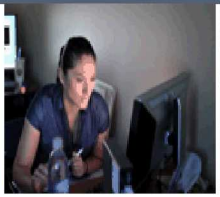
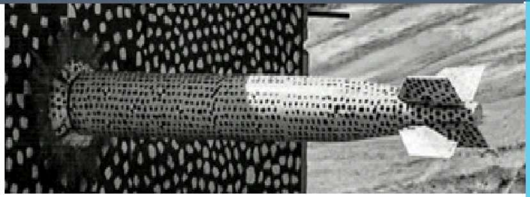
SAND2019-3232PE

Future High-Performance Networks





Ryan E. Grant
Center for Computing Research
Sandia National Laboratories
regrant@sandia.gov

PRESENTED BY

Managing the data tsunami on future generation communication networks

Center for Computing Research

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline

- General background – Why care?
- Research challenges – What's the problem?
- High speed data center networks – How to fix it?
- Self-learning networks – Why do we need them?
- Long-term Research Path – Where are we going?
- Conclusion

15 Second Research Philosophy

My Two Rules of Data Center Network Design:

#1 Avoid moving data whenever possible

#2 If you must move data do it as efficiently as possible

Background

- High performance networks (HPNs) and Cloud
  - Cost – HPC is a small market ($s)
  - Scale – HPC is a small market (volume)
  - Viability – HPC is a small market (risk)
- HPNs can only keep up if they also help Clouds
  - Clouds starting to need lower latency
  - Higher bandwidth always helps
    - Clouds catching up here already
  - Network tuning problem for HPC and Cloud

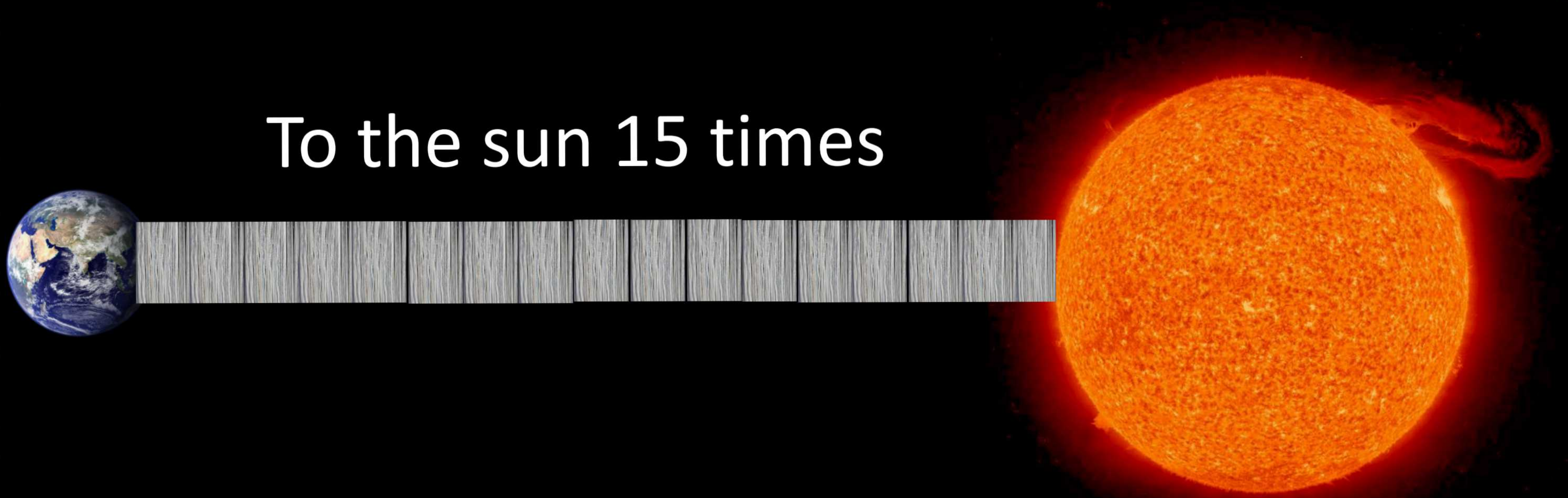
Background

- From Cisco1:
  - Annual global IP traffic will reach 3.3 ZB by 2021
  - Global IP traffic 3X by 2021 (127X 2005)
  - Smartphone traffic will exceed PC traffic by 2021
- Where does all this data go?
  - Data centers
- Data centers are the hotspots of the internet
  - HPC centers have same problem (CERN)

[1]Cisco whitepaper: Cisco Visual Networking Index: Forecast and Methodology, 2016–2021 June 6, 2017

Just how much data?

