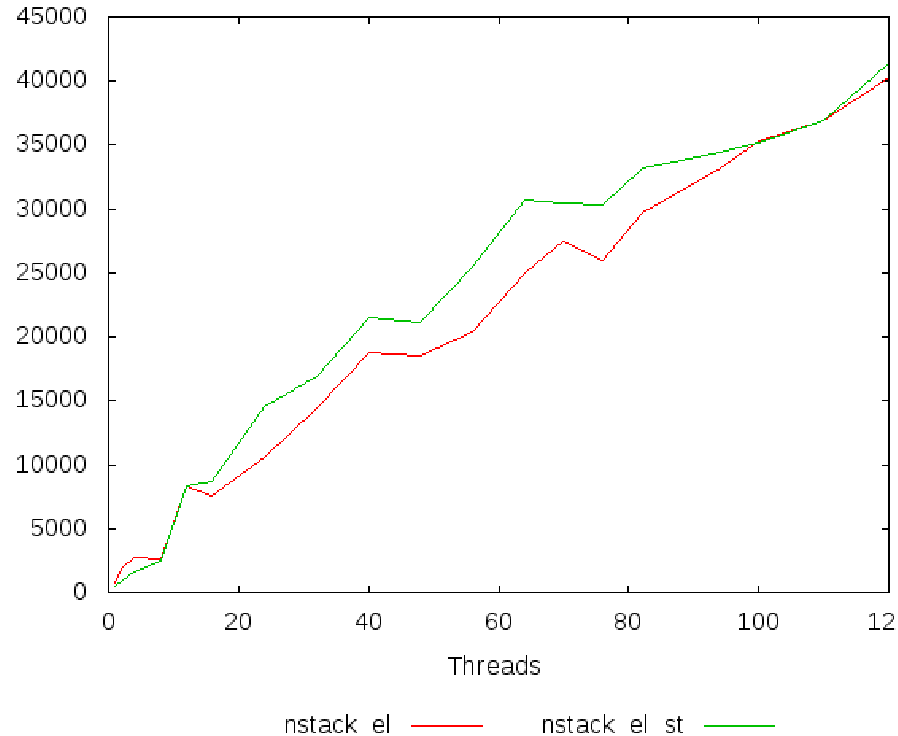
Number of Slots



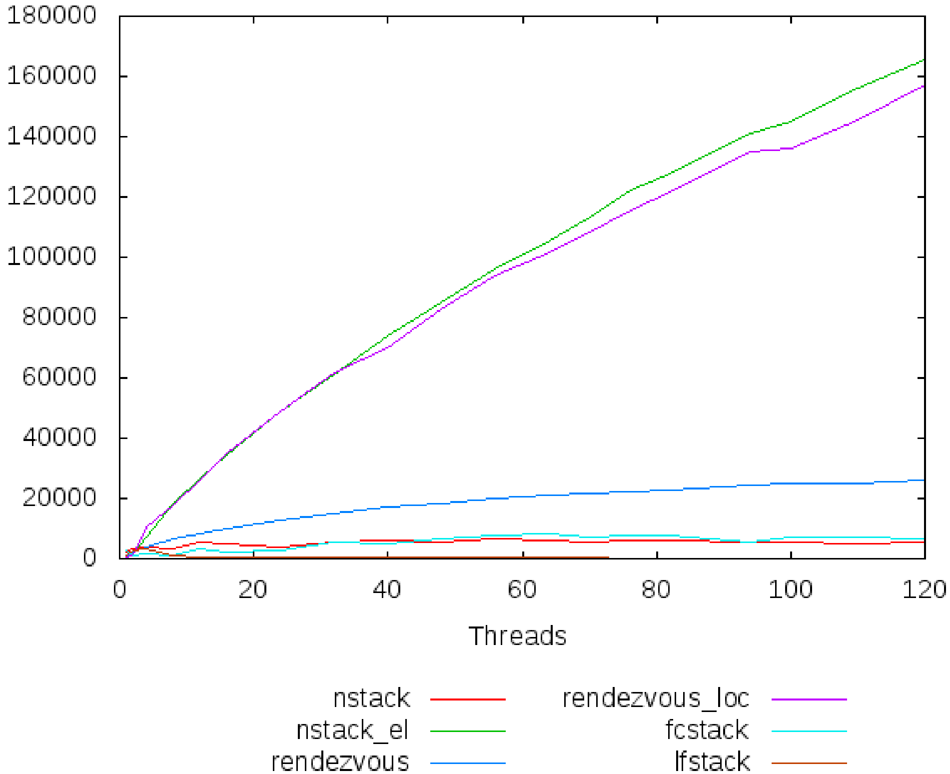
50% push 50% pop

Throughput (Better)

90% push 10% pop



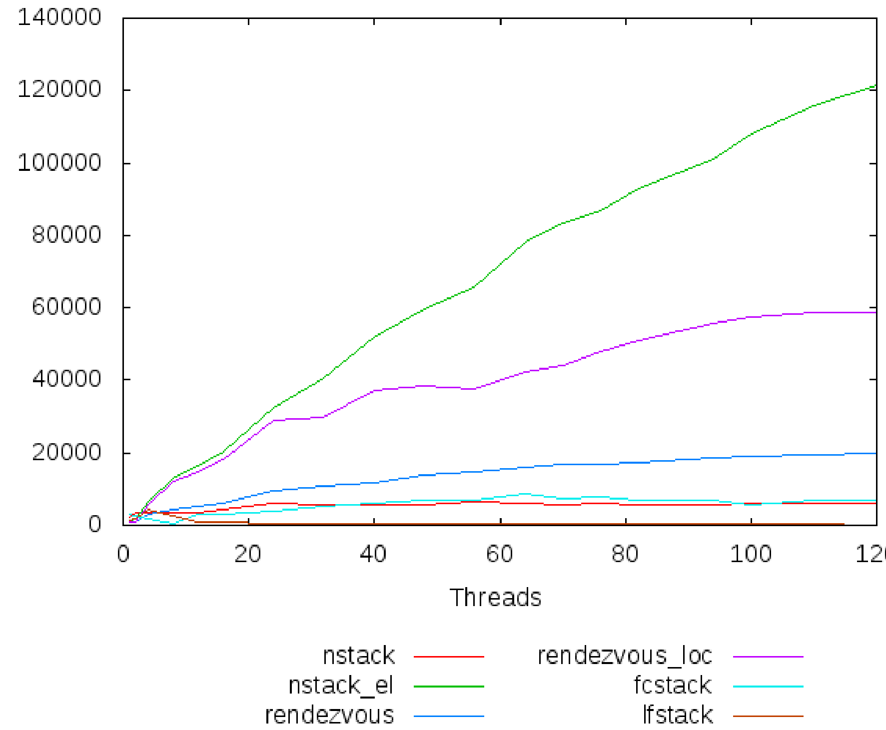
Workloads: Balanced vs. Unbalanced



50% push 50% pop

Throughput (Better)

70% push 30% pop



Advantages

- Memory and cache locality
- Reduced bus traffic
- Increased parallelism through elimination

Drawbacks

- Communication cost between clients and server thread
  - Insignificant compared to the benefits
- Serializing otherwise parallel data structures
  - Parallelism through elimination
- Elimination opportunities decrease as workload more unbalanced

Open Questions

- Are there other data structures where we can use delegation and elimination?
- Are there data structures where direct access is much better?
- What can we do for those data structures?

Thank you! Questions?

