SAND2016-6556PE

Improving Power and Performance in HPC Networks

Taylor Groves

Sandia National Laboratories

Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

UNM CCR Center for Computing Research

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

My Past

- From San Antonio/Boerne
- Went to Texas State in San Marcos for B.S.
- Completing Ph.D. at UNM (Fall)
- Year-round intern at Sandia Labs now

My Family

My Vacation

Sandia National Laboratories

Flesh-eating bacteria scare along Gulf Coast has locals on alert

Infections from *Vibrio vulnificus* are rare and there is no official tracking of cases - but some people have started to cobble together their own ideas

Research Interests

- HPC/Scalable Systems,
- Networks, Communications,
- Modeling and Simulation,
- Monitoring Systems

Past/Present Work

- Wireless Sensor Networks (Xiao Chen @ TX State)
- MRNet Overlay Network (Dorian Arnold @ UNM)
  - Performance Modeling of Tree-based Data Aggregation
  - Lightweight bootstrapping on clusters (LIBI)
- Scalable Network Monitoring (Yihua He @ Yahoo!)
- **RDMA's Impact on Performance** (Ryan Grant @ Sandia)
- MPI benchmarking (Matt Dosanjh and Ryan Grant @ Sandia)
- Network Topology Design & Power/Performance Tradeoffs
  - (Ryan Grant, Scott Hemmert, Simon Hammond @ Sandia)

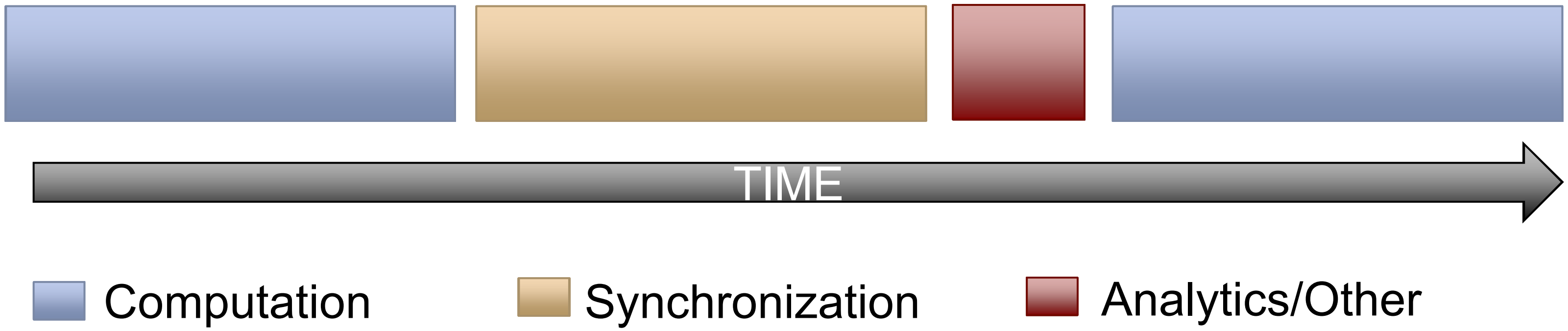
NiMC: Characterizing and Eliminating Network-Induced Memory Contention

Taylor Groves (UNM/SNL), Ryan E. Grant (SNL), Dorian Arnold (UNM) ***IPDPS 2016***

Extension with Aaron Gonzales (UNM/TripAdvisor).

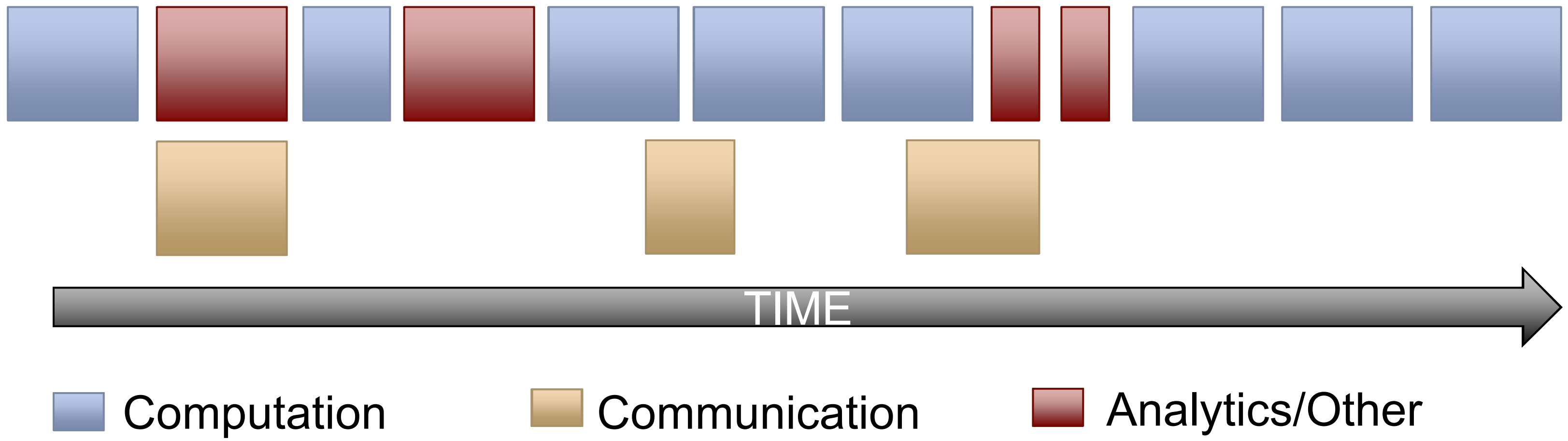
Journal in submission

Traditional HPC



- Bursty workloads
- Synchronous communication models
- Contention for shared resources, e.g. memory, networks
- Processes operating in private address space

Future HPC:

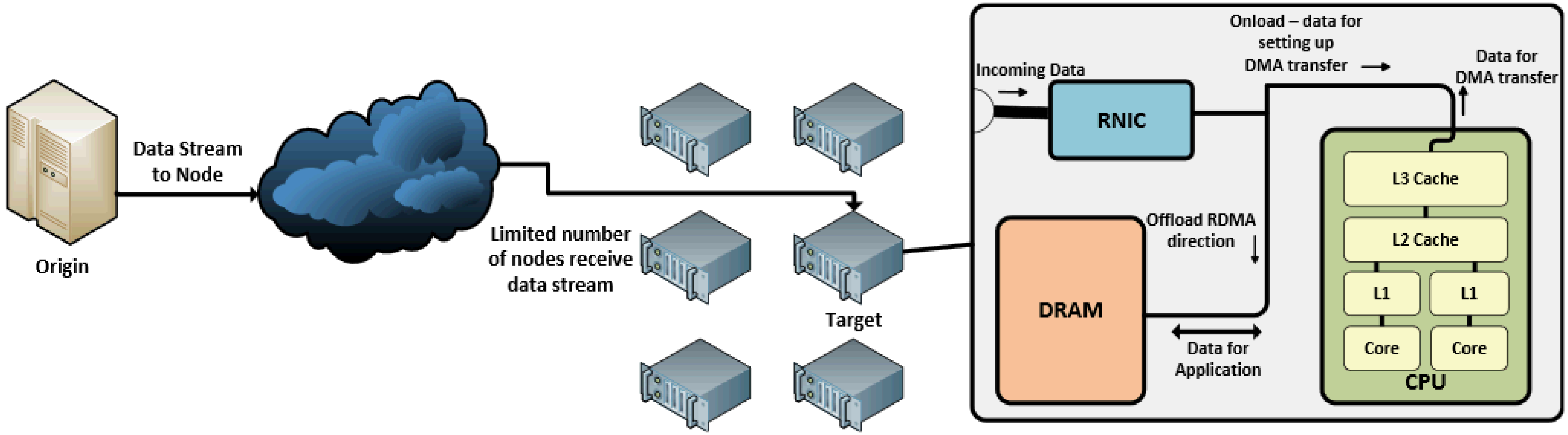


- Asynchronous many-task models
- Partitioned Global Address Space
- Efforts to further parallelize memory and communication
- Analytics to improve effective resource management

Most of these techniques want to leverage Remote Direct Memory Access (RDMA)

Background (RDMA)

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



- However, there's a downside.

Increased Contention for Memory

Sandia National Laboratories

"Fir0002/Flagstaffotos"

What is NiMC?

Network-induced Memory Contention:

Contention for local memory resources due to asynchronous communication originating from a remote node

Primary Goal

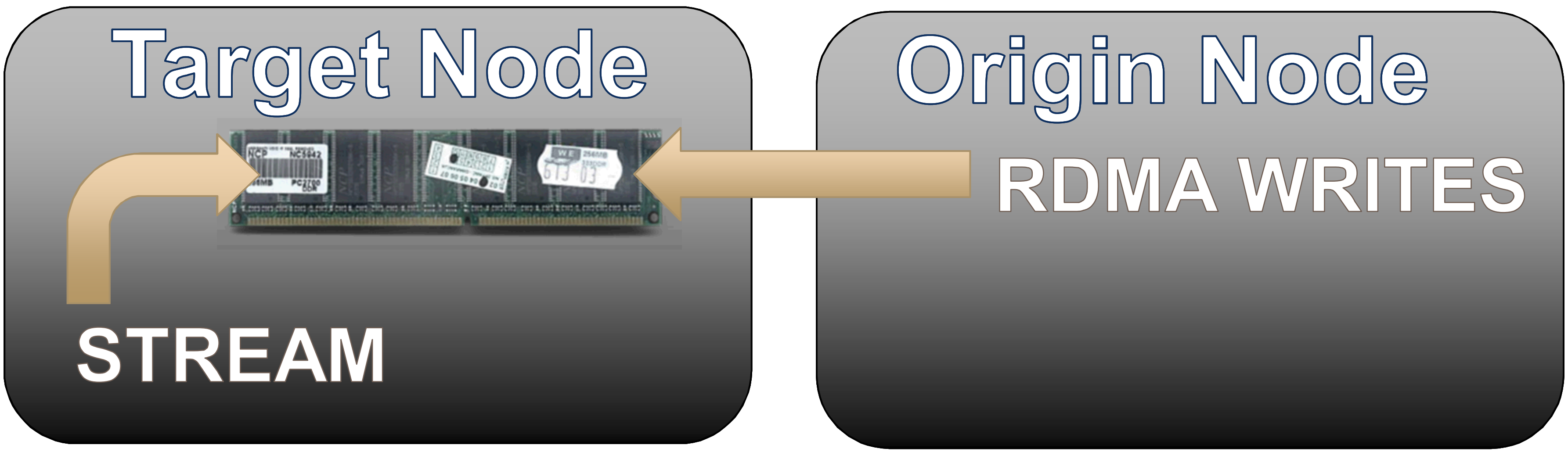
Evaluate the impact of RDMA on modern systems

1. Network-induced Memory Contention?
2. Characteristics on range of architectures?
3. Application Impact
4. Solution(s)?