- If you printed text files:

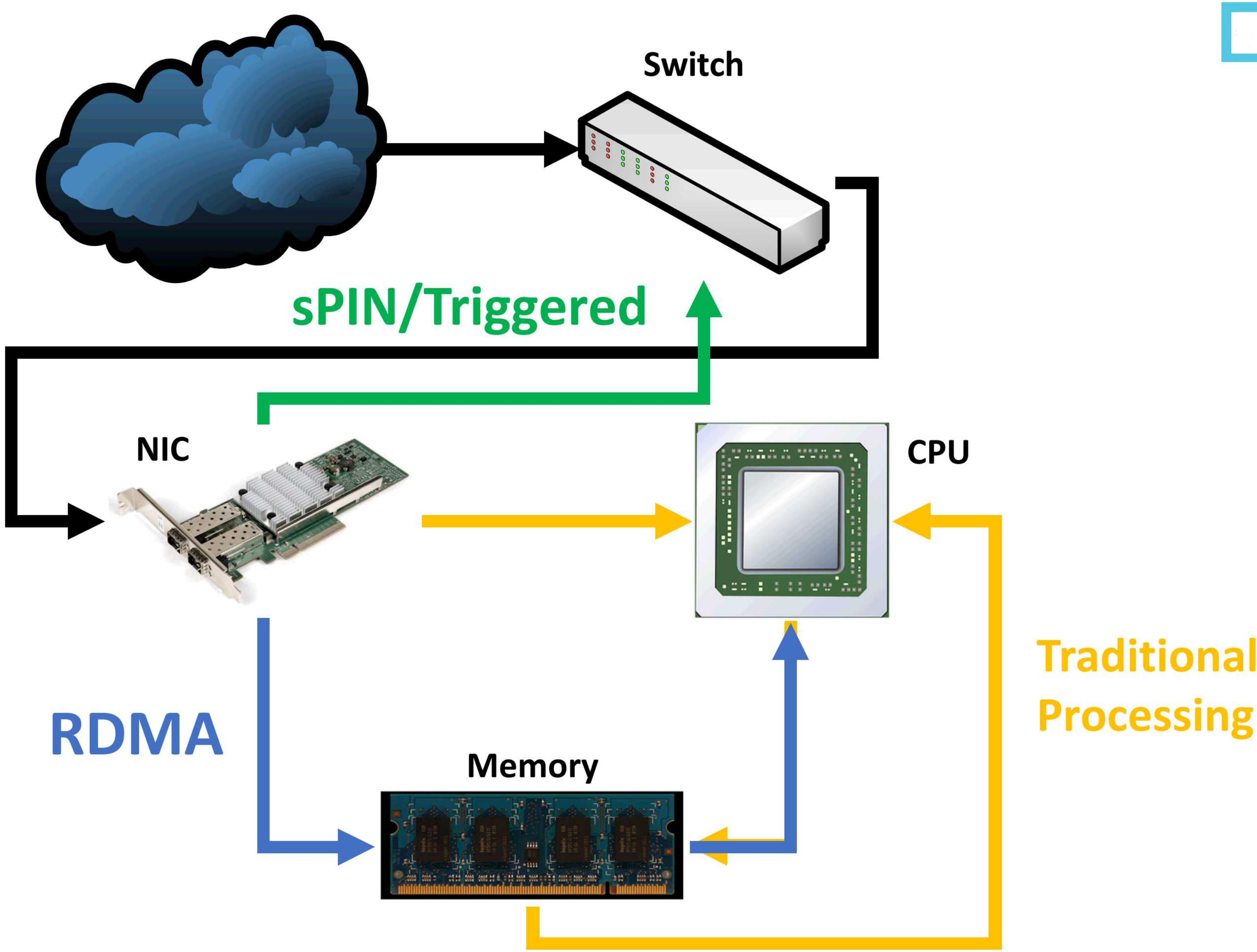


How do we handle all this data?

