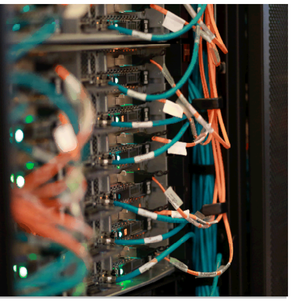
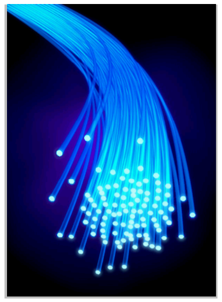
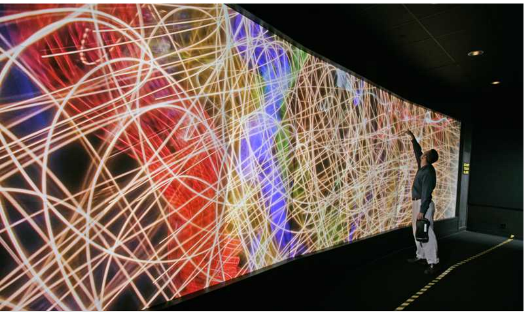
SAND2016-3309C

NiMC: Characterizing and Eliminating Network-Induced Memory Contention

Taylor Groves, Ryan E. Grant and Dorian Arnold

Sandia National Laboratories

Exceptional service in the national interest

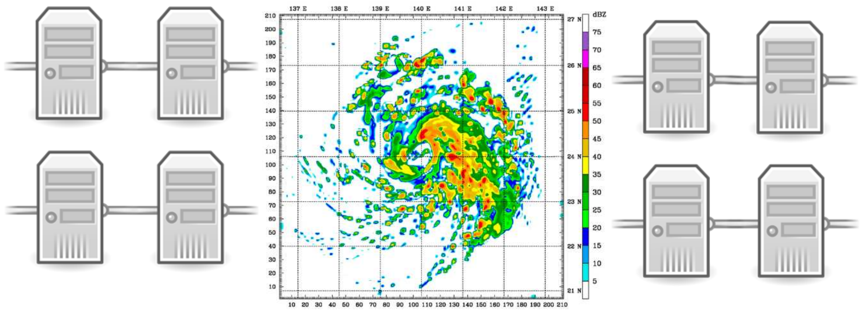
U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

High Performance Computing

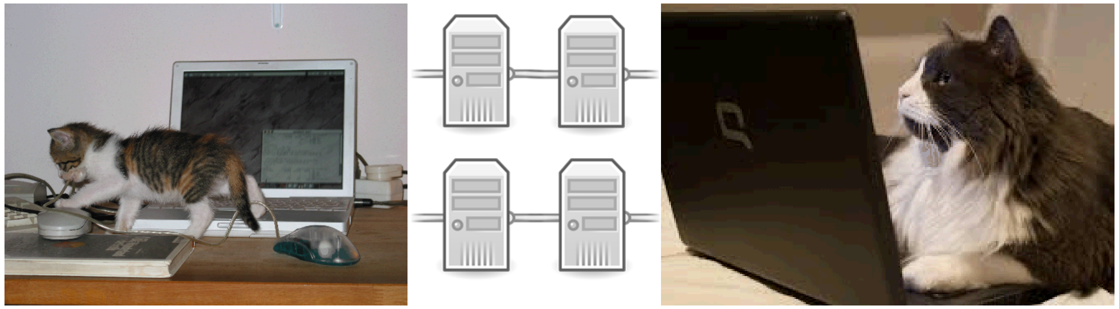
Concentrated computational power solving a common problem

N computers : 1 problem



Connected system solving disparate, independent problems

N computers : M problems



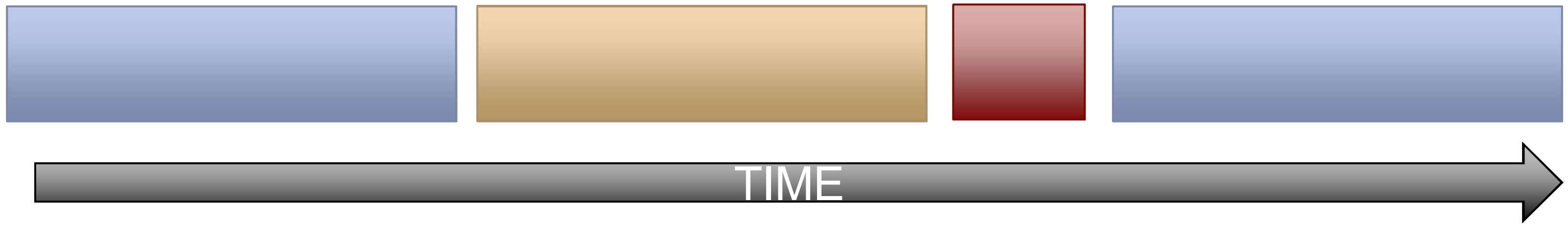
Background

- High Performance Computing is bottlenecked:
  - Bursty workloads
  - Synchronous communication models
  - Contention for shared resources, e.g. memory, networks
  - Processes operating in private address space

