High Performance Computing

Shota Kuroda 15M37093

16/01/18

Outline

- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

Outline

- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

Paper

- Scalable Multi-Access Flash Store for Big Data Analytics
  - Sang-Woo Jun, Ming Liu, Kermin Elliott Fleming, Arvind
  - *FPGA'14,* February 26–28, 2014, Monterey, CA, USA.

Outline

- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

Abstract

- we examine an architecture for a scalable distributed flash store which aims to overcome the limitation, transportation of large amount of data from hard disk to DRAM in two ways.
  1. the architecture provides a high-performance, high-capacity, scalable random-access storage.
  2. it permits some computation near the data via a FPGA-based programmable flash controller.

- The average latency for user software to access flash store is less than 70µs, including 3.5µs of network overhead.

Outline

- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

Introduction

- Hard disks has been a limitation factor in big data systems.
- With the recent advancement of low latency and high bandwidth flash devices as alternatives to disks, the performance bottleneck has shifted from the storage device to the network latency and software overhead.
- Another facet of high performance storage systems research is providing a computation fabric on the storage itself.
- we propose a novel high-performance storage architecture, which we call BlueDBM.

Introduction(Cont.)

- BlueDBM's goal
  1. Low Latency, High Bandwidth
  2. Scalability
  3. Low-Latency Hardware Acceleration
  4. Application Compatibility
  5. Multi-accessibility

Outline

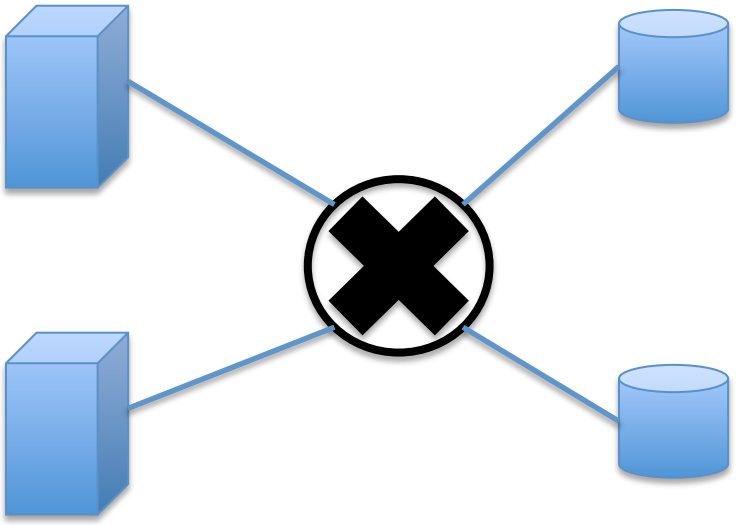
- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

Related Work

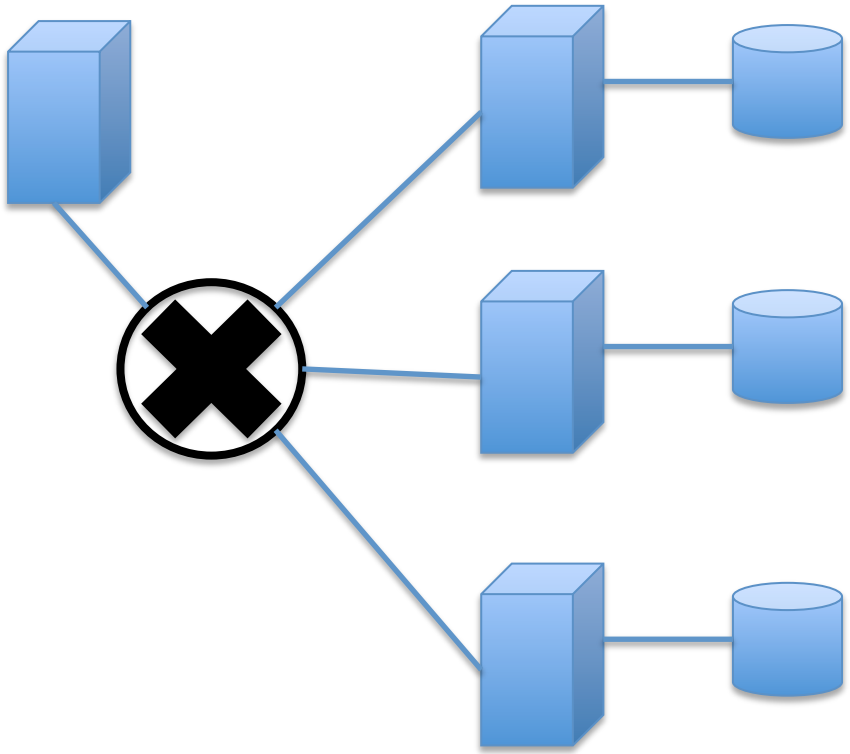
- Storage systems that require high capacity are usually constructed in two ways
  1. SAN
     - has milliseconds of latency comes from protocols
  2. distributed file system
     - the software overhead of concurrency control and the high-latency congestion-prone general purpose network degrades performance.
- These latencies are insignificant compared to the one of hard disks.

