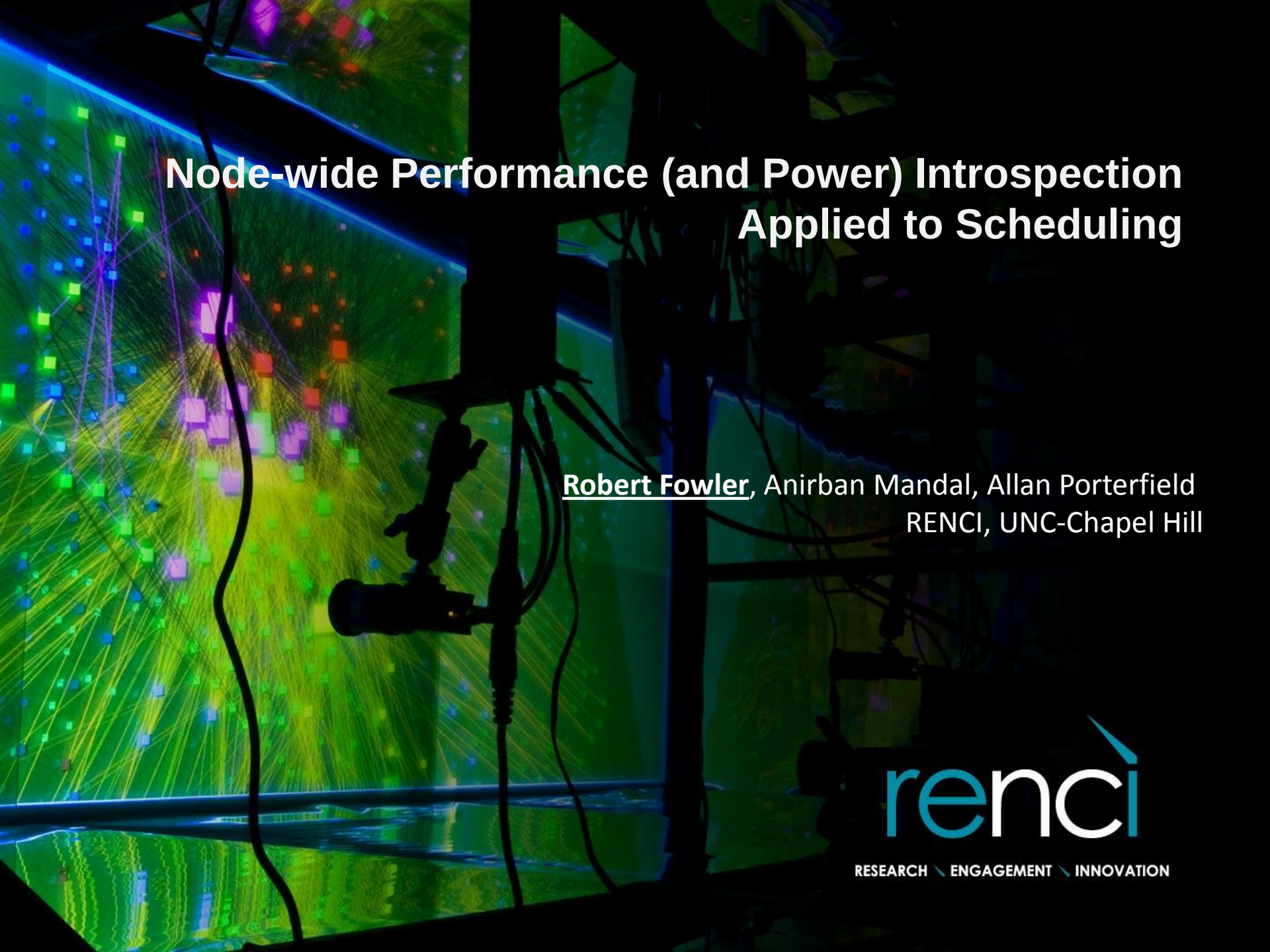


Node-wide Performance (and Power) Introspection Applied to Scheduling

Robert Fowler, Anirban Mandal, Allan Porterfield
RENCI, UNC-Chapel Hill

renci
RESEARCH \ ENGAGEMENT \ INNOVATION

Adaptive systems and applications are the future.

- Outline
  - Hardware issues:
    - The constrained evolution of processor design.
    - The hardware already adapts itself.
  - Off-chip bottlenecks: memory (and I/O).
  - Whole node measurement in support of introspection.
  - Won't actually get to adaptive, introspective scheduling.

renci

Dennard Scaling of CMOS Logic.

- Series of papers 1972-1974 by Bob Dennard and others at IBM on scaling properties of CMOS logic circuits (gates and wires!).
- Linear scaling of all transistor parameters.
  - Reduce feature size by a factor of S. Typically, 0.7/generation.
    - Including gate insulator thickness!
  - Reduce supply voltage (Vdd) by S to keep electric field constant.
  - Adjust doping of silicon gate region to compensate.
- Consequences
  - Area shrinks by S^2, C_{gate} and delay (1/f) reduced by S.
  - Power ≈ CV^2f ➔ Power per gate goes down by S^2
  - Area and power track each other so power density is unchanged.
  - For a constant die area and design density, power and power density are constant and frequency can be increased.

renci

Other Aspects of Dennard Scaling.

- Wire resistance/unit length ~ S^2
- Wire capacitance/unit length ~ 1
- RC delay/unit length (unrepeated) ~ S^2
- Die size (D) increases, so “long” wires increase by D
- Unrepeated wire delay ~ S^2D^2, repeated ~ D sqrt(S)

  →Signals cannot cross the chip in one cycle.

renci

Moore's law

Empirical observation and self-fulfilling prophesy:
Circuit element count doubles every N months. (N ~18)

- Technological explanation: Features shrink, semiconductor dies grow.
- Dennard scaling: Gate delays decrease. Wires are relatively longer/slower.
  - Dennard scaling has not been perfect in practice and is coming to an end.
- In the past, the focus has been making "conventional" processors faster.
  - Faster clocks
  - Clever architecture and implementation → instruction-level parallelism.
  - Clever architecture (speculation, predication, etc), HW/SW Prefetching, and massive caches ease the “memory wall” problem.
- Problems:
  - Faster clocks --> more power.
  - Power scaling law for CMOS: $P = \alpha CV^2F$, but $F_{max} \sim V$ so $P \sim F^3$
    - Where α is proportional to the avg. number of gates active per clock cycle.
  - Smaller transistors + long wires → either slow clock, or pipelined communication.
  - More power goes to overhead: cache, predictors, “Tomasulo”, clock, …
  - Big dies --> fewer dies/wafer, lower yields, higher costs
  - Aggregate effect --> Expensive, power-hog processors on which some signals take 6 cycles to cross.

renci

The End of Dennard Scaling: Dark and Dim Silicon

- Vdd Scaling issues
  - Initially, designers constrained by standards: 12V, 5V, 3.3V.
  - On-board power regulation now allows Vdd to be 1V or less.
  - This is getting uncomfortably close to threshold voltages.
  - Decreasing thresholds has rapidly increased leakage current/power.
  - Decreasing f allows operation with higher thresholds.
- Gate Insulator issues
  - Thickness is now ~ 5 atoms
- Useful work and duty cycles
  - Bailey and Snyder (1988) observed that α was at most a few percent for processors. If α were much larger, chips would melt.
  - Aggressive architectures have increased α to do bookkeeping, data movement, …
- "Dark" and "dim" silicon refer to schemes to reduce α and/or f to reduce power.
  - Heterogeneous cores and purpose built modules w. power mangement.
  - Programmable logic and reconfigurable devices.

➔ We can now build chips that cannot be run at their full design potential.

renci

“Dim silicon” adaptation in X64_64

- Intel: On-chip control processor
  - “Turbo” modes throttle f when all cores are active.
    - Run power-efficient, low f, low V in highly parallel code regions.
    - Inefficient high f, high V in sequential regions.
  - Shut down cores
  - Automatic DVFS to keep within thermal budgets.
    - Aggressive turbo mode when one core is active.
    - Brief performance bursts when thermal headroom exists.
  - “Balanced” mode DVFS to trade FLOPS and Watts.
    - Reduce core frequency when code seems to be memory bound.
- AMD:
  - Similar strategies
  - Additionally, A-series can attempt to balance CPU and GPU performance and power budgets.

renci

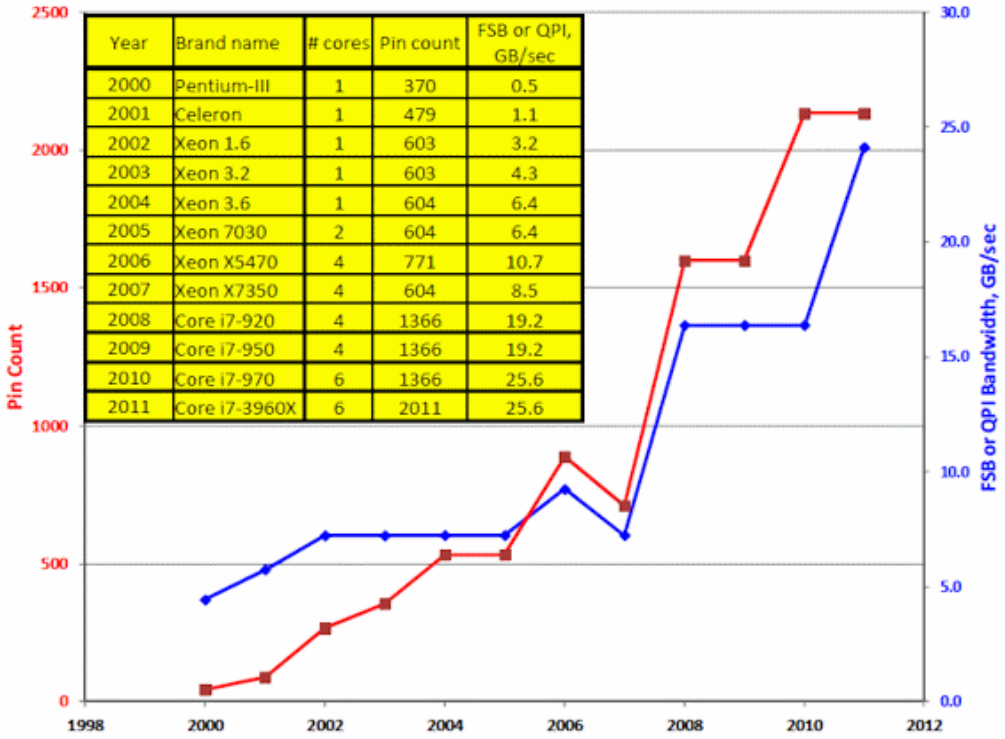
Moore's Law/Dennard Scaling for DRAM.

- As more transistors were added to processor chips, they got a lot faster.
  - Dennard scaling for faster transistors.
  - Clever architectures and on-chip concurrency.
- As more transistors were added to memory chips, they got a lot bigger.
  - Cleverness went into reliability, yield, …
  - Small transistors are fast, but weak (can't drive long wires).
  - Little increase in on chip concurrency.
  - Very low Rent's law (surface/volume ratio) exponent!

	Introduction	Size	Pins	Cycle Time	Bandwidth
DDR	2000	2 GB	168	5 ns	3.2 MB/sec
DDR2	2003	4 GB	184	3.75 ns	8.5 MB/sec
DDR3	2007(2009)	16 GB	240	5 ns	12.8 MB/sec
DDR4	2012(?)				25.6(?) MB/sec

Other Trends: Pins and GPU Memory

Year	Brand name	# cores	Pin count	FSB or QPI, GB/sec
2000	Pentium-III	1	370	0.5
2001	Celeron	1	479	1.1
2002	Xeon 1.6	1	603	3.2
2003	Xeon 3.2	1	603	4.3
2004	Xeon 3.6	1	604	6.4
2005	Xeon 7030	2	604	6.4
2006	Xeon X5470	4	771	10.7
2007	Xeon X7350	4	604	8.5
2008	Core i7-920	4	1366	19.2
2009	Core i7-950	4	1366	19.2
2010	Core i7-970	6	1366	25.6
2011	Core i7-3960X	6	2011	25.6



Year	GPU	Shader Procs (# Cores)	PCIe Speed	GDDR Bandwidth
2003	GeForce PCX4300	2	40	5.33
2004	GeForce PCX5300	5	40	5.33
2005	GeForce 6500	7	40	5.33
2006	GeForce 7900GT	64	40	42.2
2007	GeForce 8800GT	112	80	57.6
2008	GeForce 9800GT	112	80	57.6
2009	GeForce GTX280	240	80	141.7
2010	GeForce GTX480	480	80	177.4
2011	GeForce GTX580	512	80	192.4

renci

Little's Law applied to Memory.

