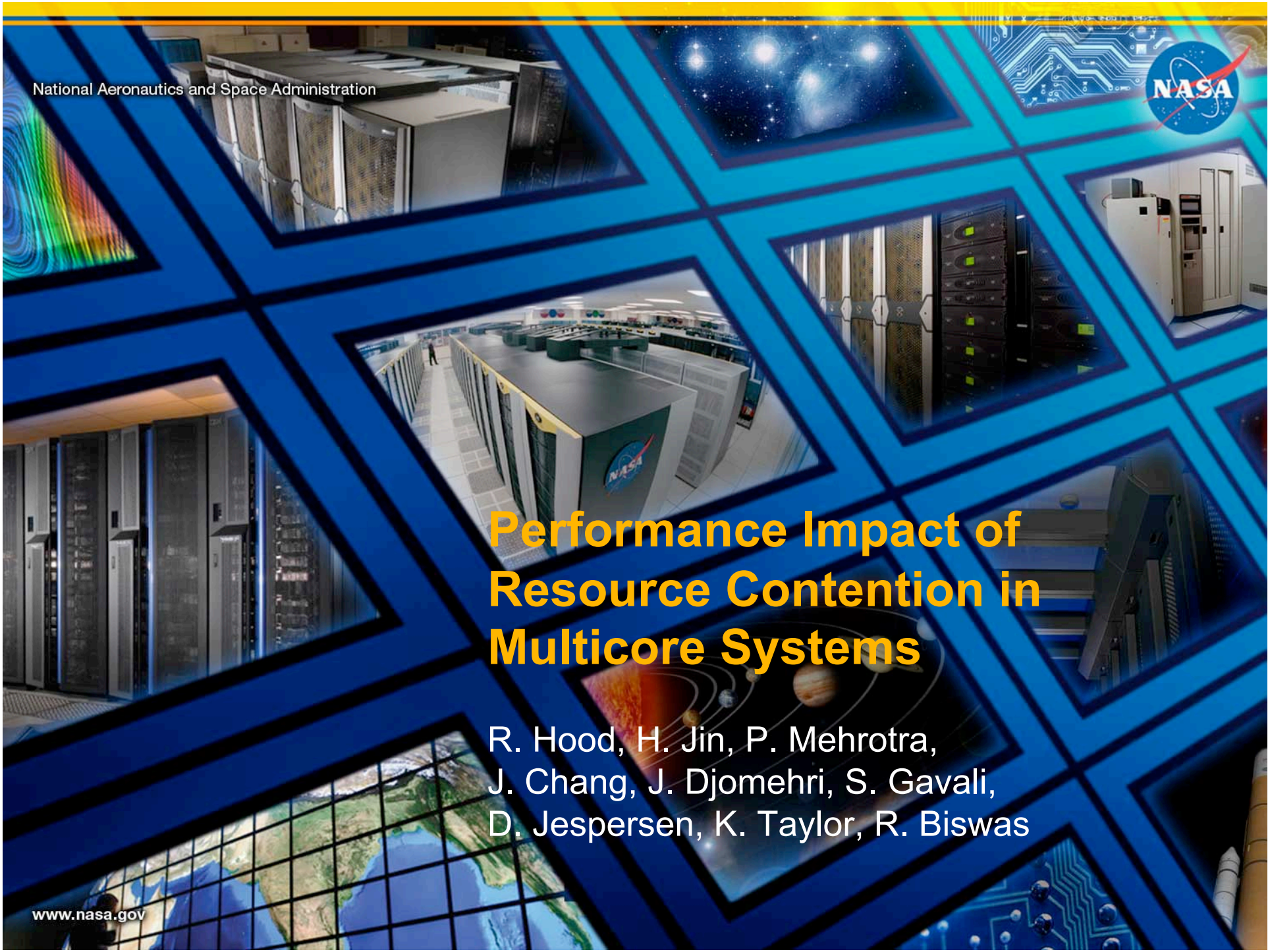
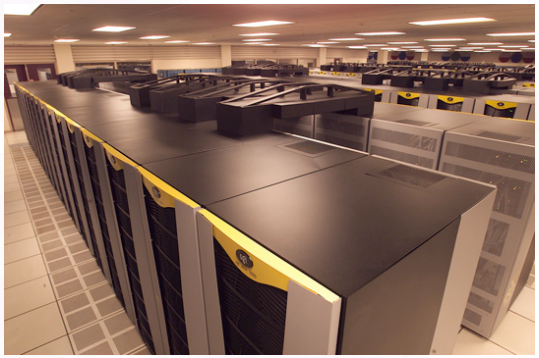
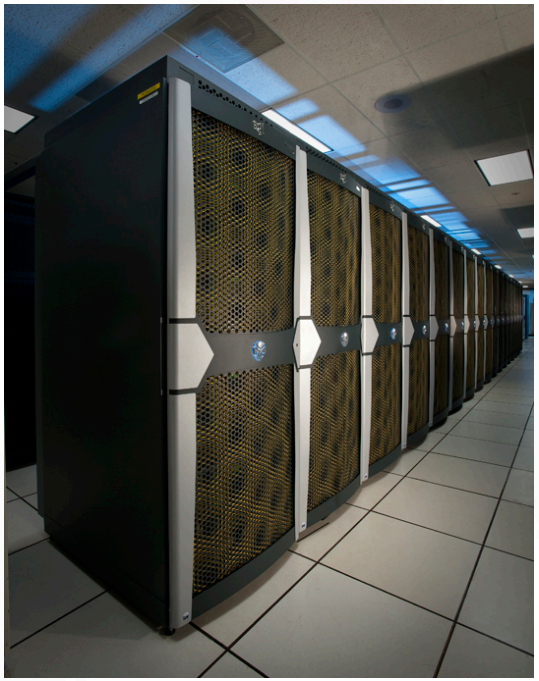