- Data center processing adds even more data to local network
- Key concerns: bandwidth and latency
- Leading edge – high performance networks
  - Mostly for scientific computing
  - Expensive
  - Only a few systems ever use them
  - Lucky if you can sell more than 10 big systems

What makes HPNs fast?

- Not like a normal network interface (NIC)
- Can write data directly to memory
  - No CPU or OS involvement
  - No copying (zero copy)
  - Called “OS Bypass” or user-level NICs
- Handles some message processing
  - Checksums (correctness)
  - Matching (data steering)


Switch
sPIN/Triggered
NIC
CPU
RDMA
Memory
Traditional
Processing

What makes HPNs fast?

- Switches provide minimal features
    - Fast – switching times in nanoseconds
- Efficient send-side
    - Command queues are fast
    - Addresses are known in advance
    - User-level drivers
        - No kernel level delays

Why not HPNs everywhere?

- Expensive
- Compute is cheap
- Sockets code can be slow
  - Negates the benefit of an HPN
- No need yet (but it’s coming)
  - Not compelling business case for all uses
  - Latency is still acceptable
  - Consumer devices don’t need too much bandwidth

So why do we care?

- Cloud will start dictating HPN design!
- Wireless 5G and 6G driving up demand
  - Latency way down, bandwidth up
- Machine learning everything
  - Unacceptable latency?
    - Alexa – wait a couple seconds - annoying
    - Self-driving car – deadly
  - Humans can wait a long time
  - Other computers cannot

What’s so difficult? Make it faster

- Sockets
  - Legacy – super easy to code
  - Everything coded for it
- But…An onload model design
  - Onload – CPU does it
- Latency needs means we need offload
  - Offload – NIC does it
- Fundamentally different designs

Making it faster

- Design an offload model
  - Easy to onload something designed for offload
  - Not easy to offload and onload design
- TCP offload engines are complicated
  - Also expensive
- Can we do better?
  - Don’t re-write code!