- Classic law/lemma in queuing theory
  - (mean # in system/queue) = (arrival rate) (mean residence time)
- Communication (memory) restatement
  - (concurrency) = (bandwidth) (latency)

→ To increase bandwidth without decreasing latency, you have to increase the concurrency of the system

  - Wider channels to send more bits per operation.
  - Overlapping, i.e., pipelined, operations.

Bottleneck → bandwidth plateaus, queuing latency dominates.

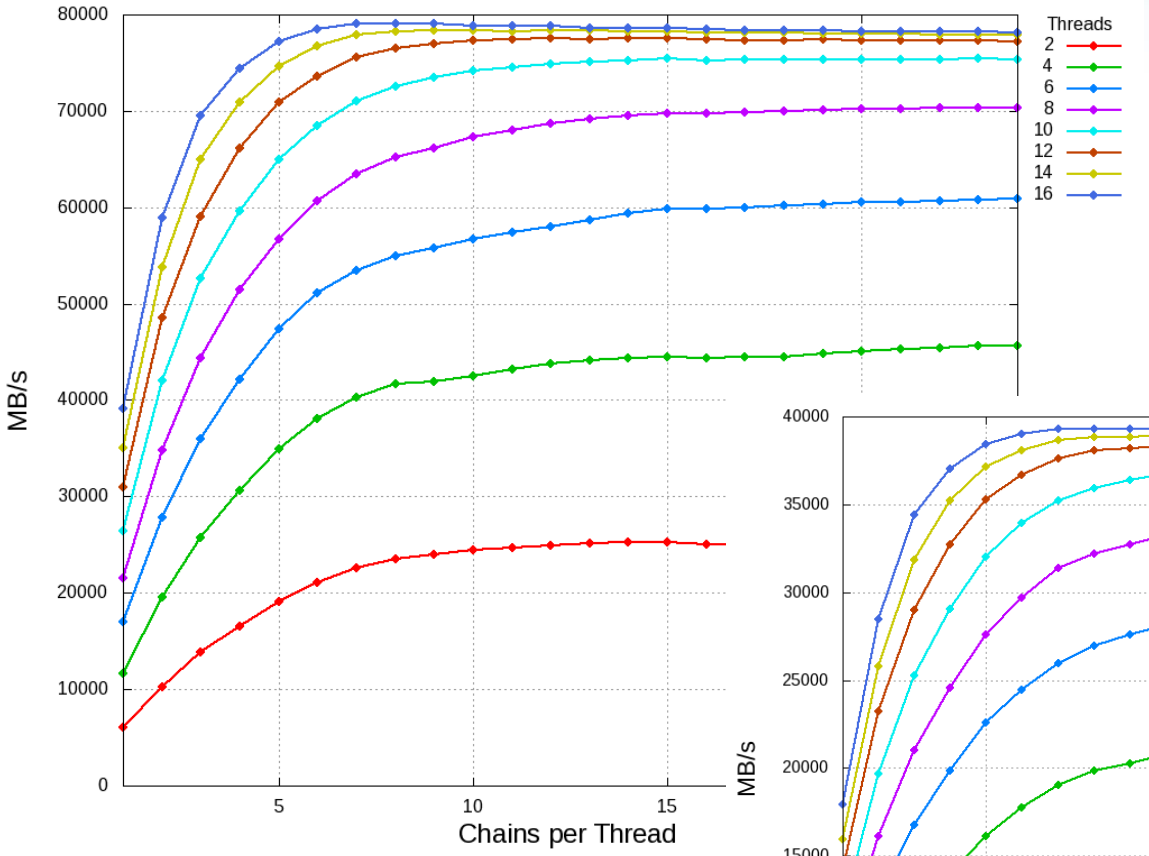
renci

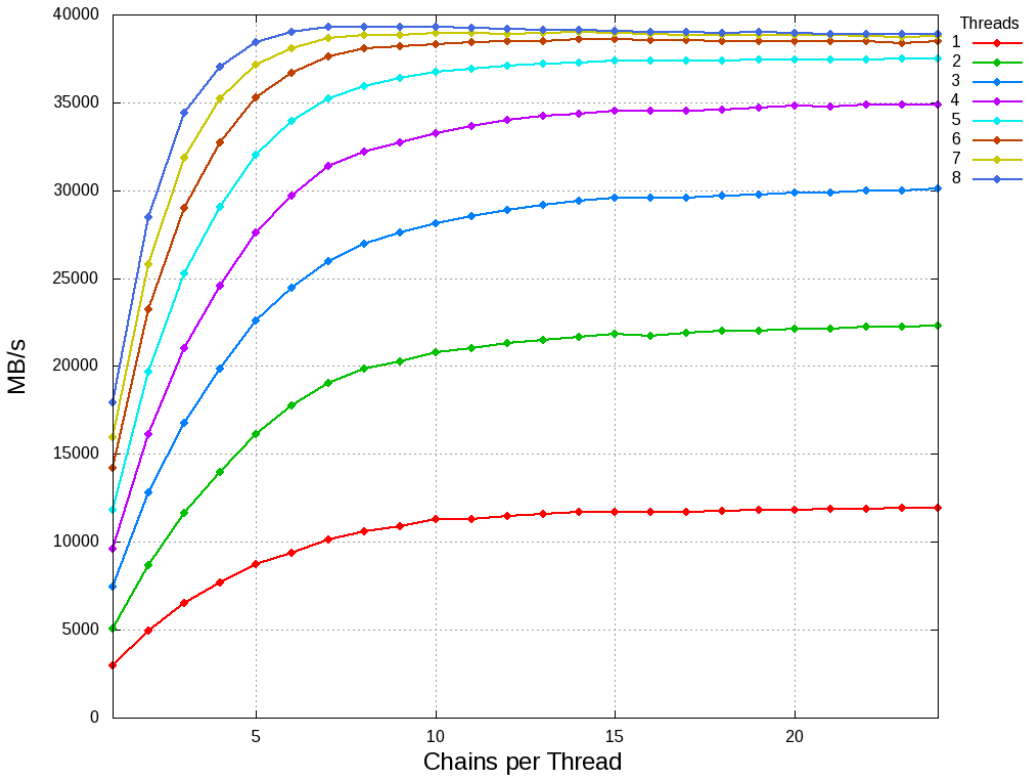
pChase

- Developed by Pase and Eckl @IBM
- Multi-threaded benchmark used to test memory throughput under carefully controlled degrees of concurrent accesses
- Each thread executes a controllable number of ‘pointer-chasing’ operations – a memory-reference chain
  - Pointer to the next memory location is stored in the current location. Grow and randomize chain to defeat cache, prefetch.
  - Dereference pointers in k independent chains concurrently, then use them.
- K=1 case measures memory latency.
- Large-k bandwidths are comparable to STREAM measurements at “common” optimization levels.
- Our Modifications
  - Added wrapper scripts around pChase to iterate over different numbers of memory reference chains and threads
  - Added affinity code to control thread and data placement
- Available at http://pchase.org

renci

E5-2680 in “maxperf” mode.



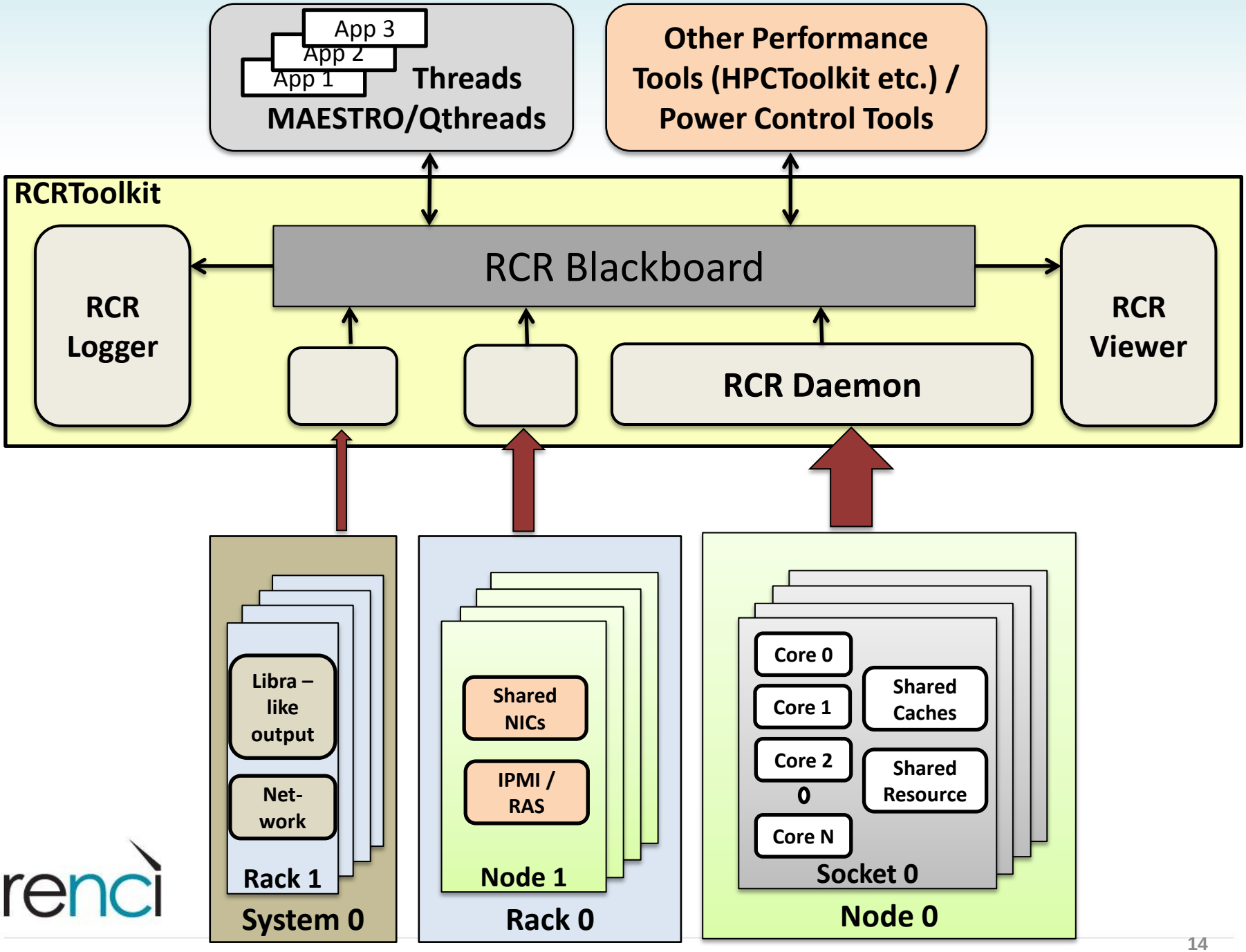


renci

Resource Centric Performance Reflection: RCRToolkit

- Performance measurement and analysis tool that focuses on shared resources in a system
  - Information and analysis should help applications and system code introspectively, in real-time, to adapt to bottlenecks, power, thermal events.
- RCRToolkit consists of
  - RCRblackboard
  - Several clients for the RCRblackboard
- RCRblackboard
  - Shared memory region (or, currently Google protocol buffers resident in memory) for real time use by producers and consumers of node- and system-wide performance information
  - Information organized in a hierarchy that reflects hardware structure
  - Coordination managed by the RCRblackboard protocol – multiple regions, each owned by a single writer.

renci


App 3
App 2
App 1
Threads
MAESTRO/Qthreads
Other Performance Tools (HPCToolkit etc.) / Power Control Tools
RCRToolkit
RCR Logger
RCR Blackboard
RCR Viewer
RCR Daemon
Libra – like output
Net-work
Rack 1
System 0
Shared NICs
IPMI / RAS
Node 1
Rack 0
Core 0
Core 1
Core 2
0
Core N
Shared Caches
Shared Resource
Socket 0
Node 0

Core RCRblackboard Clients

- RCRdaemon
  - Uses Linux MSR driver in conjunction with configuration information accessible through libpfm4 (and other sources) to access off-core HPM counters.
  - Off-core counters in PCI CONFIG space coming soon.
  - Can co-exist with tools using Linux PerfEvents using on-core counters.
  - Configuration file specifies which HPM events to monitor, as well as a set of derived measures (“meters”) computable with simple arithmetic.
- RCRlogger and RCRviewer
  - Read RCRblackboard information and output a log for post-execution analysis
  - On-line monitoring.
- MAESTRO thread scheduler
  - Adaptive, locality-aware scheduling of over-partitioned applications, e.g., OpenMP tasks, loops with guided self-scheduling.
  - Monitors shared resource usage and adapts scheduling during periods of high utilization
  - Writes scheduling decision information to RCRblackboard, including some source attribution at the level of OpenMP loops and tasks.