National Aeronautics and Space Administration
NASA
Performance Impact of Resource Contention in Multicore Systems
R. Hood, H. Jin, P. Mehrotra,
J. Chang, J. Djomehri, S. Gavali,
D. Jespersen, K. Taylor, R. Biswas
www.nasa.gov

Commodity Multicore Chips in NASA HEC





- 2004: *Columbia*
  - Itanium2 based; dual-core in 2007
  - Shared-memory across 512+ cores
  - 2GB / core
- 2008: *Pleiades*
  - Harpertown-based (UMA architecture)
  - Shared memory limited to 8 cores
  - Mostly 1GB / core; some runs at 4ppn
- 2009: *Pleiades* Enhancement
  - Nehalem-based (NUMA architecture)
  - 8 cores / node
  - Improved memory bandwidth
  - 3GB / core

Background: Explaining Superlinear Scaling

- Strong scaling of OVERFLOW on a Xeon (Harpertown) cluster

Number of MPI Ranks	8ppn	4ppn
16	16.24	7.29
32	6.96	3.40
64	3.09	1.75
128	1.49	0.91
256	0.74	0.47

superlinear

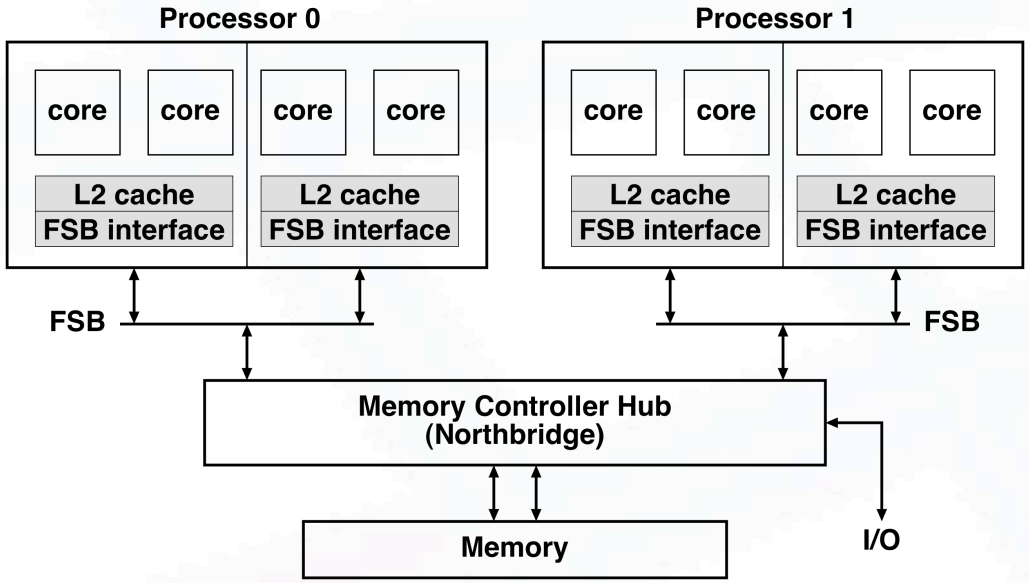
- Our traditional explanation:
  - With twice as many ranks, each rank has ~half as much data
  - Easier to fit that smaller working set into cache
- Still superlinear when run "spread out" to use only half the cores
  - Work/rank constant, but resources doubled
  - Is cache still the explanation?
- In general, what sort of resource contention is there?