References

- A Scalable Lock-free Stack Algorithm

http://www.inf.ufsc.br/~dovicchi/pos-ed/pos/artigos/p206-hendler.pdf

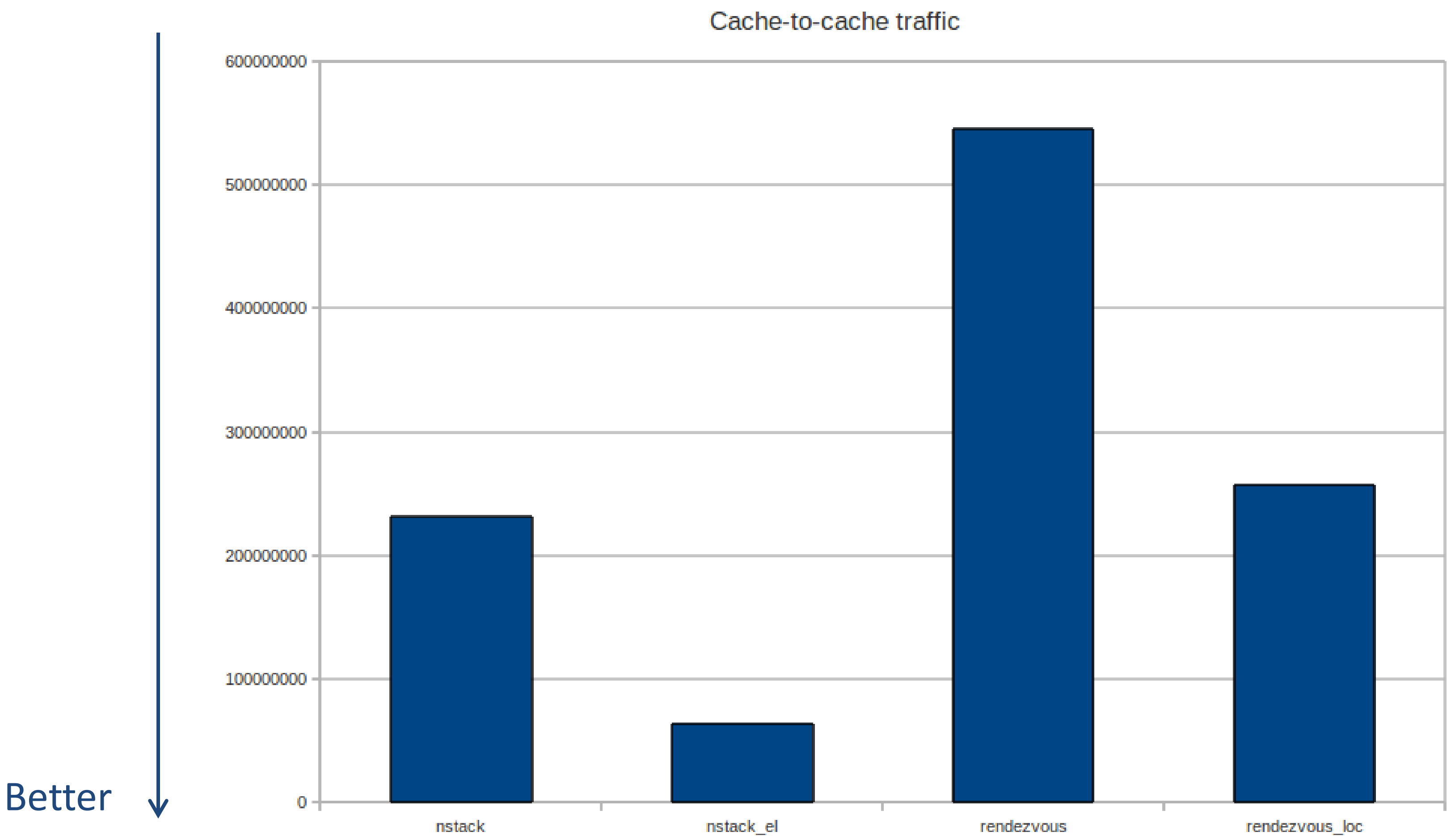
- Flat Combining and the Synchronization-Parallelism Tradeoff

