Performance and power consumption evaluation of concurrent queue implementations in embedded systems

Lazaros Papadopoulos, Ivan Walulya, Paul Renaud-Goud, Philippas Tsigas, Dimitrios Soudris and Brendan Barry

Distributed Computing and Systems
Chalmers university of technology

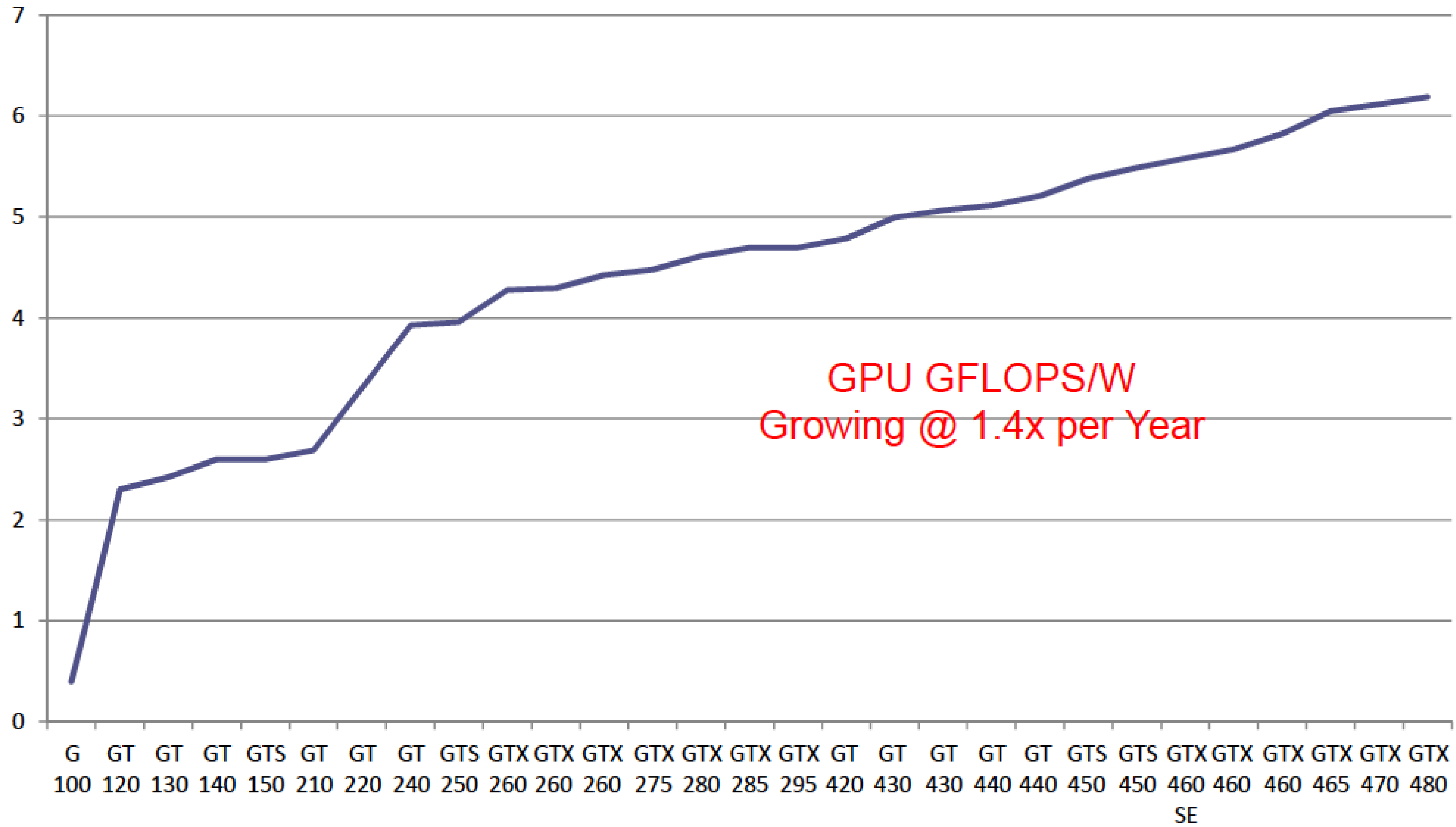
Movidius

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Computer Science

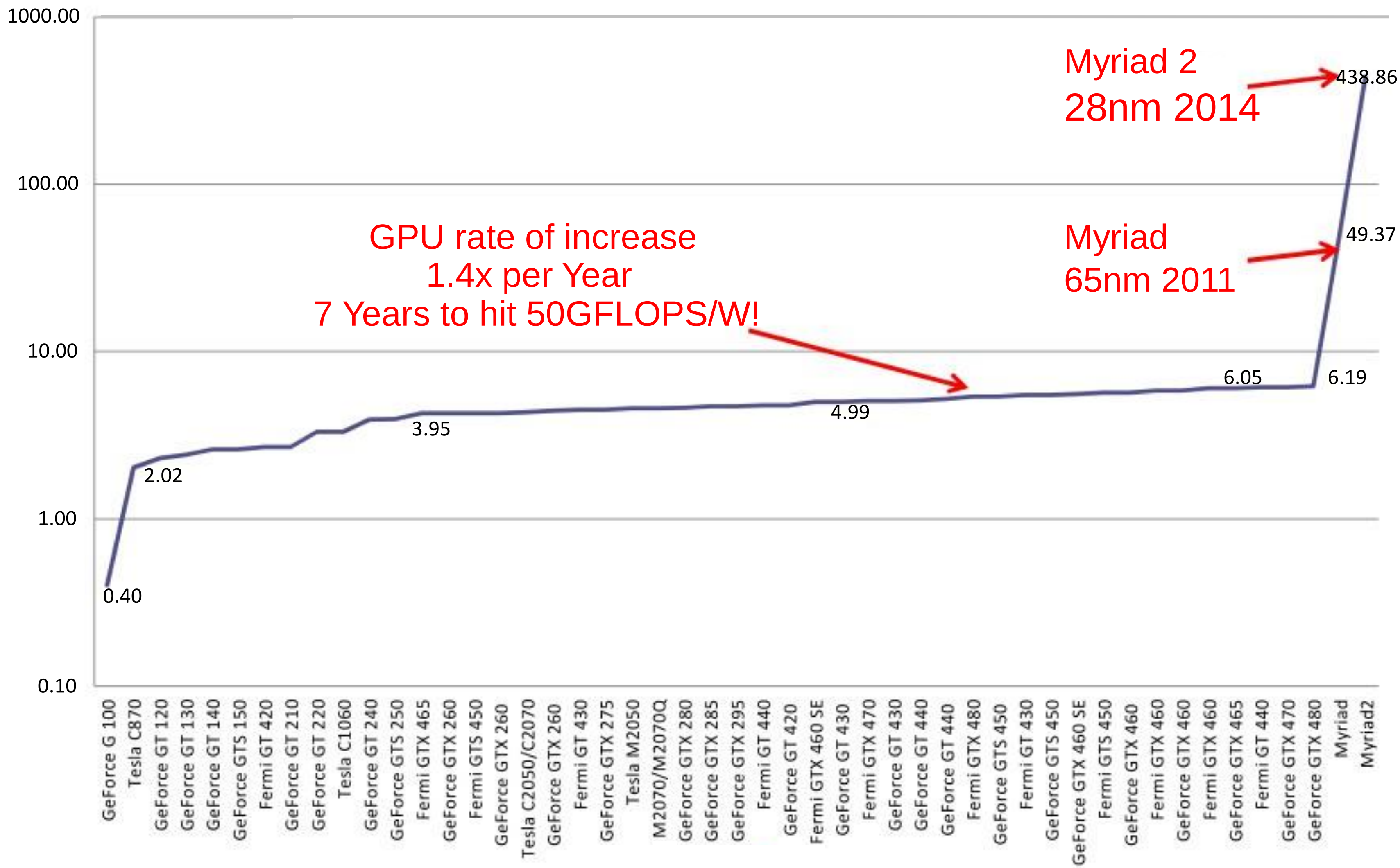
"Watt's Next?"

- Power consumption
  - – Design decisions
  - – Performance/watt metric
- Improvements in compute performance
  - More power budget
  - Cooling problems

