A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services

ISCA 2014

Microsoft – Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou[1], Kypros Constantinides[2], John Demme[3], Hadi Esmaeilzadeh[4], Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck[5], Stephen Heil, Amir Hormati[6], Joo-Young Kim, Sitaram Lanka, James Larus[7], Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, Doug Burger

[1] Microsoft and University of Texas at Austin
[2] Amazon Web Services
[3] Columbia University
[4] Georgia Institute of Technology
[5] Microsoft and University of Washington
[6] Google, Inc.
[7] École Polytechnique Fédérale de Lausanne (EPFL)
All authors contributed to this work while employed by Microsoft.

Presented by Robin Bisping
28 November 2018

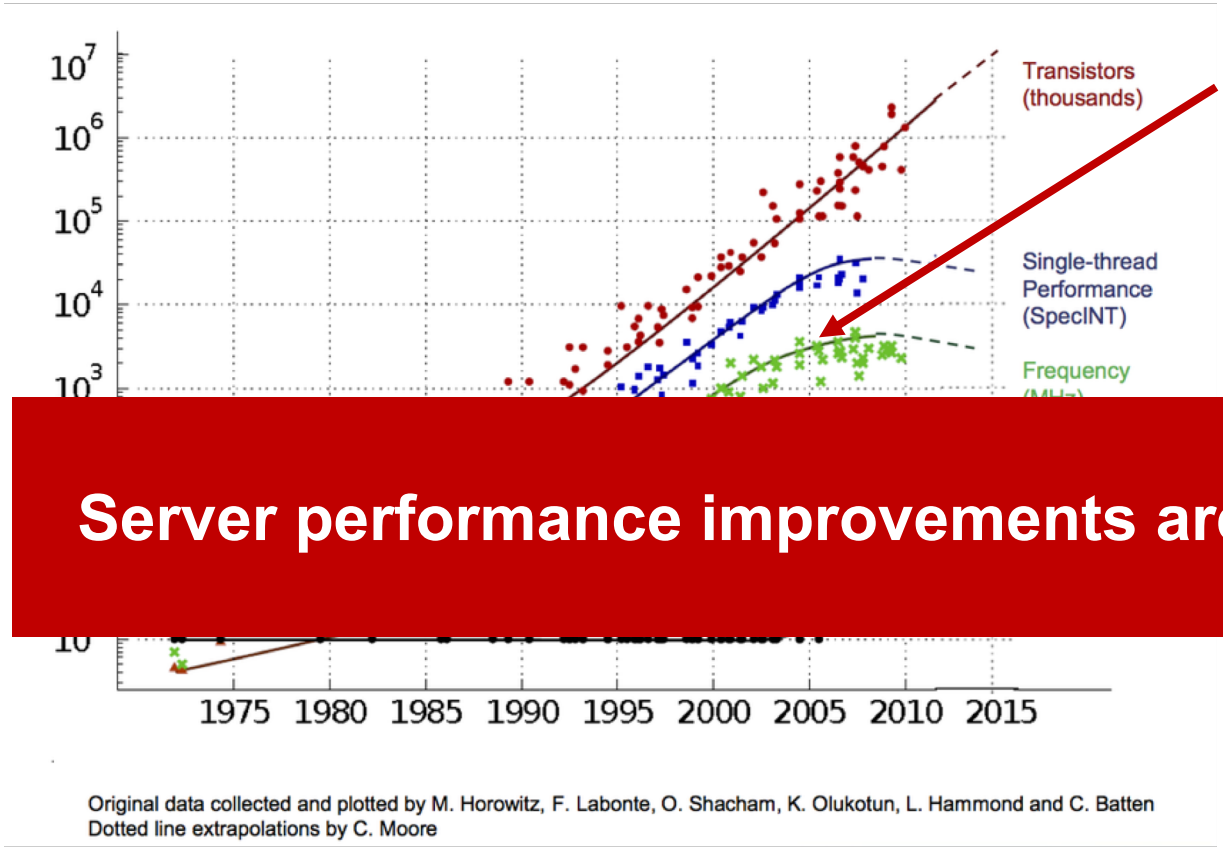
Executive Summary

- **Problem:** The clock frequency of CPUs levels off, so is the transistor count. All in all, server performance improvements are getting smaller.
- **Goal:** Accelerate datacenter services by increasing throughput and lowering latency without depending on CPU performance improvements.
- **Key approach:** Incorporating a reconfigurable fabric of FPGAs in a datacenter.
- **Results:**
  - It increases throughput by 95% at similar latency
  - It reduces tail latency by 29% while maintaining equivalent throughput

Outline

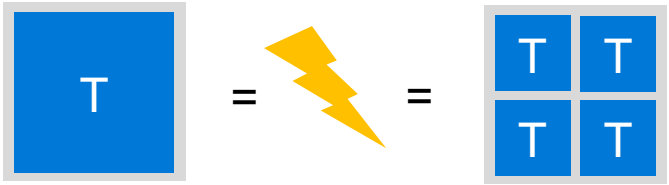
- **Introduction**
- Key Approach
- Mechanisms
- Implementation
- Evaluation
- Summary
- Strengths and Weaknesses
- Takeaways
- Ideas and Discussion

Problem



- Clock frequency slowed down in 2005
- Transistor count will eventually
- Dennard scaling broke down in 2006

Server performance improvements are getting smaller.



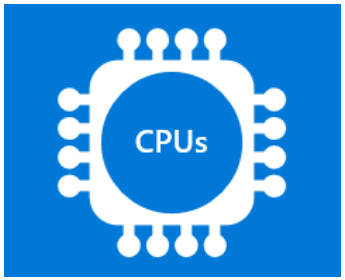
https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/
https://safari.ethz.ch/architecture_seminar/fall2018/lib/exe/fetch.php?media=dark_silicon_4.0.pdf

Goal

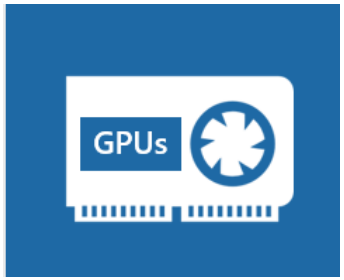
Accelerate large-scale datacenter services to achieve higher throughput and lower latency.

Use additional hardware to accelerate the datacenter services.

Hardware accelerators

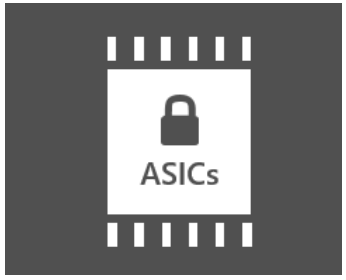


CPU



GPU

FPGA



ASIC

	CPU	GPU	FPGA	ASIC
Power efficiency	$5\ \frac{Gflops}{W}$	$20\ \frac{Gflops}{W}$	$70\ \frac{Gflops}{W}$	$> 70\ \frac{Gflops}{W}$

FPGAs are more efficient than CPUs and GPUs. And they are reprogrammable.

There are use cases where other hardware accelerators may be more beneficial.

Images: https://www.microsoft.com/en-us/research/video/inside-microsoft-fpga-based-configurable-cloud/
Numbers: https://h2rc.cse.sc.edu/2015/burger_keynote.pdf

Datacenter requirements

Efficiency and **generality** are not the only important measures in a datacenter.

- Software services are updated at least once a month
- Machines last maximally three years
- Machines are often repurposed to other services
- Little hardware maintenance is prefered

Therefore, it is desirable that all hardware is as similar as possible (**homogeneity**).