SAN & DFS

SAN



DFS



Related Work(Cont.)

- Flash, alternatives of hard disk
  - has order of 10us latency
  - hard disk's latency is order of 10ms
- As a result, the storage device is no longer a bottleneck in high capacity storage systems.
- in a disk-based distributed storage system, non-storage components are responsible for less than 5% of the total latency, while in a flash-based system, this number rises to almost 60% .

Related Work(Cont.)

- Flash's disadvantages
  - limited program erase cycles
  - coarse-grain block level erase
  - low write throughput
- Our storage architecture is similarly motivated by and designed for these flash characteristics.

Related Work(Cont.)

- CORFU attempts to build distributed file systems tailored for flash storage characteristics
  - but still suffers millisecond-level latency.
- QuickSAN have studied directly connecting flash storage to the network in order to bypass some of the soft-ware latency.
  - This brings down the latency of the system to hundreds of microseconds.

Outline

- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

System Architecture

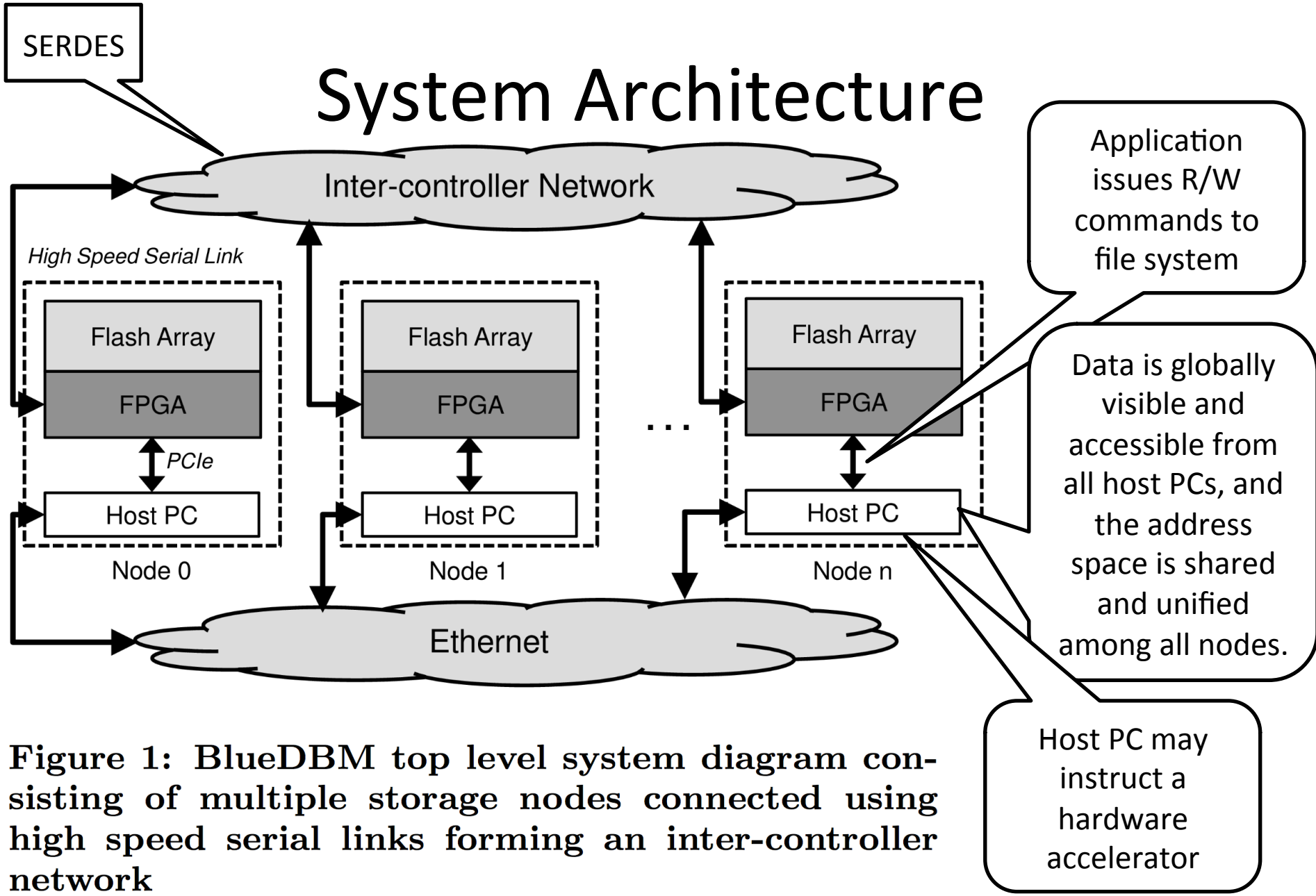


Figure 1: BlueDBM top level system diagram consisting of multiple storage nodes connected using high speed serial links forming an inter-controller network