GPU FLOPS/W Trend



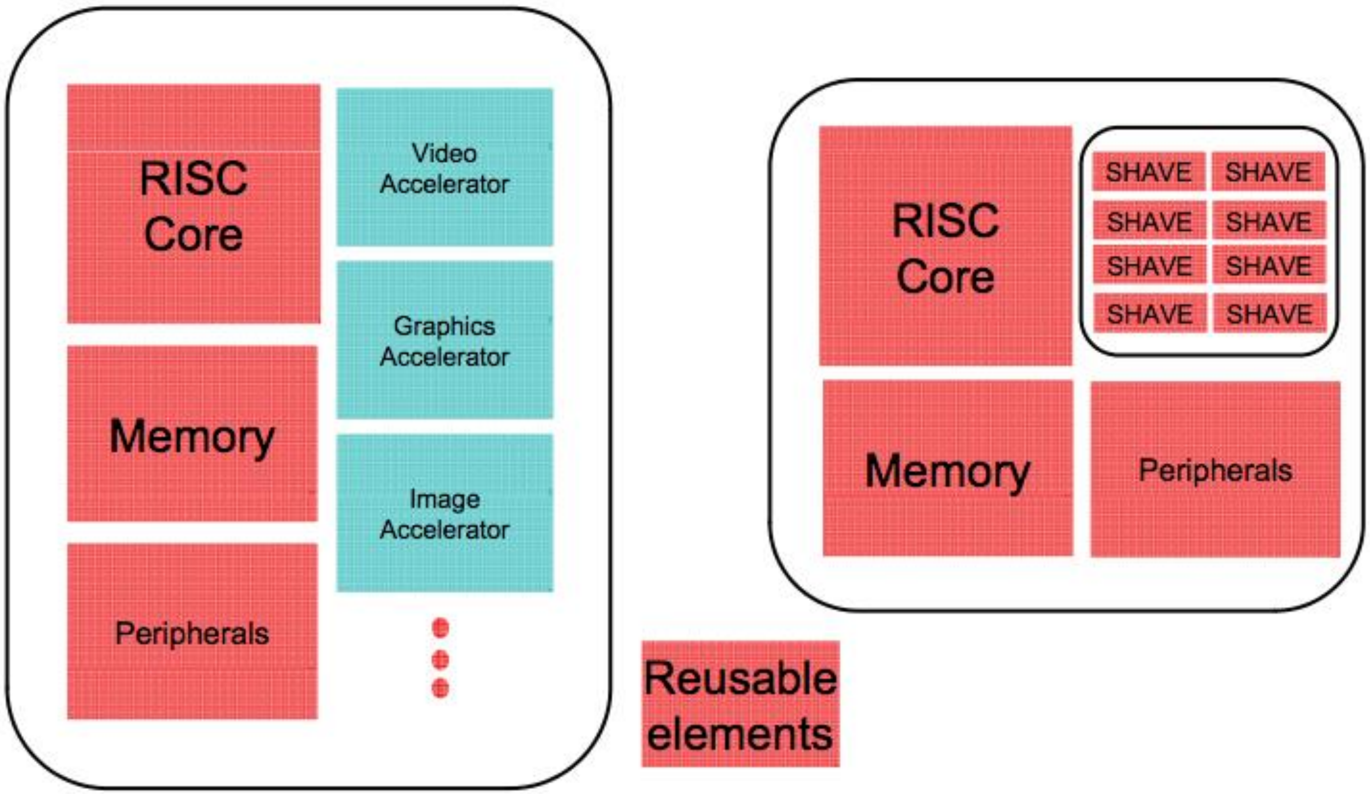
Emerging Embedded Systems Trend



Trends

Old Approach

New Approach



Always dead silicon when
not running that application

Same hardware is re-used no
matter what the application

Now that I've got an Ultra low power Compute Platform

What can I do with it?

- Potential of such low power processors for use in high end computations.
- Can they offer a solution to power problems
- Can high-performance computing techniques be deployed on these processors?

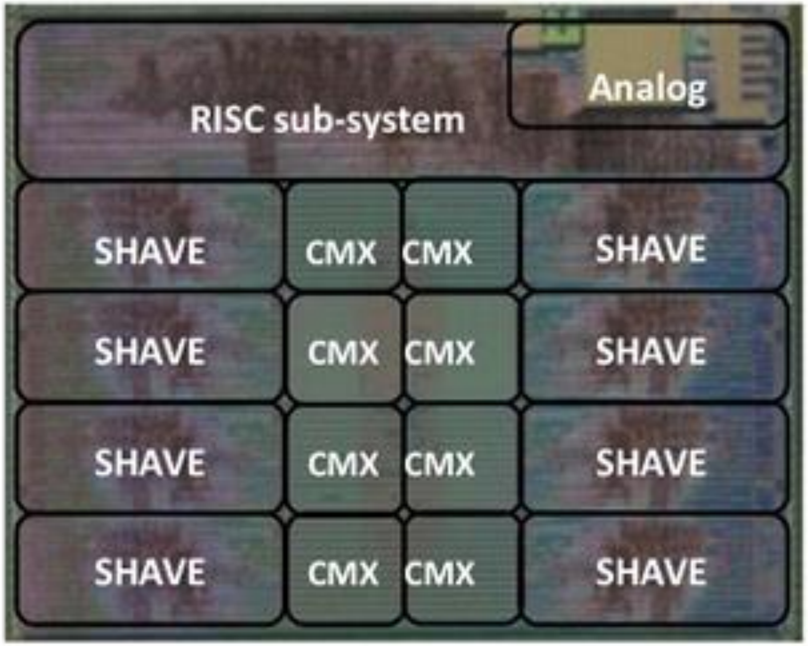
Outline

- Introduction
  - Synchronization on multi-core platforms
  - Movidius SoC
- Algorithmic Designs
- Experimental results
