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

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



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.



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 S², C_{qate} 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.



Other Aspects of Dennard Scaling.

- Wire resistance/unit length ~ S^2
- Wire capacitance/unit length ~ 1
- RC delay/unit length (unrepeated) ~ S²
- Die size (D) increases, so "long" wires increase by D
- Unrepeated wire delay ~ S²D², repeated ~ D sqrt(S)
 →Signals cannot cross the chip in one cycle.



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 \rightarrow 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 \rightarrow 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.

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.



"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.



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



0.000

2002

2004

2005

2008

2010

0.000

2012

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 \rightarrow bandwidth plateaus, queuing latency dominates.



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



E5-2680 in "maxperf" mode.



12

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 systemwide 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.





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



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?



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



Nehalem EX Uncore.



- 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.

renc

- 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.

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.



pChase TOR Occupancy on SB



pChase Power on Sandy Bridge 2600



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



"chroma" Results: RCRToolkit



"chroma" Results: RCRToolkit





"chroma": HPCToolkit hot-wired with RCRToolkit

000	hpcviewer: chroma				
פַ invcg2.cc מו פּ eoprec_clover_linop_w.cc					- 0
148 // 149 // FOR k FROM 1 TO MaxCG DO 150 //					
151 152 Double a, b, c, d;					
<pre>154 for(int k = 1; k <= MaxCG; ++k) 155</pre>					
156 // c = r[k-1] **2					
c = cp, 158 159 // o[k] -= n[k-1] **2 / < n[k]. An[k] > 1					
$\begin{array}{cccccccccccccccccccccccccccccccccccc$					0
$162 // Mp = M(u) * p$ $163 M(mp, n, P[U]S): flopcount_addFlops(M, nFlops()):$					
164 165 // d = 1 mp 1 ** 2					
166 d = norm2(mp, s);					
168 // r[k] a[k] A 169 // r[k] a[k] A + PAPI_TOT_CYC:Sum (I) ▼ P	API_L2_TCM:Sum (I)	RCR-PAP	I_L2_TCM:Sum (l) Contentio	on Percentage
170 // r - r - M(u) 7.67e+12 100 % 3 171 M(mmp, mp, MINUS); 7.67e+12 100 % 3	.83e+10 100 %	ł 3.2	0e+10 100	£	8.37e+01
172 flopcount.addFlops(-	
😵 Calling Context View 🗞 Callers View 🕂 Flat View					- 8
] 🕆 🖖 💧 fw 🕅 📰 A* 👦					
Scope	P	API_TOT_CYC:Sum (I) V	PAPI_L2_TCM:Sum (I)	RCR-PAPI_L2_TCM:Sum (I)	Contention Percentage
Experiment Aggregate Metrics	L.	7.67e+12 100 %	3.83e+10 100 %	3.20e+10 100 %	8.37e+01
Vloop at chroma.cc: 141		7.58e+12 98.7%	3.82e+10 100.0	3.20e+10 100.0	8.37e+01