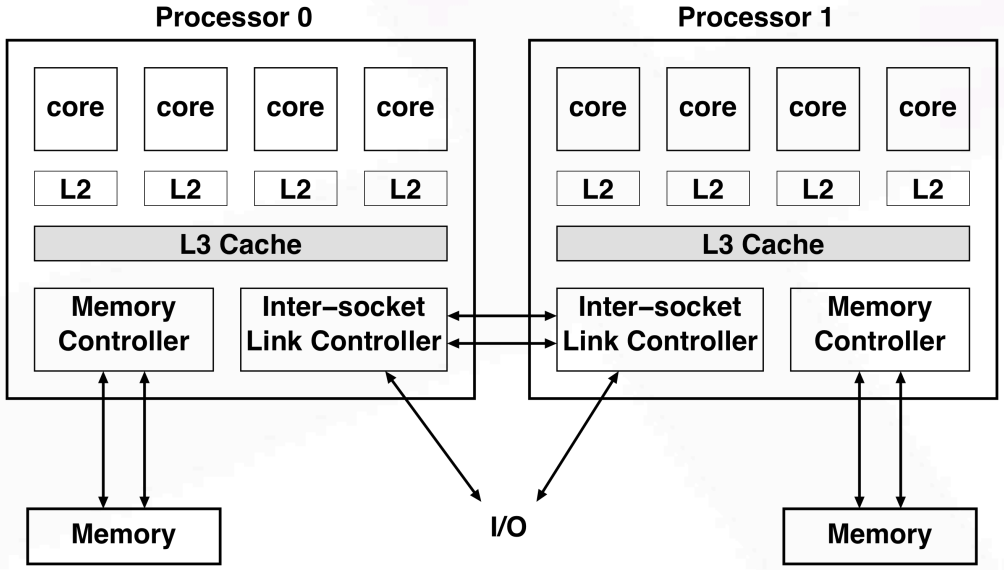
Sharing in Multicore Node Architectures

UMA-based node

Clovertown / Harpertown

- L2
- FSB
- Memory Controller



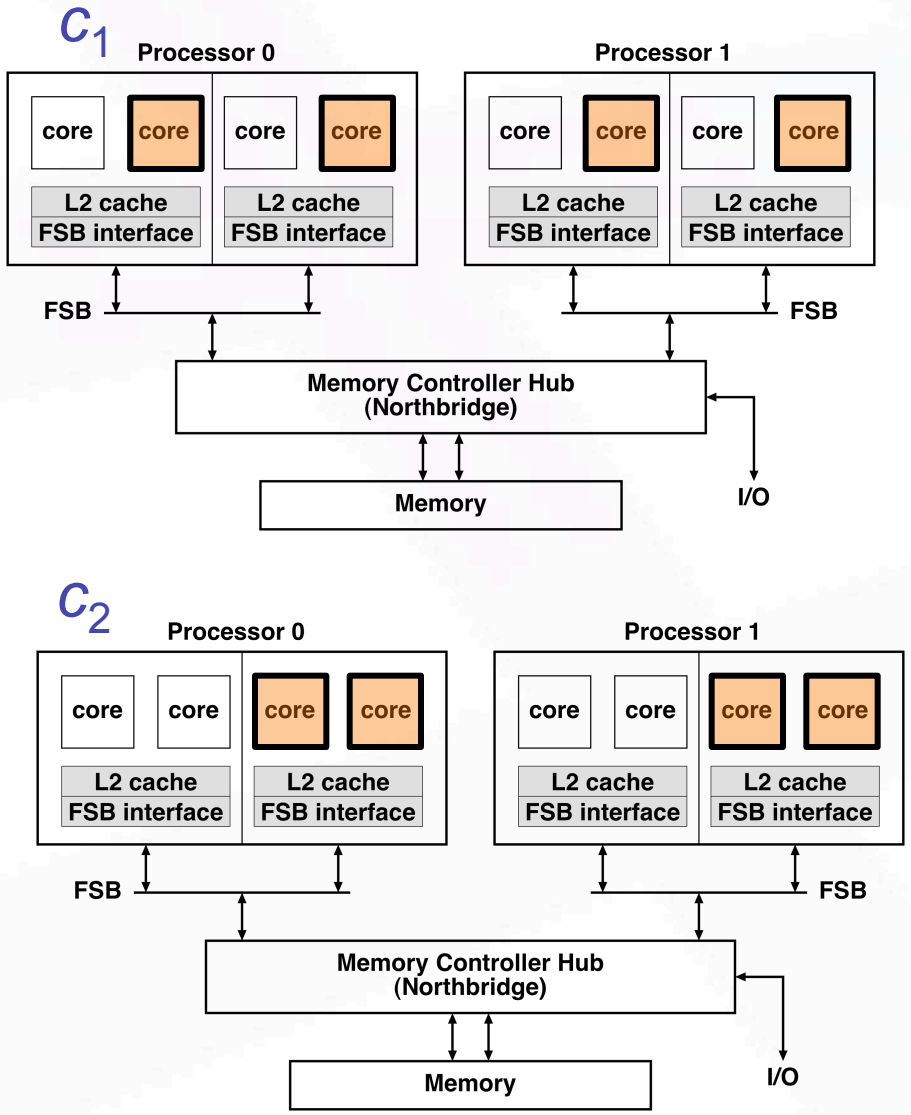


NUMA-based node

Nehalem / Barcelona

- L3
- Memory Controller
- Inter-socket Link Controller (QPI / HT3)

Isolating Resource Contention



- Compare configurations c_1 and c_2 of MPI ranks assigned to cores on a Harpertown node
  - Both use 4 cores per node
  - Communication patterns the same
- They place equal loads on:
  - FSB