Preliminary Evaluation

Test the worst case scenario

1. Run memory intensive workload
2. From a separate node, RDMA writes/puts to push as much data as possible into the machine to further increase pressure.

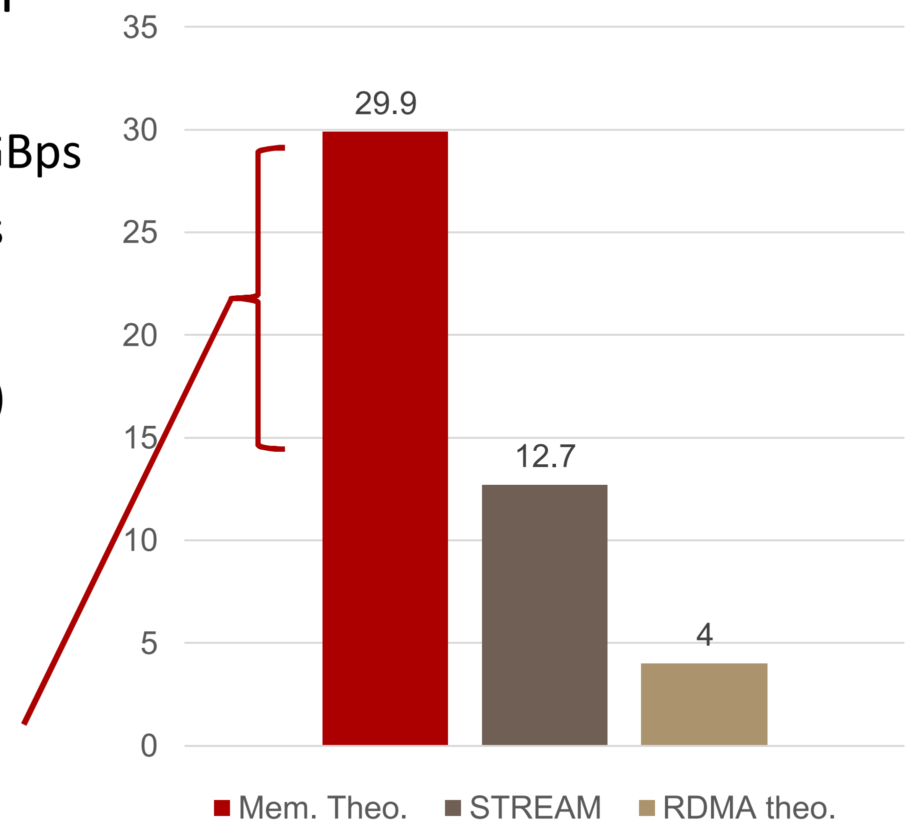


Preliminary Tests

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

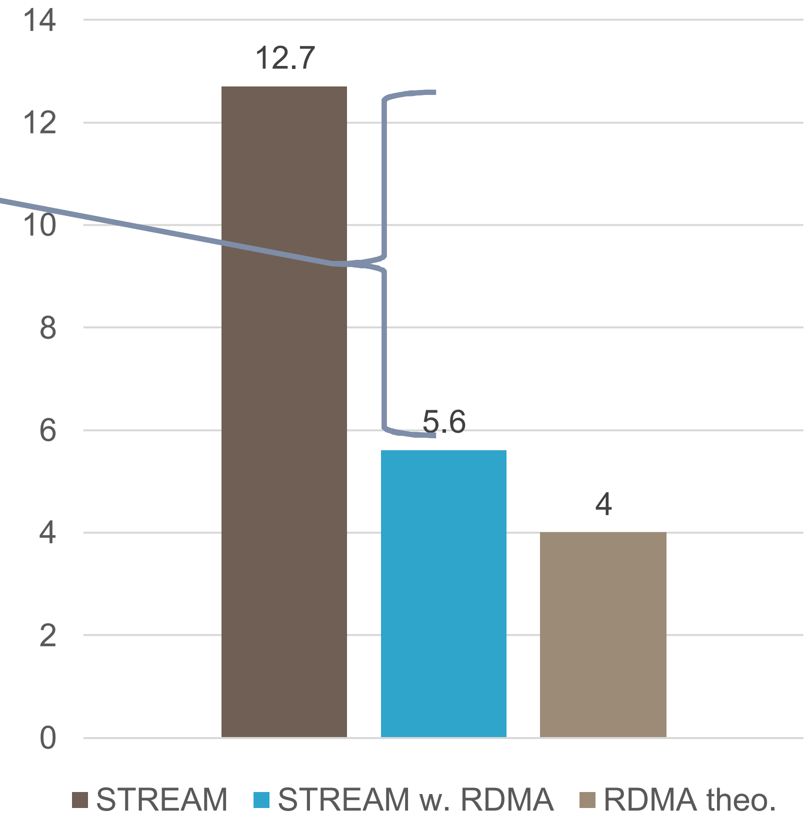
- STREAM benchmark (without RDMA)
  - Observed 12.7 GBps sustainable memory bandwidth

- 17 GBps headroom for RDMA



STREAM w. RDMA write

- However, performance worsens
- **56% penalty to STREAM**
- The penalty is greater than the total amount of RDMA
- Why is performance so bad?



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 NICs: 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 (4-core) to Xeon Phi (57-core)
- 3 variations of Infiniband Networks
  - Including onload and offload NICs
- 6 different memory frequencies
- 7 workloads of varying memory intensity

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
  - 3 offload systems see a reduction proportionate to the volume of RDMA writes

Machine	Triad no RDMA (GB/s)	Triad w. RDMA (GB/s)	Diff. (GB/s)	Diff. %
Westmere @ 800MHz, 1066MHz (offload)	12.9, 16.8	9.7, 12.8	-3.2, -4.0	-25%, -24%
Lisbon @ 800MHz, 1066MHz, 1333MHz (offload)	14, 17.9, 19.7	10.8, 14.3, 16.5	-3.2, -3.6, -3.2	-23%, -20%, -16%
Piledriver @ 1600MHz (onload)	12.4	7.4	-5	-40%
Piledriver @ 1866MHz (onload)	12.7	5.6	-7.1	-56%
SandyBridge-X2 (offload)	77.8	77.6	-0.2	0%
SandyBridge-X2 (onload)	73.4	36.1	-37.3	-51%
Xeon-Phi (on-chip, offload)	126.4	121.7	-4.7	-4%
Haswell-X2 (offload)	116.6	116.9	+0.3	0%

Further Evaluation

- Similar setup to earlier STREAM experiment
- Applications run on single node
  - We don’t want to measure contention on the network
- Injecting maximum possible amount of RDMA writes
  - 2-6 GBps

CNS

- Compressible Navier Stokes proxy app
- Stencil operations of a combustion problem
- Very lightweight (does not represent the computation)

Randy Montoya, SNL

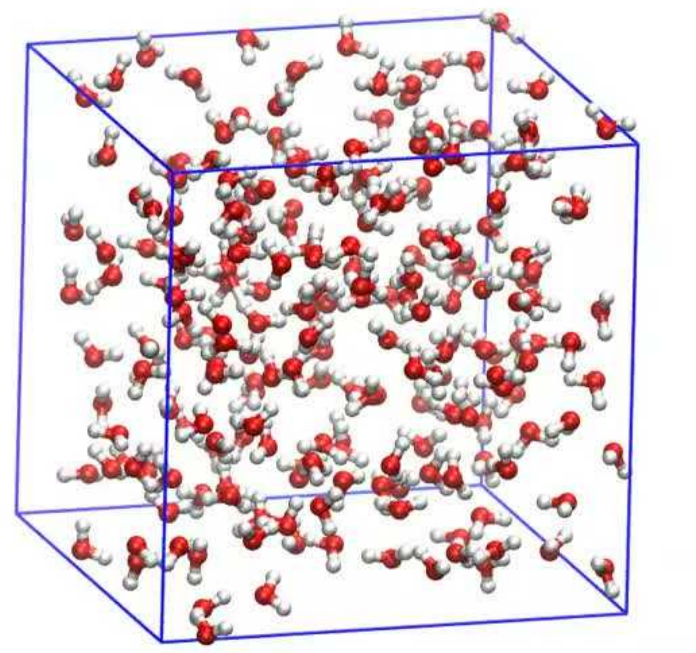
HPCCG

- Calculates conjugate gradient for a 3D chimney domain
- Mini-app, 27 point stencil
- Excellent weak scaling
- Memory intensive

LAMMPS

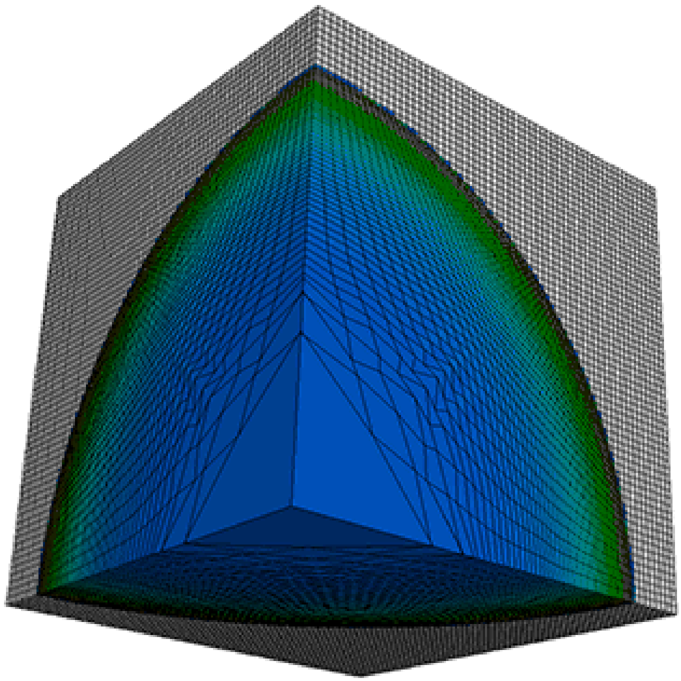
Sandia National Laboratories

- Molecular Dynamics
- Excellent weak scaling
- Nearest Neighbor Communication
- 3D Lennard-Jones melt
- 32,000 atoms per core