http://www.cs.bgu.ac.il/~hendlerd/papers/flat-combining.pdf

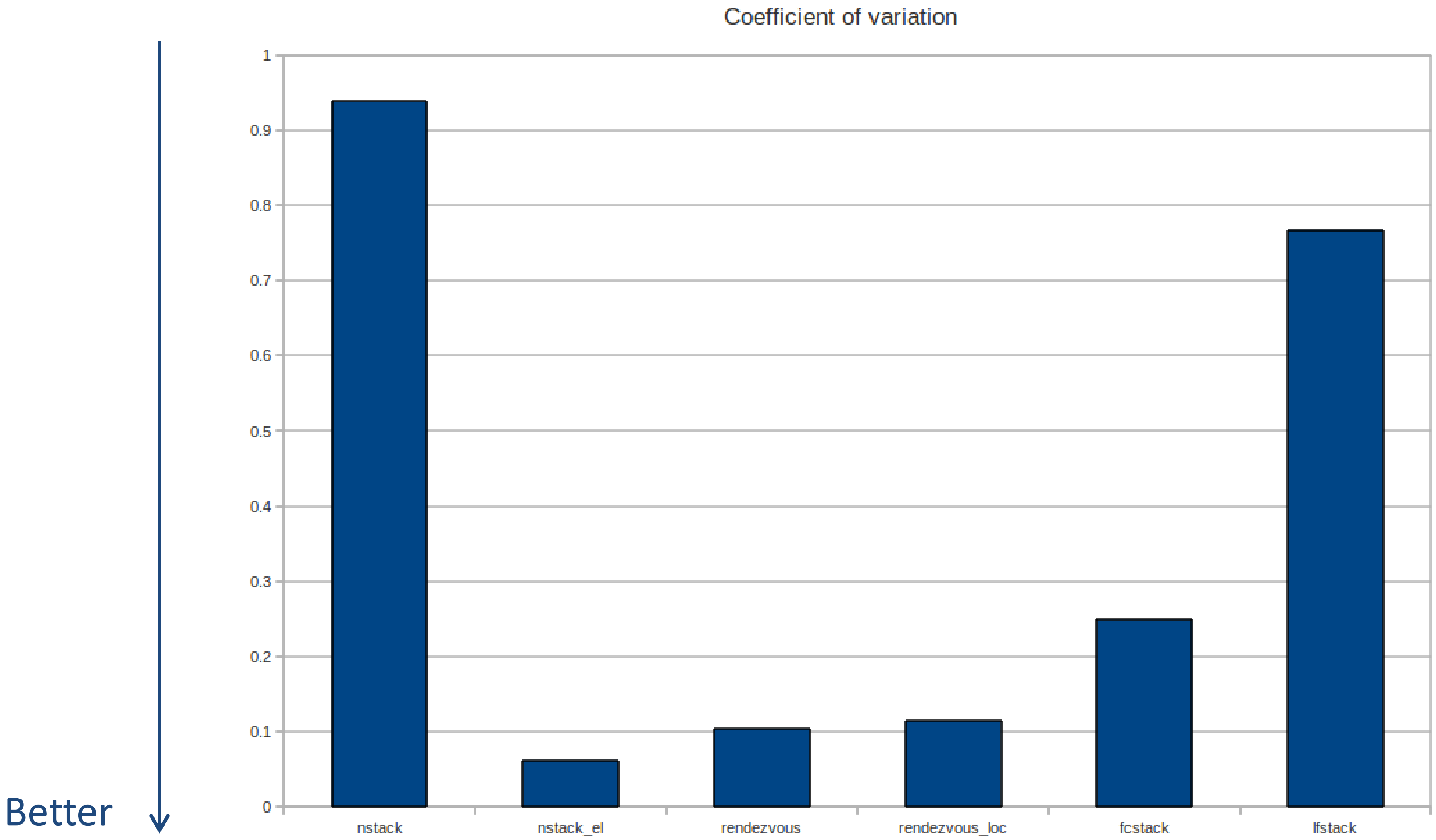
- Fast and Scalable Rendezvousing

http://www.cs.tau.ac.il/~afek/rendezvous.pdf

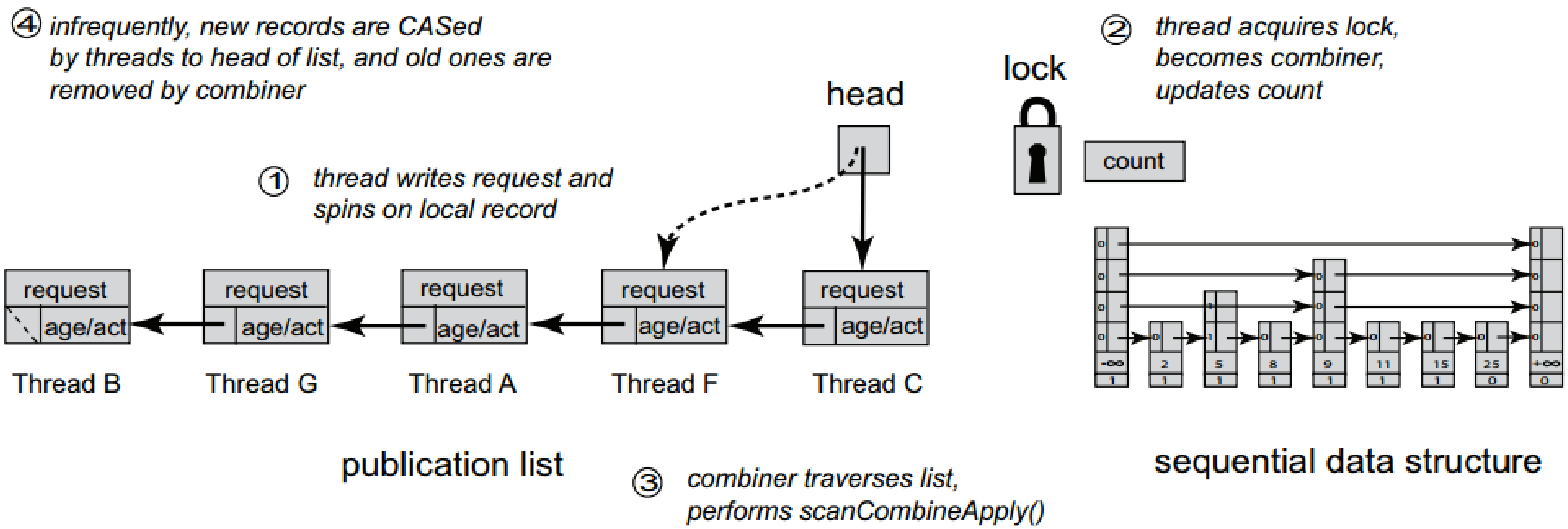
Cache to Cache Traffic



Coefficient of Variation



Flat Combining



Delegation

CLIENT

Find corresponding slot
(by NUMA node and cpuid)

Post message
Wait for response

Get response

SERVER

Loop through all slots:
If slot has message:

Take message
Process message
Send response

Time

Delegation

CLIENT

Find corresponding slot
(by NUMA node and cpuid)
try_elimination:
if (eliminate) return

Post message
Wait for response

Get response

else try_elimination

SERVER

Loop through all slots:
If slot has message:

Take message
Process message
Send response

Time

Delegation

CLIENT

Find corresponding slot
(by NUMA node and cpuid)
try_elimination:
if (eliminate) return
if (Acquire slot lock)
Post message
Wait for response

Get response
Release slot lock
else try_elimination

SERVER

Loop through all slots:
If slot has message:

Take message
Process message
Send response

Time

Open Questions

- Performance
- Scalability
- Power