  - Memory Controller
- Difference is in sharing of L2
- Compare timings of runs using these two configurations
  - Can calculate how much more time it takes when L2 shared
  - e.g. “there is a 17% penalty for sharing L2”
- Other configuration pairings can isolate FSB, memory controller

Differential Performance Analysis

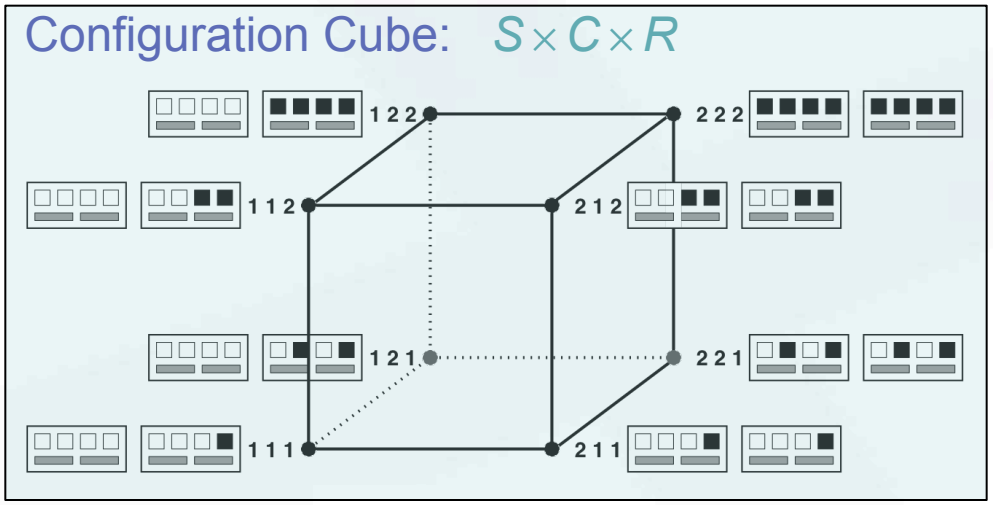
- Compare timings of runs of:
  - c_1 — a base configuration, and
  - c_2 — a configuration with increased sharing of some resource
- Compute the contention penalty, *P*, as follows:

$$P(c_1 \rightarrow c_2) = \frac{T(c_2) - T(c_1)}{T(c_1)}$$, where $T(c)$ is time for configuration c

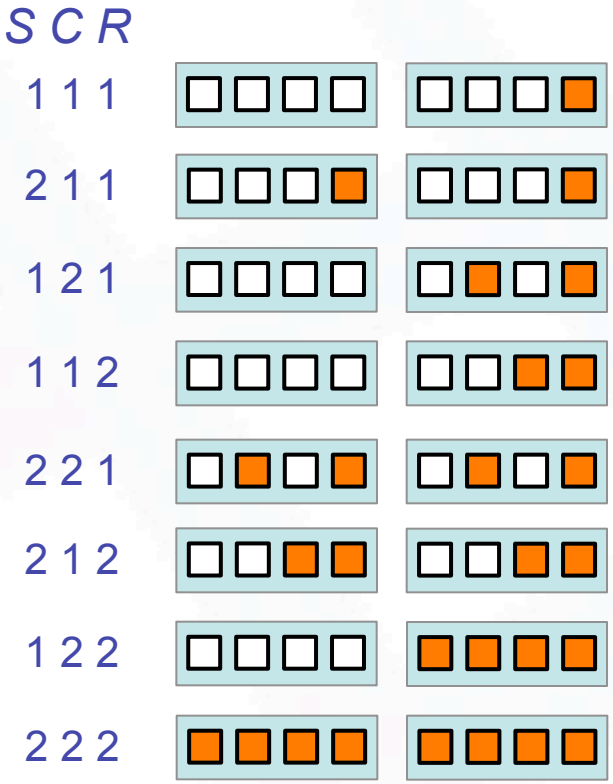
- Guidelines:
  - Isolate effect of sharing a specific resource by comparing two configurations that differ only in level of sharing of that resource
  - Minimize other potential sources of performance differences
    - Run exactly the same code on each configuration tested
    - Use a fixed number of MPI ranks in each run

Configurations for UMA-Based Nodes

- Interested in varying:
  - Number of sockets / node used *S*
  - Number of caches / socket used *C*
  - Number of active MPI ranks / cache *R*
- Label each configuration with a triple: (*S*,*C*,*R*)
- For our UMA-based nodes: *S*,*C*,*R* = {1, 2}



Node Configurations

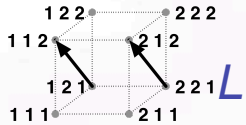


For NUMA-based nodes: *S* = {1,2}, *C* = {1}, *R* = {1,2,3,4}

- However, we use the UMA labeling for convenience

Contention Groups

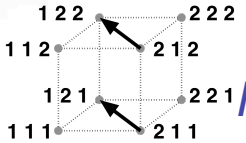
Configuration pairs to compare to isolate resource contention:



L2 *(no impact from communication)*

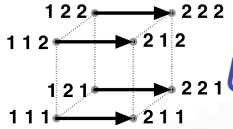
left → right

121	→	112	cores / node is the same
221	→	212	



FSB / L3+MC *(intra-node communication effect)*

211	→	121	cores / node is the same
212	→	122	



UMA:MC / NUMA: HT3, QPI *(intra- & inter-node communication effects)*

111	→	211	cores / node doubles
121	→	221	
112	→	212	
122	→	222	

Experimental Approach

- Run a collection of benchmarks and applications
  - HPC Challenge Benchmarks (DGEMM, Stream, PTRANS)
  - OVERFLOW overset grid CFD
  - MITgcm atmosphere-ocean-climate code
  - Cart3D CFD with unstructured set of Cartesian meshes
  - NCC unstructured-grid CFD
- Using InfiniBand-connected platforms that are based on multicore chips
  - UMA: Intel Clovertown-based SGI Altix cluster (hypercube)
    Intel Harpertown-based SGI Altix cluster (hypercube)
  - NUMA: AMD Barcelona-based cluster (fat tree switch)
    Intel Nehalem-based SGI Altix cluster (hypercube)
- Each application uses a fixed MPI rank count of 16 or larger
- Use placement tools to control process-core binding
- Take medians from multiple runs
  - Methodology results with ±1–2% contribution to penalty

Sample Contention Results

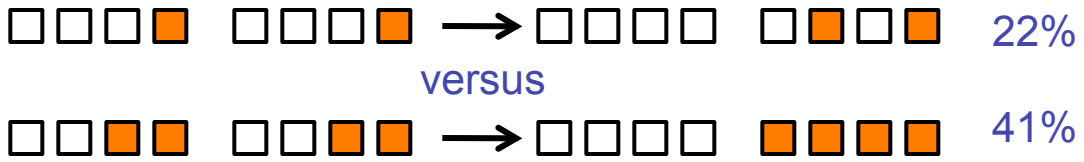
Max Penalty for Sharing Resource	ST_Triad	PTRANS	MITgcm	Cart3D
Clovertown				
○ L2 cache	1 – 3%	0 – 1%	13 – 16%	-1%
○ Front-side bus	44 – 56%	1 – 26%	14 – 41%	3 – 9%
○ Memory controller	22 – 24%	7 – 21%	10 – 27%	1 – 12%
Harpertown				
○ L2 cache	5%	-1%	24%	2 – 4%
○ Front-side bus	81 – 88%	28 – 44%	50 – 71%	22 – 41%
○ Memory controller	-2 – 3%	-4 – 9%	5 – 6%	0 – 5%
Barcelona				
○ L3 + memory controller	22 – 69%	6 – 21%	27 – 79%	7 – 14%
○ HT3	2 – 7%	2 – 18%	0 – 1%	-2 – 1%
Nehalem				
○ L3 + memory controller	50 – 95%	6 – 9%	24 – 67%	4 – 17%
○ QPI	-1 – 3%	-9 – 35%	2 – 6%	1 – 6%

Sample Contention Results

Max Penalty for Sharing Resource	ST_Triad	PTRANS	MITgcm	Cart3D
Clovertown				
○ L2 cache	1 – 3%	0 – 1%	13 – 16%	-1%
○ Front-side bus	44 – 56%	1 – 26%	14 – 41%	3 – 9%
○ Memory controller	22 – 24%	7 – 21%	10 – 27%	1 – 12%
Harpertown				
○ L2 cache	5%	-1%	24%	2 – 4%
○ Front-side bus	81 – 88%	28 – 44%	50 – 71%	22 – 41%
○ M			%	0 – 5%
Barcelo				
○ L			9%	7 – 14%
○ H			%	-2 – 1%
Nehalem				
○ L			7%	4 – 17%
○ Q			%	1 – 6%

Why the range of penalty values?

- Each penalty calculated using 2 or 4 pairs of configurations
- High side is (generally) from the denser configuration



Sample Contention Results

Max Penalty for Sharing Resource	ST_Triad	PTRANS	MITgcm	Cart3D
Clovertown				
○ L2 cache	1 – 3%	0 – 1%	13 – 16%	-1%
○ Front-side bus	44 – 56%	1 – 26%	14 – 41%	3 – 9%
○ Memory controller			10 – 27%	1 – 12%
Harpertown				
○ L2 cache			24%	2 – 4%
○ Front-side bus			50 – 71%	22 – 41%
○ Memory controller			5 – 6%	0 – 5%
Barcelona				
○ L3 + memory controller	22 – 69%	6 – 21%	27 – 79%	7 – 14%
○ HT3	2 – 7%	2 – 18%	0 – 1%	-2 – 1%
Nehalem				
○ L3 + memory controller	50 – 95%	6 – 9%	24 – 67%	4 – 17%
○ QPI	-1 – 3%	-9 – 35%	2 – 6%	1 – 6%