Introduction(Again)

- BlueDBM's goals are achieved by this architecture.
  1. Low Latency, High Bandwidth
     - by using parallel flash chips, PCIe and high-speed transceivers coupled with a thin networking protocol
  2. Scalability
     - scalability through homogeneous nodes and a network protocol that maintains low latency over multiple hops and is topologically flexible
  3. Low-Latency Hardware Acceleration
     - by providing a hook to software to invoke accelerator operations on data without passing through host.
  4. Application Compatibility
     - by providing a generic file system and exposing the abstraction of a single unified address space to the applications
  5. Multi-accessibility
     - multi-accessibility by providing multiple entry points to storage via many host PCs

System Architecture(Cont.)

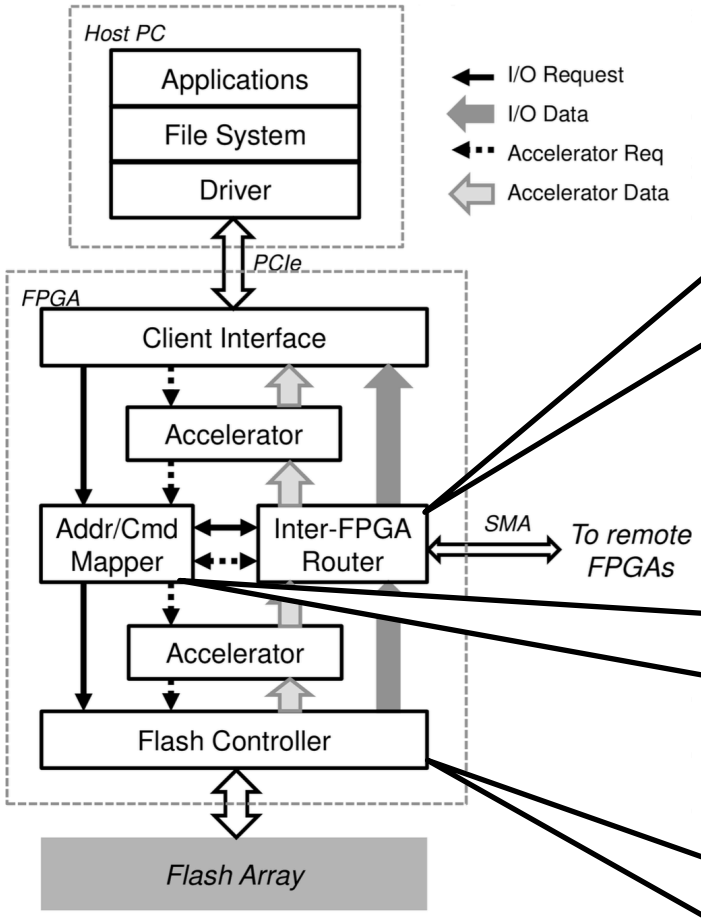


Figure 2: Hardware and software stack of a single node

The router component implements a thin protocol for communication over the high speed inter-FPGA SERDES links.