- Conclusions

Concurrent Data Structures

- Hardware support
- Mutexes
  - Scalability
  - Busy Waiting
- Non-blocking
  - Atomic hardware primitives (e.g. LL/SC, CAS)
  - Good progress guarantees (lock/wait-freedom)
  - Scalable
- Message-passing techniques from HPC domain

Myriad architecture



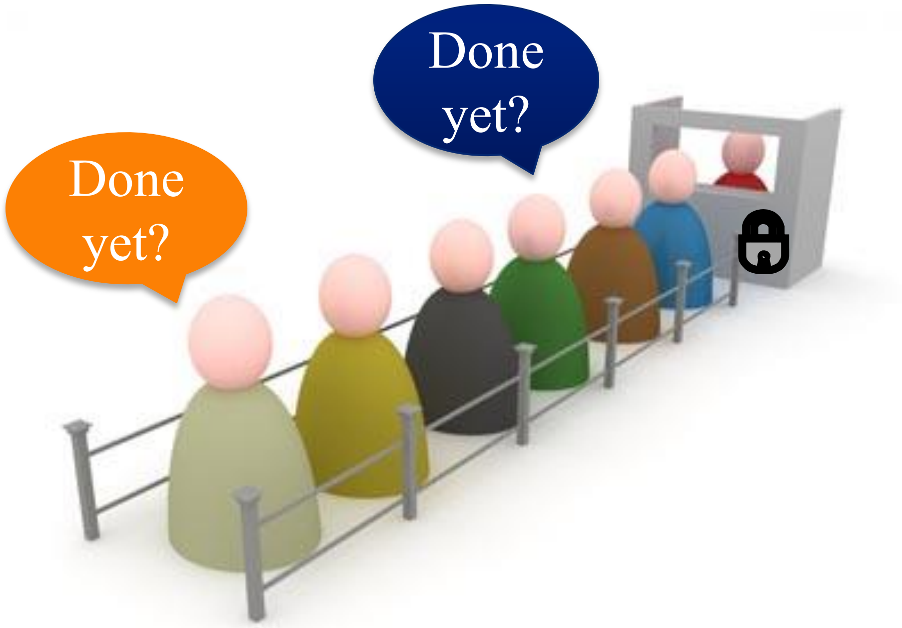
- Processors:
  - 32-bit general purpose RISC SPARC processor (LEON).
  - 8 SHAVE (Streaming Hybrid Architecture Vector Engine) processors for computational processing.
- Memory:
  - CMX (Connection Matrix): 1 MB on-chip RAM (with 128KB per SH AVE core)
  - SDRAM: 64MB.
- Synchronization support on Myriad: Mutexes, FIFO registers