Lulesh

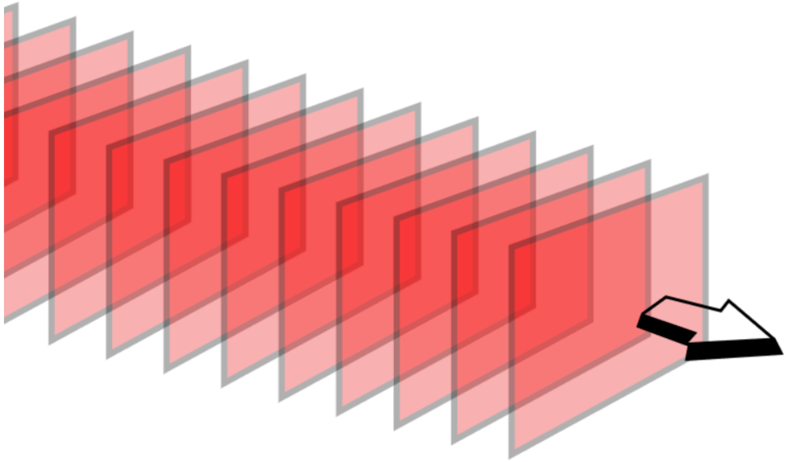
- Explicit Hydrodynamics code
- Solves simple Sedov blast problem
- Indicative of solvers in ALE3D
- Multiple kernels



codesign.llnl.gov/lulesh.php

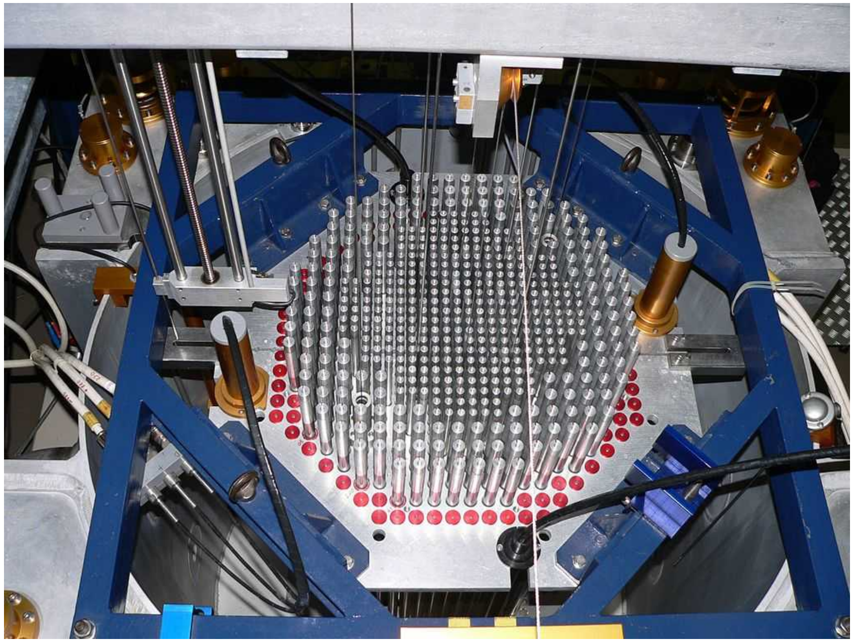
SNAP

- Neutral particle transport application
- Update to Sweep3D
- Number of neutral particles in multi-dimensional space
- Communication follows a wave propagation



XSBench

- Nuclear reactor core Monte-Carlo particle transport simulation
- Memory intensive
- Not designed for scaling (not used for multi-node)
- Single communication (reduction at the end)



https://en.wikipedia.org/wiki/Nuclear_reactor#/media/File:Crocus-p1020491.jpg

Small Scale Results (Sandy-Onload)

What about CNS?

Why is LAMMPS more impacted than STREAM?

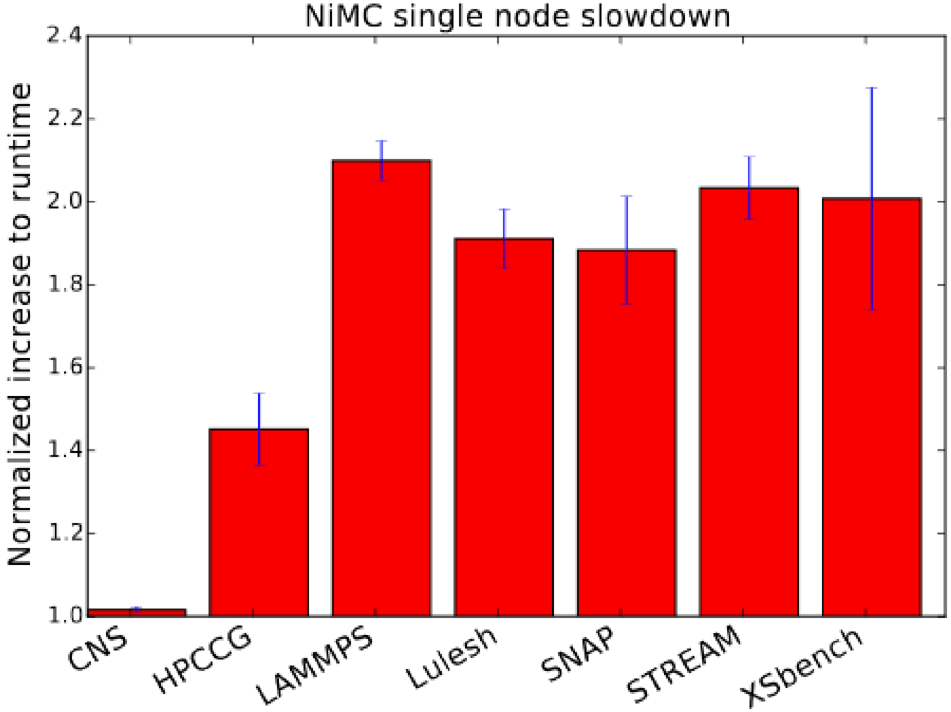


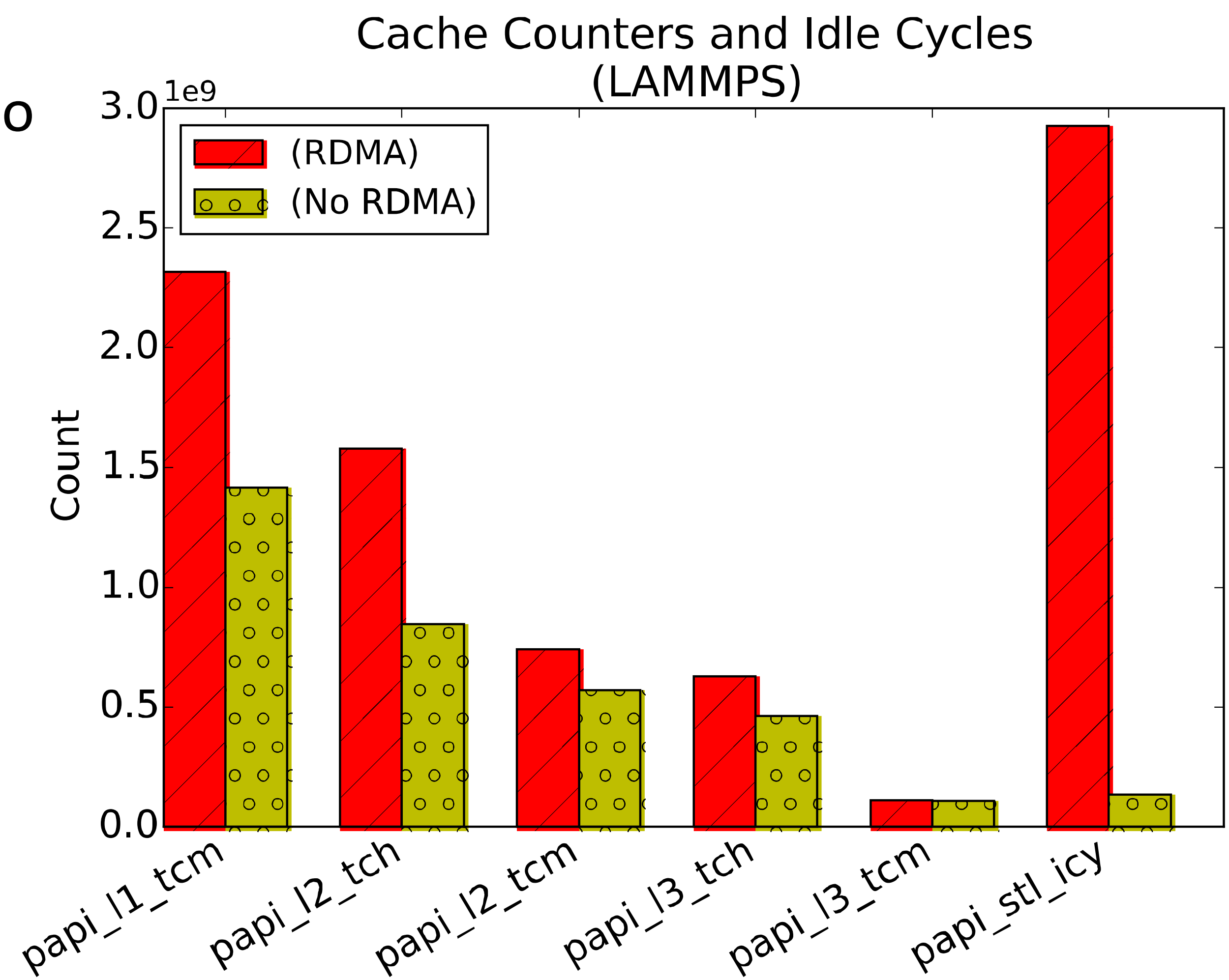
Fig. 3: Normalized impact of NiMC on single node runs.