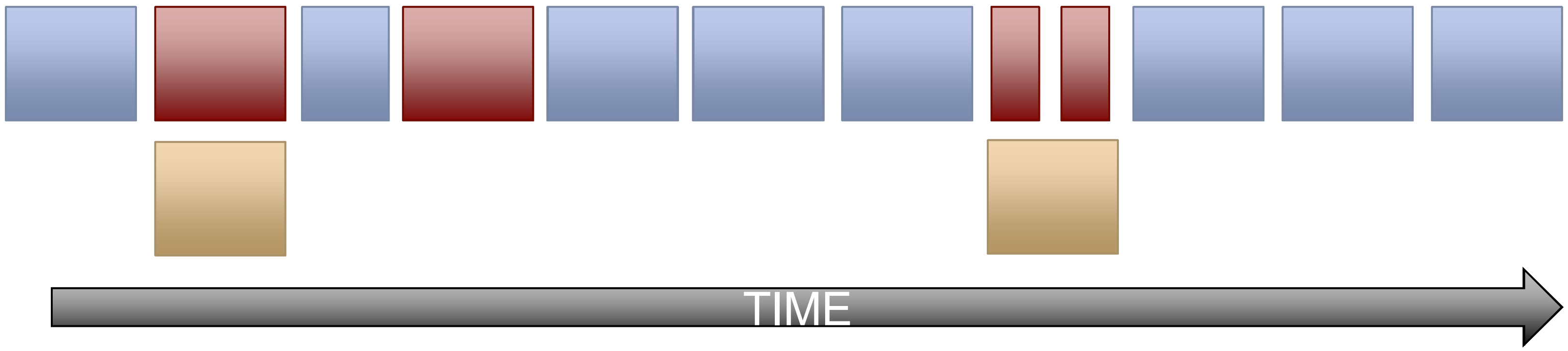
- Community proposed improvements:
  - Asynchronous many-task models
  - Partitioned Global Address Space, Remote Direct Memory Accesses
  - Efforts to further parallelize memory and communication
  - Smaller diameter networks with higher connectivity

Background (Overlap)

- Traditionally



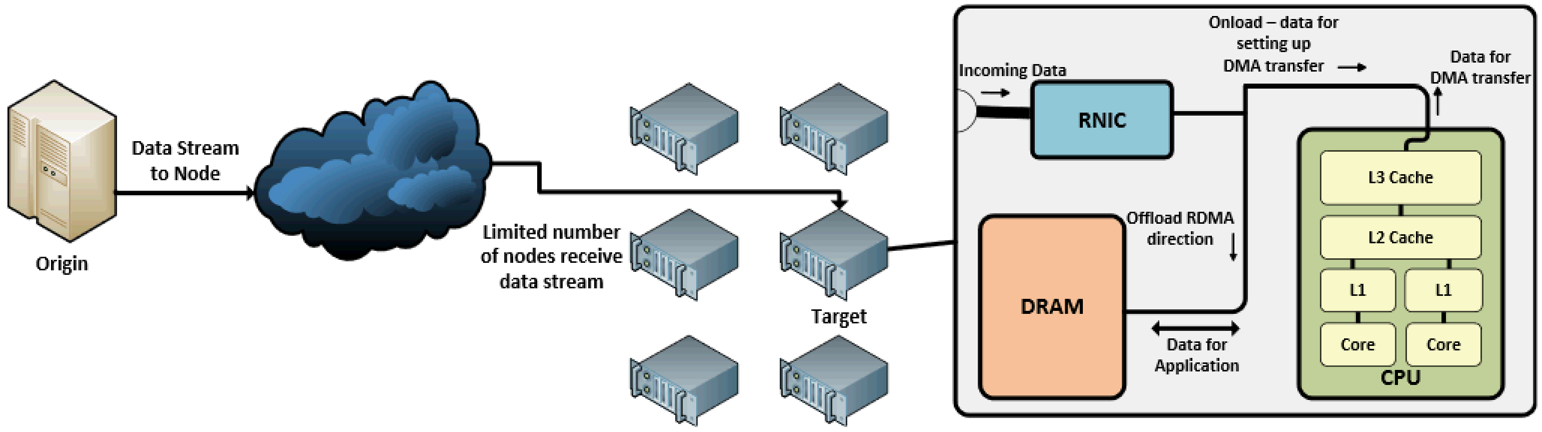
- Effort to move towards



🟦 Computation 🟫 Synchronization 🟥 Analytics/Other

Background (RDMA)

