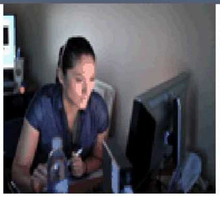
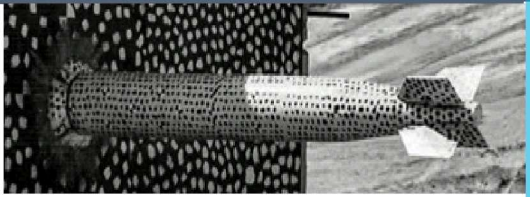
SAND2020-0853PE

Intelligent High-Performance Networks Via INCA





PRESENTED BY

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

Center for Computing Research

Outline

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

15 Second Research Philosophy

Our 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

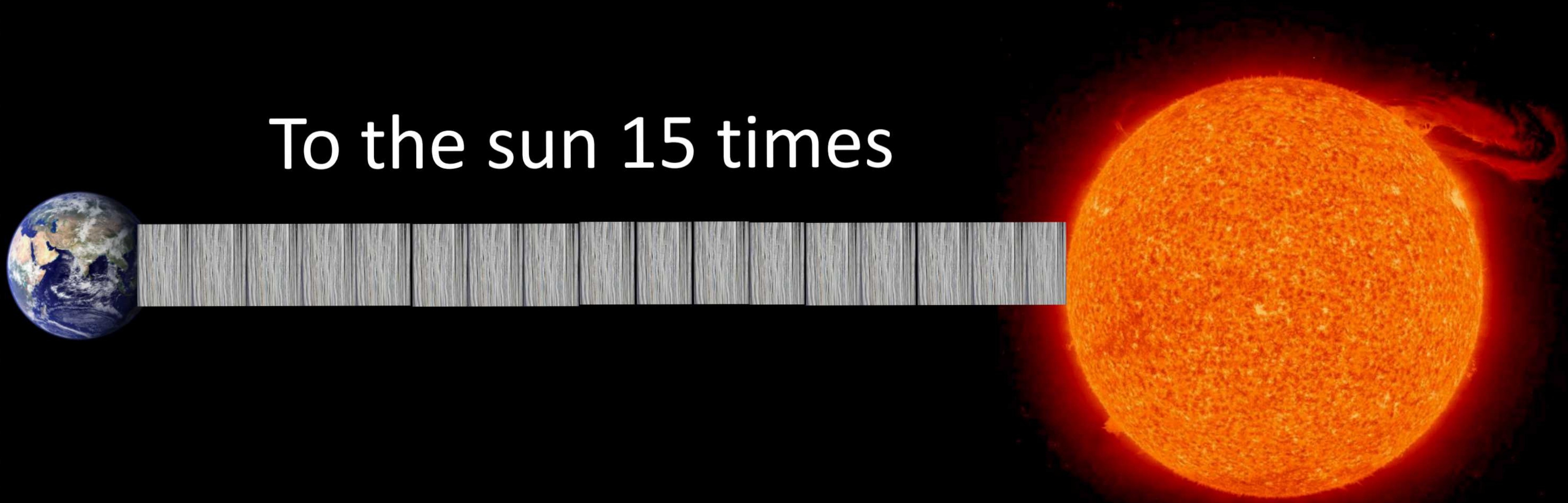
Background

- From Cisco[1]:
  - 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:

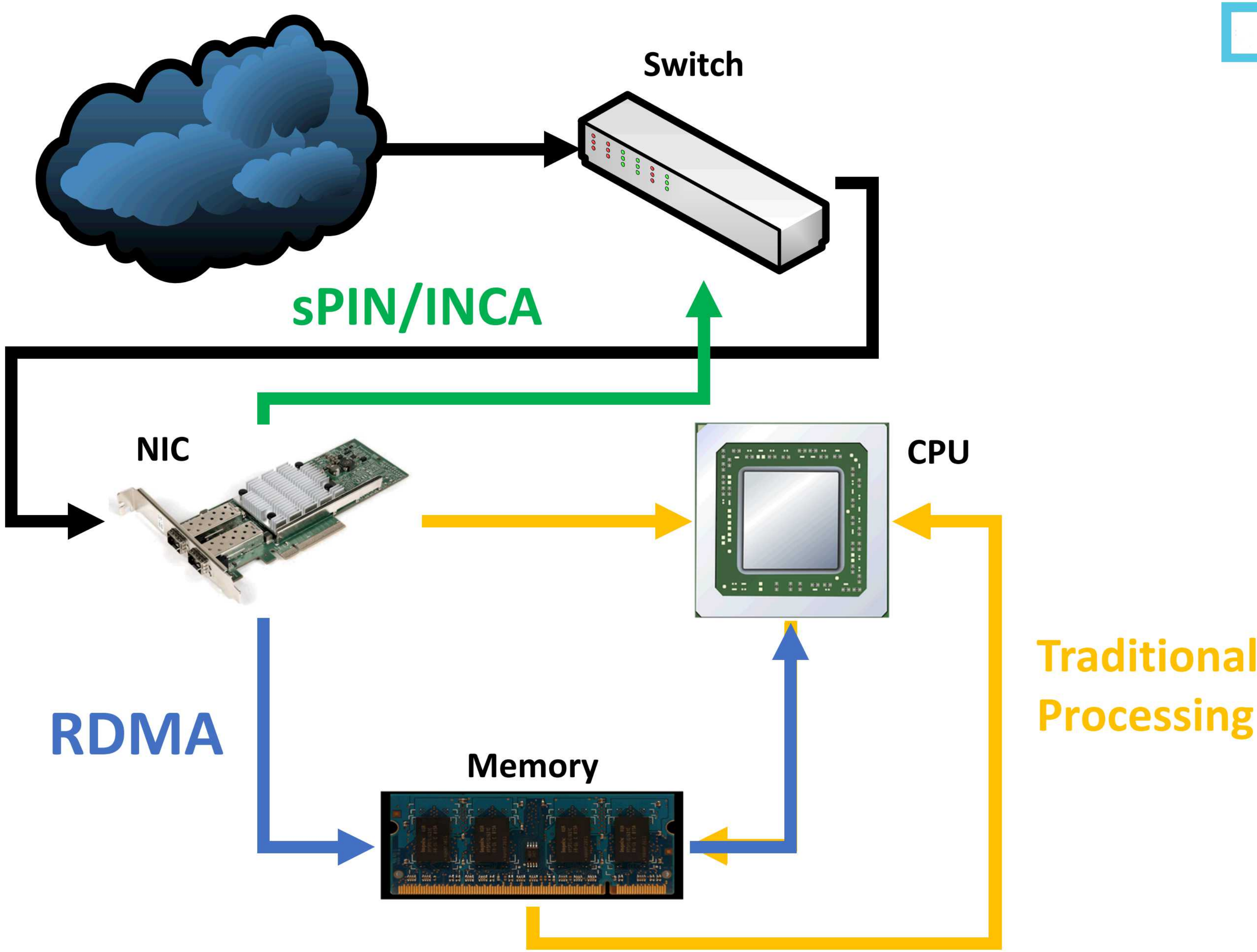


How do we handle all this data?

- 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/INCA
NIC
CPU
Traditional Processing
RDMA
Memory

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
  - But: network speeds are driving chip package sizes which is leading to lots of extra silicon available

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 an onload design
- TCP offload engines are complicated
  - Also expensive
- Can we do better?
  - Don’t re-write code!