Algorithmic Designs

- Single Lock
- Double Lock
- Client-Server
- Remote Core Locking - RCL

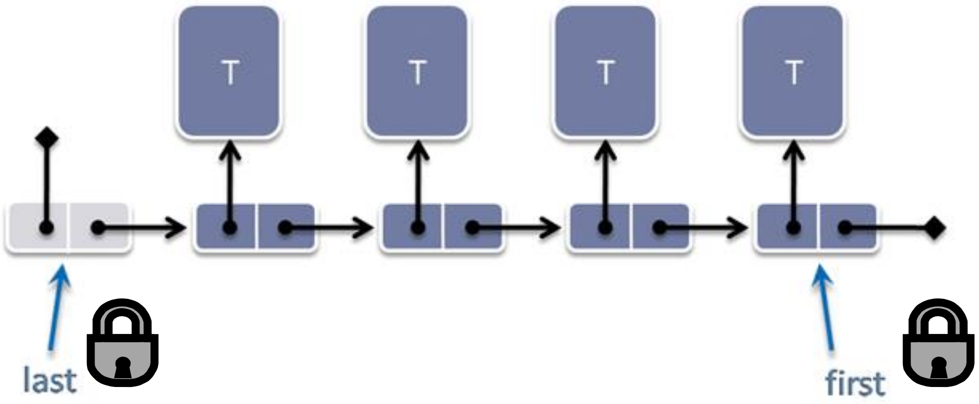
Single Lock

- No concurrency
- Busy waiting
- No Scalability



Multiple Locks

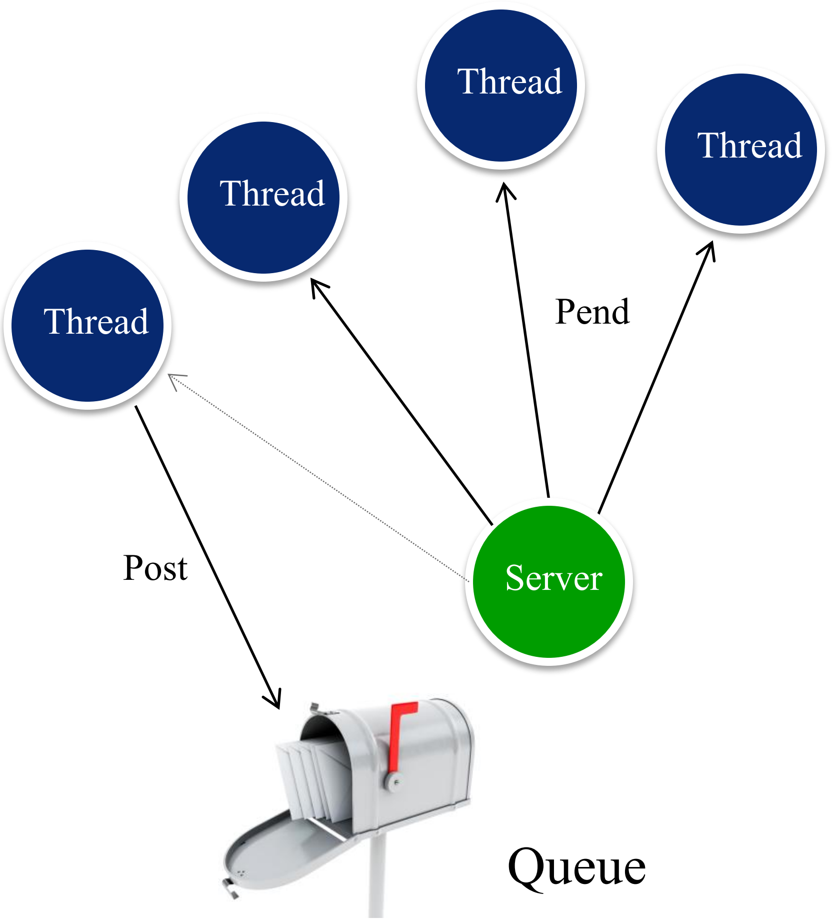
- Better concurrency
- Improved scalability
- Busy waiting



Client-Server arbitration (C-S)

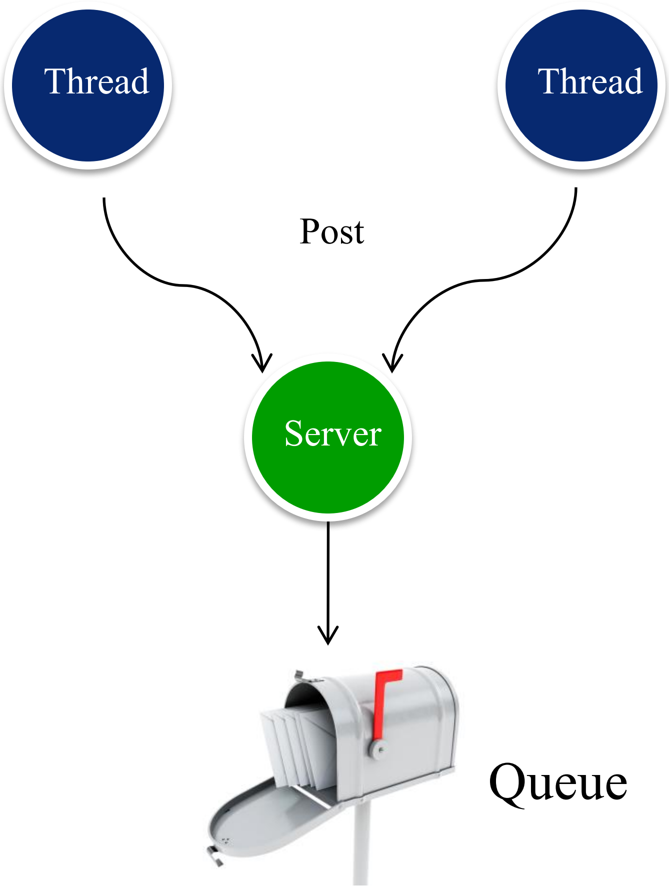
- Request for access
- Spin on local variable