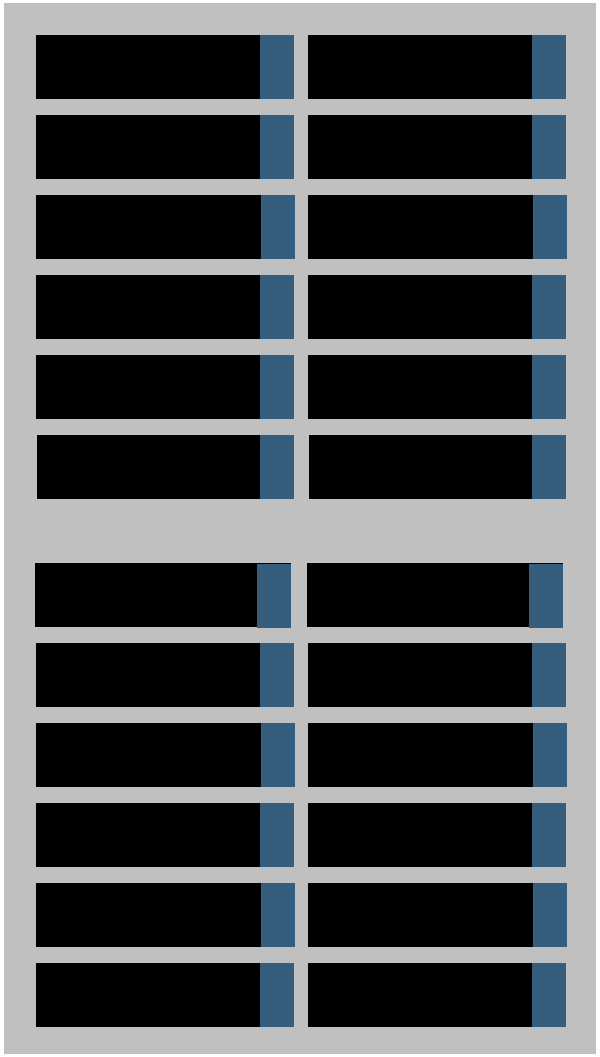
Novelty

- Several papers have already been published before which examine the possibility of accelerating CPUs with FPGAs in large-scale systems
- However, they claim to be the first who have deployed FPGAs at datacenter scale in 2012
- Introduces new mechanisms for using FPGAs in large-scale systems

Outline

- Introduction
- **Key Approach**
- Mechanisms
- Implementation
- Evaluation
- Summary
- Strengths and Weaknesses
- Takeaways
- Ideas and Discussion

Catapult v1



Server rack

In their approach (catapult v1) one **FPGA** is placed **in each server** and connected to one CPU. Groups of FPGAs are connected in an inter-FPGA network with each other.

Benefits:

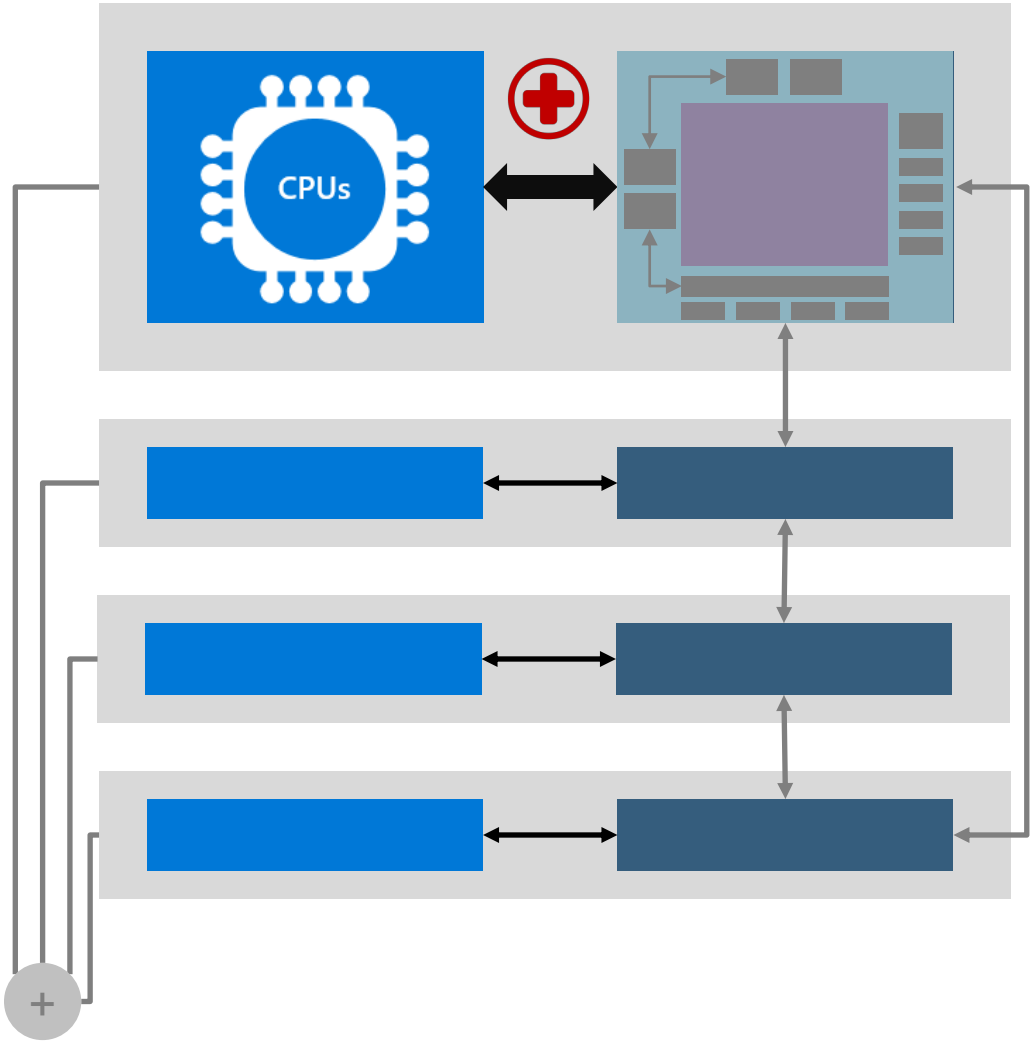
- Flexible
- Homogeneous
- No single point of failure
  - Logic can be relocated

= FPGAs

Outline

- Introduction
- Key Approach
- **Mechanisms**
- Implementation
- Evaluation
- Summary
- Strengths and Weaknesses
- Takeaways
- Ideas and Discussion

Mechanisms



1. **Software interface**
   - How to interface the FPGA with the CPU?
2. **Shell architecture**
   - How to organize the programmable area?
3. **Software infrastructure**
   - How to ensure correct operation?
   - How to recover from failures?
   - How to debug?