Research Challenges

- Our 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…

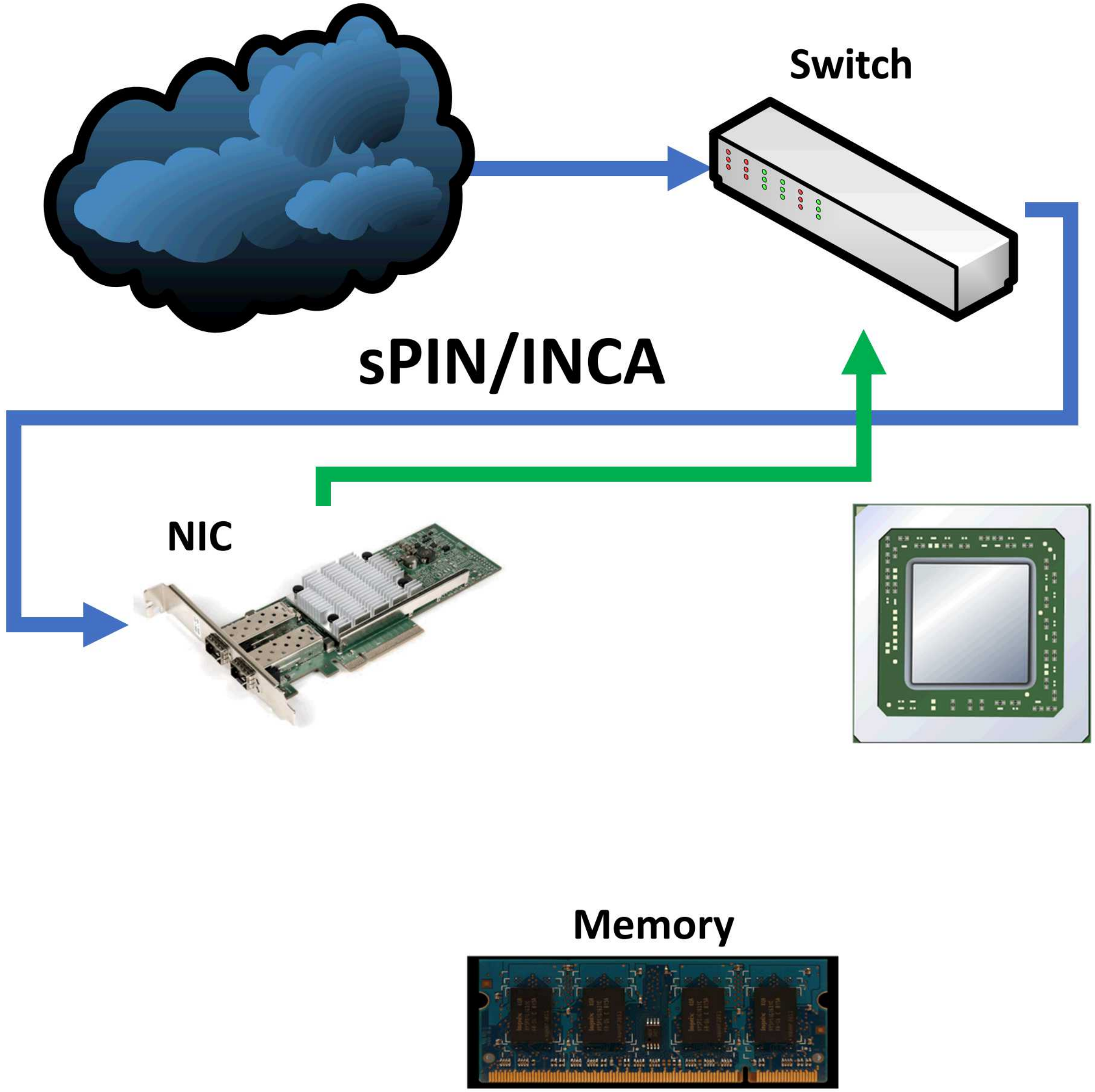
Our Two Rules of Data Center Network Design:

#1 Avoid moving data whenever possible

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

Goal: 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


Switch
sPIN/INCA
NIC
Memory

Problem = Opportunity

- HPNs need large die edge area to drive network speeds
- Large perimeter means large area
  - Don’t need all of this die area for NIC logic
- HPNs are also driving to much smaller process size very quickly
  - From 10s of nm to 5-7 nm
- Explosion of area that is unused
  - Use this to do work on data

Compute in Network

- Packet processing engines
  - Only work on packets flowing through network
- HPNs are like highways
  - Rush hour is only a small fraction of total usage time
  - 80-90% idle or low activity
- Traditional in-network compute
  - Unlimited data coming in, so limited time to work on it
- INCA approach
  - Limited data coming in, unbounded time to work on it