Hot-wiring HPCToolkit to leverage RCRToolkit

- Experimental extension to HPCToolkit to act as an RCRblackboard client
- Goal: combine third-person, system-wide performance measures with “first-person” call stack profiling.
- Accomplished by
  - Command line extension to specify events to split based on RCR predicates.
  - Modifying HPCToolkit’s event sampling code to examine RCRblackboard for existence of shared resource bottlenecks.
  - Monitoring whether the sampled event occurs when shared resource utilization exceeds threshold set in RCRdaemon configuration
- Original HPCToolkit event is split into two sub-events that can be viewed using an unmodified HPCViewer

renci

Hot-wiring HPCToolkit (2)

- RCR augmented HPCToolkit metrics
  - Extended command line argument to “hpcrun” command
  - “hpcrun” event specification string was extended by adding a suffix string, delimited by # and ^, which corresponds to a RCRToolkit derived system-wide metric with a boolean value
  - Example: If PAPI_L2_TCM is the base event and is passed as an extended specification with hpcrun, and if the RCRToolkit derived metric is a threshold variable corresponding to full utilization of a memory channel, we will have two HPCToolkit metrics – one for normal PAPI_L2_TCM events and other for PAPI_L2_TCM events that occurred during memory contention
  - Currently, user has a dictionary of mapping between contentious events and shared memory locations. (Symbolic specification via GPB “real soon now”)
  - Augmented metric appears with a “RCR-” prefix in HPCViewer, for example RCR-PAPI_L2_TCM
  - Could go beyond simple predicates. Multiple sub-events? Integrate RCR metrics rather than just a simple histogram?

renci

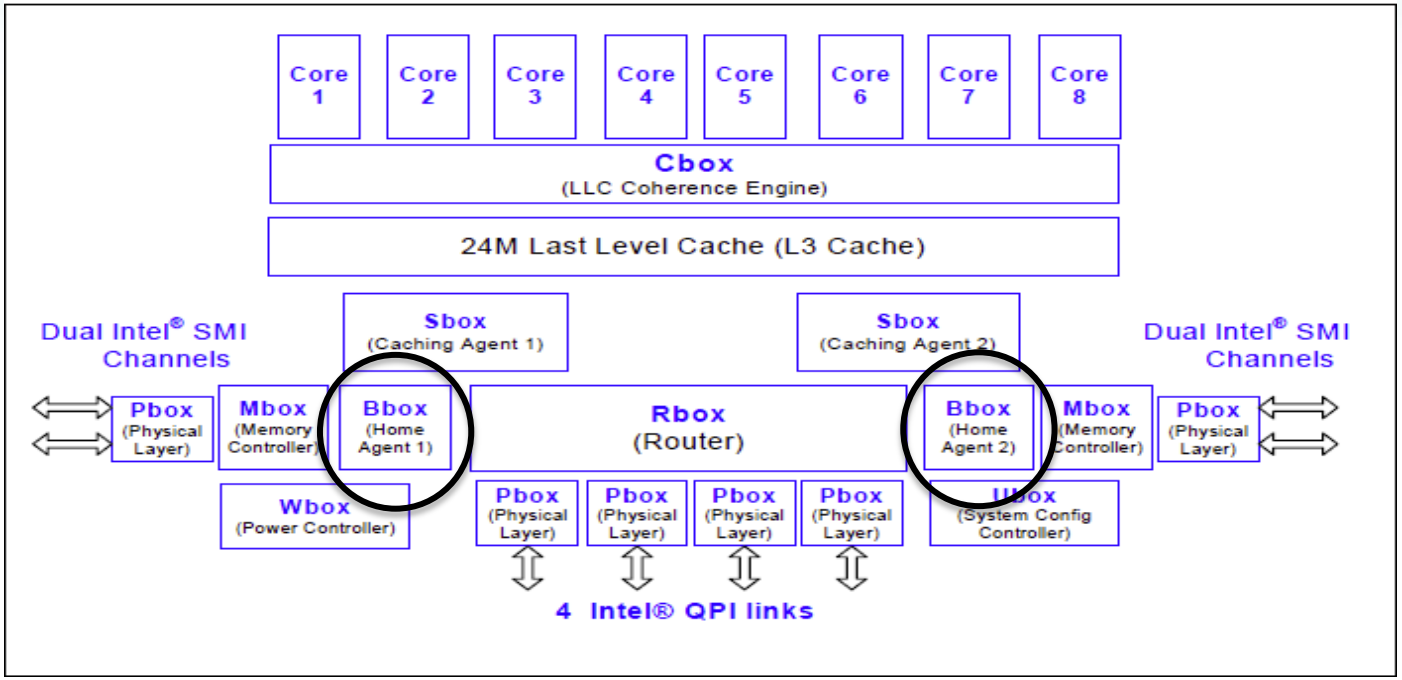
Examples

- Analyzed memory performance of three applications/benchmarks
  - Memory performance: significant source of bottlenecks on multi-core systems
  - Lattice QCD “chroma”, Lattice Boltzmann Magneto HydroDynamics (LBMHD), and Barcelona OpenMP FFT
- Each run is simultaneously measured by
  - RCRToolkit
  - Modified HPCToolkit hot-wired with RCRToolkit
- Two systems
  - Dell PowerEdge M910 test system (MMQ)
  - Two socket E5-2680 system

renci

Nehalem EX Uncore.

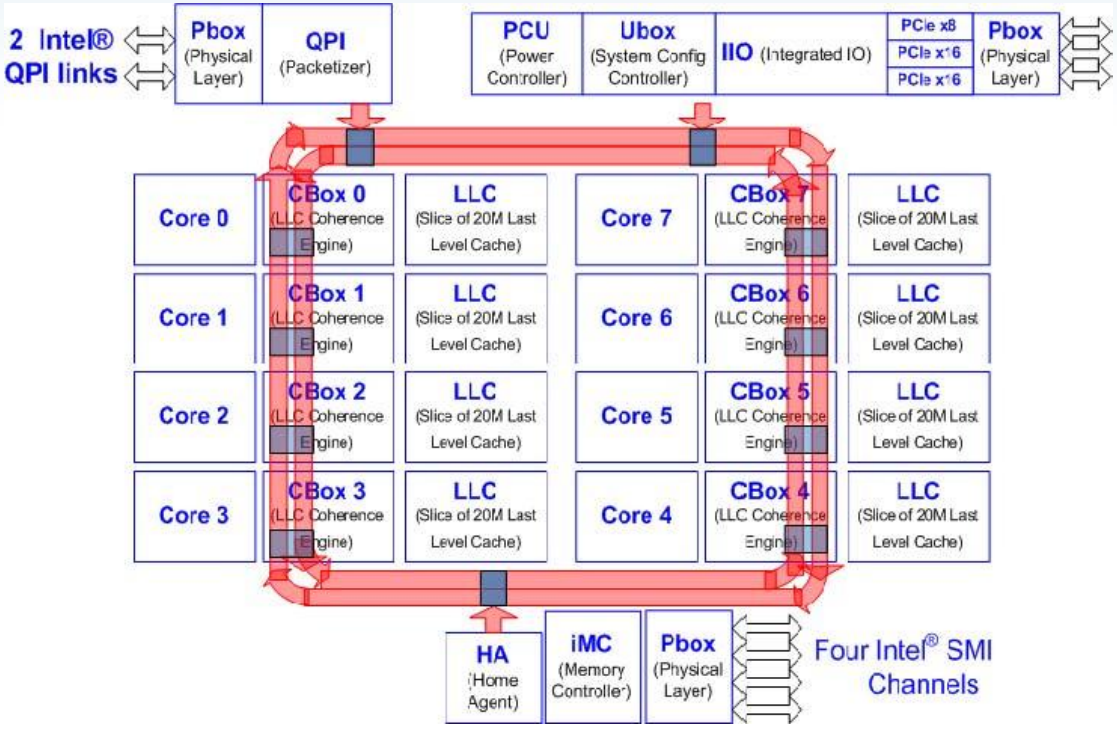
Figure 2-2. Intel® Xeon® Processor 7500 Series Block Diagram



- B-box has an In Memory Table (IMT) that tracks all in-flight memory block operations and ensures that they are all unique
- Uncore counter, IMT_VALID_OCCUPANCY tracks number of unique entries in the IMT == number of concurrent memory operations in flight
- RCRdaemon calculates IMT Average Occupancy = (count * 32 /cycles)

renci

Sandy Bridge E5-26xx



- Rich collection of uncore boxes.
- LLC is split into 8 pieces, each behind a CBox
- PMUs for CBoxes, Ubox, and PCU are accessible through MSR space.
- All other PMUs are in the PCI CONFIG space.
- LLC/Memory requests are queued in a per CBox TOR.
- We measure energy (power) consumption and TOR occupancy.