- Remote Direct Memory Access (RDMA)
  - Bypass the CPU and access memory directly
- Facilitates overlap between communication and computation



- The only problem is that we are adding additional traffic on the memory subsystem

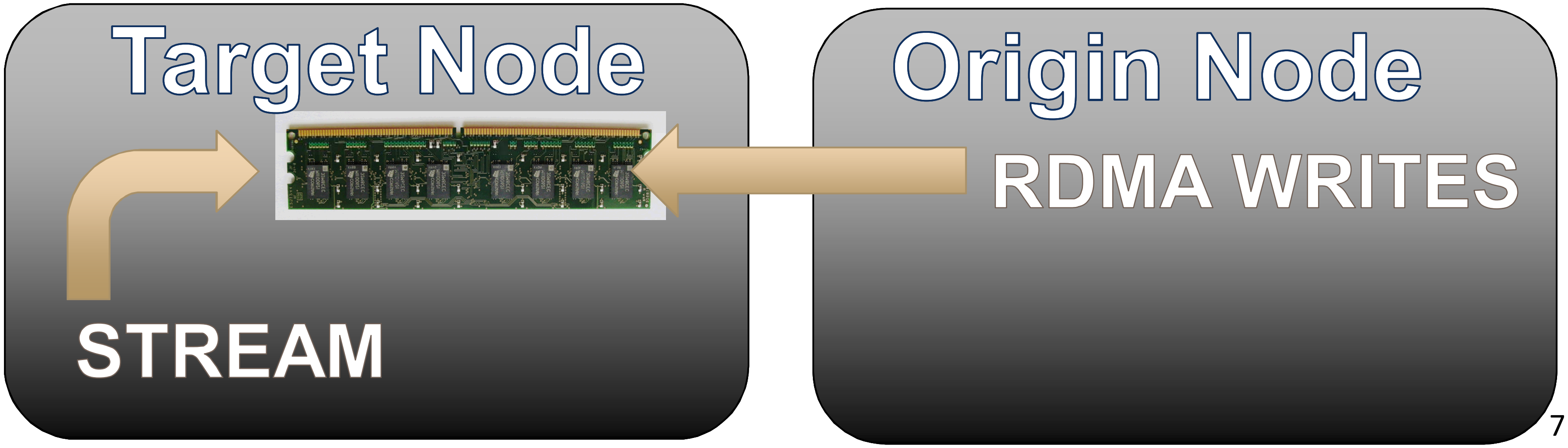
Primary Goal

Evaluate the impact of RDMA on modern systems

- Does this create Network-induced Memory Contention?
- What performance characteristics do we see on varying memory, CPU and network architectures?
- How does this impact different application workloads in the future?
- Given a performance degradation, can we identify solutions?

Preliminary Evaluation

- Test the worst case scenario
  1. Run memory intensive workload
  2. From a separate node, use RDMA writes to push as much data as possible into the machine to further increase pressure.



Preliminary Tests

- Experiments on small cluster of AMD Piledriver (4 cores)
  - Theoretical Network bandwidth 4GBps
  - Theoretical Memory Bandwidth 29.9 GBps

- STREAM benchmark (sustainable memory bandwidth)
  - Observed 12.7 GBps sustainable memory bandwidth
  - Modern CPU's don't fully utilize the theoretical memory bandwidth
  - Just 42% of theoretical memory bandwidth for this CPU
  - This leaves 17.2 GBps of unutilized memory bandwidth (more than enough to fit the 4GBps of RDMA bandwidth)

- Expectation is that RDMA will have no real impact on STREAM

STREAM + RDMA write

- STREAM sustained bandwidth reduced from 12.7 to 5.6 GBps
  - **Reduction to 44% of original STREAM performance**

- Somehow the RDMA writes are causing massive interference

Possible Culprits

- Memory Controllers: how is RDMA traffic distributed across different memory controllers?
  - Subtle policies like open page row-buffer management
- Memory Channels: ganged vs unganged
- CPU processing from Onload NIC's: some portion of packet processing is handled by the CPU
  - In our experiments this was never more than 2% of a single core
- Other overlooked factors

Further Evaluation

- **Need more results to draw meaningful conclusions**

- 7 different CPU architectures
  - Ranging from Westmere to Xeon Phi
- 3 variations of Infiniband Networks
  - Including onload and offload NICs
- 6 different memory frequencies
- 7 workloads of varying memory intensity
  - STREAM, CNS, HPCCG, LAMMPS, Lulesh, SNAP, XSBench