INCA

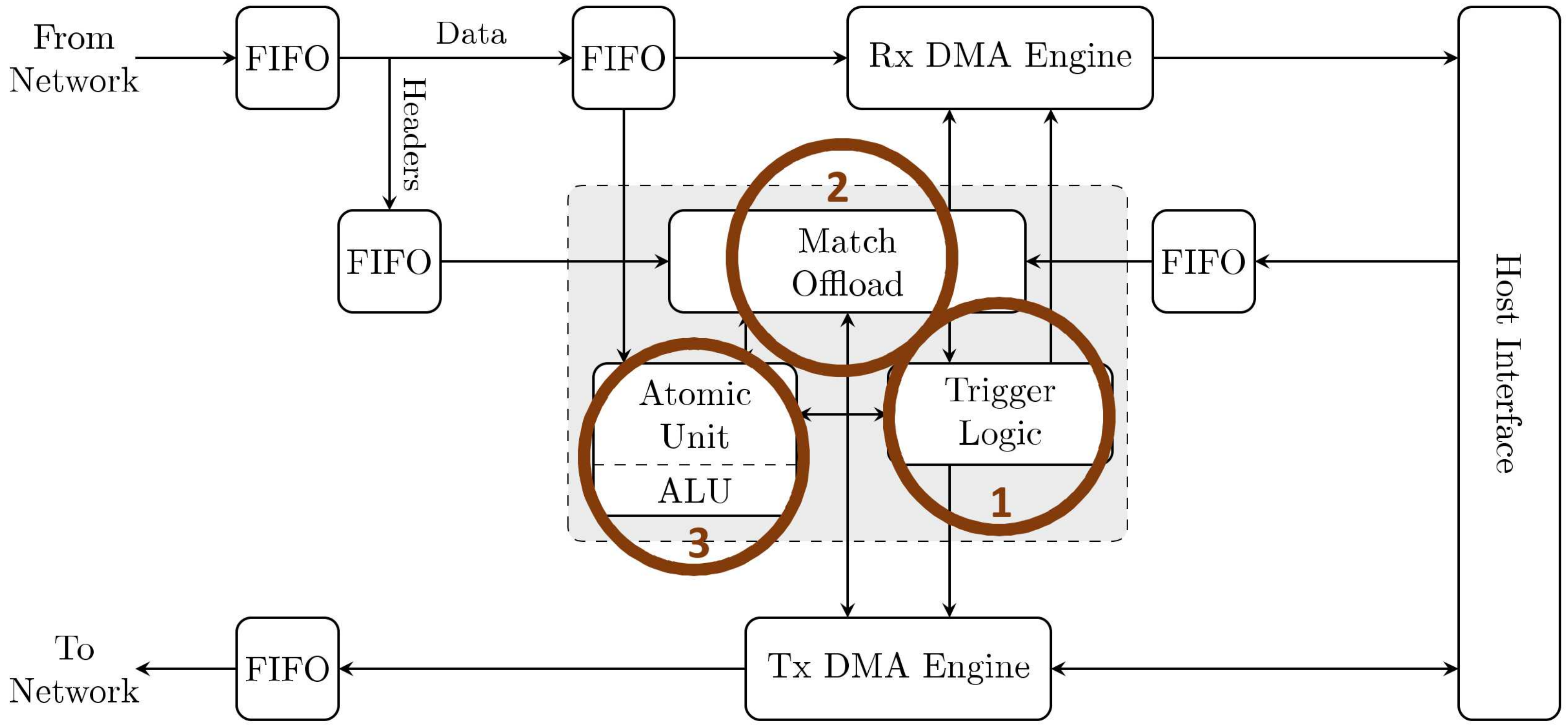
- Two strategies for offloading:
  - In-pipeline compute capabilities
  - ‘Processing near NIC’

- A third way:
  - Leverage existing application-specific, in-pipeline resources
  - Provide general-purpose compute capabilities
  - Put resources to work when network idle
  - Avoid imposing deadlines