Images: https://www.microsoft.com/en-us/research/video/inside-microsoft-fpga-based-configurable-cloud/

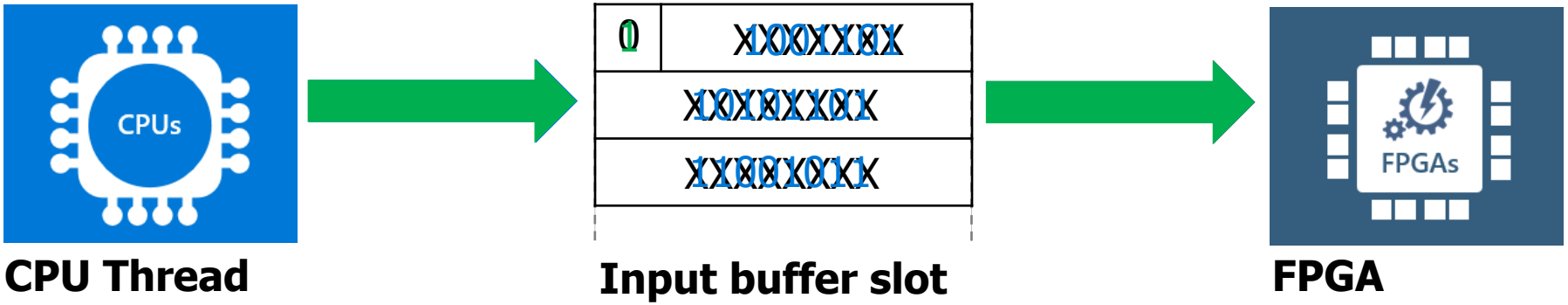
Mechanism 1: Software interface

Communication interface between CPU and FPGA must satisfy:

- **Minimize disruptions to the motherboard**
  - Interface via PCIe
- **Low latency**
  - Avoid system calls
  - Allocate input and output buffer in non-paged, user-level memory. Supply base pointer to to the FPGA
- **Support for CPU threads**
  - Buffers are divided into slots
  - Each thread is statically assigned exclusive access to one or more slots

Mechanism 1: Software interface

Sending data to FPGA:

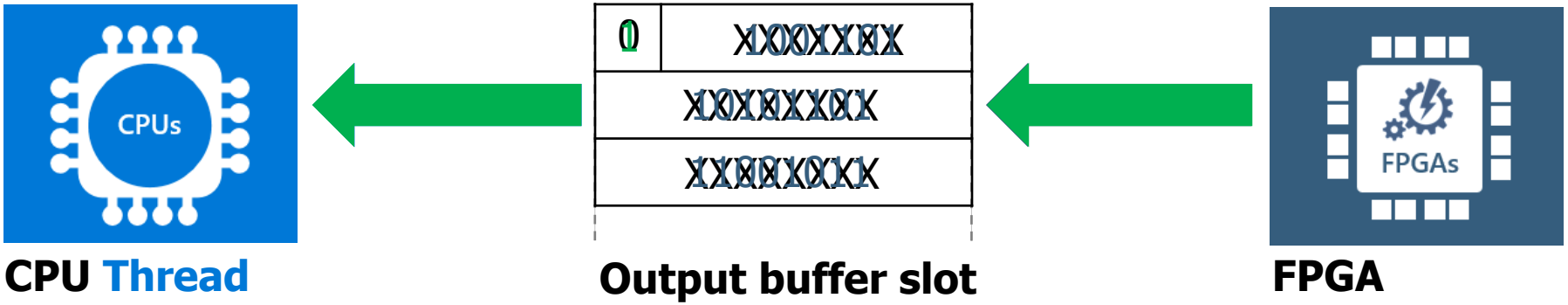


1. Thread checks that its slot is empty
2. Thread writes data into the input buffer
3. Thread sets the full bit for its slot
4. FPGA fairly selects a slot. Fairness is achieved by taking periodic snapshots of the full bits
5. FPGA reads data from the input buffer
6. FPGA clears the full bit

Images: https://www.microsoft.com/en-us/research/video/inside-microsoft-fpga-based-configurable-cloud/

Mechanism 1: Software interface

Readback from FPGA:



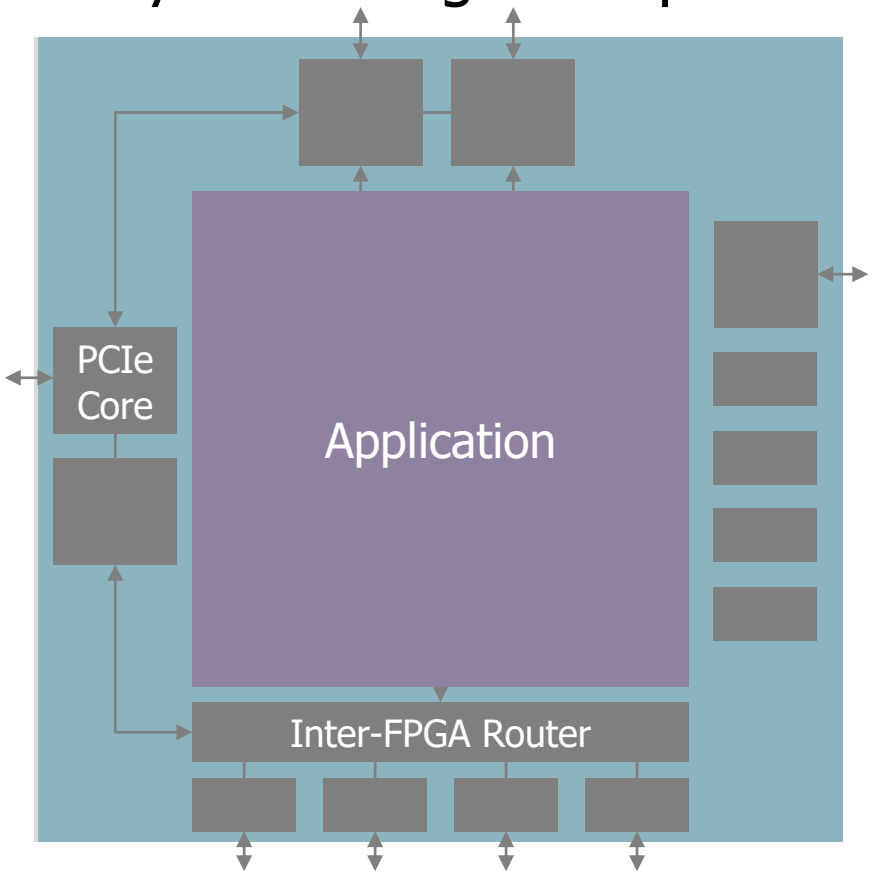
1. FPGA checks that the output slot is empty
2. FPGA writes data into the output buffer
3. FPGA sets the full bit for that slot
4. FPGA generates an interrupt to notify the consumer thread
5. Thread reads data from the output buffer
6. Thread clears the full bit

Images: https://www.microsoft.com/en-us/research/video/inside-microsoft-fpga-based-configurable-cloud/

Mechanism 2: Shell architecture

Call to action:

- System integration may not be portable to other platforms
- System integration places a burden on the user



Solution:

The programmable logic is divided into two parts.

- **Shell**: Reusable portion of programmable logic
- **Role**: Application logic

The user can focus on writing the application.

Mechanism 3: Software infrastructure

The software infrastructure needs to

- Ensure correct operation
- Detect failures and recover
- Support debugging

Two services are introduced for these tasks:

- **Mapping manager**: Configures FPGAs with correct application images
- **Health monitor**: Is invoked when there is a suspected failure in one or more systems

These services run on servers.

Mechanism 3.1: Correct operation

An FPGA being reprogrammed while the system is otherwise up may cause instability.