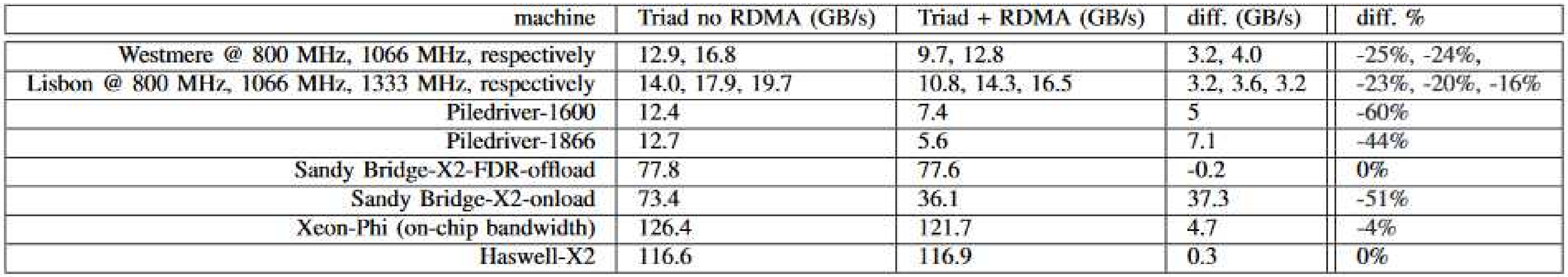
Further Evaluation

- Similar setup to the preliminary STREAM experiment
- All of our workloads for these experiments run on a single node
  - We don’t want to delay the application due to contention for network resources
- Injecting the maximum possible amount of RDMA writes

STREAM results

- 6 out of 8 systems see degradation of STREAM bandwidth
  - 4-56% reduction in sustainable bandwidth
  - Most noticeable for systems with onload NIC's
  - Older offload systems, where sustainable bandwidth is near theoretical see a reduction proportionate to the volume of RDMA writes

TABLE II: STREAM Triad Bandwidth with and without RDMA-NiMC



machine	Triad no RDMA (GB/s)	Triad + RDMA (GB/s)	diff. (GB/s)	diff. %
Westmere @ 800 MHz, 1066 MHz, respectively	12.9, 16.8	9.7, 12.8	3.2, 4.0	-25%, -24%,
Lisbon @ 800 MHz, 1066 MHz, 1333 MHz, respectively	14.0, 17.9, 19.7	10.8, 14.3, 16.5	3.2, 3.6, 3.2	-23%, -20%, -16%
Piledriver-1600	12.4	7.4	5	-60%
Piledriver-1866	12.7	5.6	7.1	-44%
Sandy Bridge-X2-FDR-offload	77.8	77.6	-0.2	0%
Sandy Bridge-X2-onload	73.4	36.1	37.3	-51%
Xeon-Phi (on-chip bandwidth)	126.4	121.7	4.7	-4%
Haswell-X2	116.6	116.9	0.3	0%

Small Scale Results (Sandy-Onload)

Why is LAMMPS more impacted than STREAM?

What about CNS?