INCA

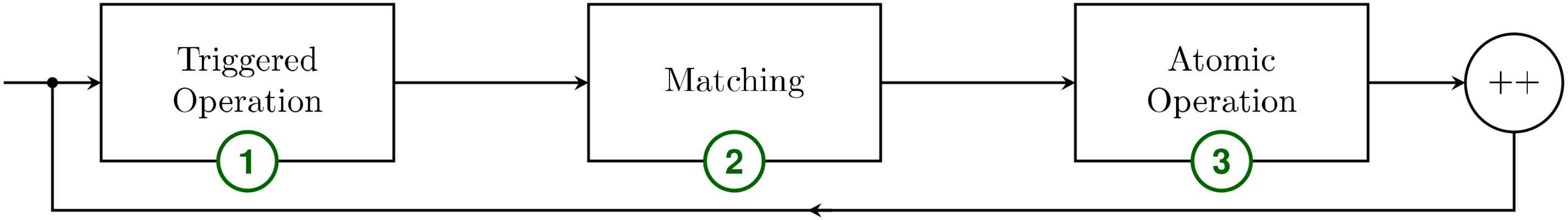
	Inside Pipeline	Outside Pipeline
Deadline	Myrinet Quadrics SPiN (SC’17) Atos BXI Broadcom Stingray Azure	
Deadline-Free	**INCA**	Mellanox Bluefield

INCA



1. Triggered Operations 2. Message Matching 3. Atomic operations

INCA



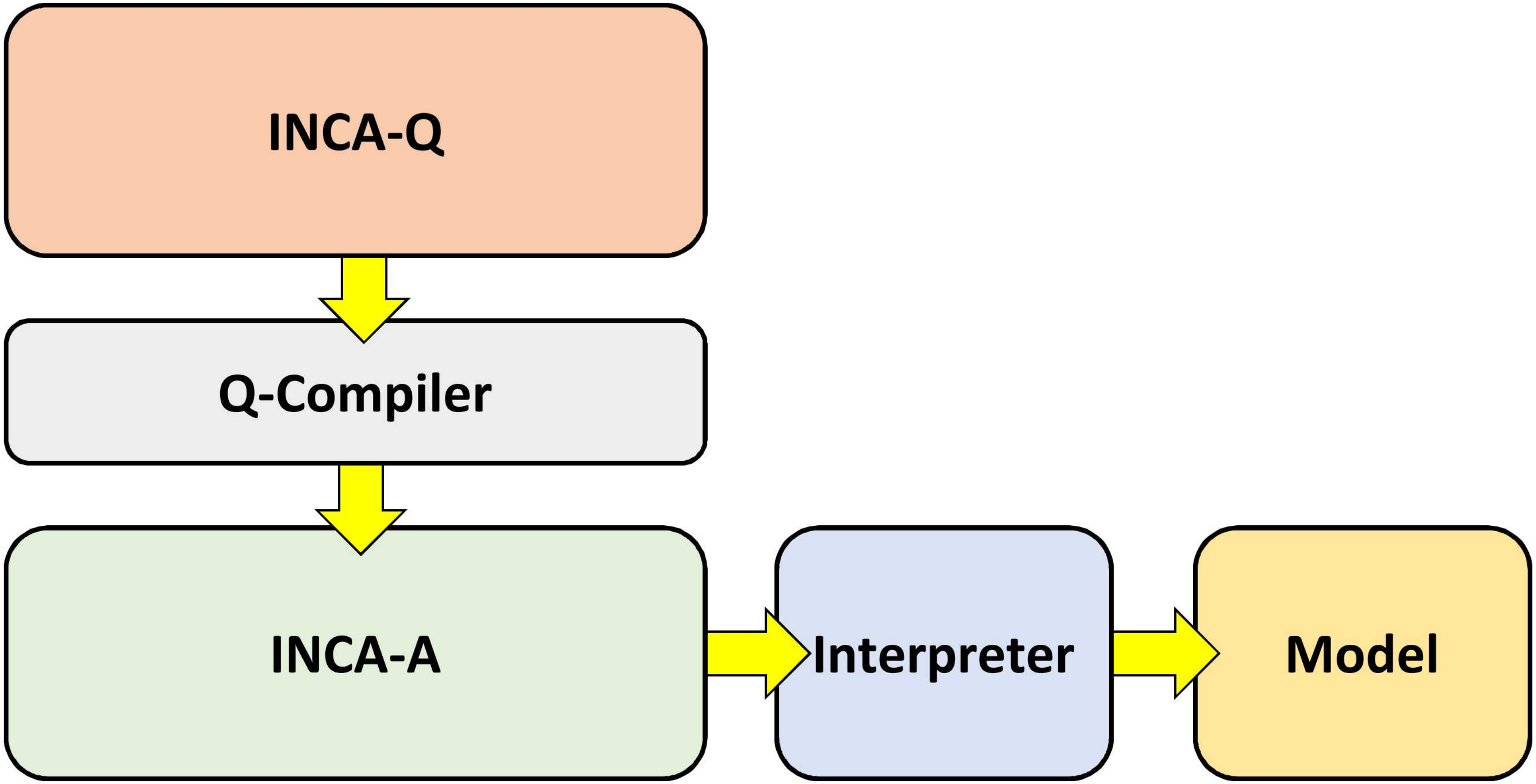
1. Triggered operation generates message containing 1st argument.