A tale of two applications –

MITgcm: Substantial penalties for socket's memory channel and for cache

Cart3D: Designed & tuned to make effective use of cache

Sample Contention Results

Max Penalty for Sharing Resource	ST_Triad	PTRANS	MITgcm	Cart3D
Clovertown				
○ L2 cache	1 – 3%	0 – 1%	13 – 16%	-1%
○ Front-side bus	44 – 56%	1 – 26%	14 – 41%	3 – 9%
○ Memory controller	22 – 24%	7 – 21%	10 – 27%	1 – 12%
Harpertown				
○ L2 cache	5%	-1%	24%	2 – 4%
○ Fr			50 – 71%	22 – 41%
○ M			5 – 6%	0 – 5%
Barcelon				
○ L3			27 – 79%	7 – 14%
○ HT			0 – 1%	-2 – 1%
Nehalem				
○ L3 + memory controller	50 – 95%	6 – 9%	24 – 67%	4 – 17%
○ QPI	-1 – 3%	-9 – 35%	2 – 6%	1 – 6%

Why would the L2 penalty go up?

Clovertown L2: 4MB

Harpertown L2: 6MB

- Apparently 4MB not enough but 6MB is
- Small penalty for Clovertown from comparing poor performance to poor performance

Sample Contention Results

Max Penalty for Sharing Resource	ST_Triad	PTRANS	MITgcm	Cart3D
Clovertown				
○ L2 cache	1 – 3%	0 – 1%	13 – 16%	-1%
○ Front-side bus	44 – 56%	1 – 26%	14 – 41%	3 – 9%
○ Memory controller	22 – 24%	7 – 21%	10 – 27%	1 – 12%
Harpertown				
○ L2 cache	5%	-1%	24%	2 – 4%
○ Front-side bus	81 – 88%	28 – 44%	50 – 71%	22 – 41%
○ Memory controller	-2 – 3%	-4 – 9%	5 – 6%	0 – 5%
Barcelona				
○ L3 + memory controller	22 – 69%			
○ HT3	2 – 7%			
Nehalem				
○ L3 + memory controller	50 – 95%	6 – 9%	24 – 67%	4 – 17%
○ QPI	-1 – 3%	-9 – 35%	2 – 6%	1 – 6%

Why is there an HT3 / QPI penalty for Stream on NUMA?

- Snooping for cache coherency?
- Nehalem QPI has snoop filtering

Sample Contention Results

Max Penalty for Sharing Resource	ST_Triad	PTRANS	MITgcm	Cart3D
Clovertown				
○ L2 cache	1 – 3%	0 – 1%	13 – 16%	-1%
○ Front-side bus	44 – 56%	1 – 26%	14 – 41%	3 – 9%
○ Memory controller	22 – 24%	7 – 21%	10 – 27%	1 – 12%
Harpertown				
○ L2 cache	5%	-1%	24%	2 – 4%
○ Front-side bus	81 – 88%	28 – 44%	50 – 71%	22 – 41%
○ Memory controller	-2 – 3%	-4 – 9%	5 – 6%	0 – 5%
Barcelona				
○ L3 + me			%	7 – 14%
○ HT3				-2 – 1%
Nehalem				
○ L3 + me			%	4 – 17%
○ QPI	-1 – 3%	-9 – 35%	2 – 6%	1 – 6%

Architectural observations:

Clovertown ➔ Harpertown

- Clear reduction in Memory Controller penalties
- FSB becomes more of a bottleneck