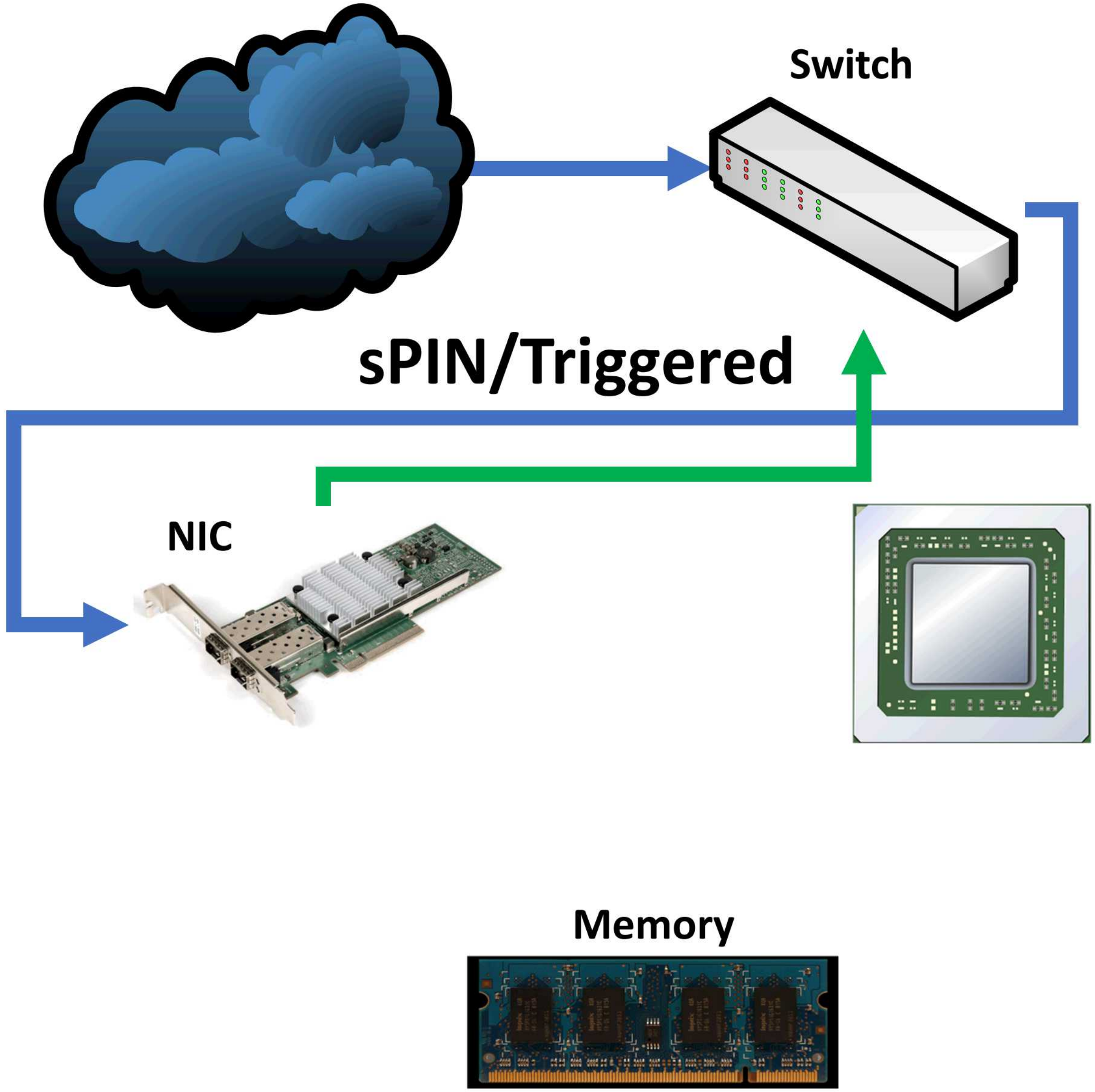
Research Challenges

- My research is to make HPNs:
  - Useful – compatible with sockets
  - Fast – best exotic networks
  - Reliable – best off the shelf hardware
  - Flexible – software defined networking flexible
  - Adaptable – adaptive to conditions
  - Deployable – not just in data centers
- We can learn from what Cloud does better
  - Clouds are reliable…

My Two Rules of Data Center Network Design:

#1 Avoid moving data whenever possible

#2 If you must move data do it as efficiently as possible


Switch
sPIN/Triggered
NIC
Memory

Avoid moving data

- Challenges in HPC and data centers – Power/Energy
- Moving data is expensive
  - In time and energy
- Orders of magnitude more energy to move from component to component
- Avoid copies
- Work on data where it is

My Two Rules of Data Center Network Design:

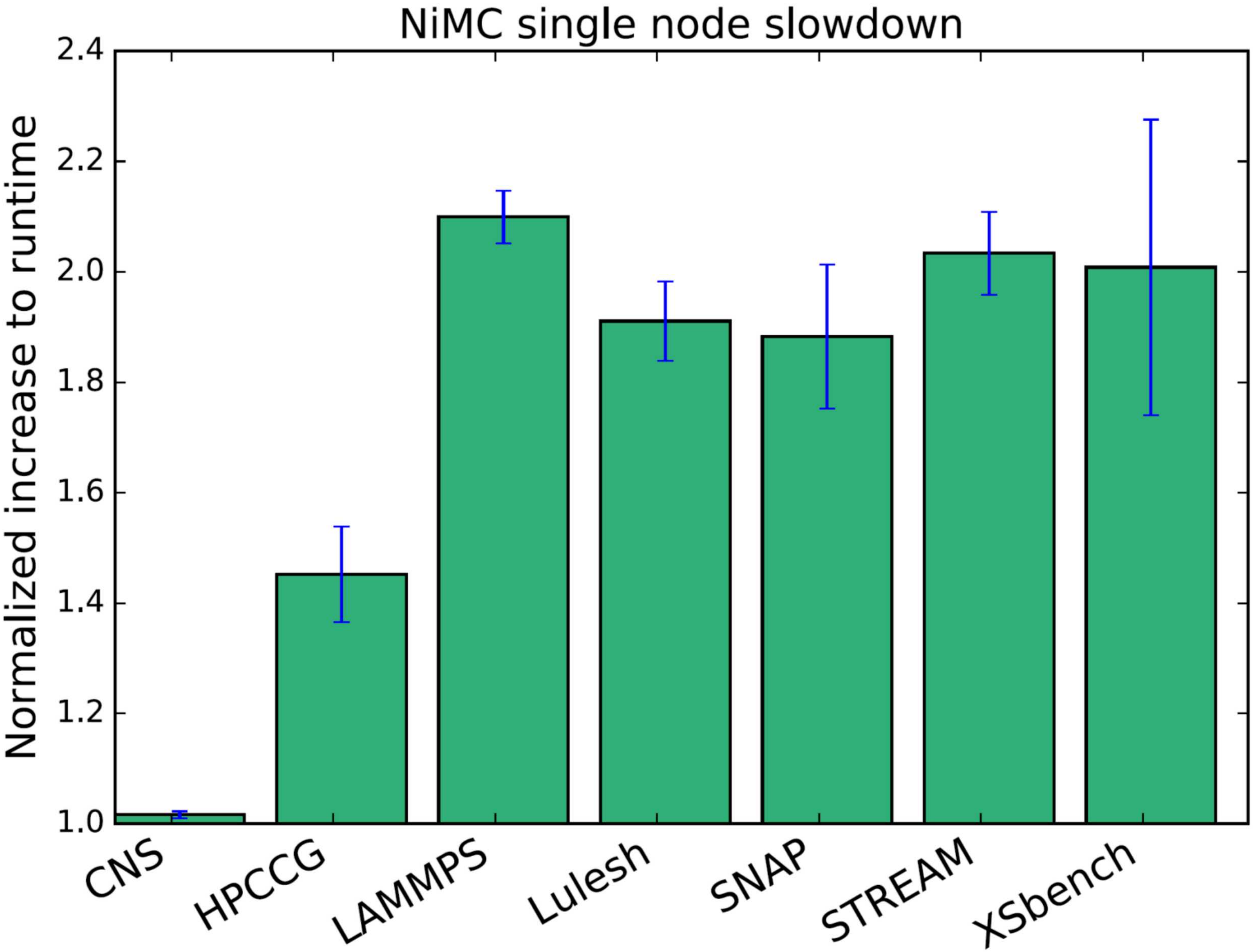
#1 Avoid moving data whenever possible

#2 If you must move data do it as efficiently as possible

Research Challenges

- Fast and Useful:
  - RDMA – Direct Memory Access
  - Competes with applications
- Hypothesis:
- Contention for memory resources should be observable and significant
- Corollary:
- If contention exists, we can avoid it