- **Issue:** It can appear as a failed PCIe device. This raises a non-maskable interrupt
- **Solution:** The driver behind the reprogramming must disable non-maskable interrupts

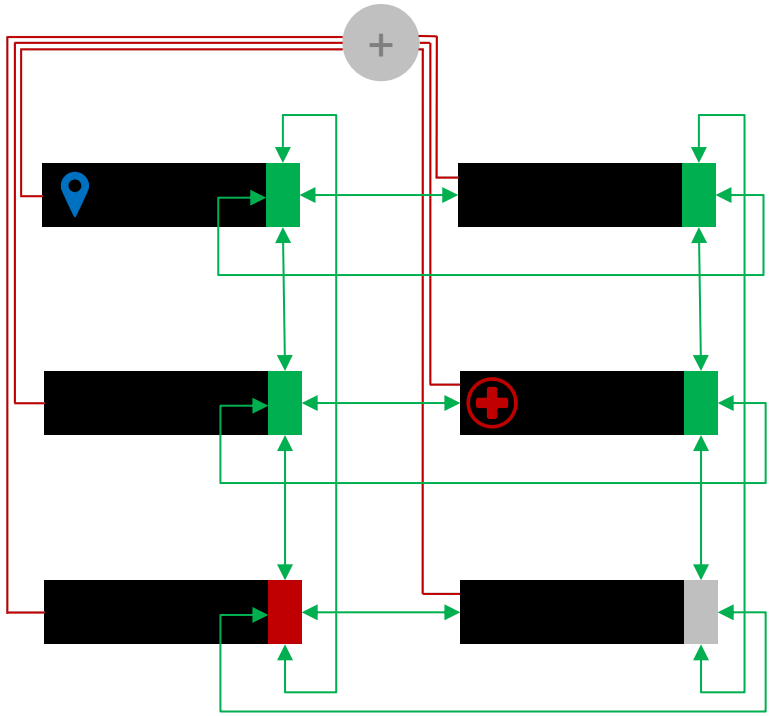
- **Issue:** It may corrupt its neighbors by randomly sending traffic that appears valid
- **Solution:** The FPGA to be reprogrammed sends a TX Halt message, indicating that the neighbors should ignore all further traffic until the link is reestablished

Mechanism 3.1: Correct operation

An FPGA being reprogrammed while the system is otherwise up may cause instability.

- **Issue:** Reprogramming may not occur synchronously across servers. Thus, a reprogrammed FPGA cannot trust its neighbors
- **Solution:** Each FPGA comes up with RX Halt enabled. The Mapping Manager tells each server to release RX Halt once all FPGAs are programmed

Mechanism 3.2: Failures and recovery



Server:

Switch or router:

Ethernet network:

FPGA:

Inter-FPGA network:

1. Health monitor is invoked and queries each machine
2. Servers respond with data about their FPGA and links
3. Health monitor invokes the mapping manager
4. Mapping manager determines where to relocate the logic
5. Mapping manager goes through the reprogramming

Mechanism 3.3: Debugging support

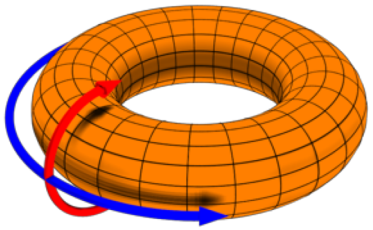
- **Flight data recorder** captures important information during run-time
  - Information is stored in on-chip memory
  - Due to limited capacity it can only capture the most recent events
- **Circular buffer** records the most recent head and tail flits of packets going through the router
  - Flits are link-level packets
  - Head flit holds information about the packets route
  - Tail flit contains the end of the packets payload
- The captured data can be streamed out during the health status check (via PCIe)

Outline

- Introduction
- Key Approach
- Mechanisms
- **Implementation**
- Evaluation
- Summary
- Strengths and Weaknesses
- Takeaways
- Ideas and Discussion

Implementation

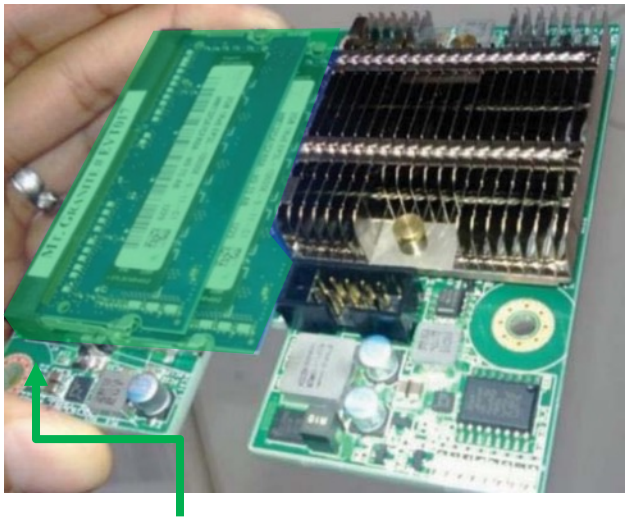
- 1632 servers are deployed in 17 racks
- Fabric is implemented in 34 half-racks, which consists of 48 servers each
- Network topology: 2D, 6x8-torus
  - Tradeoff between routability and cabling complexity
- The protocol "serial lite III" is used for inter-FPGA communication



Images: https://h2rc.cse.sc.edu/2015/burger_keynote.pdf (left); Wikipedia (right)

Implementation

- FPGA board is placed in the back of the server
  - There, the heat cannot disrupt other components



- Memory is needed to accommodate certain services since the FPGA resides in I/O space

Workload

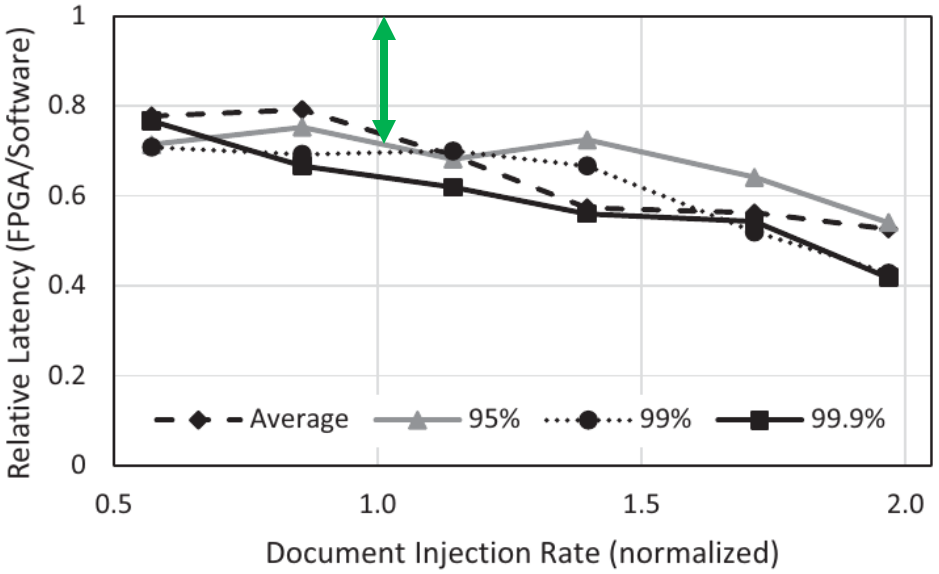
- A fraction of Bing's ranking service was ported onto the catapult fabric.