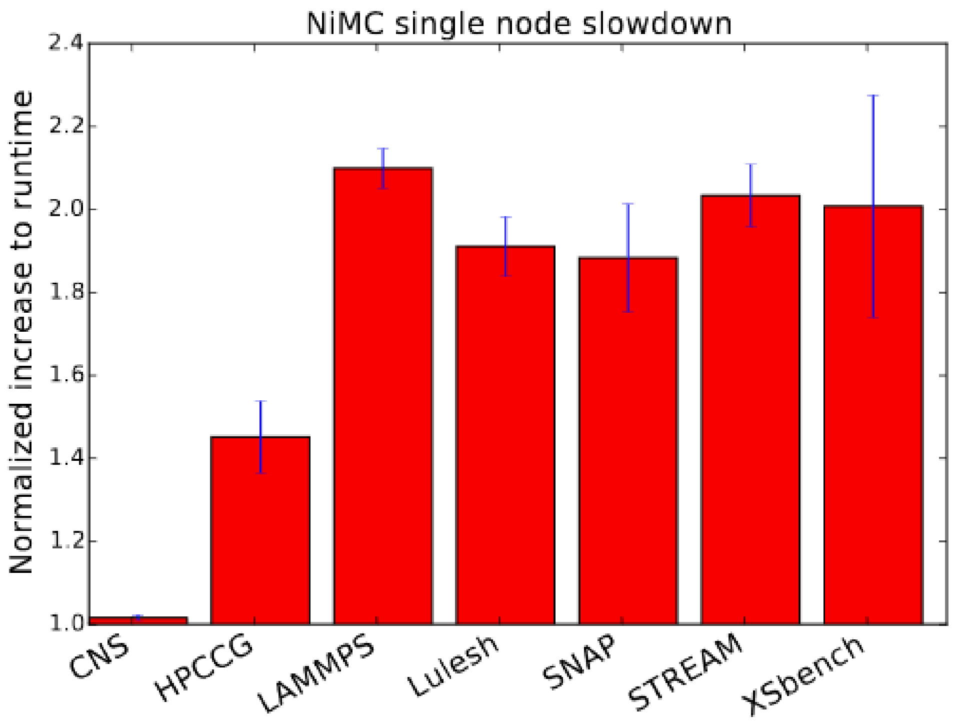


Fig. 3: Normalized impact of NiMC on single node runs.

Diving in with OpenSpeedShop

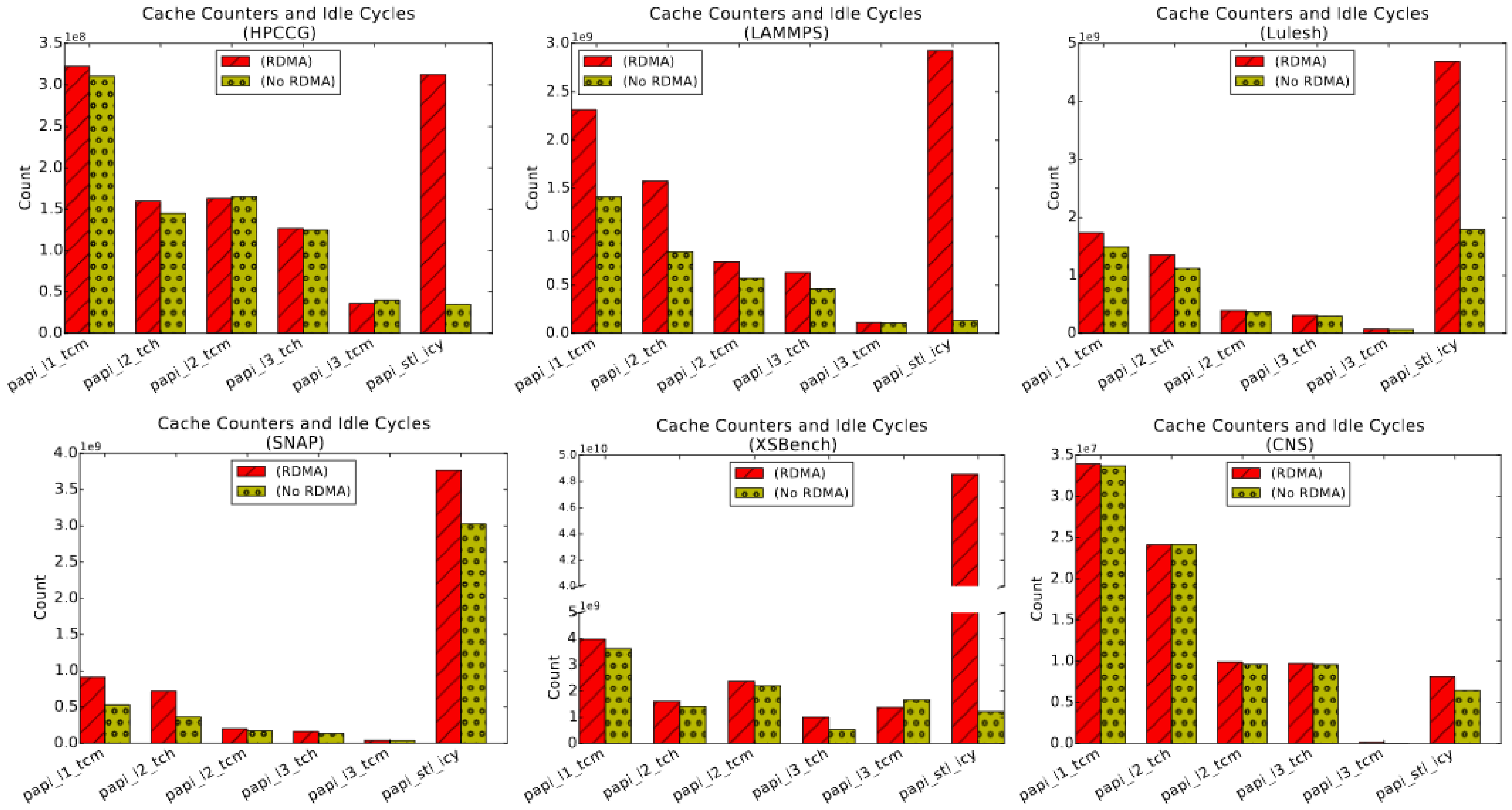
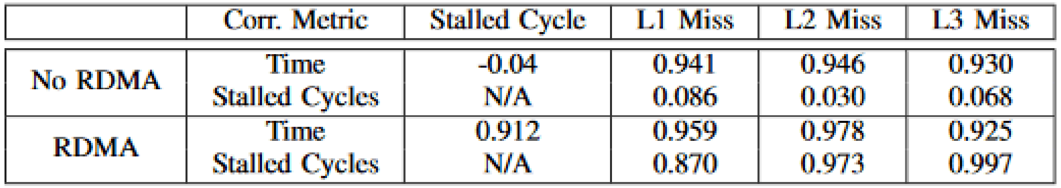


