Lightning Talks
Platform Lab Students
Stanford University
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
PISCES: A Programmable, Protocol-Independent Software Switch

Muhammad Shahbaz, Sean Choi, Ben Pfaff, Changhoon Kim, Nick Feamster, Nick McKeown, and Jennifer Rexford

PISCES programs are on average about 40 times shorter than equivalent OVS programs and incur a forwarding performance overhead of only about 2%.
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
SLIK: Scalable Low Latency Indexes for a Key Value Store
SLIK: Scalable Low Latency Indexes for a Key Value Store
SLIK: Scalable Low Latency Indexes for a Key Value Store

1. Colocation
SLIK: Scalable Low Latency Indexes for a Key Value Store

1. Colocation
SLIK: Scalable Low Latency Indexes for a Key Value Store

1. Colocation
SLIK: Scalable Low Latency Indexes for a Key Value Store

1. Colocation
2. Independent
SLIK: Scalable Low Latency Indexes for a Key Value Store

1. Colocation
2. Independent
SLIK: Scalable Low Latency Indexes for a Key Value Store
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. **Samuel Grossman**: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
### Existing Work

<table>
<thead>
<tr>
<th>Properties of Graph Problems</th>
<th>Scalability Optimizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Irregular graph data</td>
<td>● Partitioning algorithms</td>
</tr>
<tr>
<td>● Difficult to partition</td>
<td>● Dynamic scheduling, load balancing</td>
</tr>
<tr>
<td>● Unpredictable access pattern</td>
<td>● Sharing and synchronization optimizations</td>
</tr>
</tbody>
</table>
Existing Work

Properties of Graph Problems

- Irregular graph data
- Difficult to partition
- Unpredictable access pattern

Modern Hardware Features

- Vector processing units
- Sequential memory accesses
- Prefetchers
- NUMA
Grazelle

Properties of Graph Problems

- Irregular graph data
- Simple and easy to partition
- Predictable access pattern

Modern Hardware Features

- Vector processing units
- Sequential memory accesses
- Prefetchers
- NUMA
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
Making RAMCloud Writes Even Faster

GOAL: Make writes durable asynchronously with consistency

- Server returns to clients before making writes durable
- Behavior is still consistent
  - Block reads during durable => all reads are consistent
  - Client retries if server crashes => linearizable if client is alive

- Desired Performance
  - Write latency: 15 µs => 6 µs
  - Better throughput: ~2x server throughput

Poster & Talk: tomorrow 1:25pm
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
Nimbus: Scalable Fast Cloud Computing

Omid Mashayekhi
(omidm@stanford.edu)

Hang Qu
Philip Levis

June 2016
Nimbus: Scalable Fast Cloud Computing

2008

I/O bound
data analytics

MapReduce
Hadoop

10s  1s  100ms  10ms  1ms

Task Length
Nimbus: Scalable Fast Cloud Computing
Nimbus: Scalable Fast Cloud Computing
Nimbus: Scalable Fast Cloud Computing

- I/O bound data analytics (MapReduce, Hadoop)
- In memory data analytics (Spark, Naiad)
- Optimized data analytics (Spark 2.0, Common IL, C++)

Task Length:
- 2008: 10s
- 2012: 1s
- 2016: 100ms, 10ms, 1ms

Runtime Overhead (s):
- 20 workers: ≈1 s
- 50 workers: ≈1 s
- 100 workers: ≈1 s

Number of Workers
Nimbus: Scalable Fast Cloud Computing

50X faster tasks → Only 2X speedup at scale
Control Plane is the next bottleneck in cloud frameworks.
Execution Template is an abstraction for dynamic control plane optimizations.
Nimbus: Scalable Fast Cloud Computing

- **Control Plane** is the next bottleneck in cloud frameworks.
- Execution Template is an abstraction for dynamic control plane optimizations.
- Nimbus with execution templates runs 16-40X faster than leading frameworks at even higher scales.