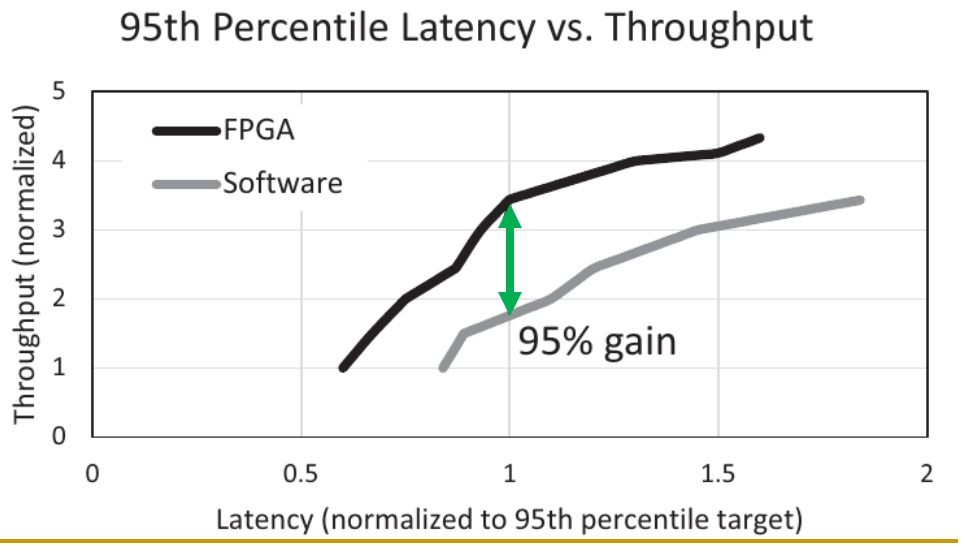
Outline

- Introduction
- Key Approach
- Mechanisms
- Implementation
- **Evaluation**
- Summary
- Strengths and Weaknesses
- Takeaways
- Ideas and Discussion

Evaluation

- Reduces worst-case latency by 29% in the 95th percentile distribution
- The improvement in latency increases at higher injection rates
- The FPGA achieves a 95% gain in throughput relative to software





Evaluation

- In average, 58% of the programmable area was used
- The average clock frequency was 154 MHz
  - This is much lower than in conventional CPUs
  - Leaves room for improvement
- Power consumption was increased by 10%
- Total cost of ownership was increased by 30%

Outline

- Introduction
- Key Approach
- Mechanisms
- Implementation
- Evaluation
- **Summary**
- Strengths and Weaknesses
- Takeaways
- Ideas and Discussion

Summary

- **Problem:** Server performance improvements are getting smaller.
- **Goal:** Accelerate datacenter services by increasing throughput and lowering latency without depending on CPU performance improvements.
- **Key approach:** Incorporating a reconfigurable fabric of FPGAs in a datacenter.
- **Results:**
  - It increases throughput by 95% at similar latency
  - It reduces tail latency by 29% while maintaining equivalent throughput
  - Power consumption increases by 10%
  - Total cost of ownership increases approximately by 30%

Outline

- Introduction
- Key Approach
- Mechanisms
- Implementation
- Evaluation
- Summary
- **Strengths and Weaknesses**
- Takeaways
- Ideas and Discussion

Strengths

- Proposes a viable way to tackle the problem
  - FPGAs have the potential to be further improved. For example, in terms of configurable area and clock frequency
- Proposes an elastic fabric for dynamically allocating FPGAs to address the needs of datacenters
- Simple but effective way of ensuring low-latency communication between FPGA and CPU
- Evaluates the fabric with a real-world application
- Relevant topic
  - Data is "doubling in size every two years"[1]. Faster servers are demanded
- Paper is clear structured and well written

[1] "The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things", IDC, April 2014 (https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm)

Weaknesses

- Hides some implementation details and does not explain some decisions in detail
  - How does the Mapping Manager work?
  - Why did they choose PCIe as interface?
  - What if more traffic arrives than can be processed at a stage?
- Number of FPGAs which can communicate with each other is limited to one pod
- Requires additional cabling (additional complexity)
- Software must be rewritten to work on this architecture
- Proposes only a way to overcome the problem as long as FPGAs can be further improved

Outline

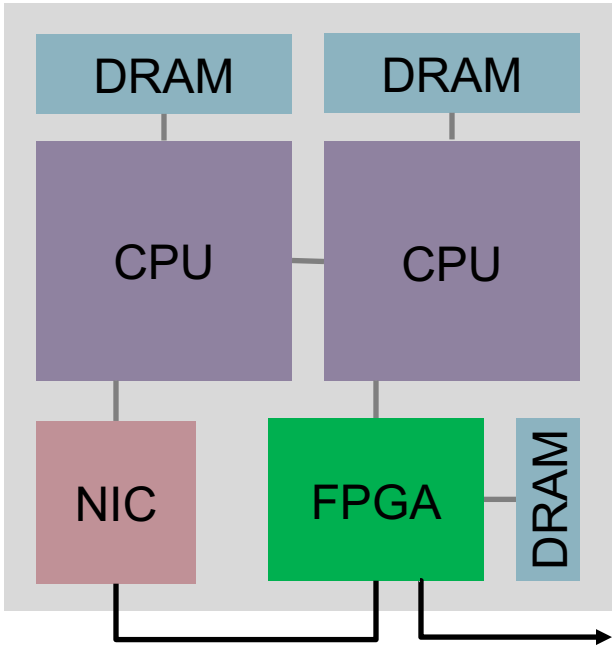
- Introduction
- Key Approach
- Mechanisms
- Implementation
- Evaluation
- Summary
- Strengths and Weaknesses
- **Takeaways**
- Ideas and Discussion

Takeaways

- Server performance improvements are getting smaller
- Reconfigurable fabrics of FPGAs are a viable path forward to accelerate large-scale datacenter services as server performance improvements slow down
- FPGAs are being deployed in datacenters nowadays

Beyond the paper

- Microsoft improved the fabric further (**Catapult v2**)



  - The FPGA can be used as a compute or a network accelerator locally
  - Services can also be mapped to remote FPGA resources
  - «A Cloud-Scale Acceleration Architecture»; Caulfield, Cheng, et al.; IEEE ISM; October 2016

- Almost every new server from Microsoft uses Catapult v2
  - Microsoft expects to be able to improve server performance with FPGAs until 2030

Outline

- Introduction
- Key Approach
- Mechanisms
- Implementation
- Evaluation
- Summary
- Strengths and Weaknesses
- Takeaways
- **Ideas and Discussion**

Ideas and discussion

Do you have any questions?

Ideas and discussion

- **What do you think of specialized servers?**
- Efficiency gains for specific workloads
- Loss of homogeneity in datacenter
- Slowdown of datacenter services evolution
  - Greater changes may be incompatible with the specialized servers

Ideas and discussion

- **How can the fabric be further improved?**
- Tighter coupling between CPU and FPGA
  - For example with Intel's Front Side Bus or QuickPath interconnect
- Using faster memory technology (e.g. SRAM)
- Enhancing debugging by increasing available space for storing events

Ideas and discussion

- **Can you think of other ways of incorporating FPGAs in datacenters?**
- **Or other ways of accelerating datacenter workloads?**

A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services

ISCA 2014

Microsoft – Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou[1], Kypros Constantinides[2], John Demme[3], Hadi Esmaeilzadeh[4], Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck[5], Stephen Heil, Amir Hormati[6], Joo-Young Kim, Sitaram Lanka, James Larus[7], Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, Doug Burger