Fig. 4: Comparing performance counters for single node runs of benchmarks and proxy-apps with and without NiMC.

Evidence of Cache Pollution

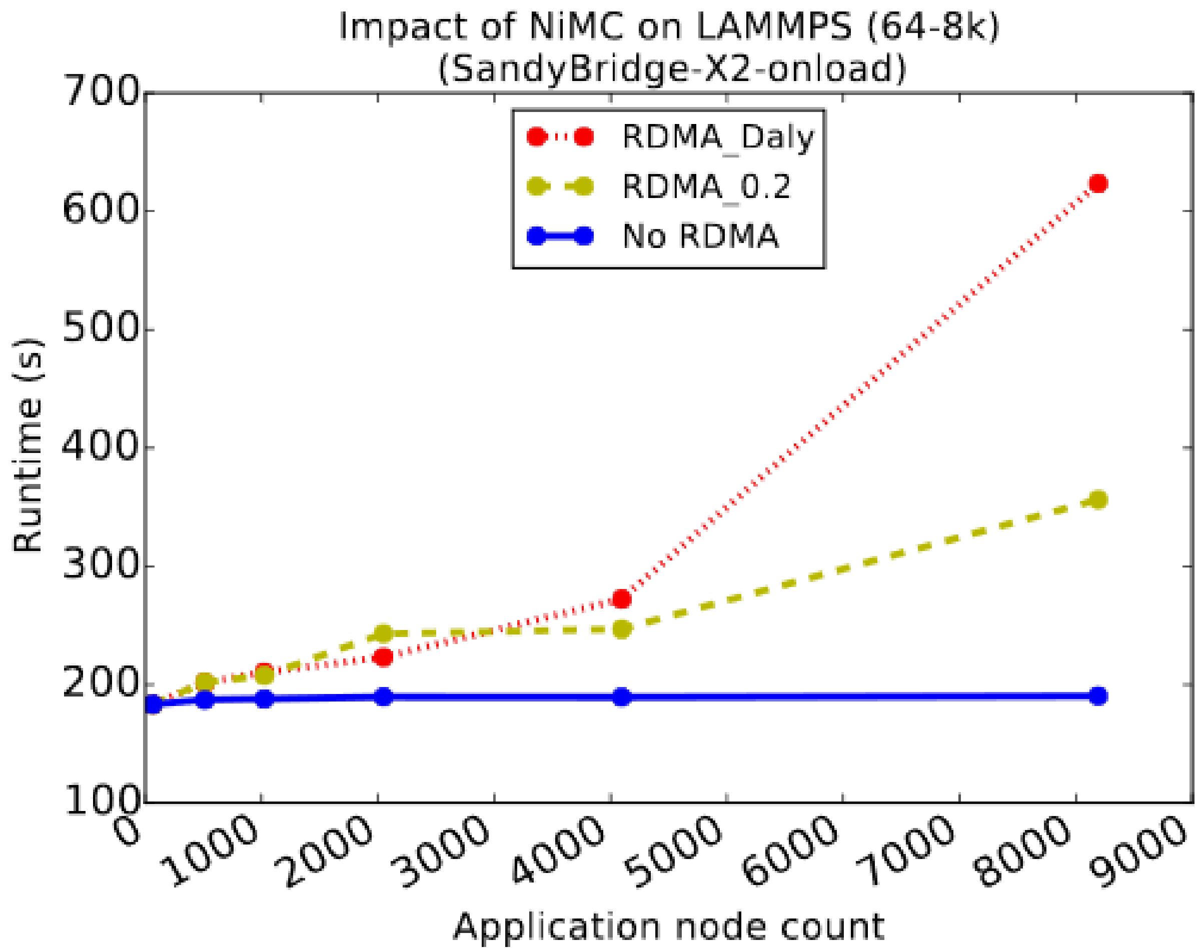
- In the absence of RDMA writes
  - No real correlation between stalled cycles and any of the cache misses
  - No real correlation between stalled cycles and runtime
- Add in RDMA writes
  - Strong correlation between Stalled Cycles and misses throughout the cache hierarchy
  - Correlation between runtime and L1 Misses becomes larger