More on the execution templates at the poster session...
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
High Speed Networks Need Proactive Congestion Control

Context

- Network speed: 10 → 100 Gb/s
- 1 MB / 100 Gb/s = 80 μs

<table>
<thead>
<tr>
<th>Network Speed</th>
<th>Typical Flow Completion Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Gb/s</td>
<td>70-80 RTTs</td>
</tr>
<tr>
<td>40 Gb/s</td>
<td>17-20 RTTs</td>
</tr>
<tr>
<td>100 Gb/s</td>
<td>7-8 RTTs</td>
</tr>
</tbody>
</table>

Reactive Congestion Control

1. Adjust Flow Rate
2. Measure Congestion
3. Adjust Flow Rate
4. Measure Congestion

High Speed Networks Need Proactive Congestion Control
High Speed Networks Need Proactive Congestion Control

Problem

- Can we use explicit information (Link Capacities, Traffic Matrix) to find the optimal flow rate allocation more quickly?
High Speed Networks Need Proactive Congestion Control

Problem

- Can we use explicit information (Link Capacities, Traffic Matrix) to find optimal flow rate allocation more quickly?

Approach -- PERC

- Distributed proactive congestion control
- Message passing between flows and links
High Speed Networks Need Proactive Congestion Control

Approach -- PERC

- Distributed proactive congestion control
- Message passing between flows and links

<table>
<thead>
<tr>
<th></th>
<th>RCP (Reactive)</th>
<th>PERC (Proactive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median</td>
<td>14 RTTs</td>
<td>4 RTTs</td>
</tr>
<tr>
<td>Tail (99th %)</td>
<td>71 RTTs</td>
<td>10 RTTs</td>
</tr>
</tbody>
</table>
High Speed Networks Need Proactive Congestion Control

P4 to NetFPGA SUME Prototype

- Implemented instance of PERC algorithm in P4 and compiled it to NetFPGA SUME board using Xilinx Labs tools
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
ReFlex:
Remote Flash $\approx$ Local Flash

Ana Klimovic    Heiner Litz
Christos Kozyrakis
Why remote flash?

➔ Enables *disaggregation*:
  ◆ Scale flash independently & elastically
  ◆ Improve resource utilization
Why remote flash?

➔ Enables *disaggregation*:
  ◆ Scale flash independently & elastically
  ◆ Improve resource utilization

Why are existing approaches not sufficient?

➔ 2 main issues
Issue 1: Performance overhead

Traditional network storage protocols like iSCSI have high overhead for flash.
Issue 2: Request Interference

Sharing flash is challenging due to request interference (e.g. read-vs-write)
How does ReFlex achieve remote ≈ local flash?

Come to our poster!
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
Fixed-Memory Crash Recovery in RAMCloud

- Current crash recovery uses unbounded memory → out-of-memory errors
- Goal: use fixed amount of memory without duplicate reads
- Don’t know how many times replica will be accessed
- Recovery masters read replicas in set order
- Solution: cyclic buffer, rate limit based on slowest master
  - Use disk read speed heuristic to decide eviction
  - Only have double reads when slowest master is more than 2x slower than fastest
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
A Deeper Look at the Latency, Throughput, and Core Utilization Space

Platform Lab Review: Allocating cores in the kernel, scheduling threads at user level.

Thread scheduling on multicore is still chaotic and bug-prone.

Many design questions spring from this basic idea:

1. How might an application take advantage of knowledge of core count?
2. How does an application, or a user space library, decide when it needs more cores or fewer cores?
3. What is a good relationship between concurrency and parallelism?
Lightning Talks

2. Song Han: “EIE: Efficient Interference Engine on Compressed Deep Neural Network”
5. Samuel Grossman: “Grazelle: Hardware-Optimized In-Memory Graph Processing”
7. Omid Mashayekhi & Hang Qu: “Nimbus: Running Fast, Distributed Computations…”
8. Yatish Turakhia: “A Genomics Co-Processor”
9. Stephen Ibanez & Eyal Cidon: “High Speed Networks Need Proactive Congestion Control”
10. Ana Klimovic & Heiner Litz: “Reflex: Remote Flash == Local Flash”
12. Henry Qin: “Core-Aware Scheduling”
13. Collin Lee: “In search of a Large-Scale Control Framework”
In search of a Large-Scale Control Framework

Goal: Simplify the development of large-scale control applications

Proposal: Logically Centralized Scalable Control

- Works for Cluster Managers, Networking, Drones?
- Declarative programming using goals, intents, states?
- Do we need internal strong consistency?

Gathering Data:

- What control systems do you deploy?
- What are your development pain points?
- What are your performance requirements?