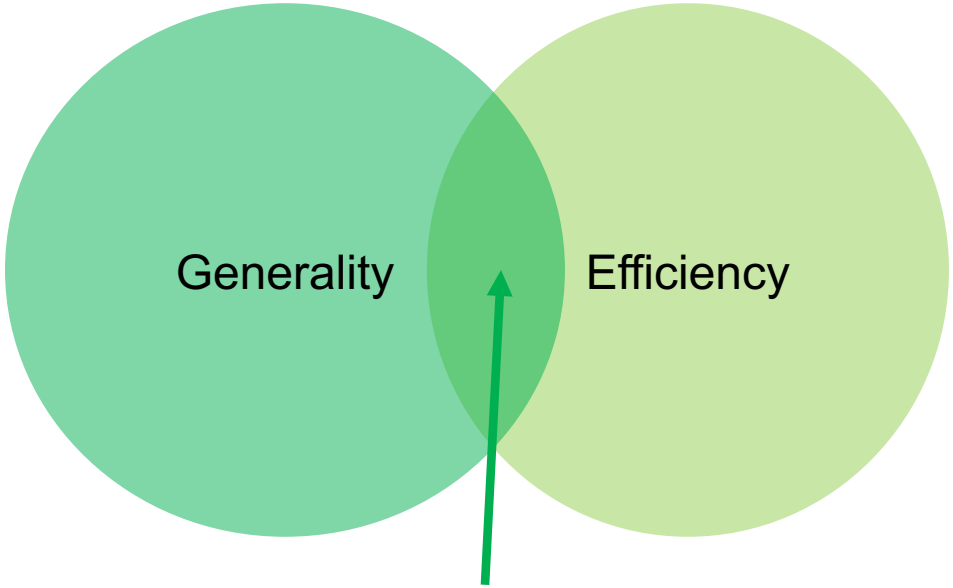
[1] Microsoft and University of Texas at Austin
[2] Amazon Web Services
[3] Columbia University
[4] Georgia Institute of Technology
[5] Microsoft and University of Washington
[6] Google, Inc.
[7] École Polytechnique Fédérale de Lausanne (EPFL)
All authors contributed to this work while employed by Microsoft.

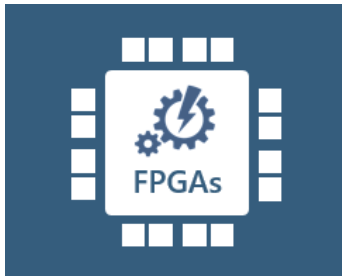
Presented by Robin Bisping
28 November 2018

Additional slides

Hardware accelerators

Paradox: Generality and efficiency are desired.



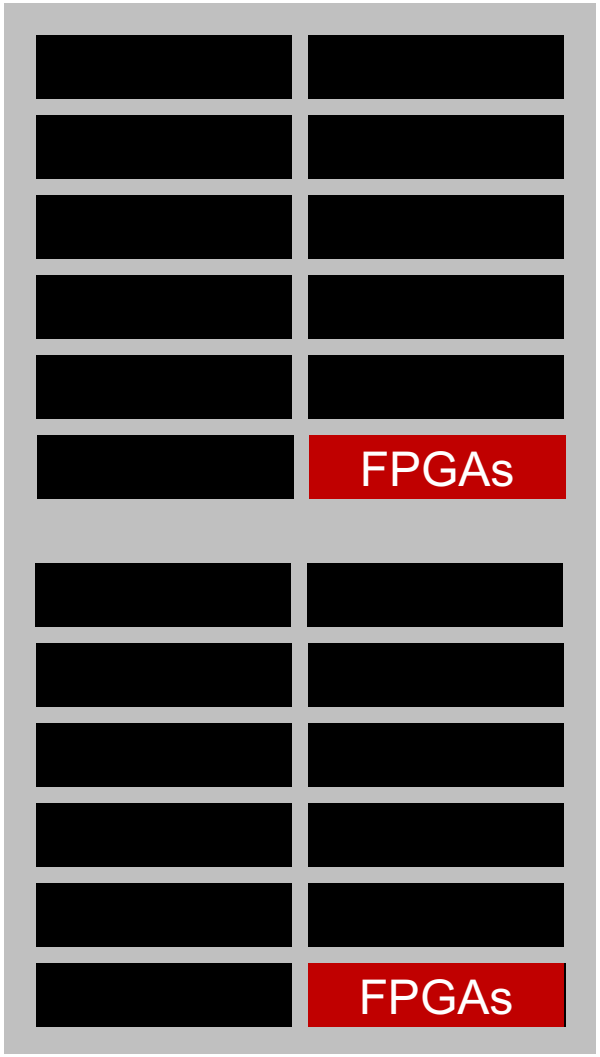


FPGAs are both. They are more efficient than CPUs and GPUs. And they are reprogrammable.

There are use cases where other hardware accelerators may be more beneficial.

Image: Microsoft

Ideas and discussion



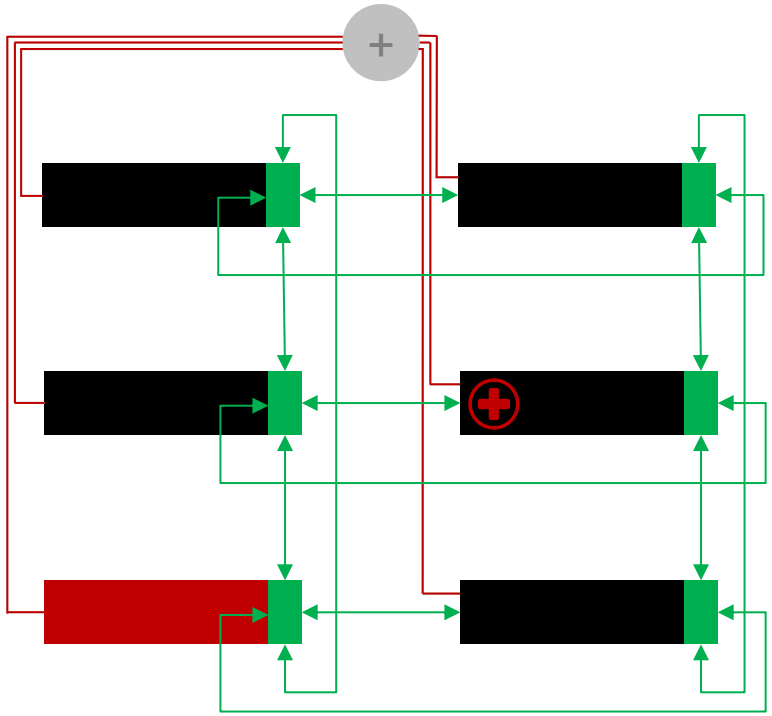
Server rack

In their first approach (catapult v0) six FPGAs are placed onto a card and houses it along with a subset of servers.

What are the drawbacks compared to catapult v1?

- Inelastic
  - Stranded capacity
- Single point of failure
- Heterogeneous

Failure detection and recovery



1. Application hangs
2. Health monitor is invoked
3. Health monitor queries each machine
4. If a server is unresponsive:
    1. Soft reboot
    2. Hard reboot
    3. Flagged for manual service and possible replacement

Server:

Switch or router:

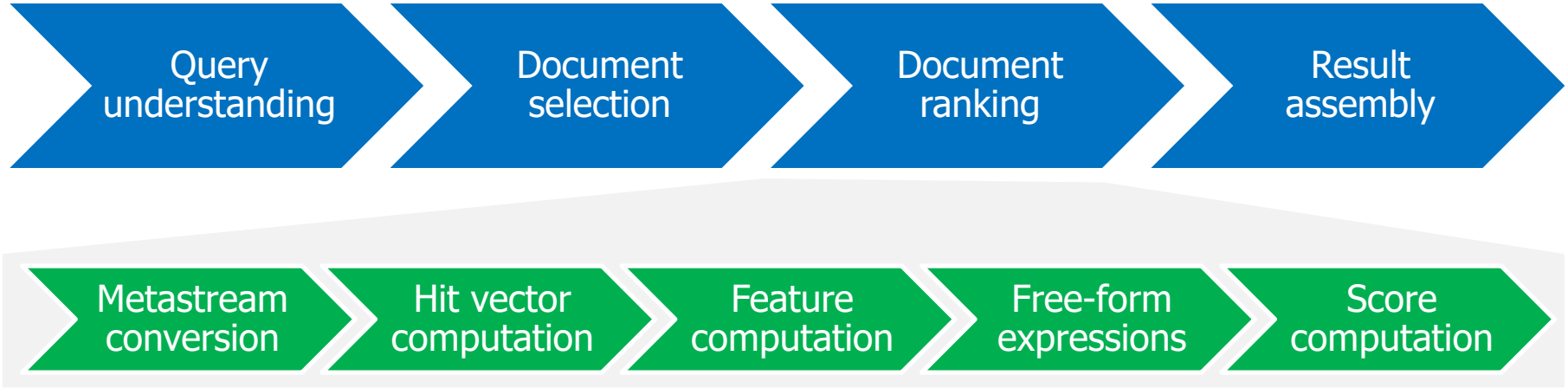
Ethernet network:

FPGA:

Inter-FPGA network:

Workload

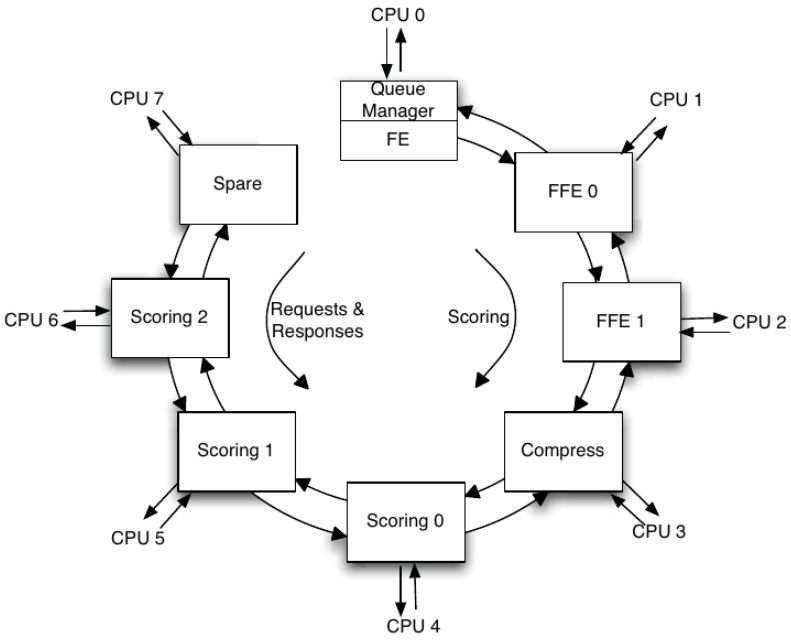
- A fraction of Bing's ranking service was ported onto the catapult fabric.

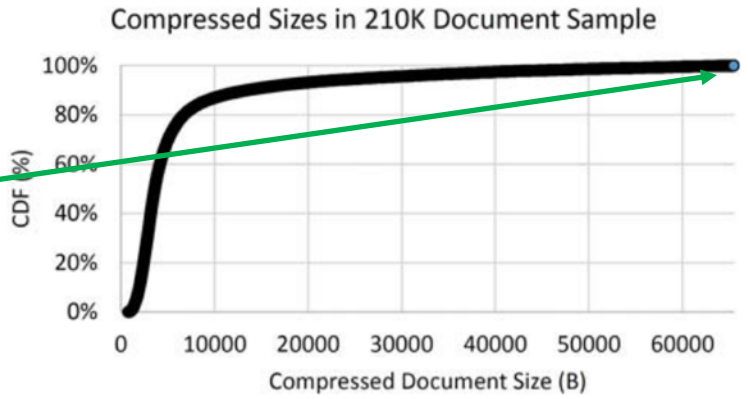


1. The document and its metadata are processed into several metastreams
2. A hit vector is computed, which describes the locations of query words in each metastream
3. Features are computed (e.g. number of occurences)
4. Free-form expressions are computed (combination of features)
5. All features are sent to a machine-learned model that generates a score

Workload

- The service maps to groups of eight FPGAs
- Documents are only transmitted in compressed form to save bandwidth
- Due to the slot based communication interface, the compressed documents are truncated to 64 KB
  - This is not an issue because 99.85% of all documents are not affected

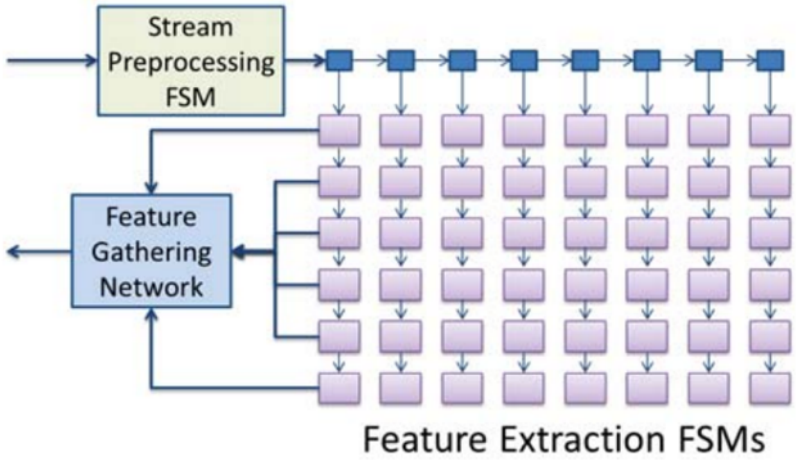




Workload

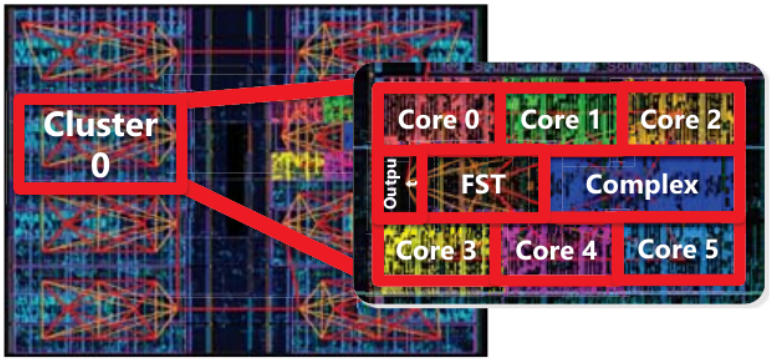
Feature extraction

- Each of the feature extraction engines can run in parallel, working on the same input stream (MISD computation).



Free-form expressions

- Custom multicore processor that is efficient at processing thousands of threads with long-latency floating point operations
- 60 cores on a single FPGA