2. Unique matching element specifies buffer containing 2nd argument and atomic.

3. Atomic unit performs specified operation and stores result.

- Program: A list of tuples of triggered operations, matching entries, and atomic operations, ordered by triggering thresholds, sharing the same counter.

INCA



INCA

Algorithm 1 INCA-Q Dot Product

```
i = 0
while i < 50 do {
    c = c + (A[i] * B[i])
    i = i + 1
}
```

Algorithm 2 INCA-A Dot Product

1 PUTL i, 0
2 PUTL r_0, i
3 LT r_0, r_0, 50
4 BLEZ r_0, 10
5 PUTL r_1, A[i]
6 MUL r_1, r_1, B[i]
7 ADD c, c, r_1
8 ADD i, i, 1
9 JMP 2
10 END

INCA

- Kernels:
  - Matrix transposition
  - Filter
  - Matrix unpack
  - Convolution
  - Linear interpolation
  - Hadamard product
  - Dot product
  - Matrix multiplication

INCA

- Model parameters:
  - 200 millions messages/second
  - Scratchpad scenario:
    - 1 MiB local scratchpad memory
    - 1ns access time
  - Negligible loopback latency
  - Vary payloads (work) from 128B to 8192B

INCA – Matrix Multiply

	Payload							
Scenario	128B	256B	512B	1024B	2048B	4096B	8192B	Average Speedup wrt scratchpad
scratchpad	7.89	30.61	53.91	213.88	400.25	1597.64	3088.59	

8KiB on 2.[illegible]GHz Haswell CPU
10.56 - 139.49 microsecs

All times microseconds

INCA – Matrix Multiply

	Payload							
Scenario	128B	256B	512B	1024B	2048B	4096B	8192B	Average Speedup wrt scratchpad
scratchpad	7.89	30.61	53.91	213.88	400.25	1597.64	3088.59	
parallel	1.13	3.61	7.07	26.59	52.92	208.86	417.68	7.68x
advanced-parallel	0.29	1.10	1.50	5.89	7.82	31.29	47.42	42.09x

8KiB on 2.3GHz Haswell CPU
10.56 - 139.49 microsecs

All times microseconds

INCA

- Use case: Application Acceleration

- (Mini)Apps
  - MiniAMR
  - MiniMD
  - MiniFE
  - LAMMPS

- Methodology:
  - Identify regions of code as candidates for INCA offloading
  - Time those candidates and the regions they appear in
  - Calculate ideal speedup assuming 100% overlap

INCA

	MiniAMR	MiniMD	MiniFE	LAMMPS
Potential speedup without code refactor		11%	2.98%	11.50%
Potential speedup with code refactor	26%	37.20%	25.70%	28.90%

INCA

- INCA pros:
  - General purpose offloaded compute capabilities
  - Compute speed increases with network speed
  - Harvest idle network resources
  - Deadline-free kernel execution
  - Fast handoff
- INCA cons:
  - Slow out of the box
    - But: Ample opportunities for acceleration through additional hardware (SIMD or more exotic)
  - Busy network = no INCA kernel progress
    - But: our goal is to make it no worse off than if no acceleration were used

Our Two Rules of Data Center Network Design:

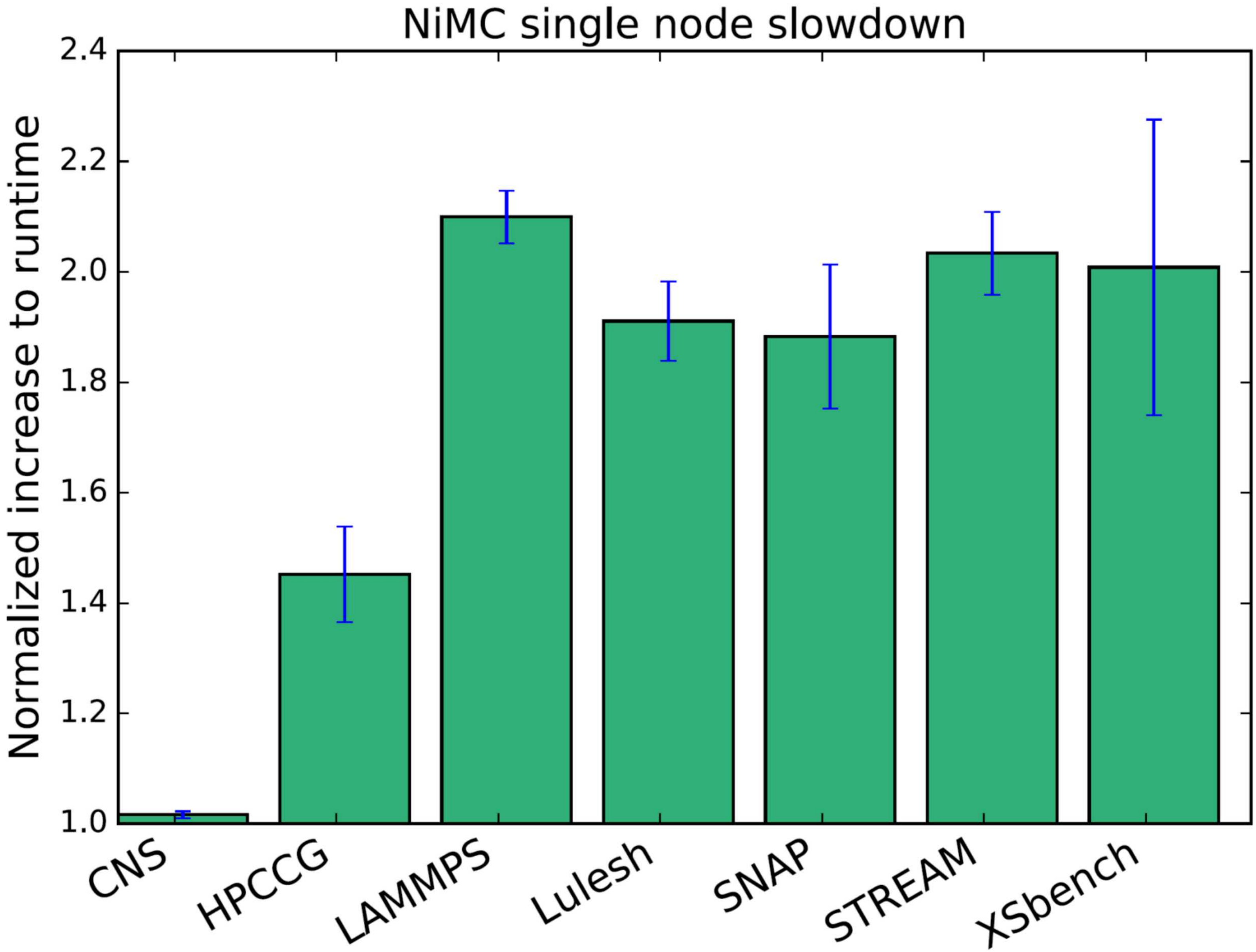
#1 Avoid moving data whenever possible

#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

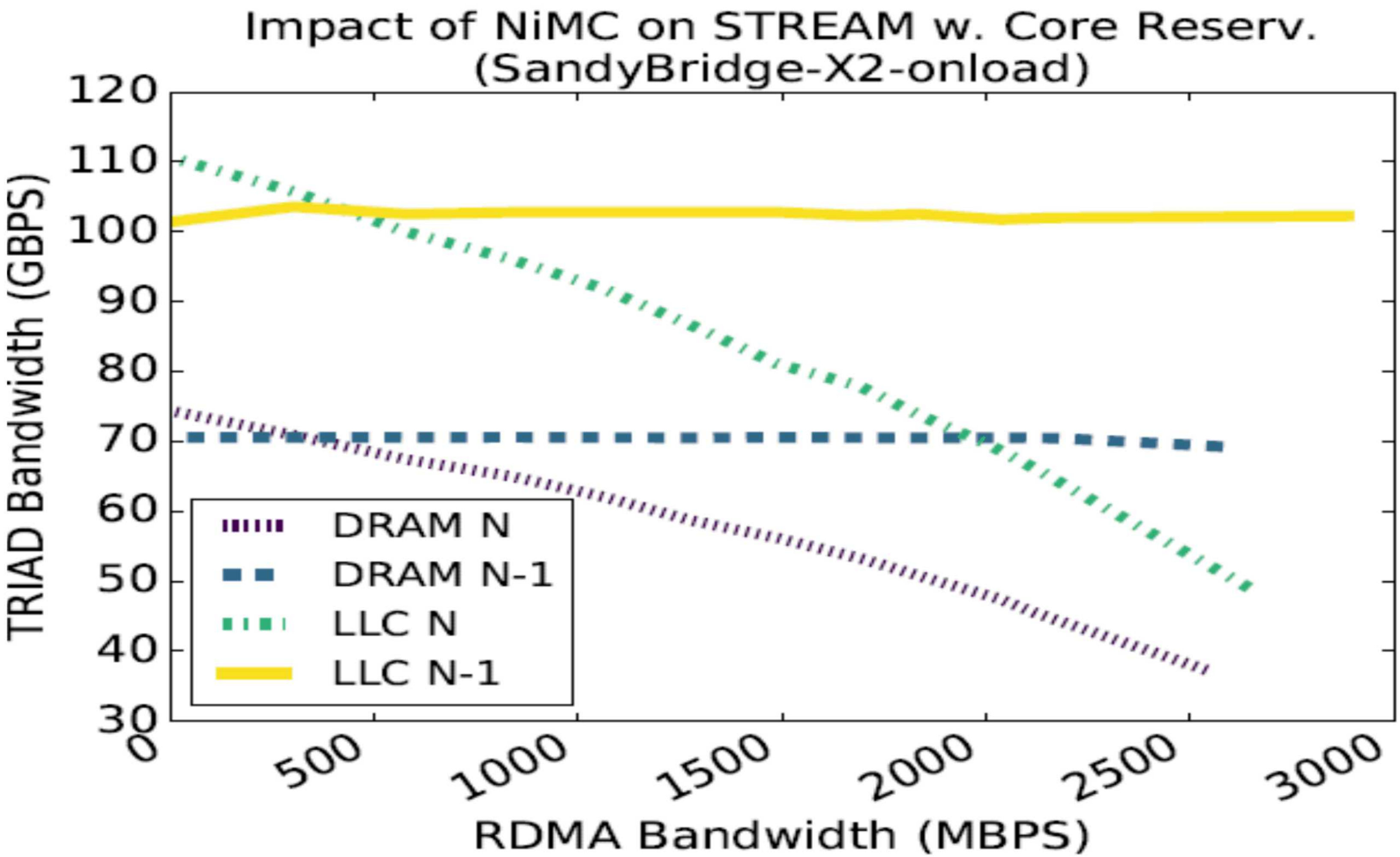
Network-induced Memory Contention



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



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

- INCA: Non-deadline based methods redefine area
  - 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
  - Solid ML use case for efficient data transfer
  - Speed grows with line rate
  - Eviction of programs
    - Reschedule – feed back into pipeline
  - No longer dependent on remote packets
  - Fully distributed in-network only programs
  - Independent network optimization programs
    - Can use even when no application assistance needed

RDMA-next

- Re-design RDMA
  - Hard to use and model mismatch exists
  - Non-coherent memory
  - Client centric resource ownership
- What can we use to redesign?
  - 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

Collaborative Opportunities

- Application Acceleration
- Accelerate Machine Learning
  - Prep data
  - Work on data
- Additional Use Cases (e.g. secure communication)
- Use simple ML Techniques
  - Good area for collaboration
- In-network distributed programs
- Data movement optimization
  - RDMA-Next

Summary

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