Vloop at chroma.cc: 223		7.46e+12 97.2%	3.73e+10 97.4%	3.13e+10 97.7%	8.39e+01
Chroma::InlinePropagator::operator()(unsigned long, QDP::XMLWriter&)		4.21e+12 54.9%	3.20e+10 83.6%	3.06e+10 95.6%	9.57e+01
Chroma::InlinePropagator::func(unsigned long, QDP::XMLWriter&)		4.21e+12 54.9%	3.20e+10 83.6%	3.06e+10 95.6%	9.57e+01
loop at inline propagator w.cc: 365		4.12e+12 53.7%	3.12e+10 81.6%	2.99e+10 93.4%	9.58e+01
loop at inline propagator w.cc: 3/4		4.12e+12 53.7%	3.12e+10 81.6%	2.99e+10 93.4%	9.58e+01
Inlined from inline propagator w.cc. 347 Indined from Action A	P-PColorVector_ODP-PCom	4.110+12 53.6%	3.120+10 81.6%	2.99e+10 93.48	9.58e+01
B Chroma: WilsonTypeFermAct <odp: olattice<odp::pspinvector<qdf<="" td=""><td>or<odp: pcolorvector<odp:<="" td=""><td>4.11e+12 53.6%</td><td>3.12e+10 81.6%</td><td>2.99e+10 93.4%</td><td>9.59e+01</td></odp:></td></odp:>	or <odp: pcolorvector<odp:<="" td=""><td>4.11e+12 53.6%</td><td>3.12e+10 81.6%</td><td>2.99e+10 93.4%</td><td>9.59e+01</td></odp:>	4.11e+12 53.6%	3.12e+10 81.6%	2.99e+10 93.4%	9.59e+01
▼	tor <qdp::pcolorvector<qdf< td=""><td>4.01e+12 52.3%</td><td>3.04e+10 79.5%</td><td>2.92e+10 91.1%</td><td>9.60e+01</td></qdp::pcolorvector<qdf<>	4.01e+12 52.3%	3.04e+10 79.5%	2.92e+10 91.1%	9.60e+01
Ioop at guarkprop4 w.cc: 71		4.01e+12 52.3%	3.04e+10 79.5%	2.92e+10 91.1%	9.60e+01
Vloop at guarkprop4 w.cc: 73		4.01e+12 52.3%	3.04e+10 79.5%	2.92e+10 91.1%	9.60e+01
♥ B Chroma::PrecFermActQprop < QDP::OLattice < QDP::PS	pinVector <qdp::pcolorvecto< td=""><td>3.84e+12 50.1%</td><td>2.90e+10 75.7%</td><td>2.80e+10 87.4%</td><td>9.67e+01</td></qdp::pcolorvecto<>	3.84e+12 50.1%	2.90e+10 75.7%	2.80e+10 87.4%	9.67e+01
▼ IB Chroma::LinOpSysSolverCG <qdp::olattice<qdp:: ■ Chroma::linOpSysSolverCG<qdp::olattice<qdp::< td=""><td>OL attice < ODB:: PC pinVector <</td><td>3.75e+12 48.9%</td><td>2.82e+10 73.7%</td><td>2.74e+10 85.7%</td><td>9.74e+01</td></qdp::olattice<qdp::<></qdp::olattice<qdp:: 	OL attice < ODB:: PC pinVector <	3.75e+12 48.9%	2.82e+10 73.7%	2.74e+10 85.7%	9.74e+01
		3.660+12 47.78	2.750+10 71.98	2.68e+10 83.78	9.740+01
loop at invcq2.cc: 163		3.48e+12 45.4%	2.61e+10 68.3%	2.55e+10 79.6%	9.75e+01
B Chroma::EvenOddPrecCloverLinOp::opera	tor()(ODP::OLattice <odp::psr< td=""><td>1.530+12 19.98</td><td>1.190+10 31.26</td><td>1.190+10.36.08</td><td>9.07e+01</td></odp::psr<>	1.530+12 19.98	1.190+10 31.26	1.190+10.36.08	9.07e+01
ce<0DP··F	SpinVector <odppcolorvect< td=""><td>1.58e+11 2.1%</td><td>1.92e+08 0.5%</td><td>1.87e+08 0.6%</td><td>9.74e+01</td></odppcolorvect<>	1.58e+11 2.1%	1.92e+08 0.5%	1.87e+08 0.6%	9.74e+01
Vloop at invco2 cc: 163	SpinVector <odp::pcolorvect< td=""><td>1.44e+11 1.9%</td><td>1.48e+08 0.4%</td><td>1.42e+08 0.4%</td><td>9.59e+01</td></odp::pcolorvect<>	1.44e+11 1.9%	1.48e+08 0.4%	1.42e+08 0.4%	9.59e+01
Hoop at invegziee. Tos		1.41e+11 1.8%	1.99e+09 5.2%	1.85e+09 5.8%	9.28e+01 🔻
] 310M of	13м 🛅]	
2 490+12 45 49 2 610	10 60 29	2 550	10 70 69		0.750 ± 01
3.400712 43.45 2.010	-TO 00.34	2.55e	-TO 13.04		9./Setul

Chroma Power and L3 queue on SB.