Finding the Culprit

- Profiled each workload on Sandy-Bridge-X2-Onload
- Ran with and without RDMA
- Collected 6 counters
  - L1 miss
  - L2 hit & miss
  - L3 hit & miss
  - Stalled cycles

Finding the Culprit

- Huge increase to stalled cycles
- Increases to L1 & L2 Miss
- Increases to L1, L2 & L3 Hit



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
- **With** RDMA writes
  - Strong correlation between Stalled Cycles and misses throughout the cache hierarchy
  - Correlation between runtime and L1 Misses becomes larger

	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

- LAMMPS, scaling up to 8,192 processes
- Ran on the SandyBridge-X2 Onload system.
- Interested in minimum runtime
  - Don't want to capture performance degradation due to nearby jobs

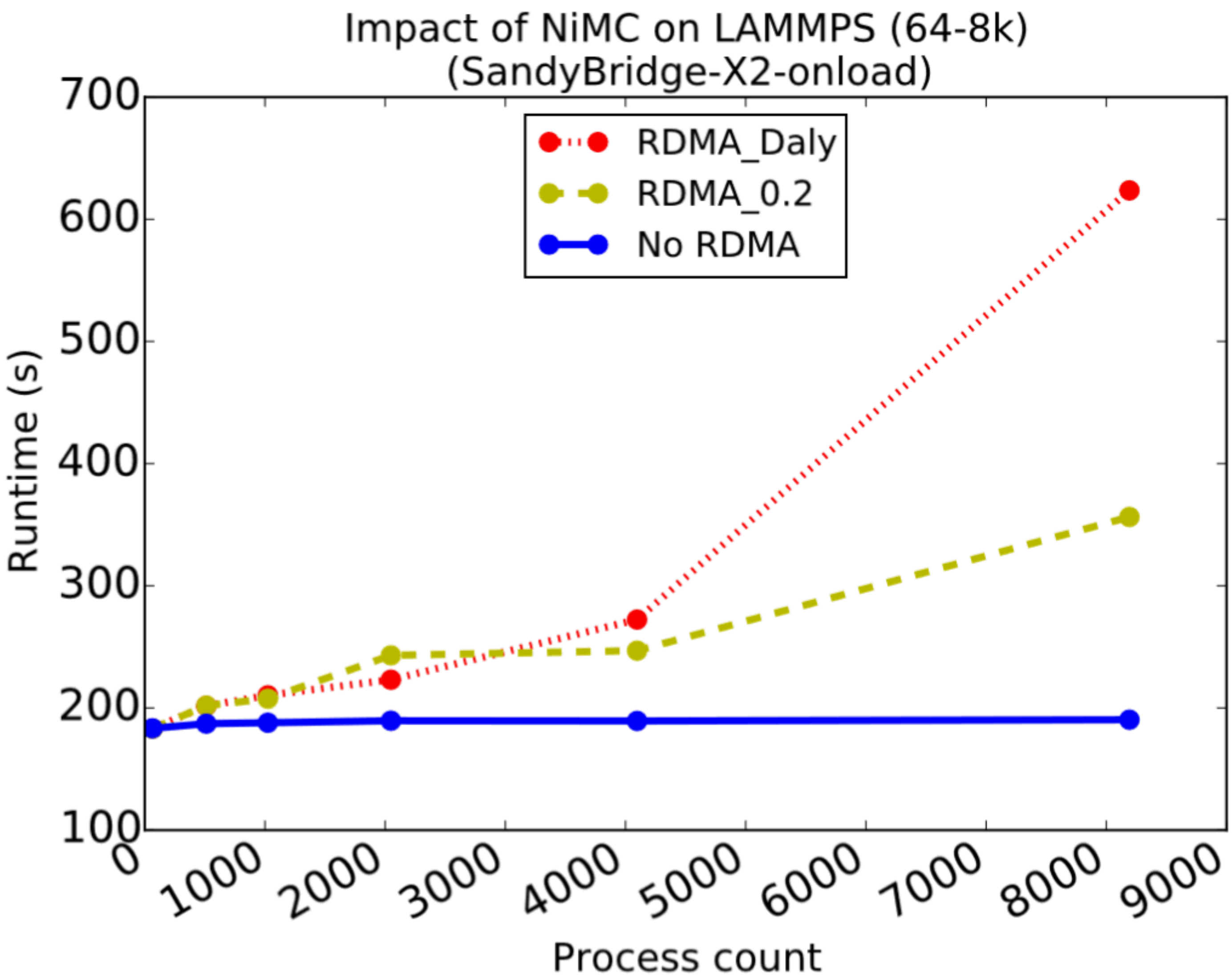
Impact at Scale

Impact of NiMC given a reasonable amount of traffic?

- Hypothetical example: uncoordinated in-memory checkpoints
  - Reduced duration of RDMA writes (1 second)
  - Only writing to a subset of nodes at any point in time:
    1. 0.2% of nodes
    2. Daly's Optimal Coordinated Checkpoint Interval as an estimate (0.2-0.5%)

Impact at Scale

RG1



Slide 33

RG1 Application Process Count

Ryan Grant, 5/17/2016

Solutions for Congestion

- Network Bandwidth Throttling

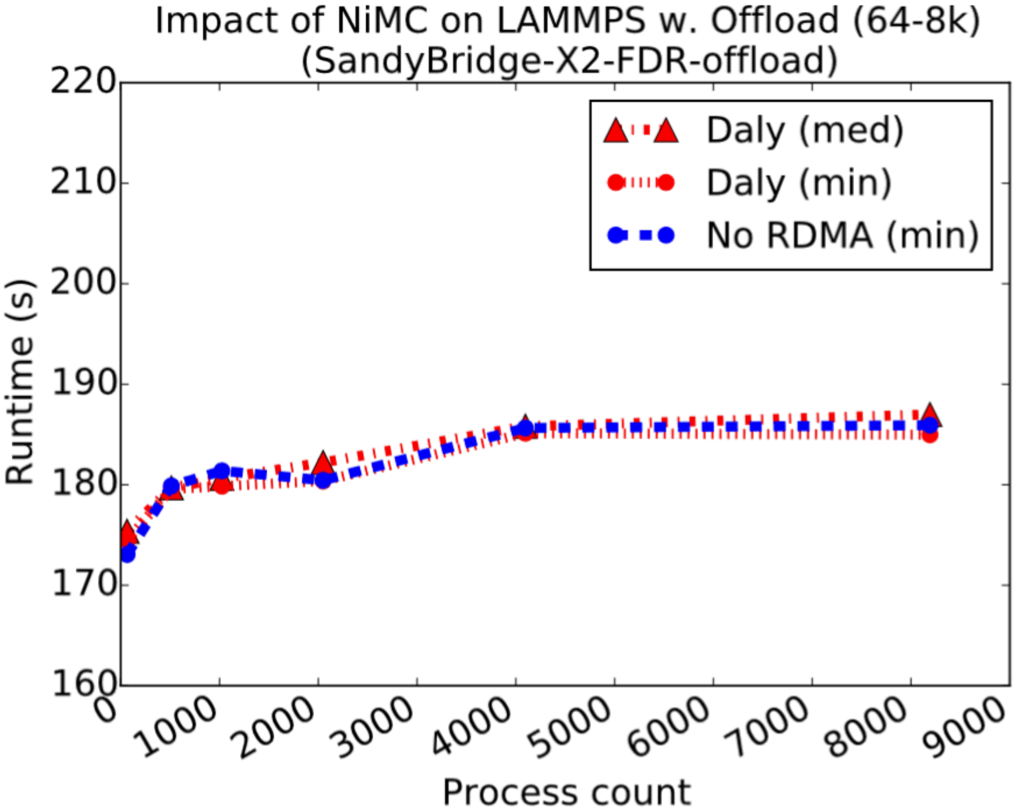
- Offload Network Cards
  - (for current-gen CPUs)

- Core Reservation

Mariordo (Mario Roberto Duran Ortiz)

Offload NIC

- Not a solution for earlier gen. CPUs (Westmere & Lisbon)
- Requires headroom between effective and theoretical memory bandwidth

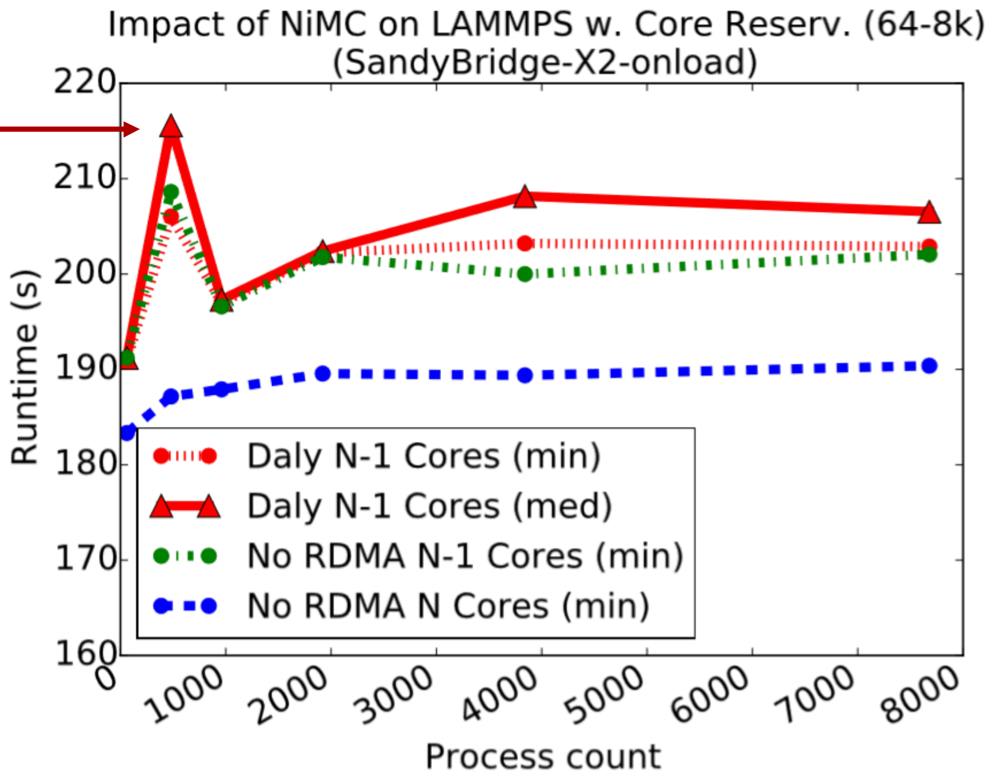


Core Reservation

- Near constant overhead (approx. 6% increase to runtime)