Sample Contention Results

Max Penalty for Sharing Resource	ST_Triad	PTRANS	MITgcm	Cart3D
Clovertown				
○ L2 cache	1 – 3%	0 – 1%	13 – 16%	-1%
○ Front-side bus			14 – 41%	3 – 9%
○ Memory controller			10 – 27%	1 – 12%
Harpertown				
○ L2 cache			24%	2 – 4%
○ Front-side bus			50 – 71%	22 – 41%
○ Memory controller			5 – 6%	0 – 5%
Barcelona				
○ L3 + memory controller	22 – 69%	6 – 21%	27 – 79%	7 – 14%
○ HT3	2 – 7%	2 – 18%	0 – 1%	-2 – 1%
Nehalem				
○ L3 + memory controller	50 – 95%	6 – 9%	24 – 67%	4 – 17%
○ QPI	-1 – 3%	-9 – 35%	2 – 6%	1 – 6%

Architectural observations:

UMA ➔ NUMA

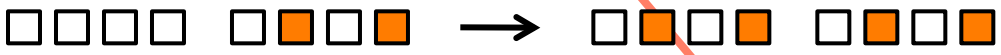
- FSB contention moves to L3 + memory controller
- Except in a few cases, little impact on HT3 / QPI

Sample Contention Results

Max Penalty for Sharing Resource	ST_Triad	PTRANS	MITgcm	Cart3D
Clovertown				
○ L2 cache	1 – 3%	0 – 1%	13 – 16%	-1%
○ Front-side bus	44 – 56%	1 – 26%	14 – 41%	3 – 9%
○ Memory controller	22 – 24%	7 – 21%	10 – 27%	1 – 12%
Harp				
			24%	2 – 4%
			50 – 71%	22 – 41%
			5 – 6%	0 – 5%
Barc				
			27 – 79%	7 – 14%
○ HT3	2 – 7%	2 – 18%	0 – 1%	-2 – 1%
Nehalem				
○ L3 + memory controller	50 – 95%	6 – 9%	24 – 67%	4 – 17%
○ QPI	-1 – 3%	-9 – 35%	2 – 6%	1 – 6%

Why an HT3 / QPI penalty?

- Memory accesses should be local to socket
- Recall: communication differences, too
  - HT3 / QPI configuration pairs:



Can also have impact on UMA memory controller penalty calculation

Effect of Communication on Penalties

- Penalty of total execution time was defined as:

$$P(c_1 \rightarrow c_2) = \frac{T(c_2) - T(c_1)}{T(c_1)}$$

- Time breaks down as: $T(c) = T_{comp} + T_{comm}$

- Break penalty down to computation & communication parts:

$$P(c_1 \rightarrow c_2) = P_{comp} + P_{comm}$$

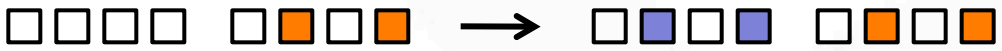
$$P_{comp} = \frac{T(comp_2) - T(comp_1)}{T(c_1)}$$

$$P_{comm} = \frac{T(comm_2) - T(comm_1)}{T(c_1)}$$

Computation & Communication in MITgcm (Clovertown) & PTRANS (Nehalem)

- Instrumented PTRANS to separate communication time
  - MITgcm already does this
- Calculated P_{comp} and P_{comm} as just discussed
  - MITgcm on Clovertown
    - Memory controller penalties from communication small
  - PTRANS on Nehalem
    - QPI penalties almost entirely due to communication

- Future work: use multiple instances of program
  - Double pressure on last level of memory hierarchy
  - No change to inter-node communication patterns



Conclusions

- New: a technique for quantifying effects of resource contention
  - Based on *differential performance analysis*
  - Determine impact due to sharing of specific resources
    e.g. L2, FSB, memory controller, HT3 / QPI
  - Tested technique on 4 multicore-based platforms,
    with 3 benchmarks and 4 applications
- Experimental observations
  - Dominant contention factor: memory bandwidth to socket
    Up to 95% for StreamTriad on Nehalem
  - Clovertown ➔ Harpertown: moved MC contention to FSB
  - UMA ➔ NUMA: socket memory bandwidth still big bottleneck
- Approach aids understanding of both applications & architectures

- OVERFLOW's "superlinear" behavior 4ppn ➔ 8ppn?
  - L2: 40% FSB: 54% MC: 3%