TABLE IV: Performance Monitoring Counter Correlations Across All Applications



	Corr. Metric	Stalled Cycle	L1 Miss	L2 Miss	L3 Miss
No RDMA	Time	-0.04	0.941	0.946	0.930
	Stalled Cycles	N/A	0.086	0.030	0.068
RDMA	Time	0.912	0.959	0.978	0.925
	Stalled Cycles	N/A	0.870	0.973	0.997

Impact at Scale



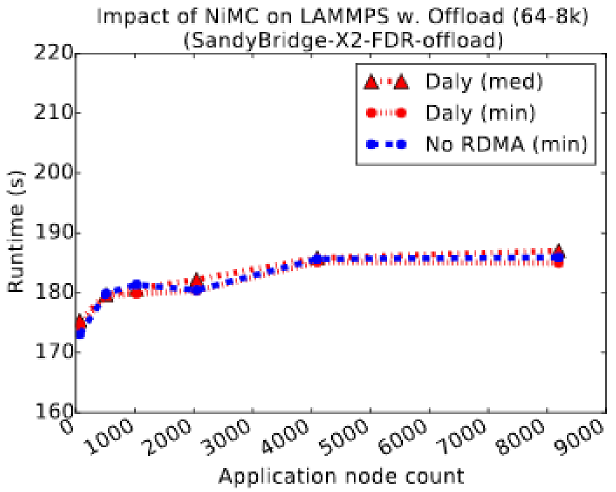
Solutions for NiMC

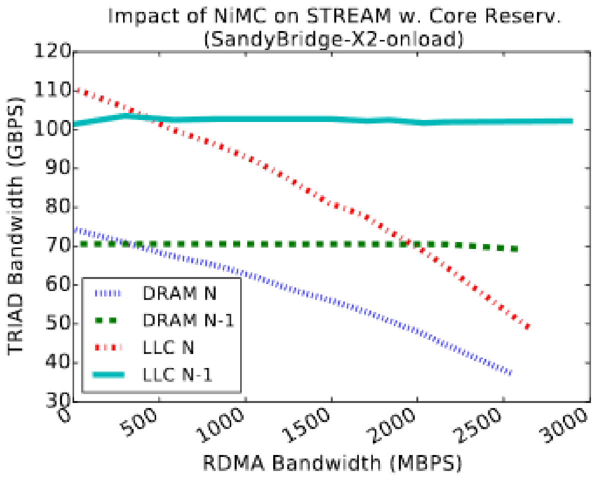
- Offload Network Cards

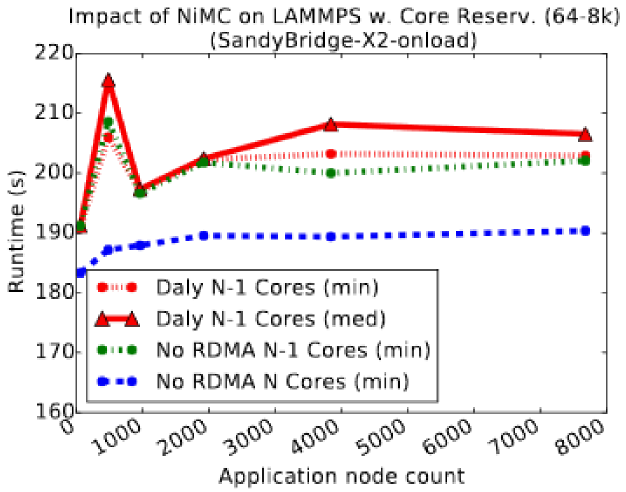
- Network Bandwidth Throttling

- Core Reservation

Solutions for NiMC







Summary of Results

- NiMC degraded performance on 75% of the evaluated systems
  - Up to 56% reduction of sustained memory bandwidth on single nodes
- 3X slowdown in LAMMPS running on an onload system with 8k processes
- Evaluated three possible solutions
  - Offload NIC's (for recent CPU's)
  - Network throttling
  - Core reservation

Acknowledgements

- Sandia National Laboratories: Center for Computing Research
- Scalable Systems Laboratory at University of New Mexico
- Texas Advanced Computing Center
- IPDPS Reviewers for comments and suggestions