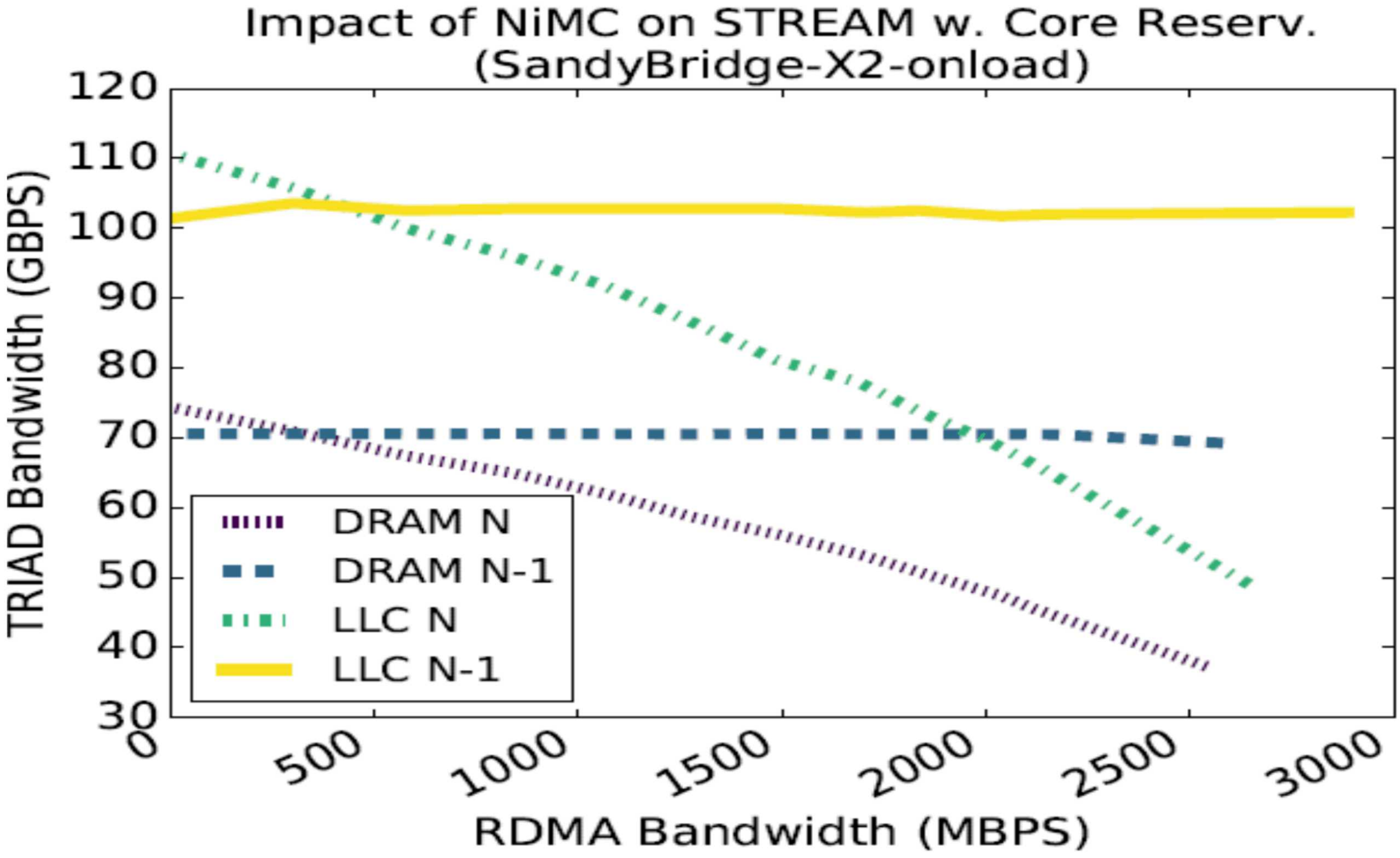
Network-induced Memory Contention



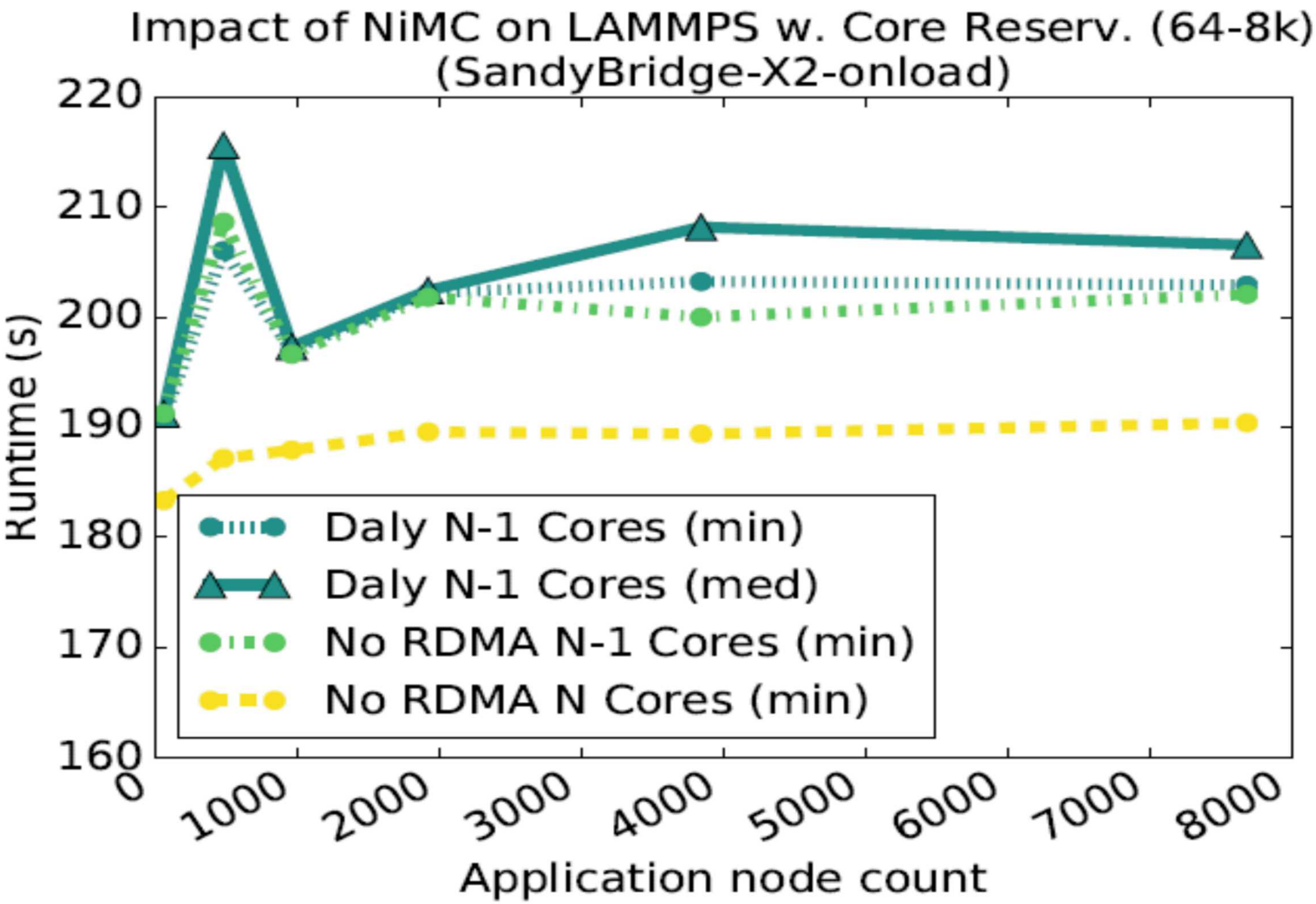
NiMC Problem

- NiMC is a problem
- Can we detect it?
  - Groves, Grant, Arnold, "NiMC: Characterizing and Eliminating Network-Induced Memory Contention", IPDPS 2016
- What can we do if we can detect it?
  - Groves, Grant, Gonzales, Arnold, "Unraveling Network-induced Memory Contention: Deeper Insights with Machine Learning" TPDS Vol 29 Iss 8

More NiMC Data



Core reservation solution



Machine Learning Results

- Online data collection is limited to performance monitoring counters
  - Can only read 3-4 at a time on most CPUs
- Choosing the right 3 characteristics
  - Can identify NiMC 99.5 times out of 100
- Worst case scenario
  - False positives
    - Unnecessary slowdowns
  - Solution – slow stream first, then allocate new resources

Vision of the Future

- sPIN: Need non-deadline based methods too
  - Leverage/extend existing hardware
  - Break deadline – allow rescheduling

- NiMC: The RDMA model is part of the problem
  - RDMA is the local memory subsystem model
  - Issues with knowing when operations complete
  - Reserved resources per peer

Research Vision

- Self-learning NICs
  - Use triggered operations
  - Current work: Turing complete
  - Speed grows with line rate
  - Eviction of programs
    - Reschedule – feed back into pipeline
  - No longer dependent on remote packets

Research Vision

- Breaking dependency on streams
  - Use spare cycles to "learn"
  - Adjust network parameters
  - Yield whenever work to do
  - NICs are idle 90%+ of the time
    - When doing compute intensive work
  - Min. speed: 1 operation per packet
  - SERDES rates mean 3+GHz speeds

Research Vision

- Accelerate Machine Learning
  - Prep data
  - Work on data
- Use simple ML Techniques
  - Good area for collaboration
- Move techniques out to the compute
  - Rule #1

RDMA-next

- Re-design RDMA
  - Memory model -> Operations model
  - Know how much data to expect
    - Build in knowledge
  - Don’t know how much data?
    - Build in buffer data thresholds
  - Abstract away specific resource allocation
    - No more reservations required

Summary

- #1 Avoid moving data whenever possible
  - sPIN today
  - Self-learning NICs tomorrow
- #2 If you must move data do it as efficiently as possible
  - NiMC and machine learning today
  - RDMA-next tomorrow

Thank you

Questions?