- Bump caused by poor mapping with 15 procs per node

- If cores are "free" this is a pretty good solution

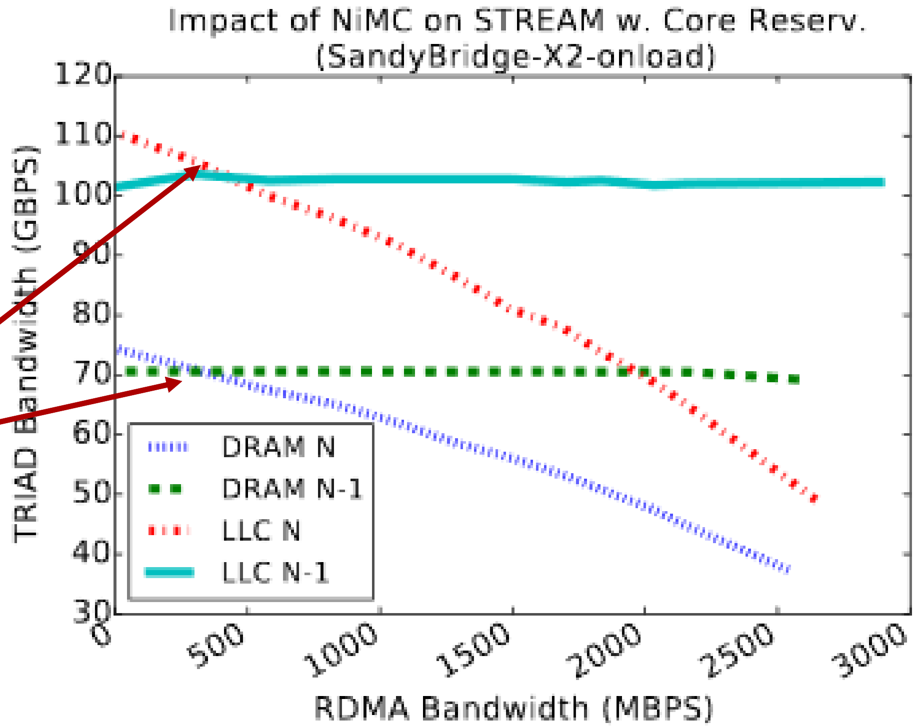


Bandwidth Throttling

- Evaluated for both LLC and DRAM

- Flat lines show core reservation

- Interesting opportunities for dynamically choosing the best solution



Key Takeaways:

- RDMA isn't free:
  - NiMC degraded performance on 6 out of 8 evaluated systems
- NiMC impact depends on architecture + workload:
  - Ranges from no impact to,
  - 3X slowdown in LAMMPS running on an onload system with 8k processes
- We can deal with NiMC, if we are conscious of its impact:
  - Offload NICs (for current CPUs)
  - Network throttling
  - Core reservation

NiMC Extension: Detect and Predict

- We know NiMC is a problem
  - (primarily for systems with onload NICs)
- 3 solutions to mitigate NiMC
- But… We don't know when to enact a solution
  - **Must be able to Detect NiMC**
- Which solution to enact (bandwidth throttling or core res.)?
  - **Must predict the impact it has on a wide range of applications**

How can we detect NiMC?

- Limited knowledge of the volume of RDMA traffic on target
- Once memory is registered, the target NIC is largely bypassing the CPU to interact with Memory
- Glean some insight from PCI-e/uncore PMU’s?
- Perhaps extract details from the driver in an onload NIC?
  - Not always available, requires privileged access

What about basic PMU’s?

- Use basic PMU’s to **detect** presence and **predict** impact?
  - L1, L2, L3, TLB, miss, hit, etc
- Readily available
- Evidence of cache pollution on machines with onload NIC’s

Random Forests get the job done

Sandia National Laboratories

Machine Learning: it's trendy (and useful)

- Statistical method to create a classification/regression model

Take the output of random forests not as absolute truth, but as smart computer generated guesses that may be helpful in leading to a deeper understanding of the problem.

-- https://www.stat.berkeley.edu/~breiman/RandomForests/cc_philosophy.htm

The Decision Tree

- Many runs of a application used to train a tree
- We have a known outcome (supervised learning)
  - e.g. a run with added RDMA or without, or a runtime
- We have a vector of features associated with a run
  - In our case we use performance counters
- Impurity measure uses features to place splits in the tree
  - Different measures of impurity like Entropy or Gini index

Random Forests

- **Many** trees instead of a single decision tree
- Classification determined by a **vote** of all trees
- Trained by N **randomly selected** (with replacement) samples
- Some subset of **randomly selected features** (counters) used to split trees

Measuring prediction error

- Out of bag (OOB) score **estimates error**
  - **Built into the algorithm**

For each sample in the bag,

1. Examine all trees not trained on the sample
2. +0 if incorrect prediction +1 if correct
3. Divide by number of trees

- Eliminates the need for separate data sets for validation

Features Sets

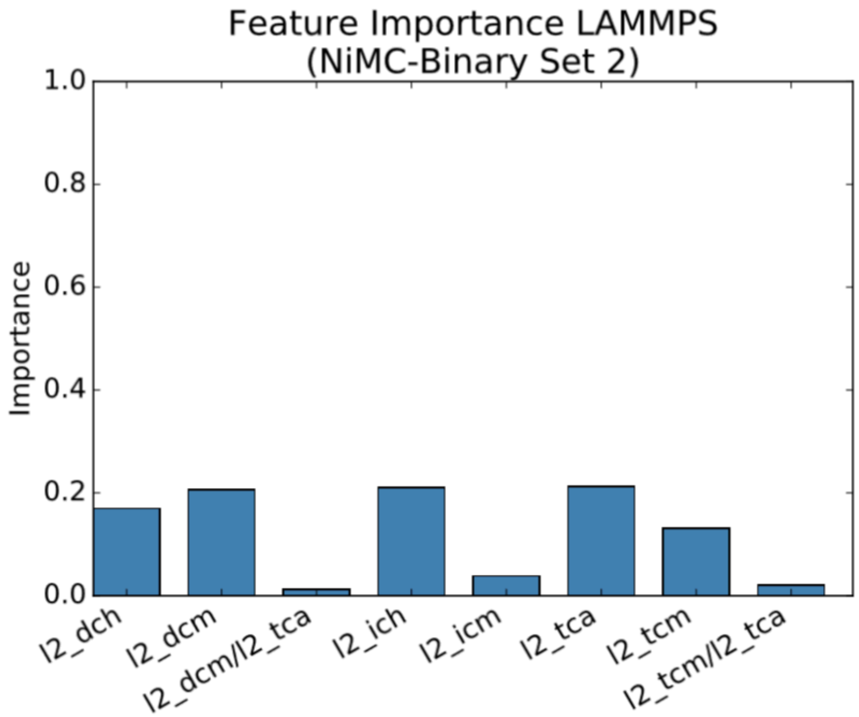
- Limited by the number of hardware counters that we can collect simultaneously
- Divided available counters into three sets and evaluated them independently
- For each set:
  - Can we detect NiMC?
  - Can we determine the volume of RDMA traffic?
  - Can we predict the application impact?

Three sets of features (counters)

- Set 1:
  - Idle cycles, L1 data cache miss L1 instruction cache miss, TLB data miss, TLB instruction miss
- Set 2:
  - L2 data cache miss, L2 data cache hit, L2 instruction cache miss, L2 instruction cache hit, L2 total cache miss, L2 total cache accesses, L2 DCM/TCA, L2 TCM/TCA
- Set 3:
  - L3 total cache miss, L3 total cache accesses, L3 instruction cache accesses, L3 data cache accesses

Feature importance

- Another bonus of Random Forest is the ability to **report feature importance**
  - *Caveat*: if features overlap in importance, one may overshadow a similar feature.



Workloads to evaluate ML

- Evaluated subset of workloads, targeting specific memory characteristics:
- CNS – Very low memory requirements, no impact from NiMC
- HPCCG – Memory intensive
- LAMMPS – Real application, with scaling data
- STREAM – synthetic memory benchmark
  - (default-DRAM and a variant that fits in L3).

Methodology Continued

- Single node runs of the workload
  - (16 procs per run, 16 threads for STREAM)
- Running on SandyBridge Onload
- Continuous RDMA writes
- 6,400 samples recorded per feature set for a given workload

Can we detect NiMC?

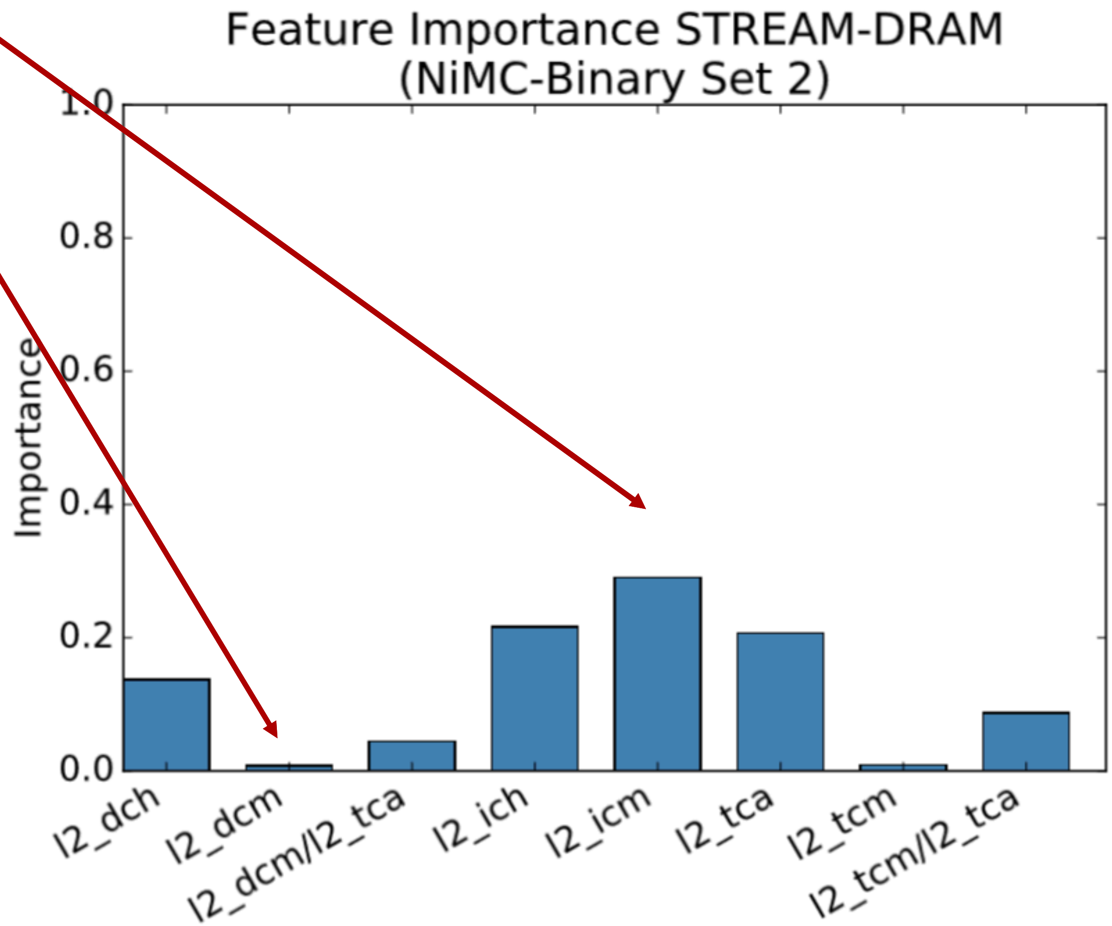
- Yes!
- CNS has a bad score, but no impact from NiMC