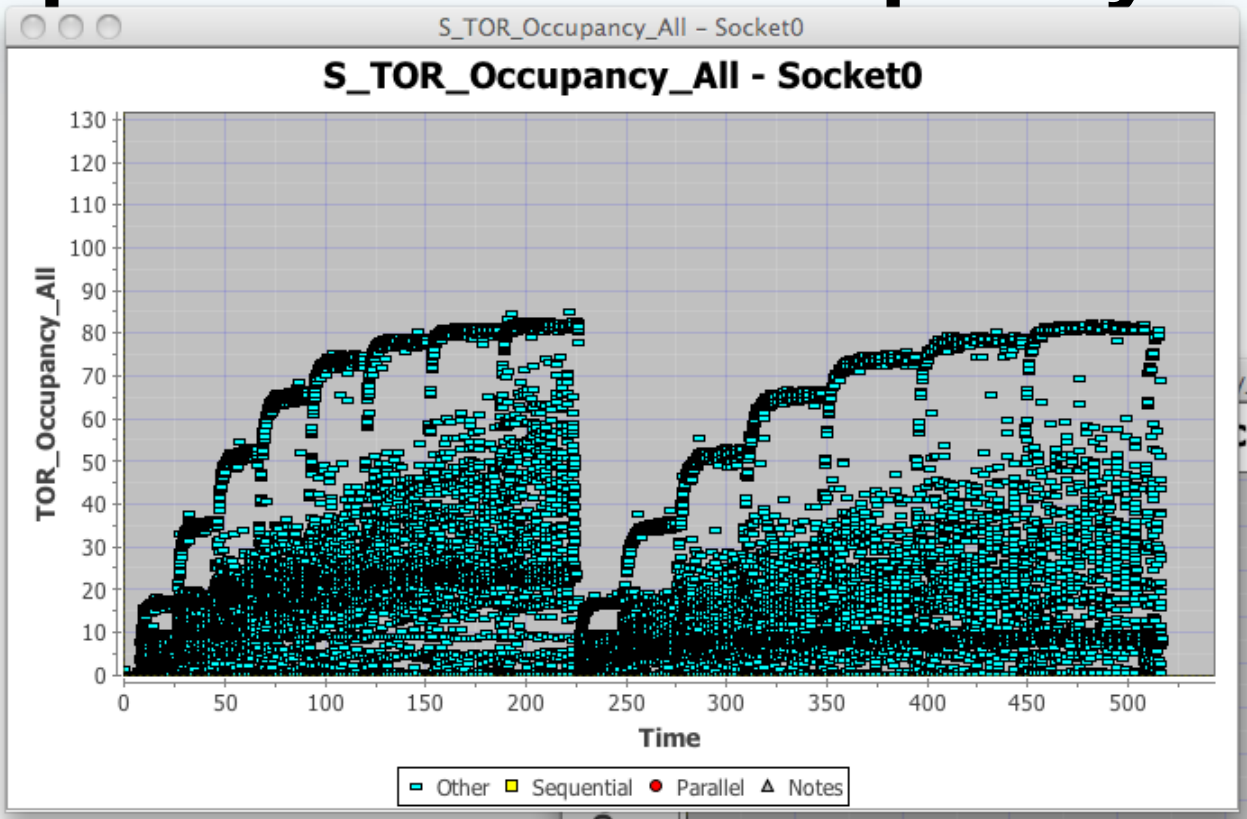
renci

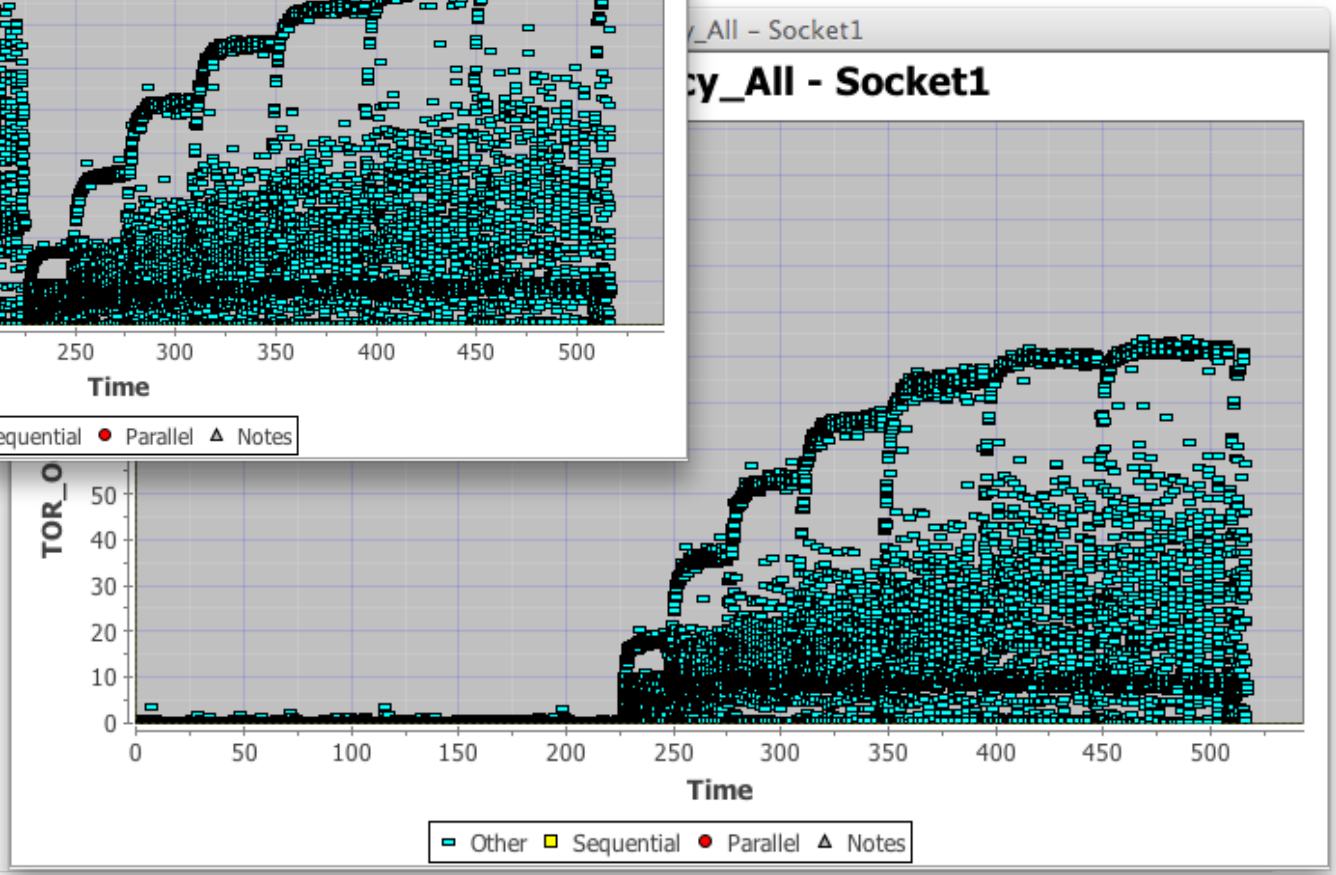
Validation and threshold selection.

- Used the pChase benchmark code to generate controlled level of concurrent memory operations
- Bandwidth is observed to increase with load up to a practical threshold after which requests are queued
- Using pChase, determined that memory contention becomes severe when IMT Average Occupancy reaches 23, and saturates at about 27
- Chose 23 as a threshold defining a memory controller bottleneck.