- Shared variables
- Hardware FIFO queues

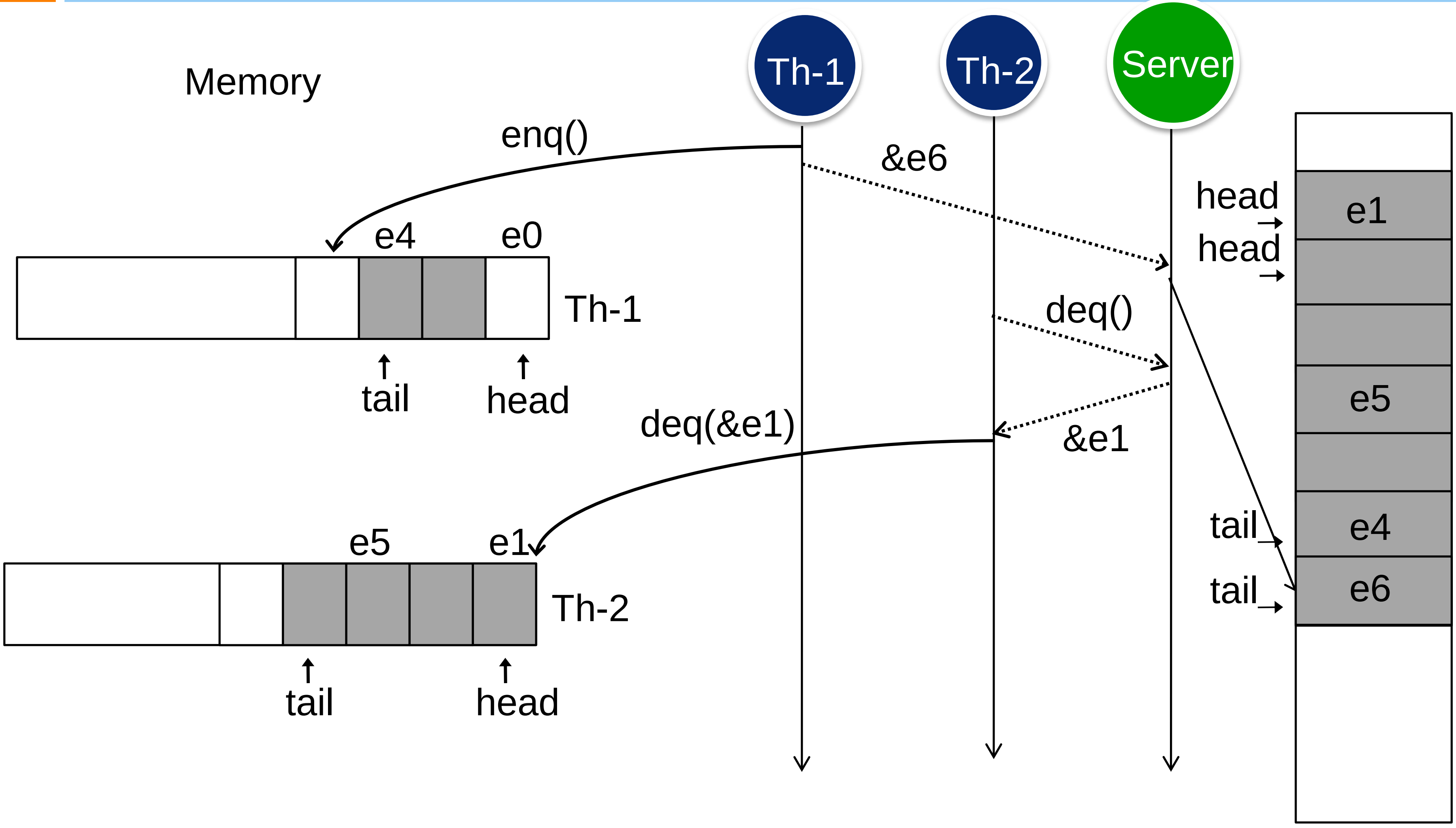


Remote Core Locking (RCL)

- Migrate Critical Section
- No shared data transfers
- Reduced Bus traffic



Client-Server



Client-Server Drawbacks

- Clients-Server communication costs
- Serialization of a concurrent data structure
- Losing one core