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



LBMHD on Nehalem EX: RCRToolkit



LBMHD Results: RCRToolkit





- -

LBMHD: HPCToolkit hot-wired with RCRToolkit

00		h	pcviewer: Ibr	nhd				
init.c 怒 🞯 bench.c								
<pre>46 int64_t EndX = guide->collisi 47 int64_t EndY = guide->collisi 48 int64_t EndZ = guide->collisi 49 for(i=0;i<=11;i++){ 50 Init_MyGrid(L->F[i],Start) 51 }</pre>	on.StartX+Parameters on.StartY+Parameters on.StartZ+Parameters (,EndX,StartY,EndY,Sta	.XDimPerThread; .YDimPerThread; .ZDimPerThread; artZ,EndZ,Up);						
<pre>51 for(i=12;i<=26;i++){ 52 for(i=12;i<=26;i++){ 53 Init_MyGrid(L->F[i] ,StartX 54 Init_MyGrid(L->G[0][i],StartX 55 Init_MyGrid(L->G[1][i],StartX 56 Init_MyGrid(L->G[2][i],StartX 57 } 58 Init_MyGrid(L->V[0] ,StartX 59 Init_MyGrid(L->V[0] ,StartX 50 Init_MyGrid(L->V[1] ,StartX 51 Init_MyGrid(L->V[1] ,StartX 52 Init_MyGrid(L->V[1] ,StartX 53 Init_MyGrid(L->V[1] ,StartX 54 Init_MyGrid(L->V[1] ,StartX 55 I</pre>	(,EndX,StartY,EndY,Sta (,EndX,StartY,EndY,Sta (,EndX,StartY,EndY,Sta (,EndX,StartY,EndY,Sta (,EndX,StartY,EndY,Sta (,EndX,StartY,EndY,Sta	artZ,EndZ,Up); artZ,EndZ,Up); artZ,EndZ,Up); artZ,EndZ,Up); artZ,EndZ,Up);						
<pre>60 Init_MyGrid(L->V[2] ,StartX 61 Init_MyGrid(L->B[0] ,StartX 62 Init_MyGrid(L->B[1] ,StartX 63 Init_MyGrid(L->B[2] ,StartX 64 Init_MyGrid(L->R ,Start) 65 } 66</pre>	, EndX, StartY, EndY, Std (, EndX, StartY, EndY, Std (, EndX, StartY, EndY, Std (, EndX, StartY, EndY, Std (, EndX, StartY, EndY, Std	artZ,EndZ,Up); artZ,EndZ,Up); artZ,EndZ,Up); artZ,EndZ,Up); artZ,EndZ,Up); artZ,EndZ,Up);						
67 voi 68 i PAPI_TOT_CYC:S	Sum (I)	PAPI_L2_TC	M:Sum (I)	RCF	-PAPI_L2	TCM:Sum (I)		Contention Percentage
⁶⁹ 1.64e+13	100 %	1.25e+11 100 % 2.96e+			10 100 %		2.36e+01	
cope Experiment Aggregate Metrics		PAPI_TOT_CYC 1.64e+13	2:Sum (I) 100 %	PAPI_L2_ 1.25e+	FCM:Sum (I) 11 100 %	RCR-PAPI_L2_TCM:S	ium (I) ▼ 100 %	Contention Percentage 2.36e+01
The second secon		1.640113	100 %	1.250	11 100 %	2.96e+10	100 %	2.360:01
Dench pthreads Vioon at bench c: 363		1.64e+13	1.25e+11 100		2.96e+10	100 %	2.36e+01	
▼ B pthread create		1.60e+13	3 97.4 $1.21e+11 96.8$		2.96e+10	96.68	2.36e+01	
▼ B⇒bench pthreads each	1.60e+13	3 97.4% 1.21e+11 96.8%		2.86e+10	96.6%	2.36e+01		
Ioop at bench.c: 273	1.59e+13	97.2%	1.21e+	11 96.8%	2.86e+10	96.6%	2.36e+01	
loop at bench.c: 274	1.59e+13	97.2%	1.21e+	11 96.8%	2.86e+10	96.6%	2.36e+01	
Vioop at bench.c: 280		1.59e+13	3 97.2% 1.21e+11 96.8%		2.86e+10	96.68	2.36e+01	
▼loop at bench	. 204 Lc: 285	1 590+13	97.28	1 210+	11 96.8%	2.860+10	96.68	2.360+01
loop at bench.c: 295		1.35e+13	82.1%	1.03e+	11 82.3%	1.57e+10	53.0%	1.52e+01
♥ Init Lattice pthreads each		9.95e+11	6.1%	9.20e+	09 7.3%	9.10e+09	30.7%	9.89e+01
▶loop at init.c: 52		7.58e+11	4.6%	7.03e+	09 5.6%	6.94e+09	23.4%	9.87e+01
► loop at Init.c: 49		1.51e+11 9.62o+10	0.9%	1.46e+ 7 10o+	09 1.28	1.45e+09	4.98	9.93e+01
▶ Init Lattice pthreads each		9.80e+11	6.0%	8.90e+	09 7.1%	3.78e+09	12.8%	4.25e+01
▶inlined fror	4.99e+11	3.0%	4.00e+	07 0.0%	4.00e+07	0.1%	1.00e+02	
inlined from bench.	c: 267	4.37e+08	0.0%					
► Init Lattice pthread	1.33e+10	0.1%						
p at bench.c: 295	1.35e+1	3 82.1%	1.0	3e+11 82	2.3%	1.57e+10	53.0%	1.52e
nit Lattice othreads each	9,95e+1	1 6.1%	9,21	0e+09 7	.38	9.10e+0.9	30.7%	9.890