renci

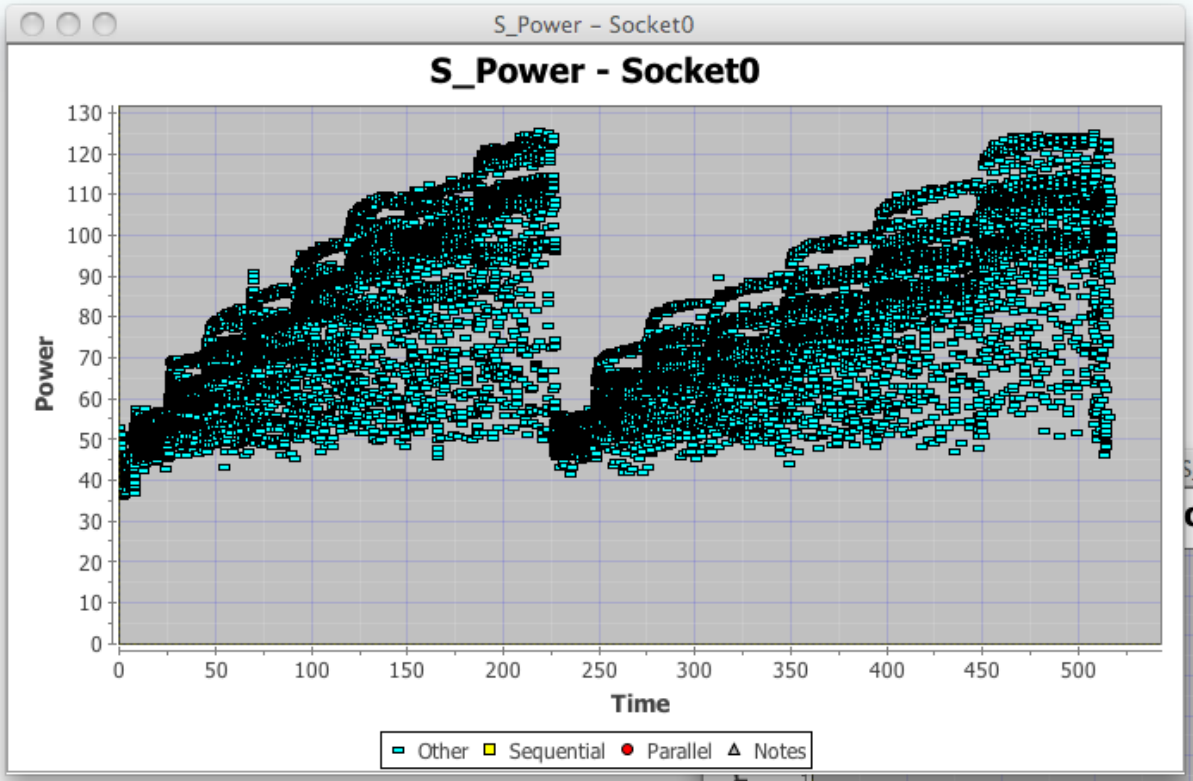
pChase TOR Occupancy on SB

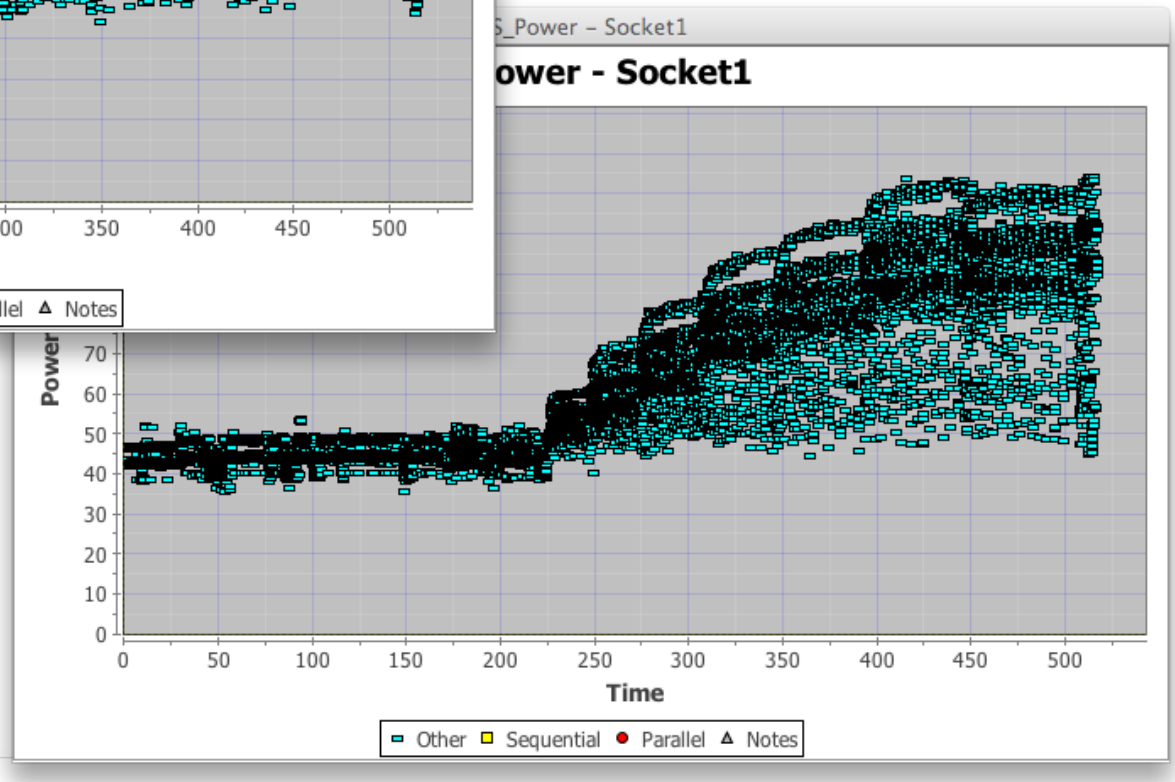




renci

pChase Power on Sandy Bridge 2600





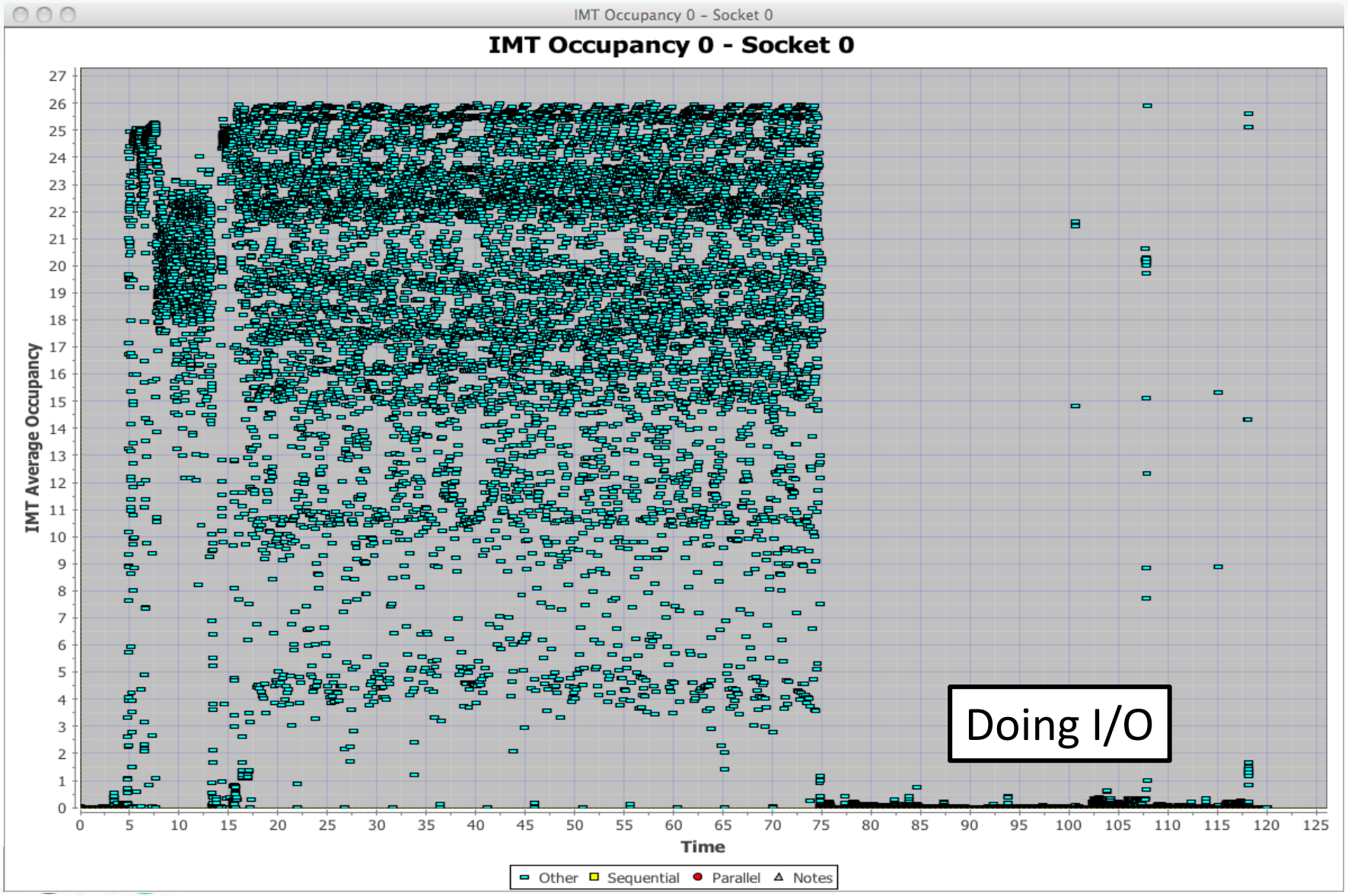
renci

Evaluation: Lattice QCD “chroma” Application

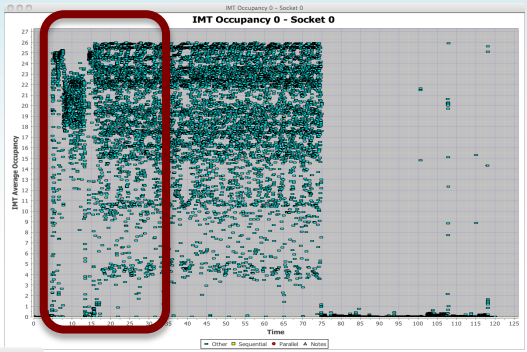
- Open-source C++ based software system from US SciDAC QCD initiative
- “chroma” is based on the QDP++ library that implements data-parallel programming constructs for lattice field theory
- Application runs used unmodified MPI build of Chroma version 3.7.3 and QDP++ version 1.35.1
  - Used ‘clover’ sample input from the FermiLab QCD benchmark suite
- Ran on all 32 cores of MMQ
  - For runs using HPCToolkit, RCRdaemon was running with a threshold for Average IMT Occupancy set to 23