Experimental evaluation

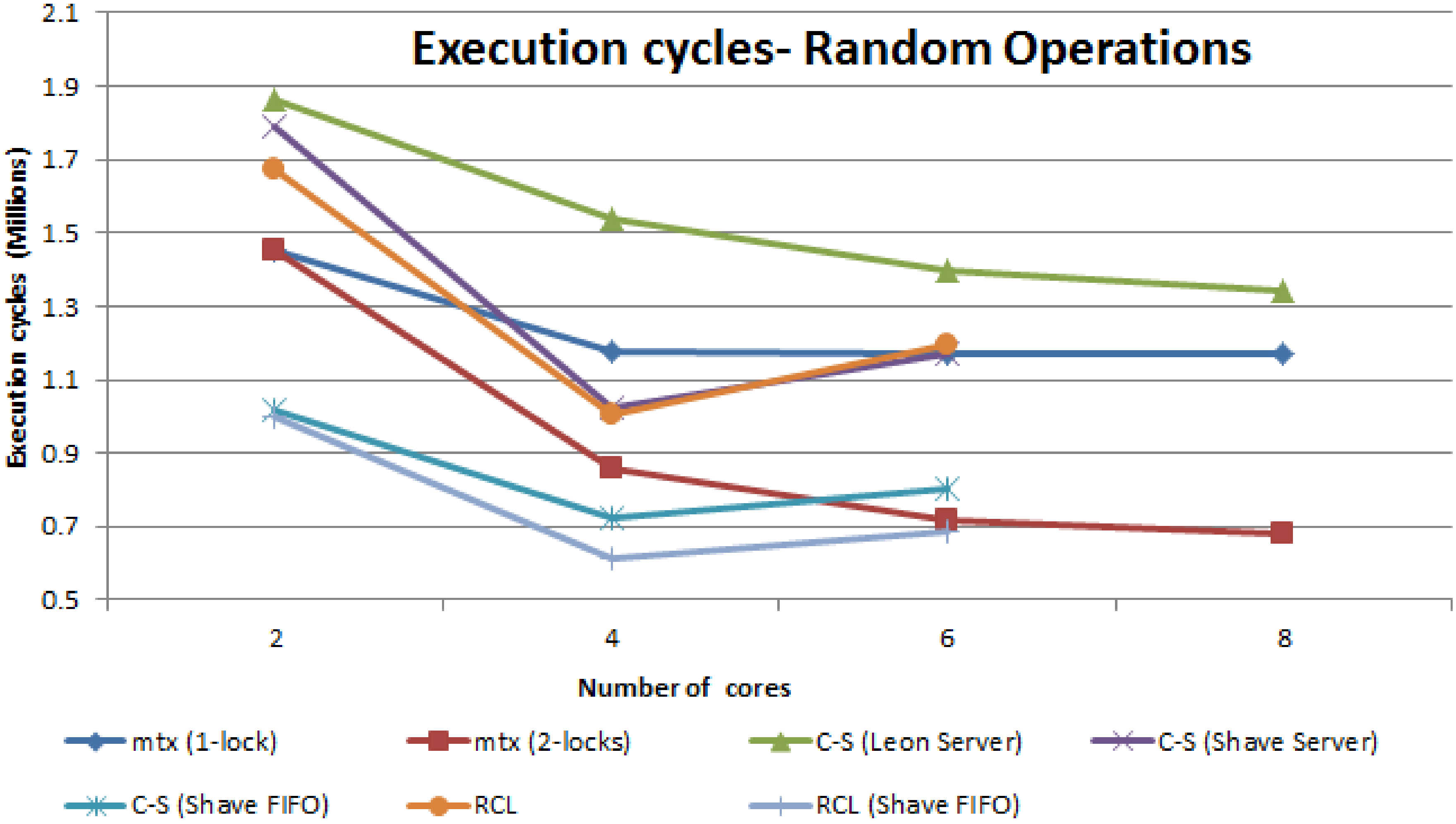
- FIFO Queues
- Cores execute Enqueue and Dequeue operations
  - High contention
- Test Configurations
  1. Random
  2. *Dedicated (N/2* Producers / *N/2* Consumers)