App/Benchmark	Set 1 Score	Set 2 Score	Set 3 Score
STREAM-DRAM	1.000	1.000	0.995
STREAM-cache	1.000	1.000	0.990
HPCCG	0.998	0.999	0.999
CNS	0.741	0.747	0.742
LAMMPS	1.000	1.000	1.000

OOB scores for forests predicting the presence of NiMC

Feature Importance (classification)

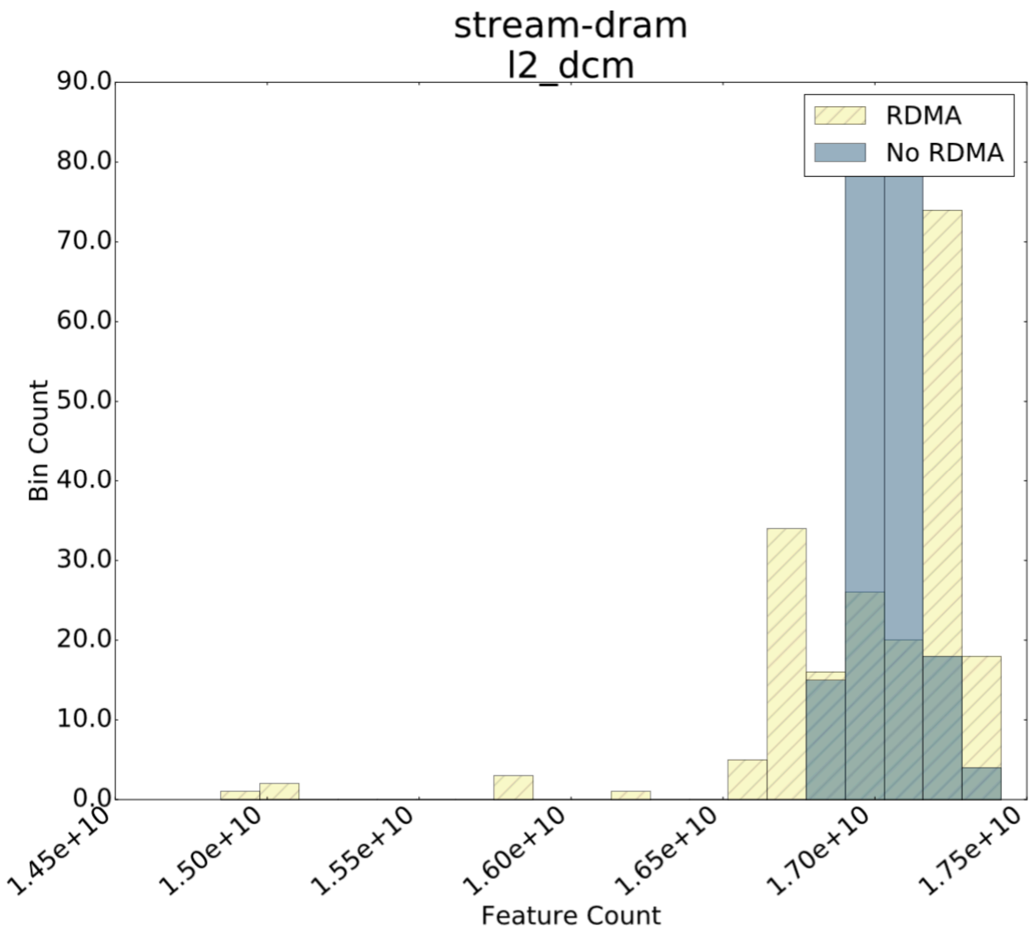
- L2 instruction cache miss -- most important
- L2 data cache miss – least important



Example 1: STREAM-DRAM feature importance

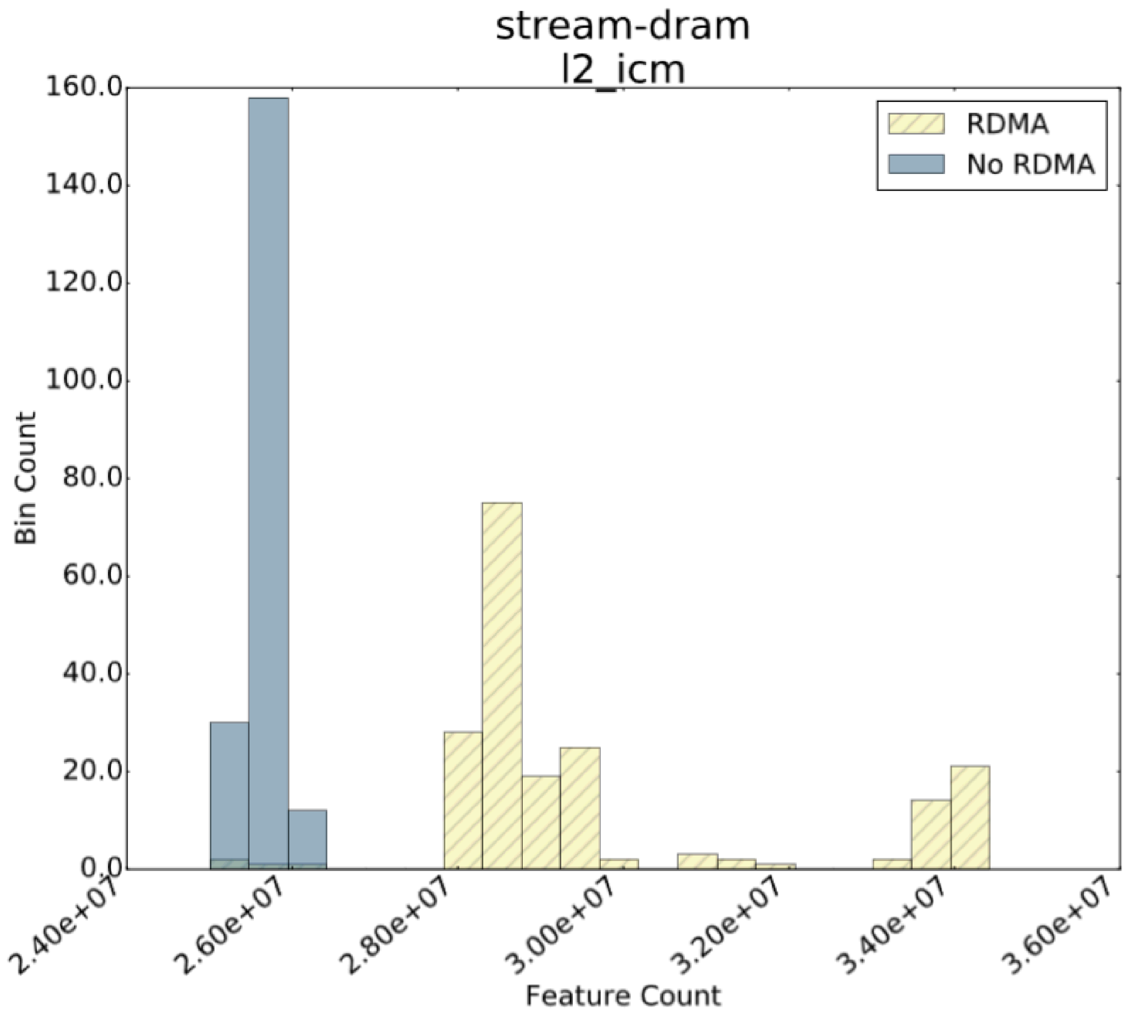
Why L2_DCM Doesn't Help

- STREAM-DRAM is designed to miss cache… alot



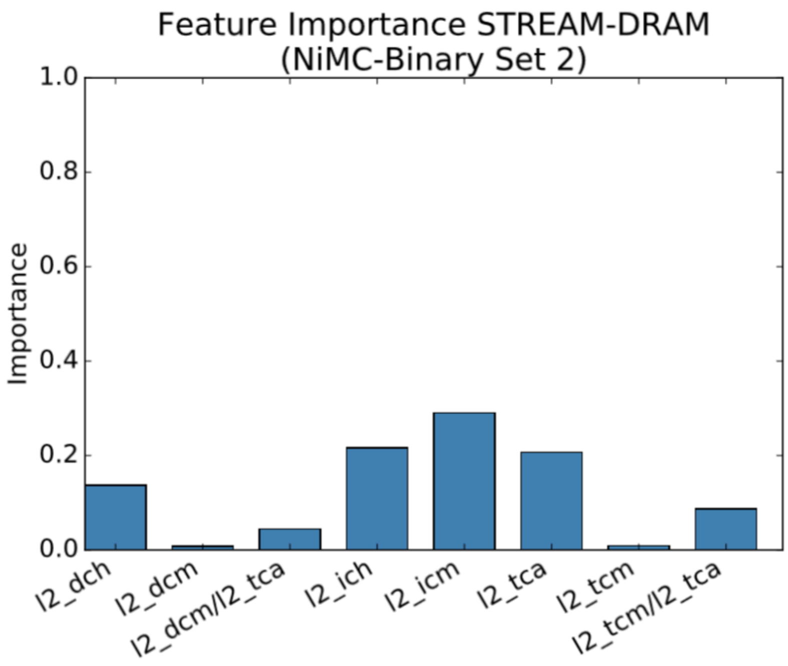
Histogram of L2_DCM for STREAM-DRAM

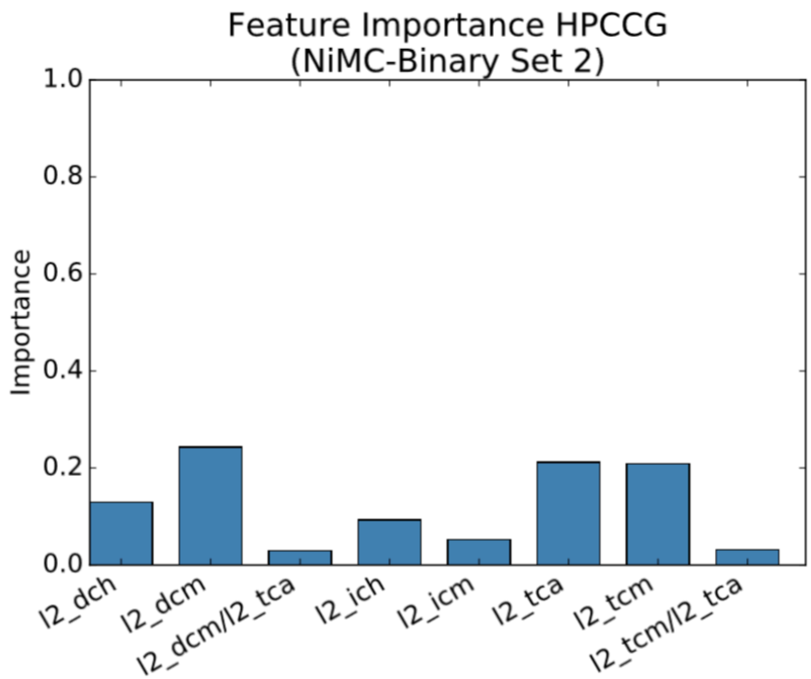
L2_ICM (Important Feature)



Universal Feature Importance?