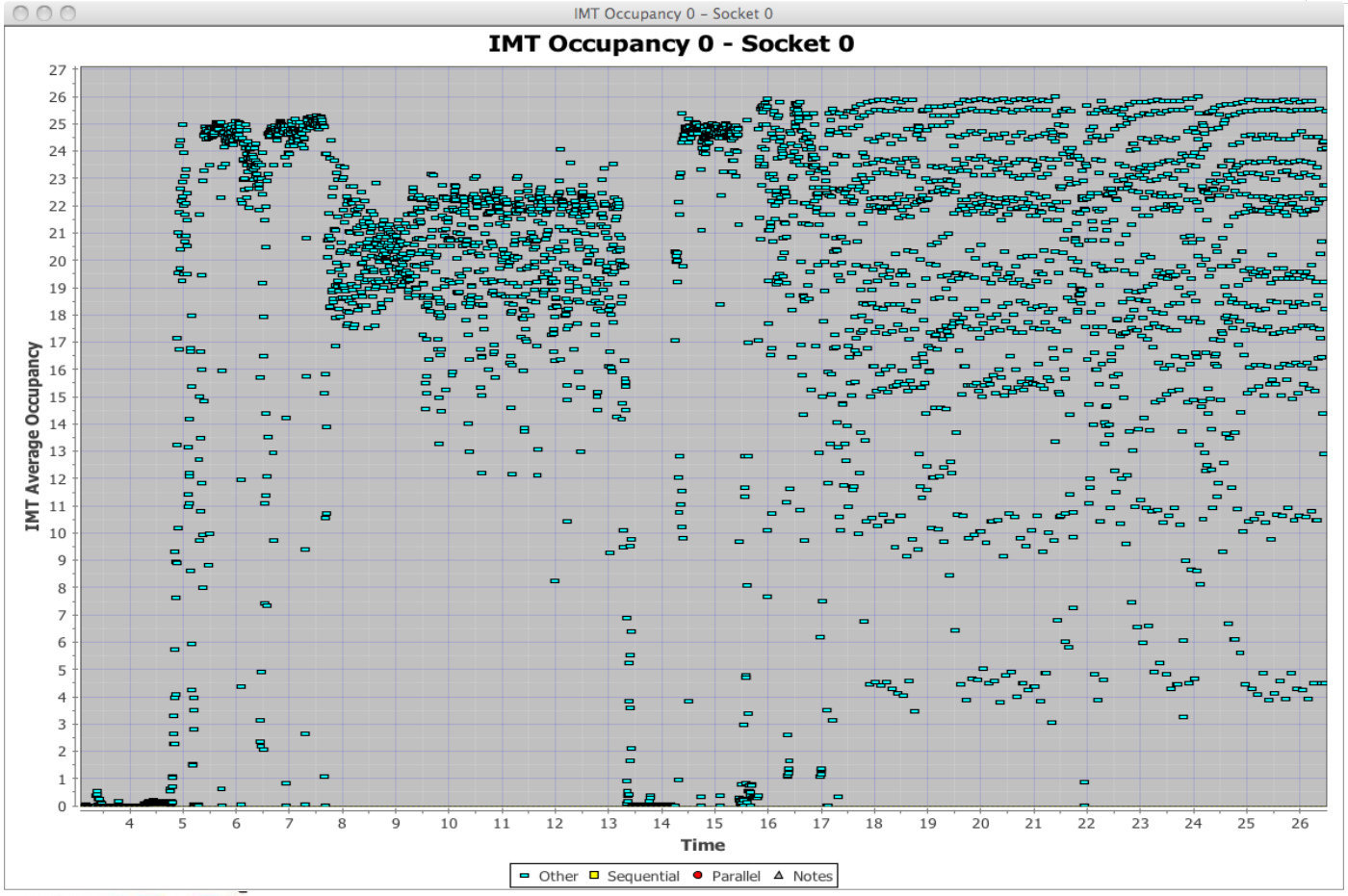
renci

“chroma” Results: RCRToolkit



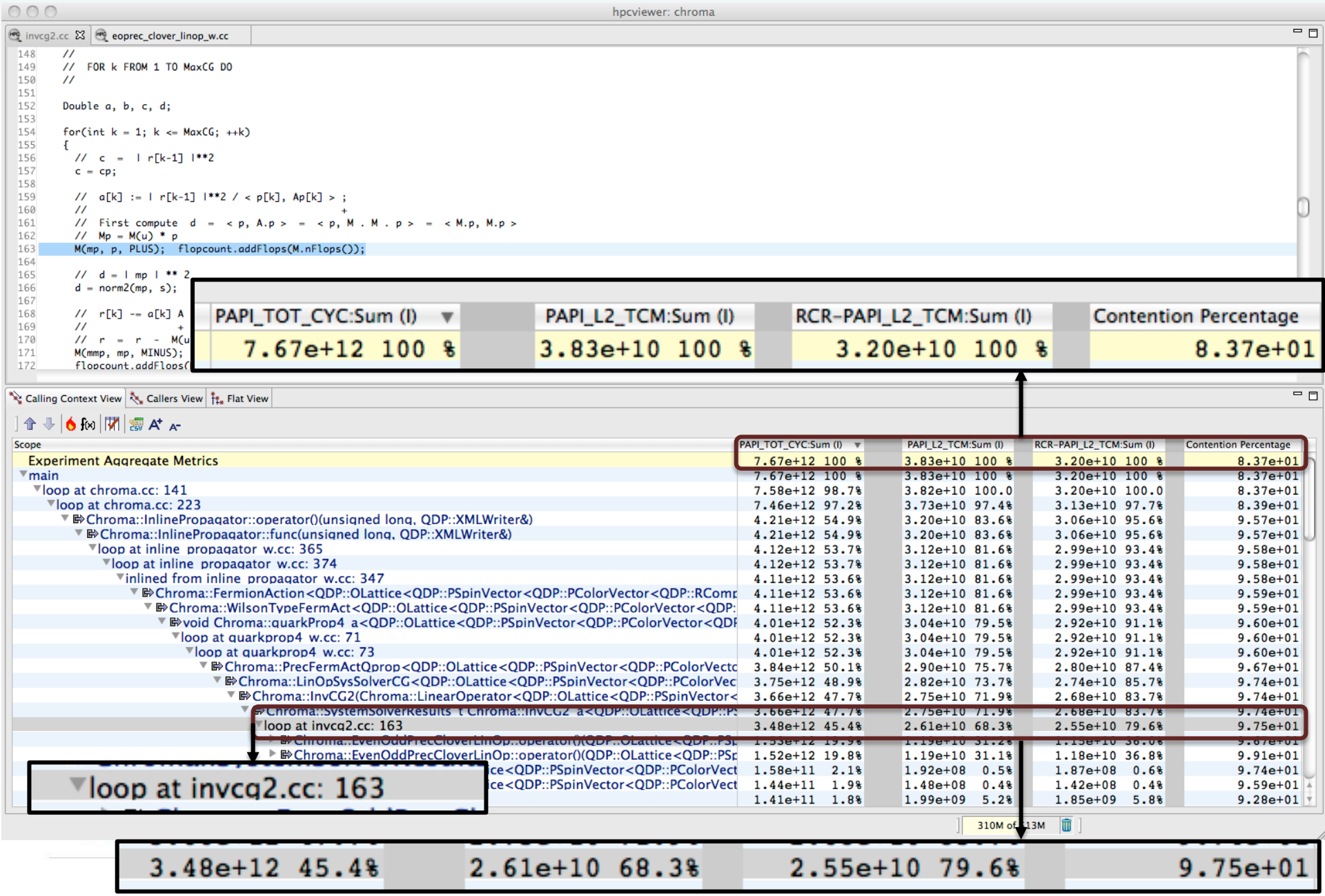
“chroma” Results: RCRToolkit



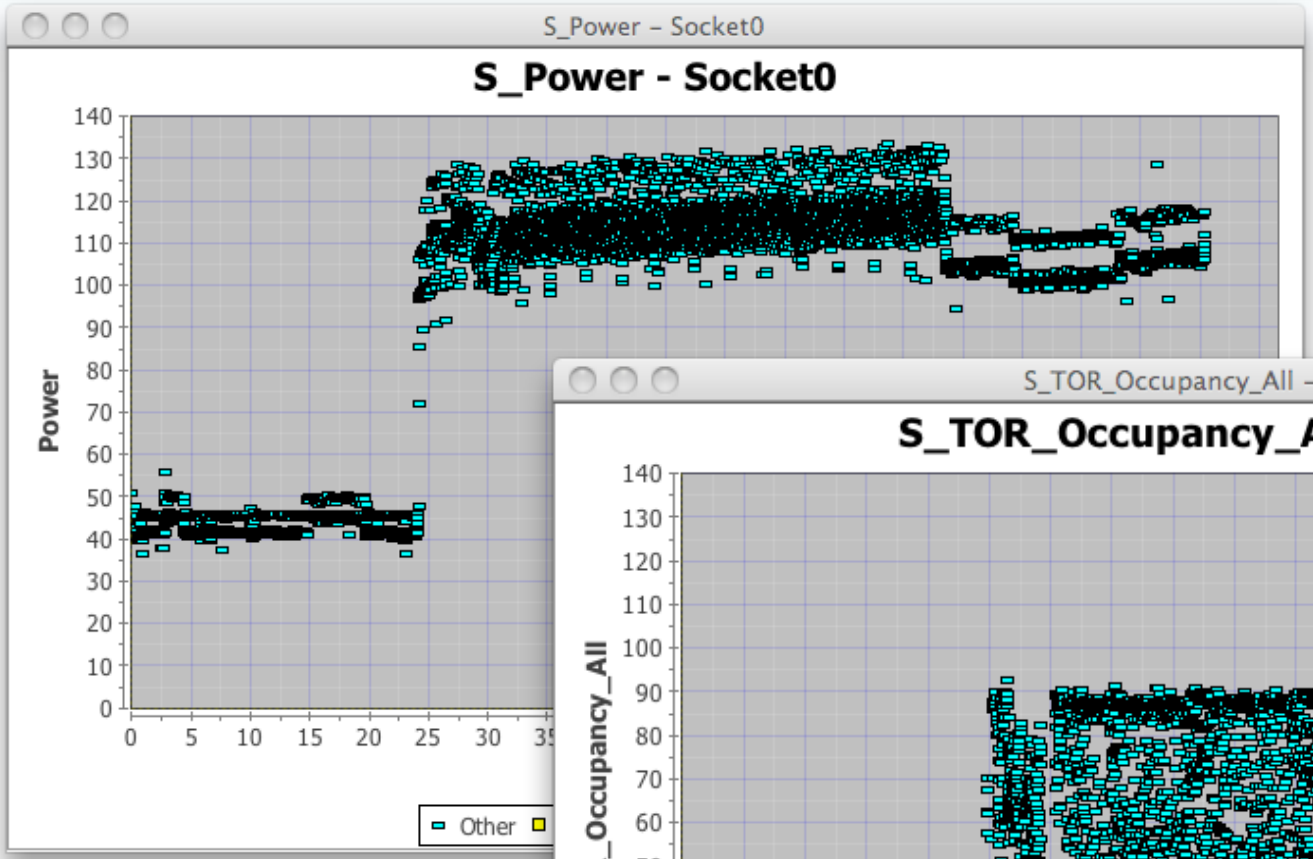


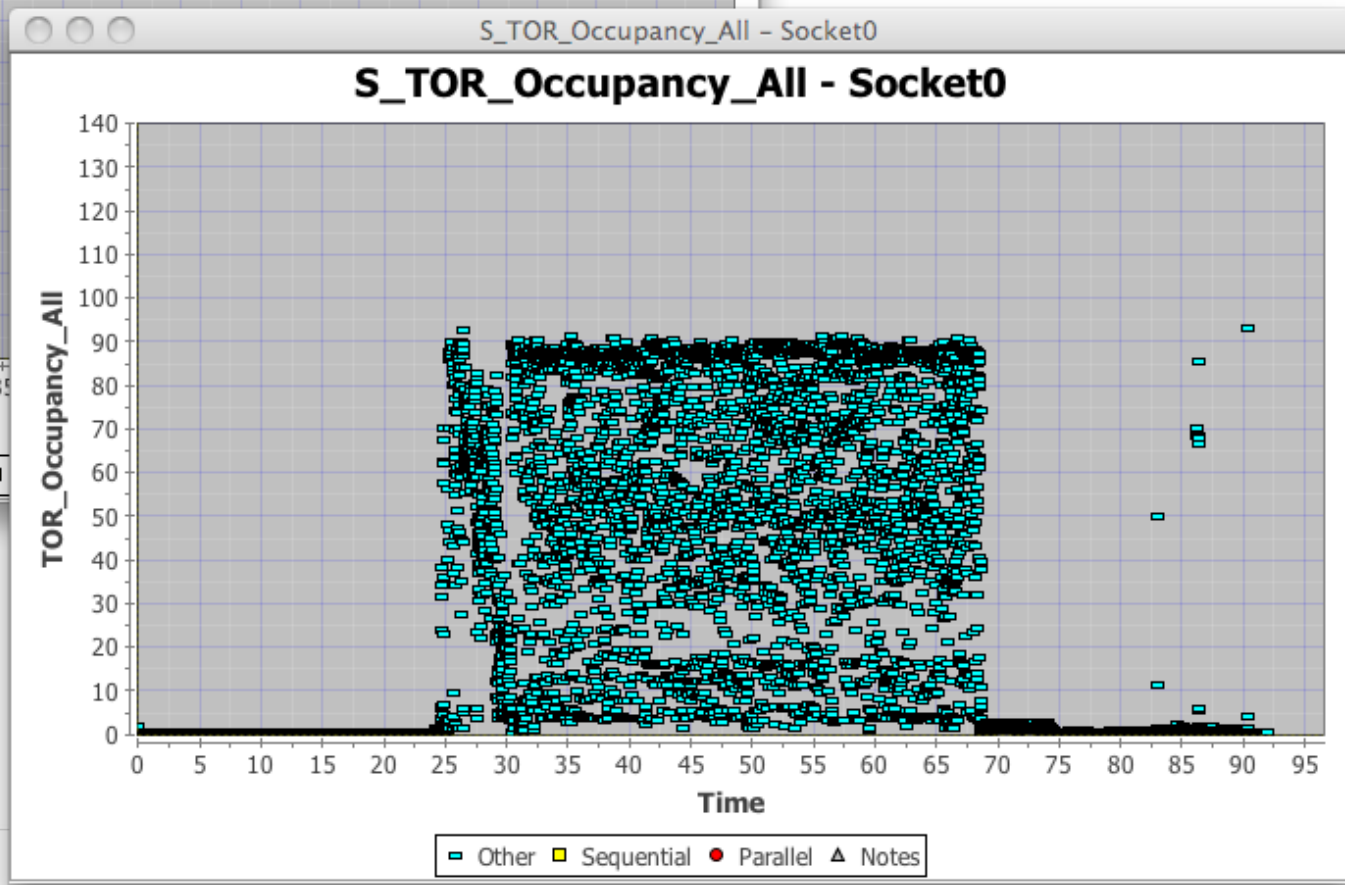
renci

“chroma”: HPCToolkit hot-wired with RCRToolkit



Chroma Power and L3 queue on SB.





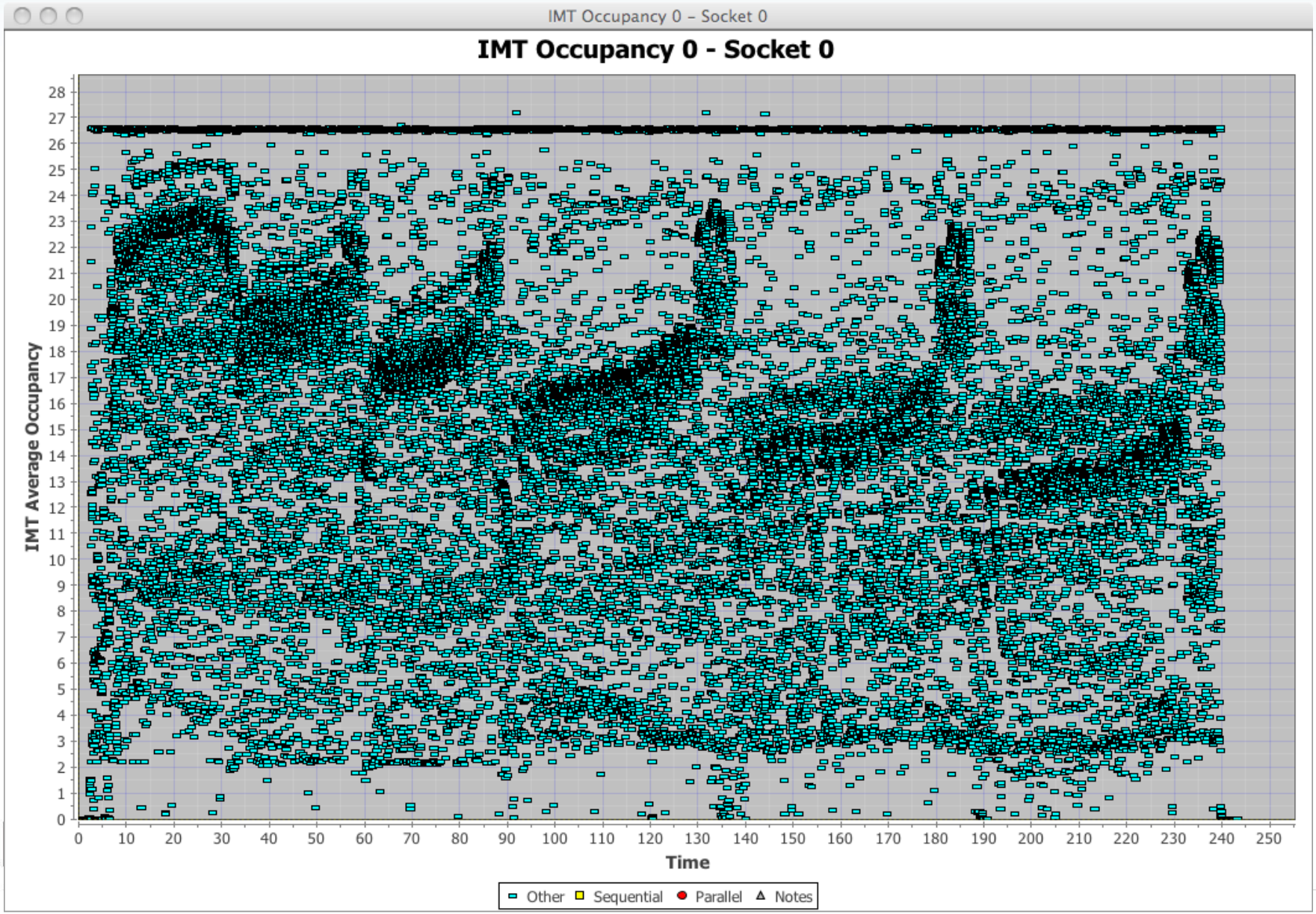
renci

Evaluation: LBMHD Benchmark

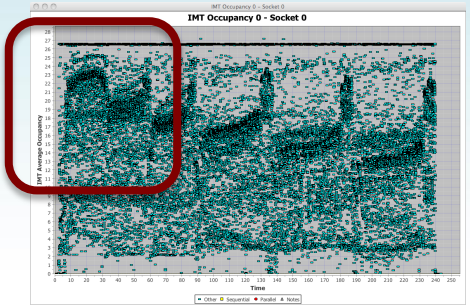
- The LBMHD benchmark models homogeneous isotropic turbulence in dissipative magneto-hydrodynamics
- LBMHD code was obtained from S. Williams at Lawrence Berkeley National Lab
- Ran on all 32 cores of MMQ
  - For runs using HPCToolkit, RCRdaemon was running with a threshold for Average IMT Occupancy set to 23