- Measured execution time in *cycles*
- Power consumption

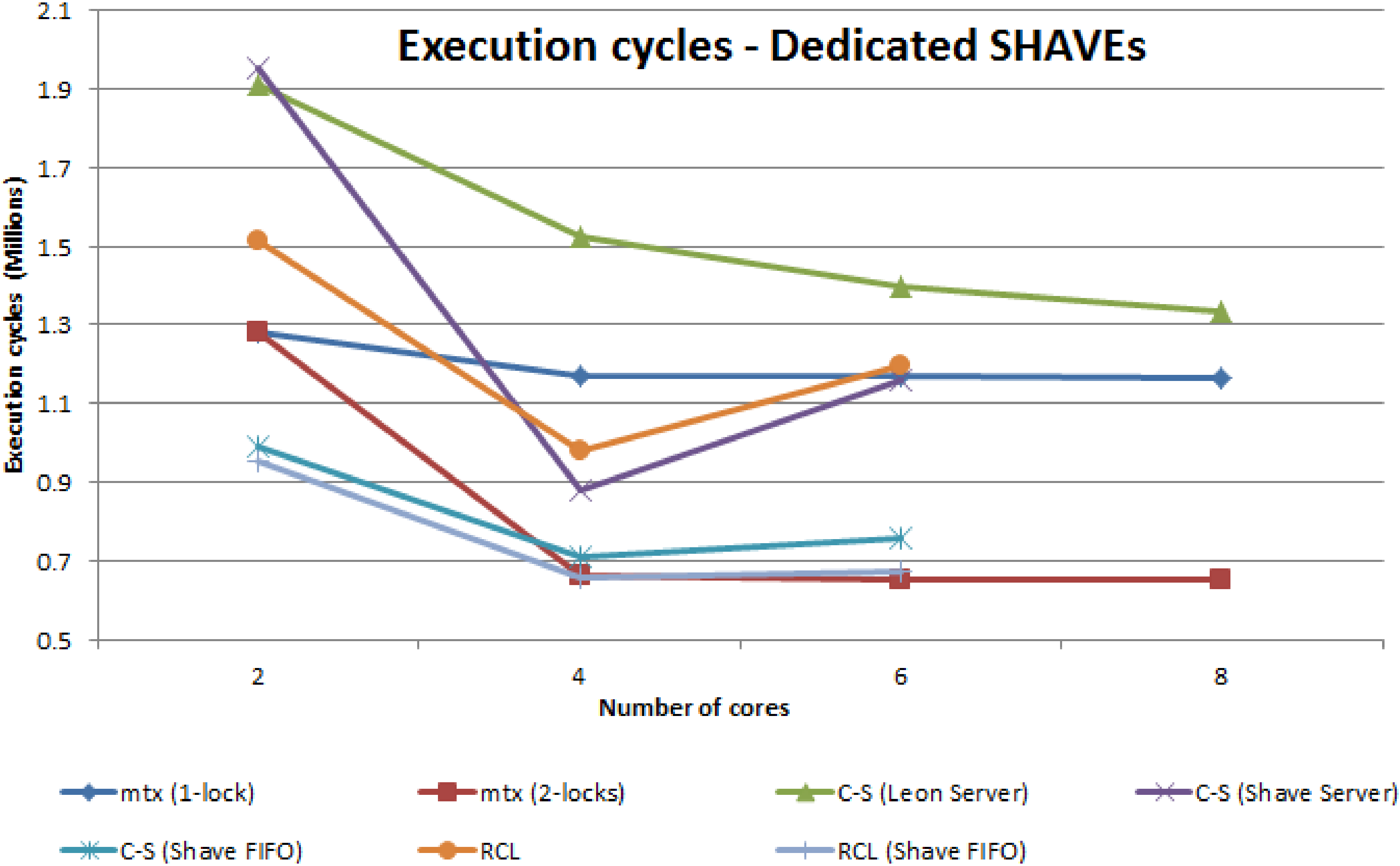
Experimental evaluation

- Single lock *mtx (1-lock)*
- implementation with 2 locks *mtx (2-locks)*
- Client-Server with Leon as server *C-S (Leon Server)*
- Shave as Server *C-S (Shave Server)*
- Shave as server using FIFO registers *C-S (Shave FIFO)*
- Remote Core Locking *RCL*
- Remote Core Locking using FIFO registers *RCL (Shave FIFO)*