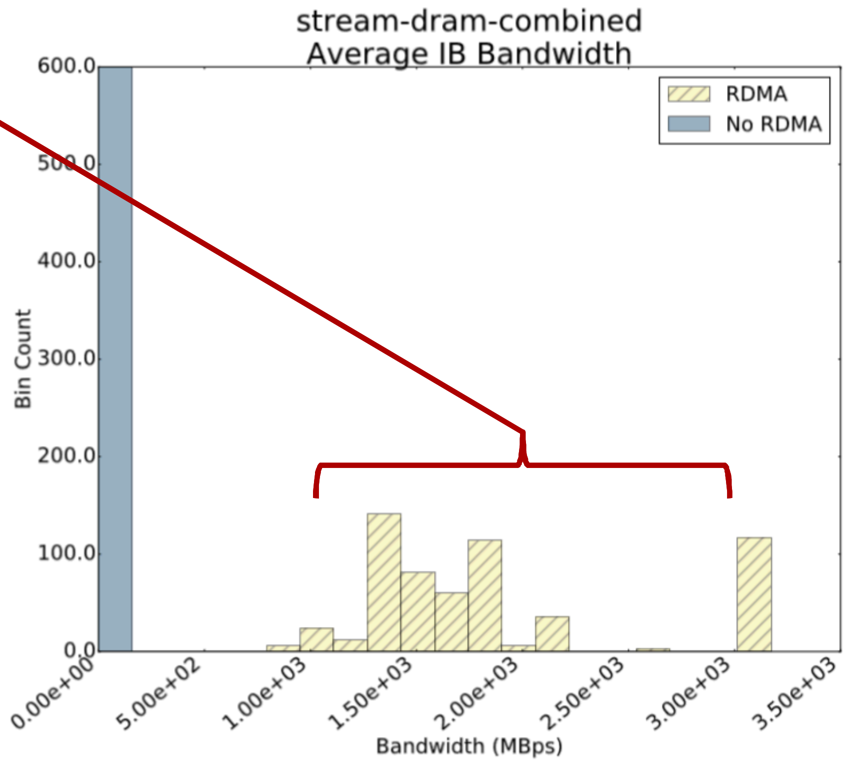
- Not really, each workload has a unique set a features that are important, this will even change as parameters shift





RF to Determine amount of RDMA?

- Not with the evaluated combination of counters.
  - Many other actors influencing counter behavior
  - RDMA Bandwidth fluctuates with nearby jobs



Histogram of bandwidth for different runs of STREAM-DRAM

Predict runtime impact?

- Surprisingly well

App/Benchmark	Set 1 Score	Set 2 Score	Set 3 Score
STREAM-DRAM	0.984	0.997	0.991
STREAM-cache	0.992	0.994	0.984
HPCCG	0.966	0.975	0.966
CNS	0.981	0.978	0.966
LAMMPS	0.990	0.995	0.967

OOB scores for runtime regression trees.

CPU Times with(out) RDMA

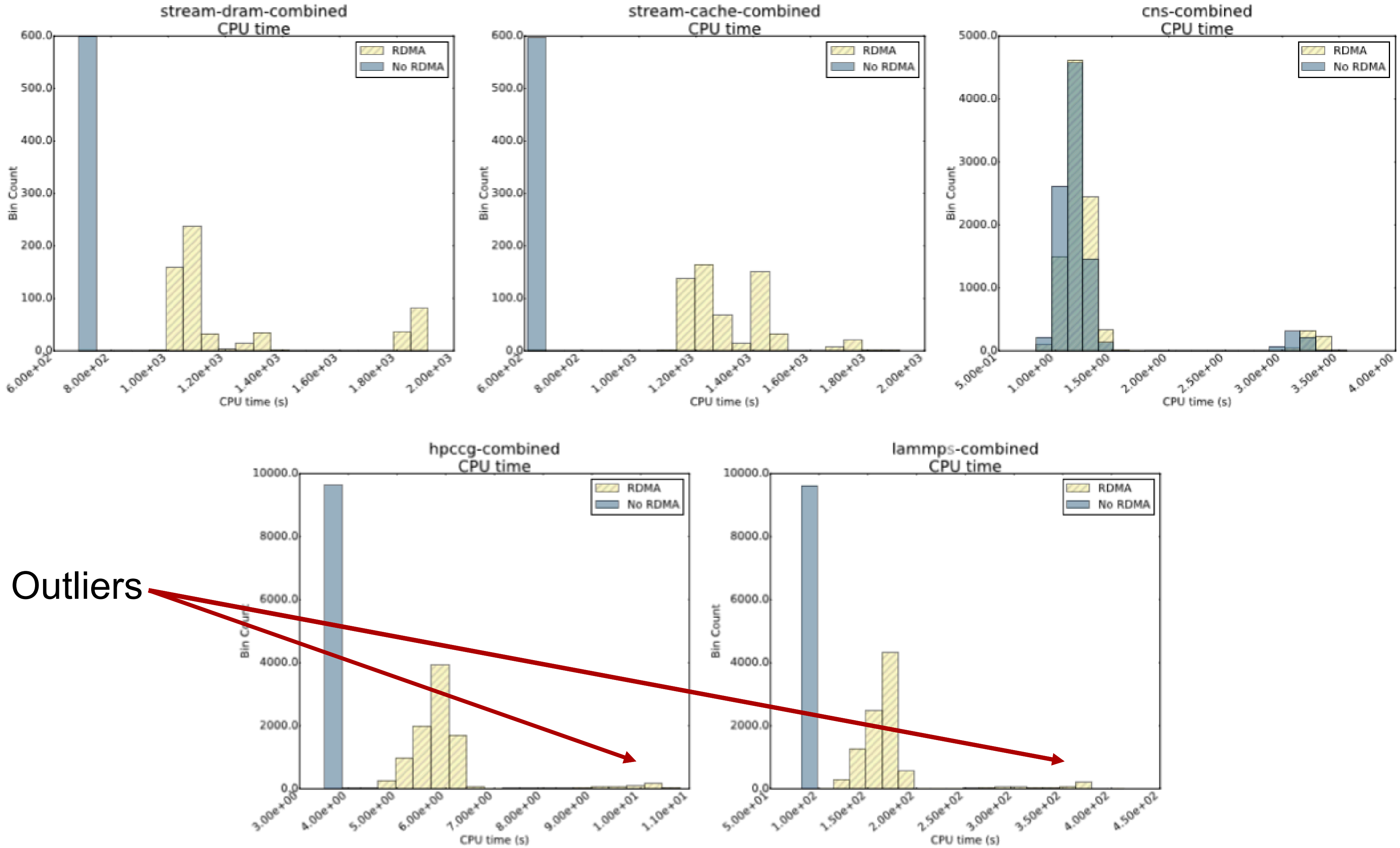
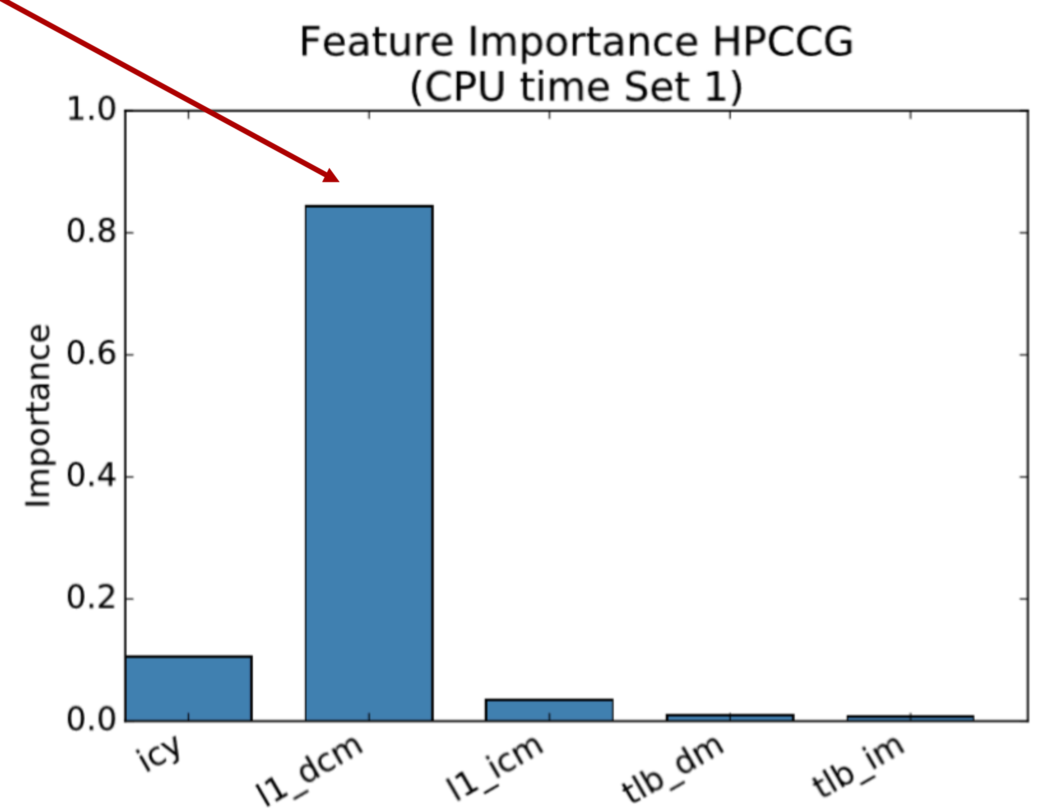


Fig. 5: Per-process CPU time (in seconds) recorded for each run of the benchmarks and applications.