renci

LBMHD on Nehalem EX: RCRToolkit

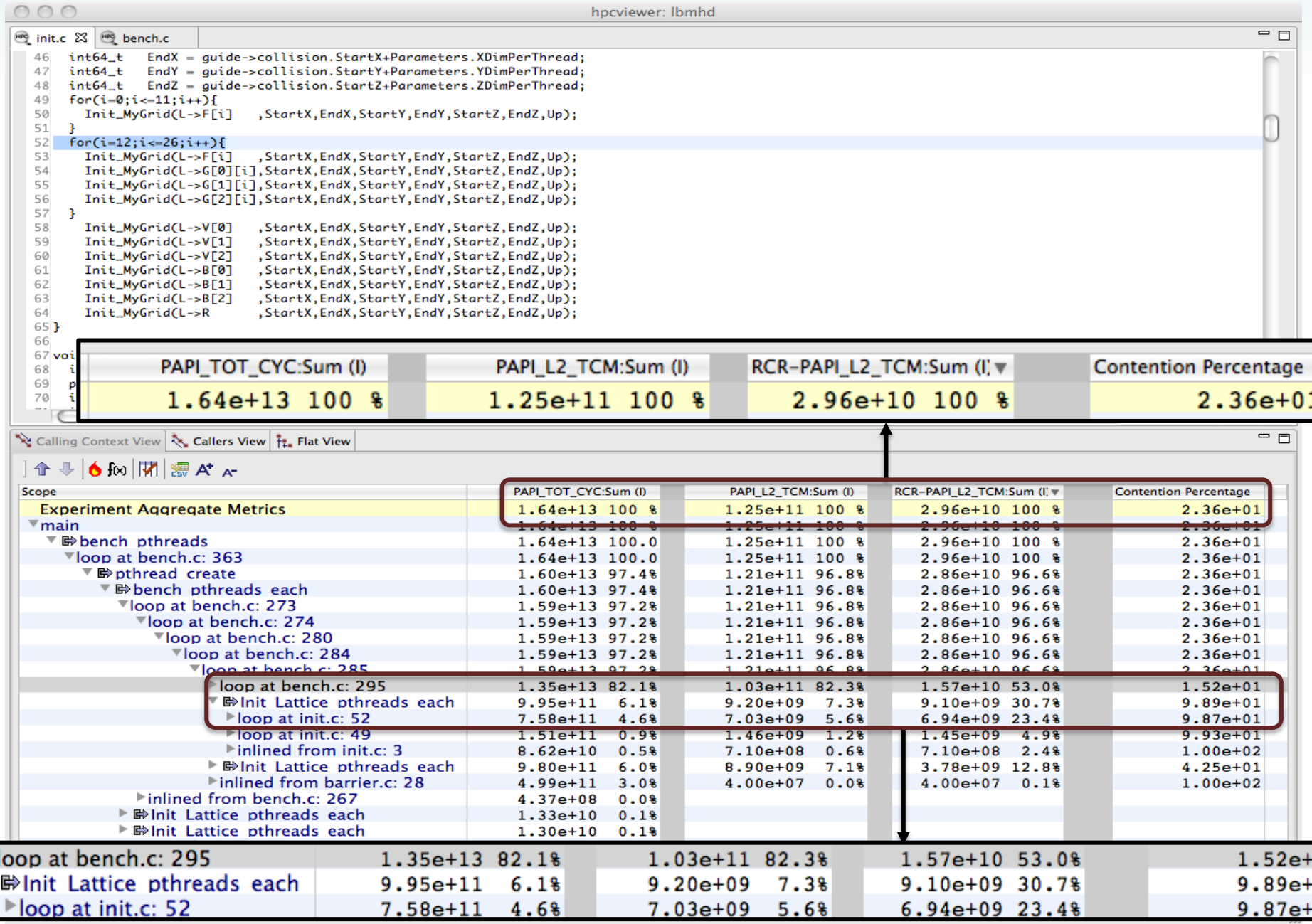


LBMHD Results: RCRToolkit



IMT Occupancy 0 – Socket 0

LBMHD: HPCToolkit hot-wired with RCRToolkit

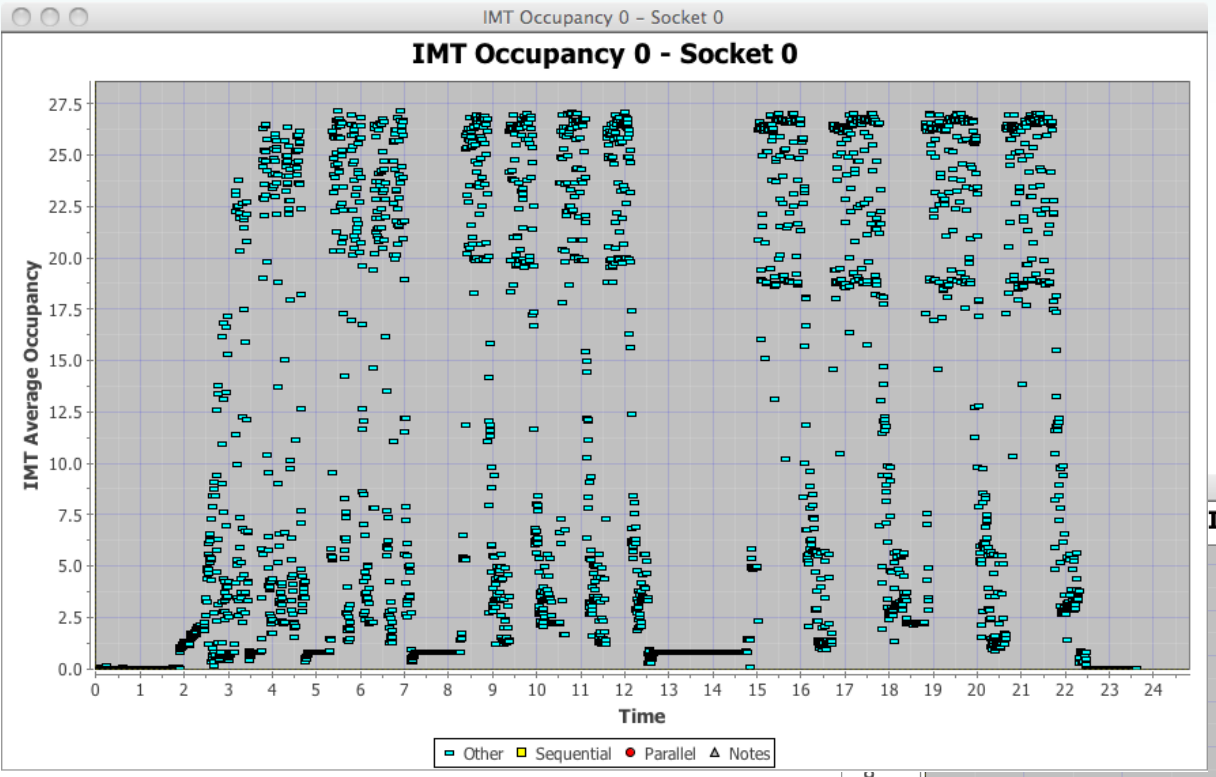


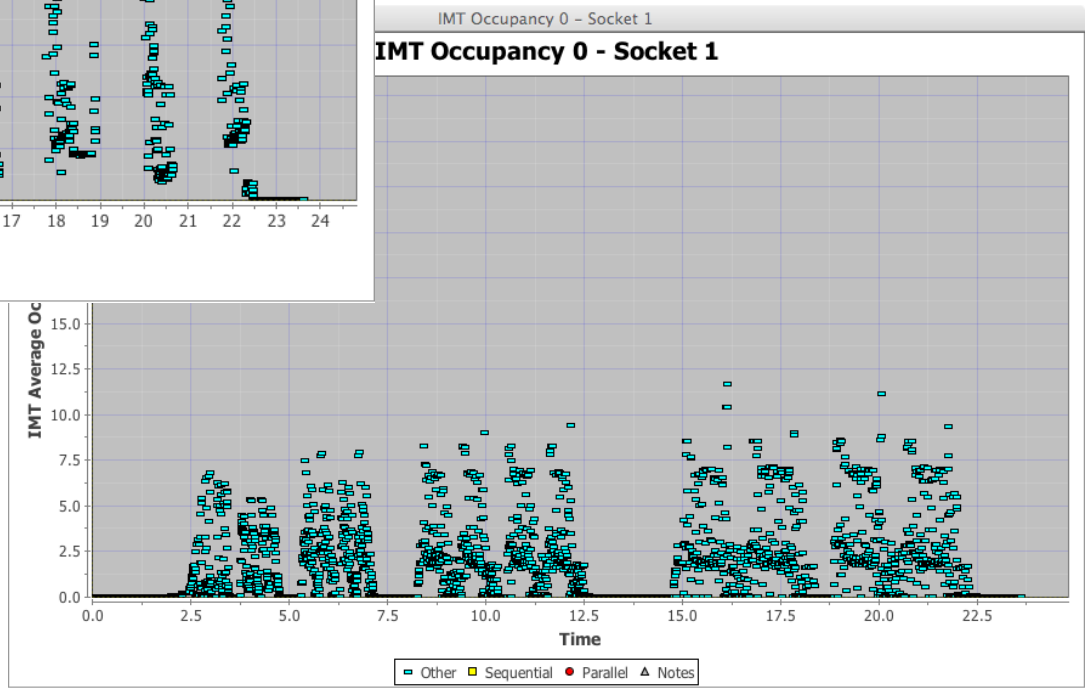
Evaluation: FFT Benchmark

- Benchmark from the Barcelona OpenMP Tasks Suite (BOTS)
- FFT computes 1-dimensional FFT of a vector of n complex values using the Cooley-Turkey algorithm
  - Recursive algorithm that divides a DFT into smaller DFTs
  - Each division generates multiple OpenMP tasks
  - Version ran was compiled with ROSE source-to-source compiler and executed with the Qthreads runtime
- Ran on all 32 cores of MMQ
  - For runs using RCRToolkit, RCRdaemon was running with a threshold for Average IMT Occupancy set to 23