Architectures

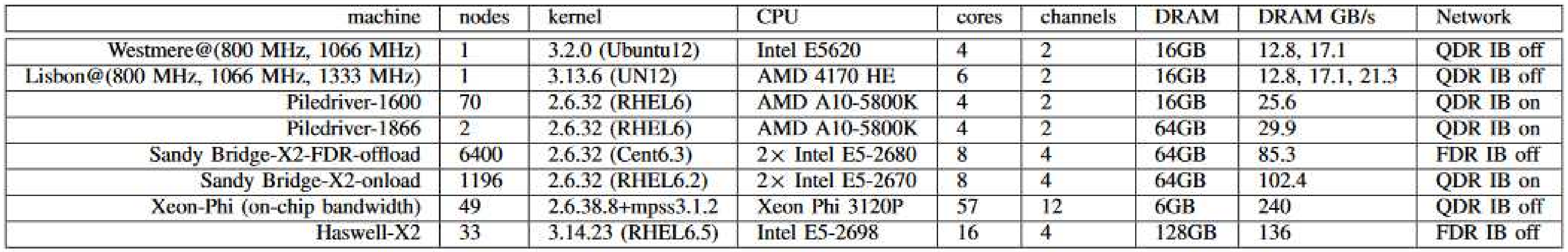
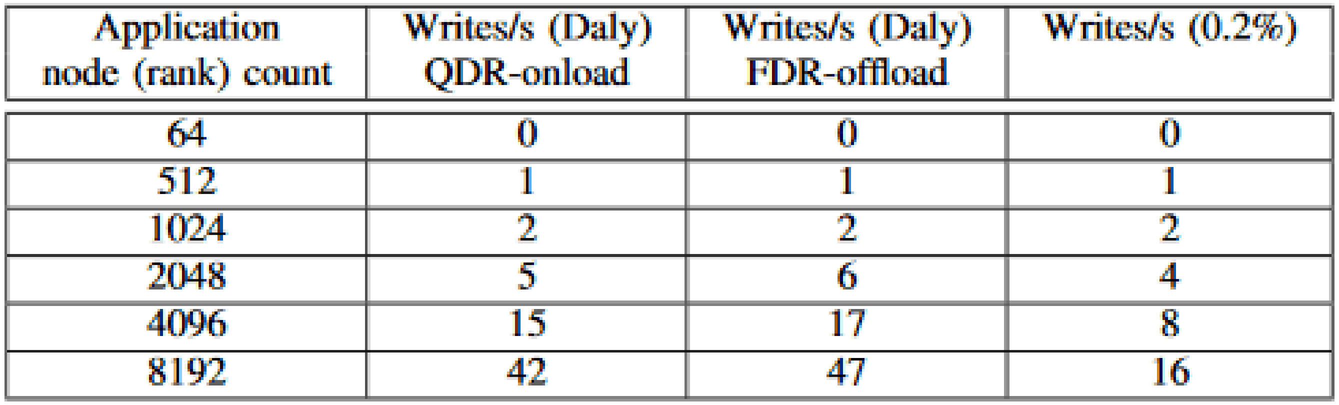


TABLE I: Evaluated Architectures

machine	nodes	kernel	CPU	cores	channels	DRAM	DRAM GB/s	Network
Westmere@(800 MHz, 1066 MHz)	1	3.2.0 (Ubuntu12)	Intel E5620	4	2	16GB	12.8, 17.1	QDR IB off
Lisbon@(800 MHz, 1066 MHz, 1333 MHz)	1	3.13.6 (UN12)	AMD 4170 HE	6	2	16GB	12.8, 17.1, 21.3	QDR IB off
Piledriver-1600	70	2.6.32 (RHEL6)	AMD A10-5800K	4	2	16GB	25.6	QDR IB on
Piledriver-1866	2	2.6.32 (RHEL6)	AMD A10-5800K	4	2	64GB	29.9	QDR IB on
Sandy Bridge-X2-FDR-offload	6400	2.6.32 (Cent6.3)	2× Intel E5-2680	8	4	64GB	85.3	FDR IB off
Sandy Bridge-X2-onload	1196	2.6.32 (RHEL6.2)	2× Intel E5-2670	8	4	64GB	102.4	QDR IB on
Xeon-Phi (on-chip bandwidth)	49	2.6.38.8+mpss3.1.2	Xeon Phi 3120P	57	12	6GB	240	QDR IB off
Haswell-X2	33	3.14.23 (RHEL6.5)	Intel E5-2698	16	4	128GB	136	FDR IB off

Number of Concurrent Writers

TABLE V: Number of concurrent RDMA writes



Application node (rank) count	Writes/s (Daly) QDR-onload	Writes/s (Daly) FDR-offload	Writes/s (0.2%)
64	0	0	0
512	1	1	1
1024	2	2	2
2048	5	6	4
4096	15	17	8
8192	42	47	16

Workloads