Feature Importance – CPU time

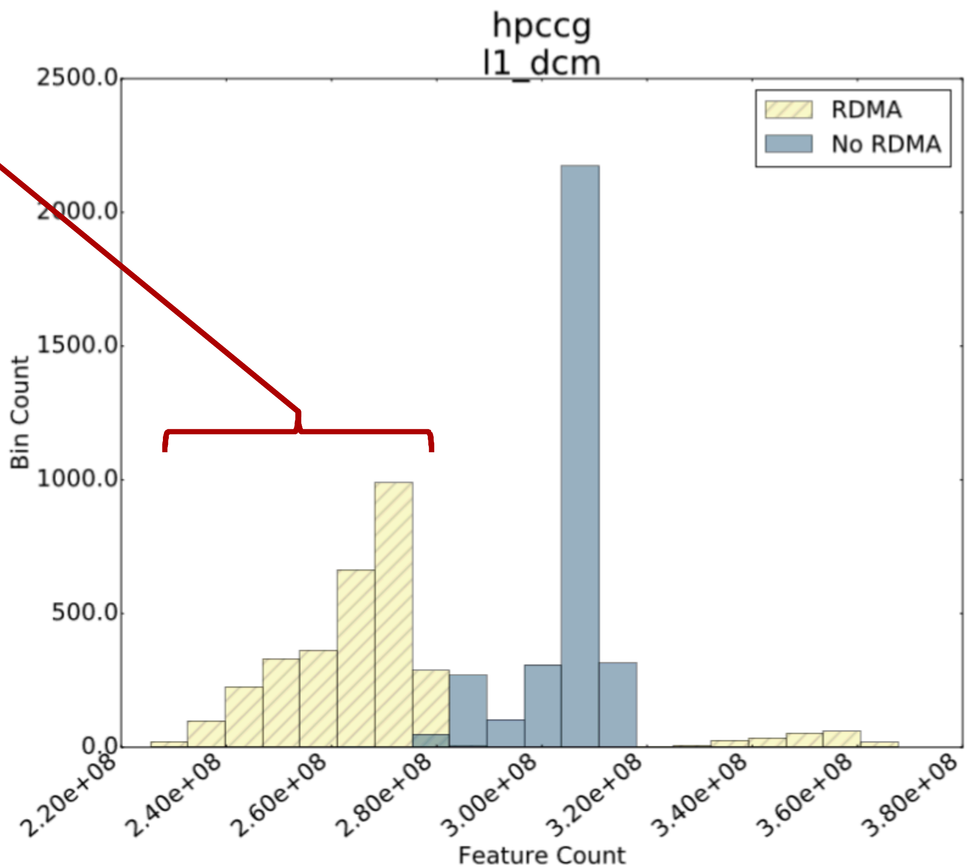
- L1_DCM is easily the most important feature from Set 1



Feature importance (Set 1) for predicting CPU time

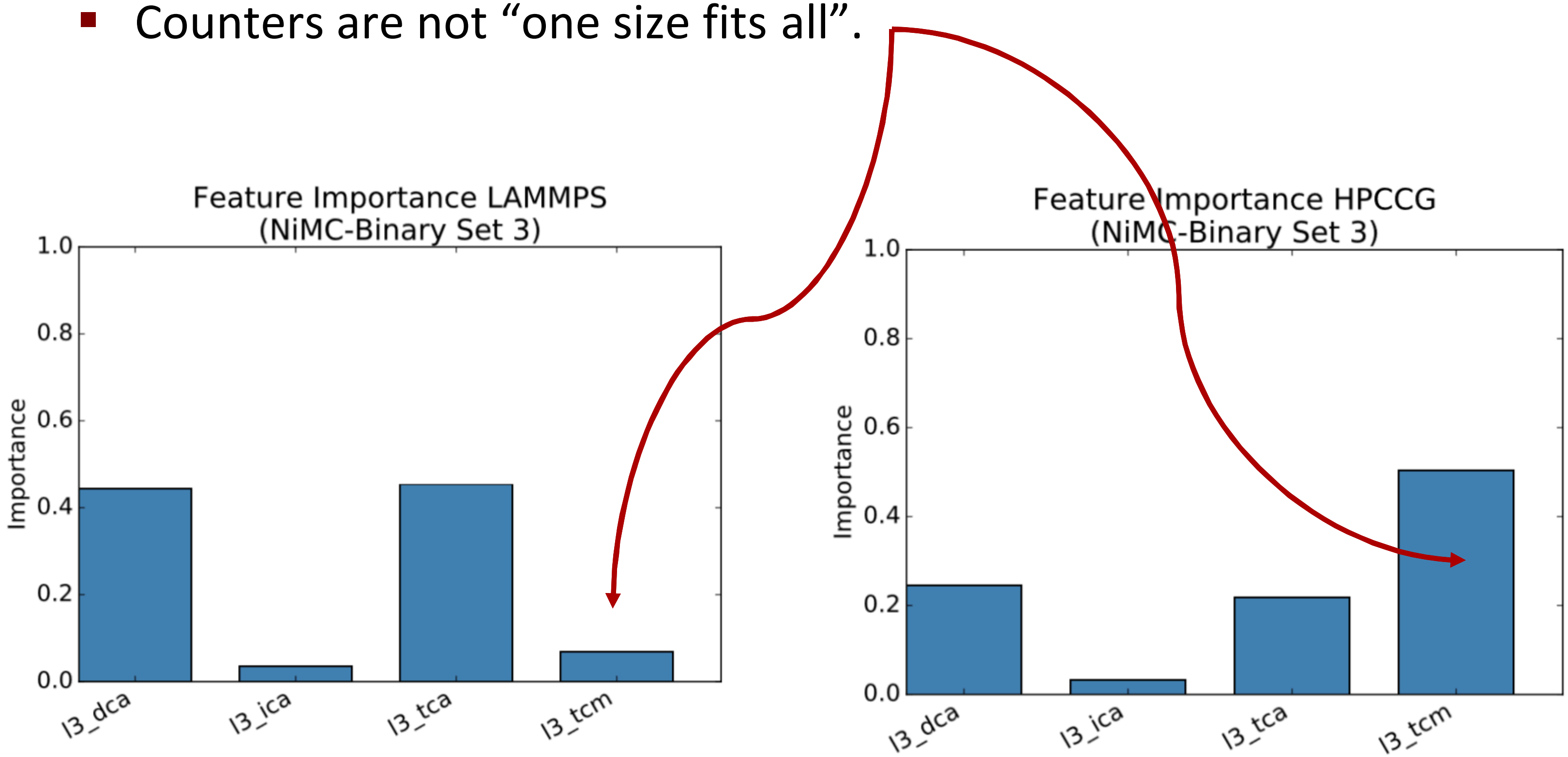
L1_DCM for HPCCG

- Wait! Why do some processes have less L1_DCMs?



Conclusions

- Workloads require customized solutions
  - Counters are not “one size fits all”.



Conclusions continued…

- Each feature set evaluated was able to detect NiMC
  - Feature sets each focused on a level of cache (L1, L2, L3)
- NiMC on onload NICs have far-reaching impact beyond just the local cache, i.e. impact in shared levels
- Furthermore, asynchronous programming models may not provide as much relief as desired
  - Even if we aren't waiting for the slowest process at a synchronization point, imbalance in the system may create bottlenecks for shared resources

Extension Part 3 (ongoing)

- Onload cards are optimized to run over PSM/PSM2
- Our experiments utilized ibverbs
- What does the full communication stack (e.g. IB + PSM + MPI) do to NiMC and RDMA performance?

Acknowledgements

- Sandia National Laboratories: Center for Computing Research
- Scalable Systems Laboratory at University of New Mexico
- Texas Advanced Computing Center

QUESTIONS?

tgroves@sandia.gov

Comm. Computation Overlap

- Bell, Christian, et al. "**Optimizing bandwidth limited problems using one-sided communication and overlap**." *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*. IEEE, 2006
- Wang, Hao, et al. "**GPU-aware MPI on RDMA-enabled clusters: Design, implementation and evaluation**." *Parallel and Distributed Systems, IEEE Transactions on* 25.10 (2014): 2595-2605.
- Subramoni, Hari, et al. "**Designing non-blocking personalized collectives with near perfect overlap for RDMA-enabled clusters**." *High Performance Computing*. Springer International Publishing, 2015.

Real-time Analytics/Viz

- Sewell, Christopher, et al. "**Large-scale compute-intensive analysis via a combined in-situ and co-scheduling workflow approach**." *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015.

- Ahern, Sean, et al. "**Scientific discovery at the exascale: report from the DOE ASCR 2011 workshop on exascale data management, analysis, and visualization**." (2011).

- Johnson, Chris, et al. "**Visualization and knowledge discovery: Report from the DOE/ASCR workshop on visual analysis and data exploration at extreme scale.**" *Salt Lake City* (2007).

Services for data staging

- Lofstead, Jay, et al. "**Extending scalability of collective io through nessie and staging**." *Proceedings of the sixth workshop on Parallel Data Storage*. ACM, 2011.
- Lofstead, Jay, Ron Oldfield, and Todd Kordenbrock. "**Experiences applying data staging technology in unconventional ways**." *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013.
- Abbasi, Hasan, et al. "**Extending i/o through high performance data services**." *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*. IEEE, 2009.
- Vishwanath, Venkatram, Mark Hereld, and Michael E. Papka. "**Toward simulation-time data analysis and i/o acceleration on leadership-class systems**." *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*. IEEE, 2011.
- Docan, Ciprian, Manish Parashar, and Scott Klasky. "**Enabling high-speed asynchronous data extraction and transfer using DART**." *Concurrency and Computation: Practice and Experience* 22.9 (2010): 1181-1204.

Other (misc.)

- Woodring, Jonathan, et al. "**On-demand unstructured mesh translation for reducing memory pressure during in situ analysis.**" *Proceedings of the 8th International Workshop on Ultrascale Visualization*. ACM, 2013.

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