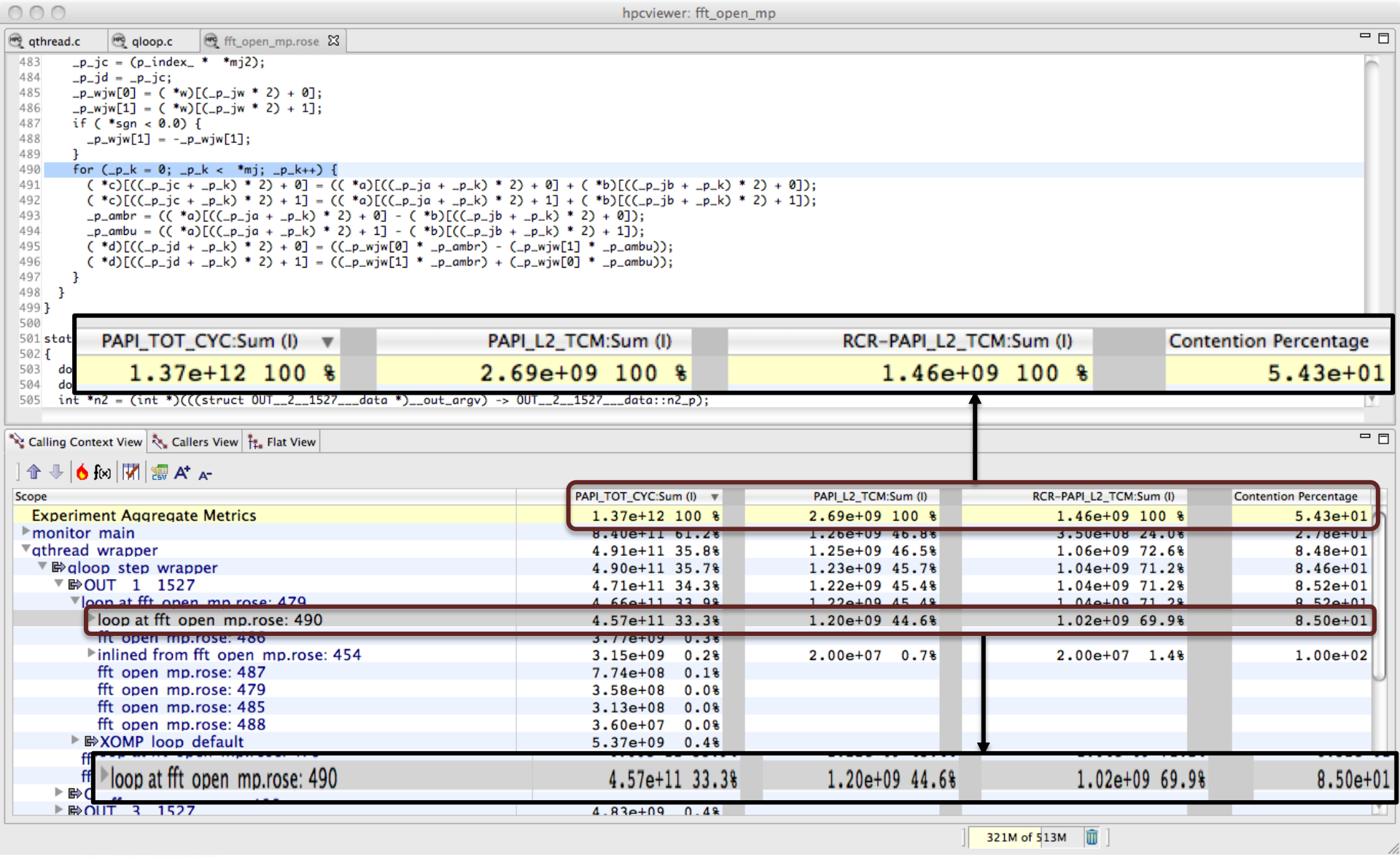
renci

FFT on Nehalem EX: RCRToolkit

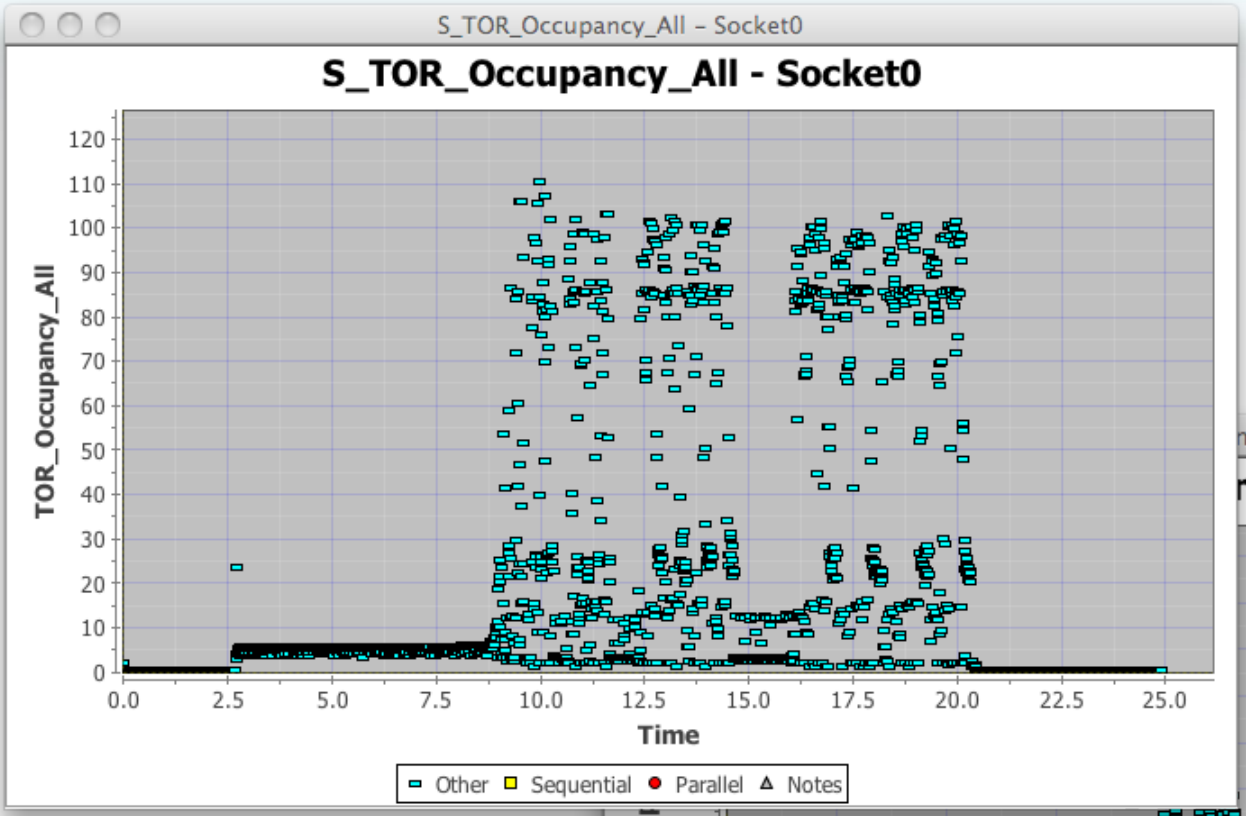




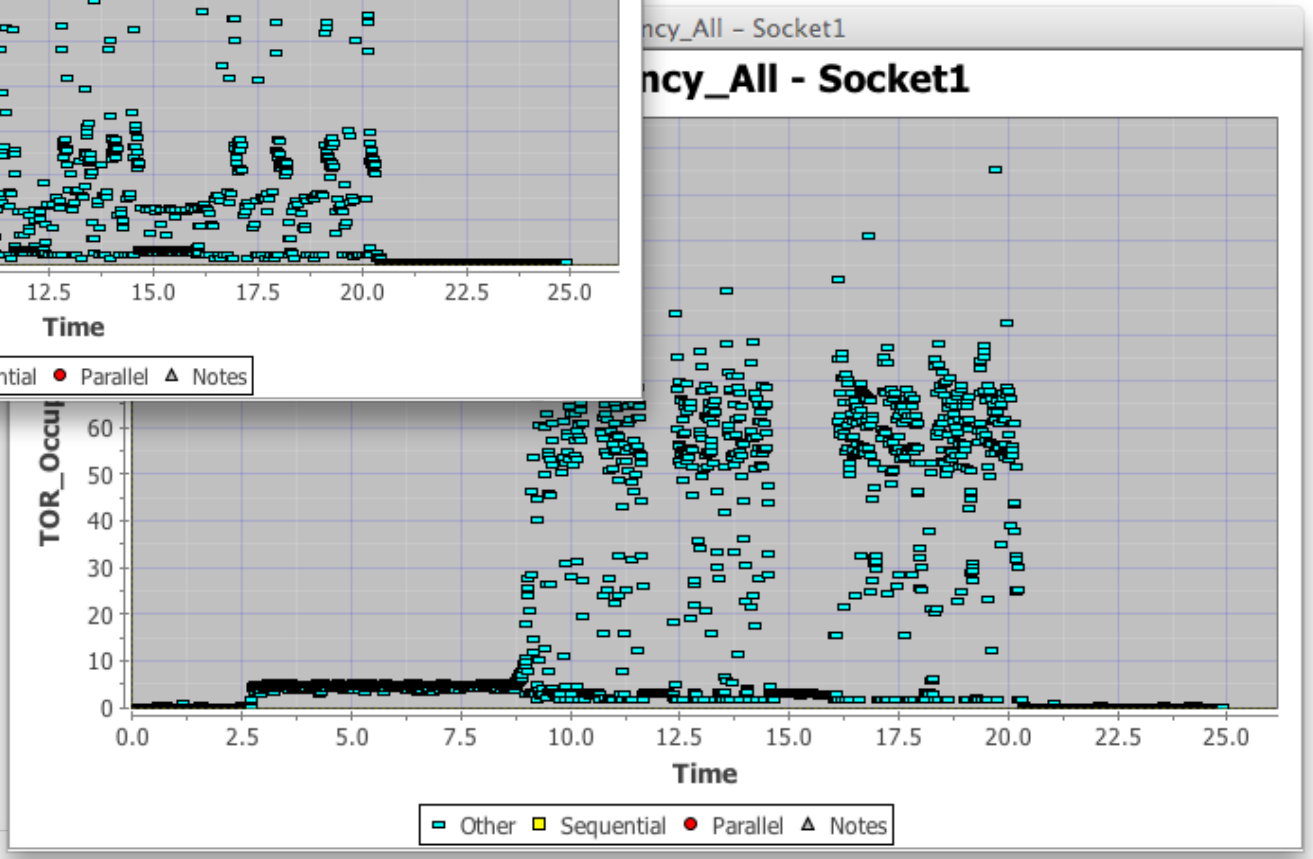
FFT: HPCToolkit hot-wired with RCRToolkit



FFT TOR Occupancy on SB

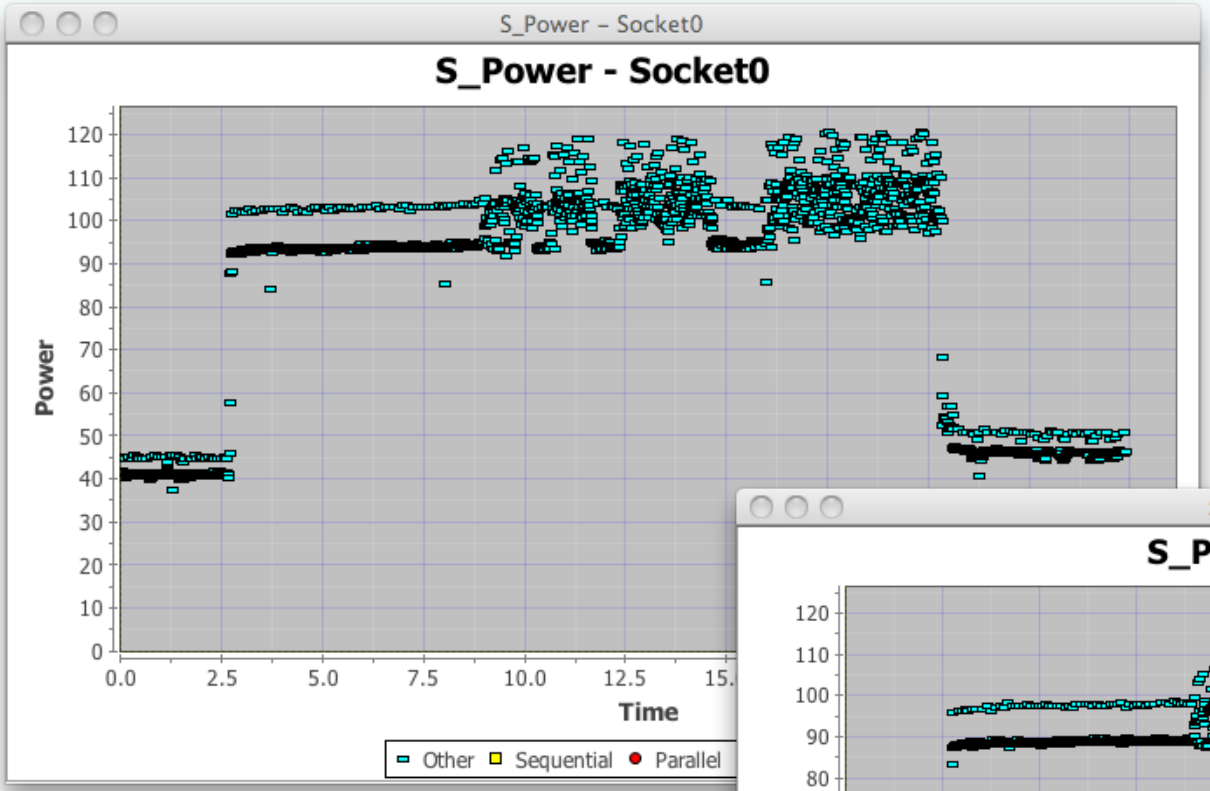


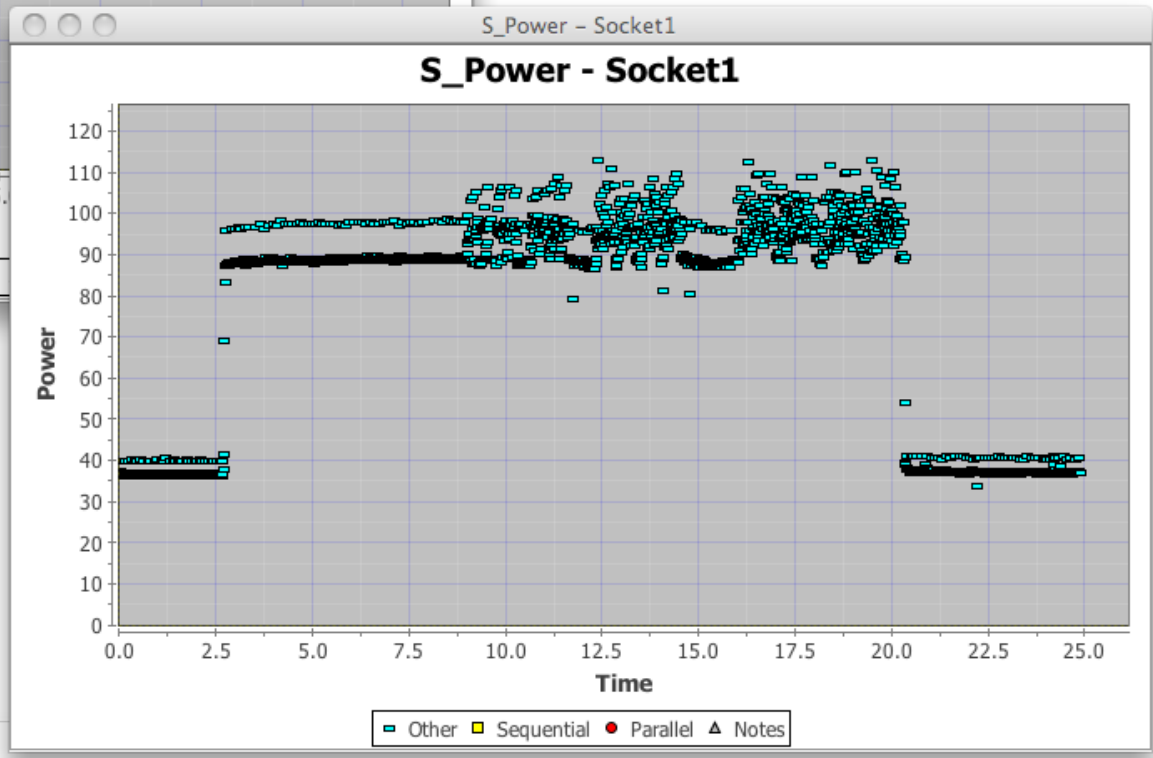
We know, from the MMQ results, that data is poorly balanced between memories. Why is there an imbalance at the TOR?



renci

FFT Power on SB





renci