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)

Cache coherency maintained between caches on different NUMA nodes
Overview

- Motivation
- Algorithms
- Results
- Conclusions
Delegation

NUMA node 0

Clients

Server

SEQ STACK

NUMA node 1

Clients
Delegation

NUMA node 0

Client 1
Client 2
Client 3
Client 4

SEQ STACK

Server
Loops through all slots

NUMA node 1

Slots

Client 5
Client 6
Client 7
Client 8
Elimination, Rendezvous
Local Rendezvous

NUMA node 0

NUMA node 1

STACK
Delegation + Elimination

NUMA node 0

Clients

PUSH

POP

Server

SEQ STACK

NUMA node 1

Clients
Delegation + LOCAL Elimination

NUMA node 0

NUMA node 1

Clients

Server

SEQ STACK

Clients
Effect of Elimination

Throughput (Better)

50% push 50% pop

90% push 10% pop
Effect of Delegation

Throughput (Better)

50% push 50% pop

90% push 10% pop
Number of Slots

Throughput (Better)

50% push 50% pop

90% push 10% pop
Workloads: Balanced vs. Unbalanced

50% push 50% pop

70% push 30% pop

Throughput (Better)
Advantages

• Memory and cache locality

• Reduced bus traffic

• Increased parallelism through elimination
Drawbacks

• Communication cost between clients and server thread  
  o Insignificant compared to the benefits  

• Serializing otherwise parallel data structures  
  o 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

• Flat Combining and the Synchronization-Parallelism Tradeoff

• Fast and Scalable Rendezvousing
  http://www.cs.tau.ac.il/~afek/rendezvous.pdf
Cache to Cache Traffic

![Cache-to-cache traffic chart]

- nstack
- nstack_el
- rendezvous
- rendezvous_loc

Better
Coefficient of Variation
Flat Combining

4. Infrequently, new records are CASed by threads to head of list, and old ones are removed by combiner.

1. Thread writes request and spins on local record.

2. Thread acquires lock, becomes combiner, updates count.

3. Combiner traverses list, performs `scanCombineApply()`.

Thread B

Thread G

Thread A

Thread F

Thread C

Publication list

Sequential data structure
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

else try_elimination

SERVER

Loop through all slots:
If slot has message:

Take message
Process message
Send response

Get response

Time

25
Delegation

CLIENT

Find corresponding slot (by NUMA node and cpuid)

try_elimination:
  if (eliminate) return
  if (Acquire slot lock)
    Post message
    Wait for response
  else try_elimination

SERVER

Loop through all slots:
  If slot has message:
    Take message
    Process message
    Send response
    Get response
    Release slot lock
    else try_elimination
Open Questions

• Performance

• Scalability

• Power