Experimental Results



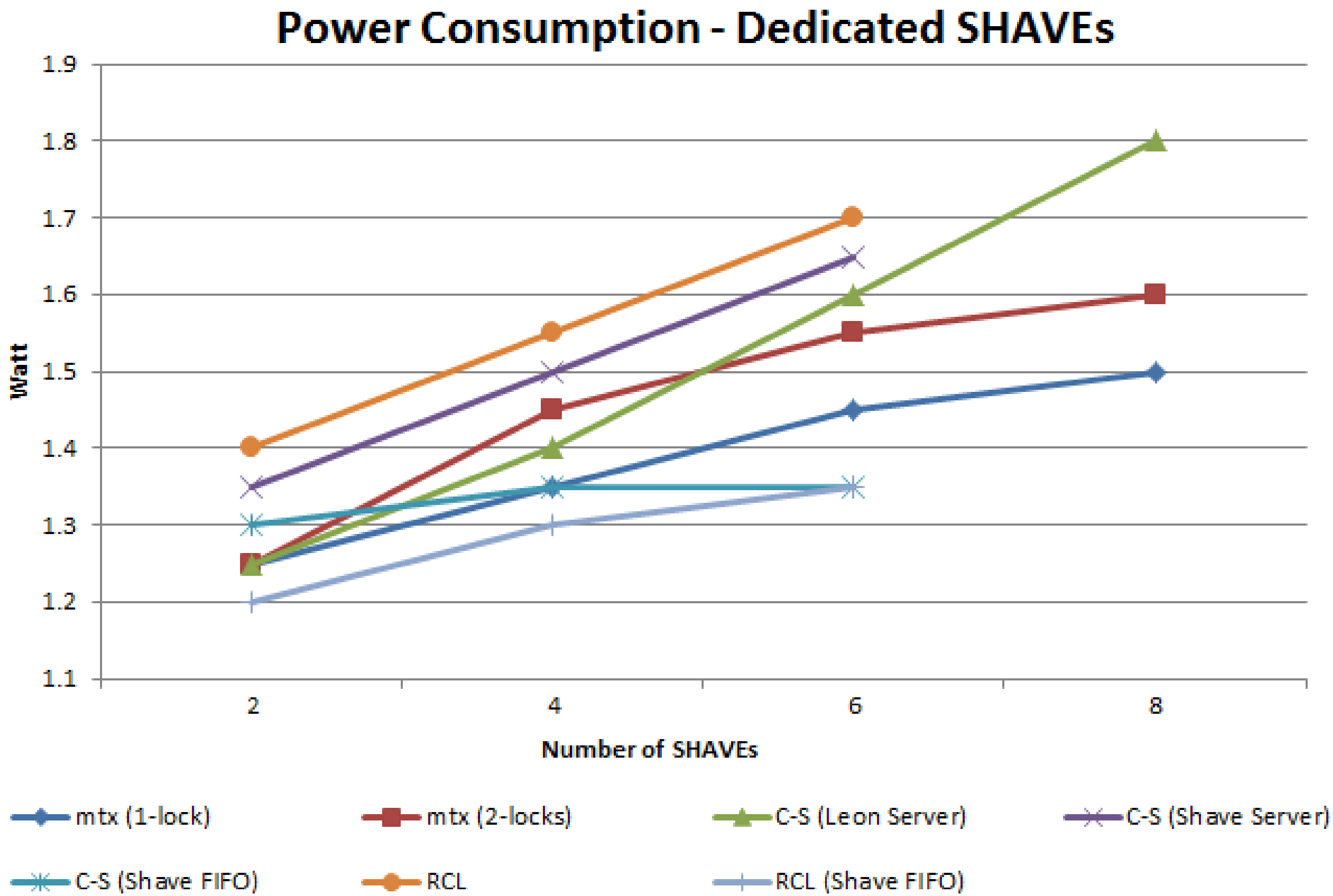
Experimental Results



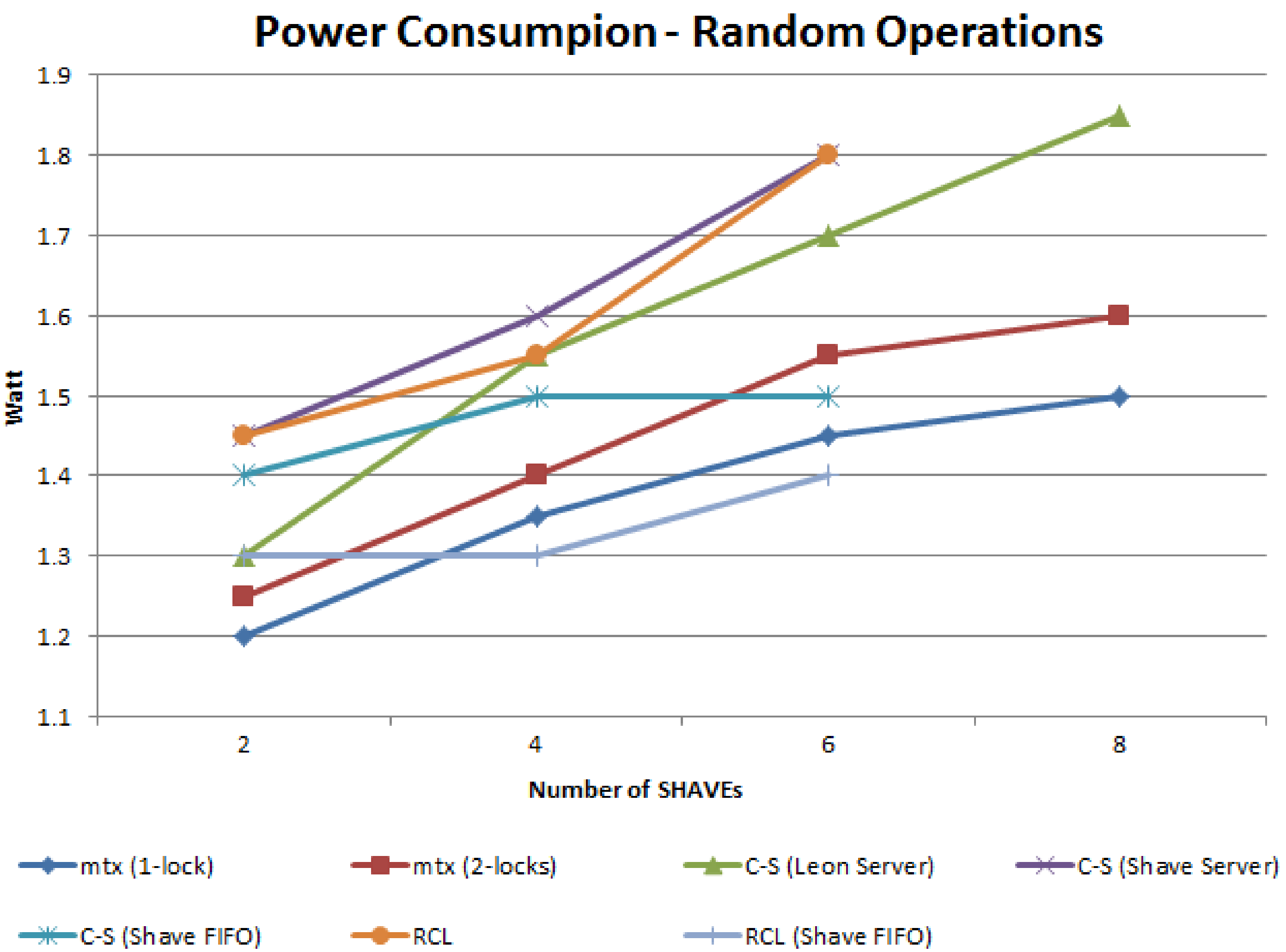
Power Consumption Evaluation

- power consumption measured using a shunt resistor connected to the power supply of the platform

Experimental Results



Experimental Results



Conclusions

- Complex data structures can be deployed on ultra low power processors
  - Exploit hardware primitives for better power values.
- With relatively low absolute performance can they be viable for high-end computing
- With 3D stacking it may become possible to stack many processors for very fast and energy-efficient communication

Questions?

Thank You!

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7) under grant agreement n°611183 (EXCESS Project, www.excess-project.eu)