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



FFT on Nehalem EX: RCRToolkit



FFT: HPCToolkit hot-wired with RCRToolkit

000	hpcviewer: fft_open	_mp						
😋 qthread.c 🛛 👻 qloop.c 🖉 fft_open_mp.rose 🛛				- 1				
<pre></pre>								
500 501 stat PAPI TOT CYC:Sum (I) PAP	LL2_TCM:Sum (I)	RCR-PAPEL2 TC	M:Sum (I) Cont	ention Percentage				
502 {				contention refeeldage				
503 do 1.37e+12 100 % 2.6	9e+09 100 %	1.46e+0	9 100 %	5.43e+01				
505 int *n2 = (int *)(((struct OUT_2_1527data *)out_argv) -> 0	UT21527data::n2_p);	1		<u>Y</u>				
Calling Context View 🗞 Callers View † Flat View								
] 🕆 🖖 🍐 fixi 🕅 📰 A* 🗛								
Scope	PAPI_TOT_CYC:Sum (I) V	PAPI_L2_TCM:Sum (I)	RCR-PAPI_L2_TCM:Sum (I)	Contention Percentage				
Experiment Aggregate Metrics	1.37e+12 100 %	2.69e+09 100 %	1.46e+09 100 %	5.43e+01				
▶monitor main	8.40e+11 61.2%	1.260+09 46.8%	3.500+08 24.0%	2.78e+01				
▼gthread wrapper	4.91e+11 35.8%	1.25e+09 46.5%	1.06e+09 72.6%	8.48e+01				
▼ ⊫>ploop step wrapper	4.90e+11 35.7%	1.23e+09 45.7%	1.04e+09 71.2%	8.46e+01				
▼ B>OUT 1 1527	4.71e+11 34.3%	1.22e+09 45.4%	1.04e+09 71.2%	8.52e+01				
▼oon at fft_open_mp_rose: 479	4 660+11 33 98	1 220+09 45 48	1 040+09 71 28	8 520+01				
loop at fft open mp.rose: 490	4.57e+11 33.3%	1.20e+09 44.6%	1.02e+09 69.9%	8.50e+01				
nt open mp.rose: 480	3.//e+09 0.3%							
Pinlined from fft open mp.rose: 454	3.15e+09 0.2%	2.00e+07 0.7%	2.00e+07 1.4%	1.00e+02				
fft open mp.rose: 487	7.74e+08 0.1%			U				
fft open mp.rose: 479	3.58e+08 0.0%							
fft open mp.rose: 485	3.13e+08 0.0%							
fft open mp.rose: 488	3.60e+07 0.0%							
SOMP loop default	5.37e+09 0.4%	*						
⊨ ⊫oop at fft open mp.rose: 490	4.57e+11 33.3%	1.20e+09 44.6%	1.02e+09 69.9%	8.50e+01				
▶ ©OUT 3 1527	4.830+09 0.4%							
		1	21M of 513M					

FFT TOR Occupancy on SB



FFT Power on SB