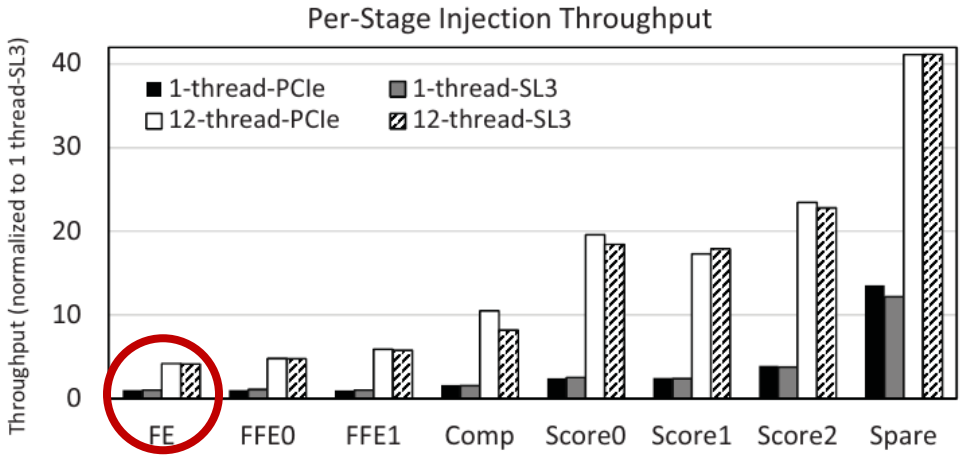


Queue manager and model reload

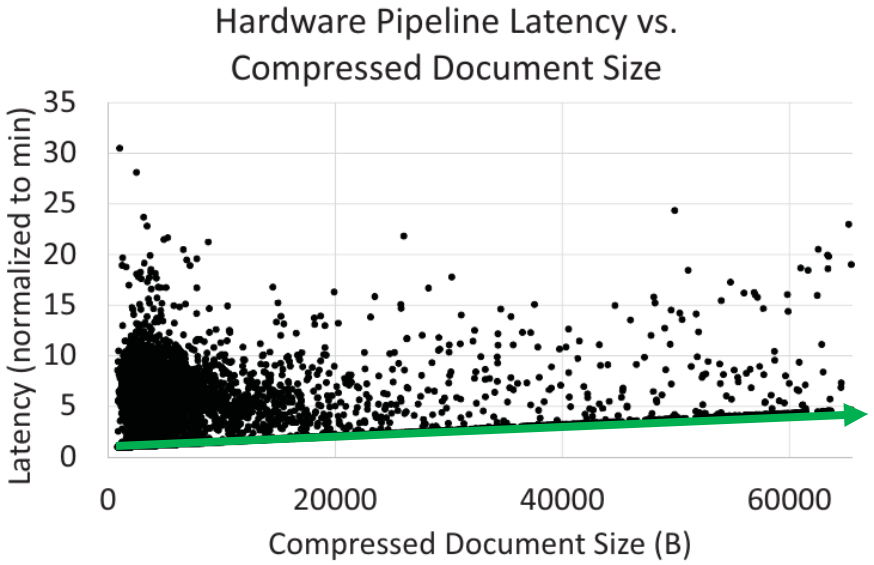
- There are many different sets of features, free forms, and scorers, called **models**. They can vary for language or query type.
  1. Documents are placed in queues of the same model at the head of the pipeline.
  2. The **queue manager** loads documents from each queue until it is empty or a timeout reached.
  3. When a new queue is selected, the queue manager sends a **model reload** command down the pipeline.
     - In the worst case, it requires all embedded RAM to be reloaded from DRAM.
     - Model reload is a slow operation relative to document processing, but fast relative to FPGA repgramming.

Evaluation

- The pipeline is limited by the throughput of the feature extraction stage.

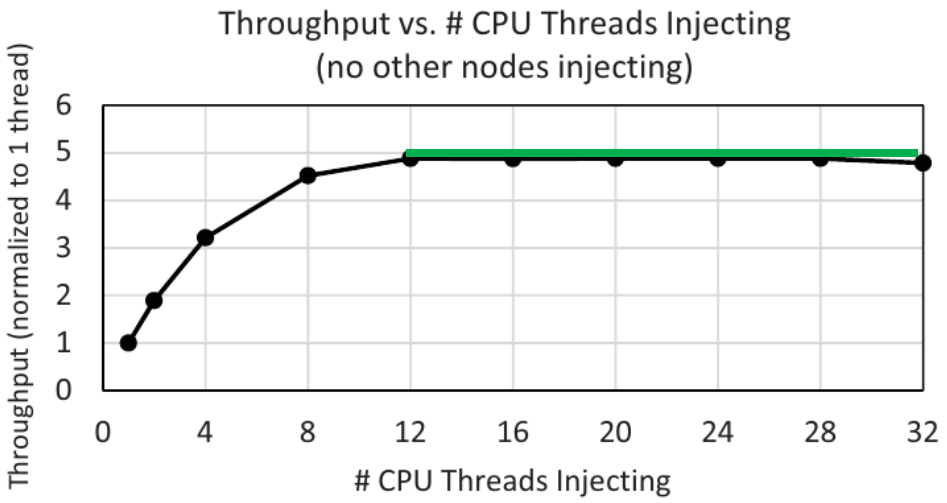


- Minimum latency is proportional to the document size.

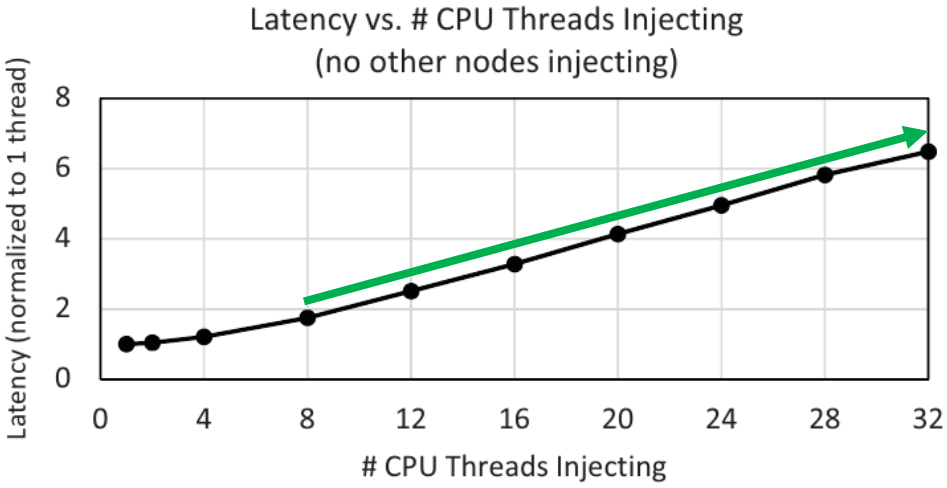


Evaluation

- Full pipeline saturation is achieved at around 12 CPU threads

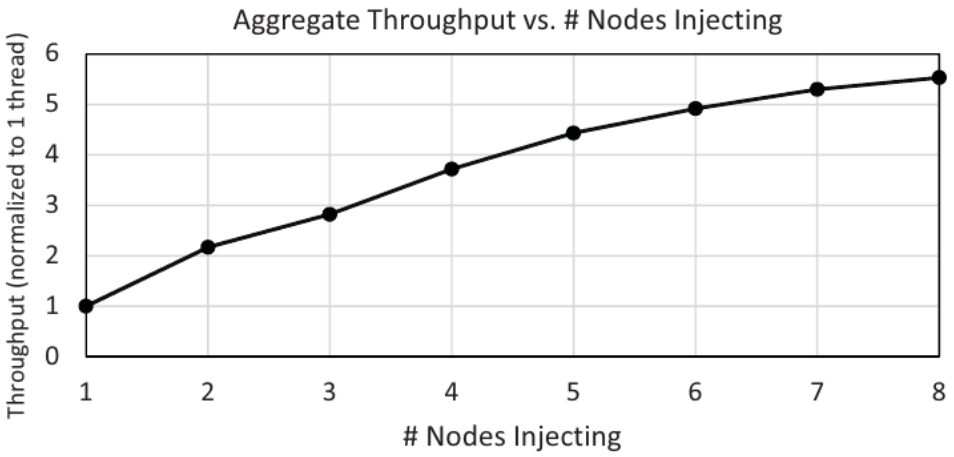


- Latency increases with each additional injecting thread



Evaluation

- When all eight servers are injecting, the peak pipeline saturation is reached



- Because the spare node must share a channel with responses, it perceives a higher latency increase