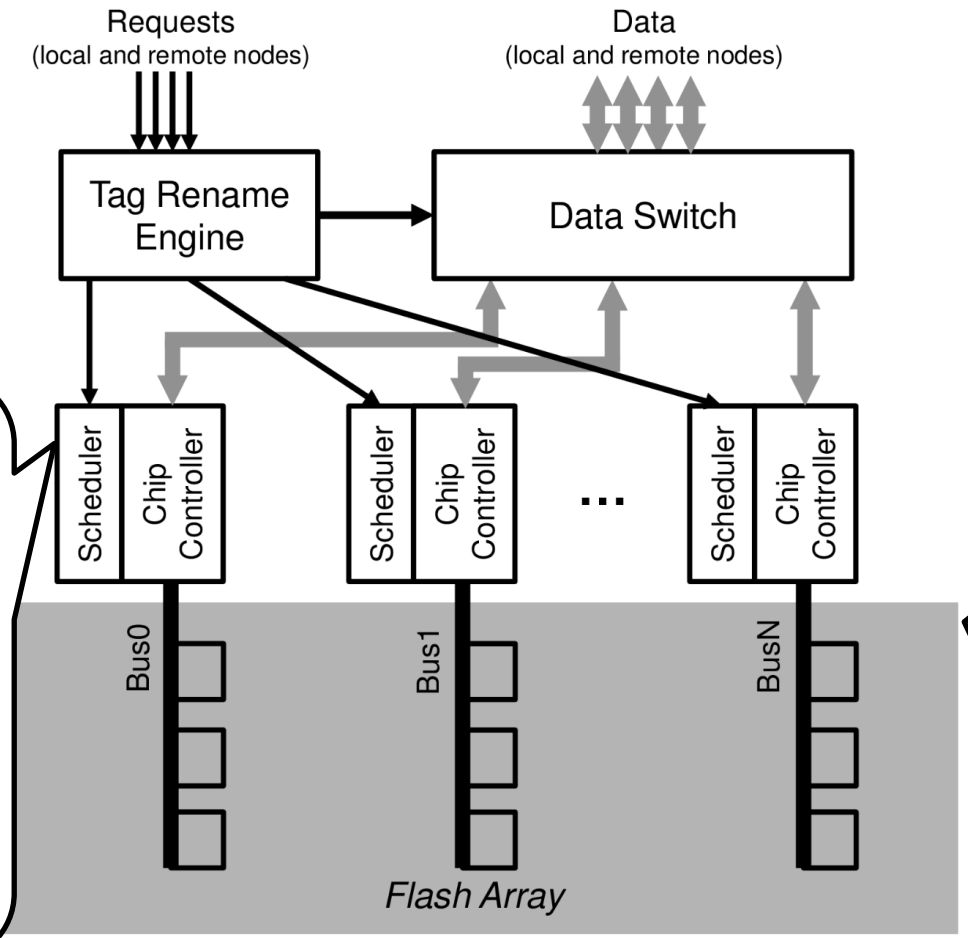
The address mapper maps areas in the logical address space to each node in the network.

The flash controller includes a simple flash translation layer to access the raw NAND chips on the flash board.

System Architecture(Cont.)

- We implemented a generic file system using FUSE
- For each I/O request, the address mapper module determines which storage node the data resides in
- the current mapping scheme focuses on utilizing as much device parallelism as possible, by striping the address space such that adjacent page addresses are mapped to different storage nodes

System Architecture(Cont.)



Not only can buses be accessed in parallel, data operations on different flash chips on the same bus may be overlapped to hide latency

We achieve this by simply permutating a portion of the logical address bits such that spatially local requests have a high probability of being mapped to different bus controllers, buses and chips thereby improving overall system throughput.

Figure 3: Flash controller featuring a scheduler and chip controller per bus, virtualized using a tagging mechanism

System Architecture(Cont.)

- there needs to be an efficient way to match the commands against the data flowing in and out of the flash chips.
- A possible solution is to implement a distributed agreement protocol between each node, but this is complex and requires additional data transfer across the network.
- Instead, we use a two-layer tagging scheme to keep track of this information.

System Architecture(Cont.)

- In the first layer, each command that is issued from a client interface is given a 8-bit tag value.
- A list of unoccupied tags are kept in a free tag queue.
- We dequeue when a new request is issued, and enqueue back when a request retires.
- On the command issuer side, this tag is correlated with information such as the request page address in a tag mapping table structure.
- When the request needs to be processed at a remote node, the tag is sent to the target node with the rest of the request information.
- However, because each node keeps a separate list of free tags, there can be tag collisions at the remote node.
- This is solved using a second layer of tagging scheme, which translates the original tag to the target node's unique local tag.
- The first layer tag is stored in another tag map table with information such as the request source and the original tag value.
- After the request has been handled, the data is sent back to the request origin node tagged with the original tag value it was requested with, so it can be reused for future operations.

Outline

- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

Inter-Controller Network

- conventional method of networking storage devices requires the storage traffic to share the host-side network infrastructure.
- reduced effective bandwidth, because the link between the host and its storage has to be shared for local and remote data.
- BlueDBM solves these issues by having a dedicated storage data network directly connecting the flash controllers to each other.
- the flash controller manages the storage device as well as the network, all data transport of words within a page could be pipelined, effectively hiding the network latency of accessing a page.

Inter-Controller Network(Cont.)

- The routing mechanism for our storage network is a packet-switched protocol
- Each node maintains a routing table of all nodes in the network, where each row contains information including which physical link a packet should take to get to that node and how many network hops away it is
- An accelerator can declare, at compile time, multiple virtual links according to its requirements
- the choice of physical communication fabric in our system is flexible

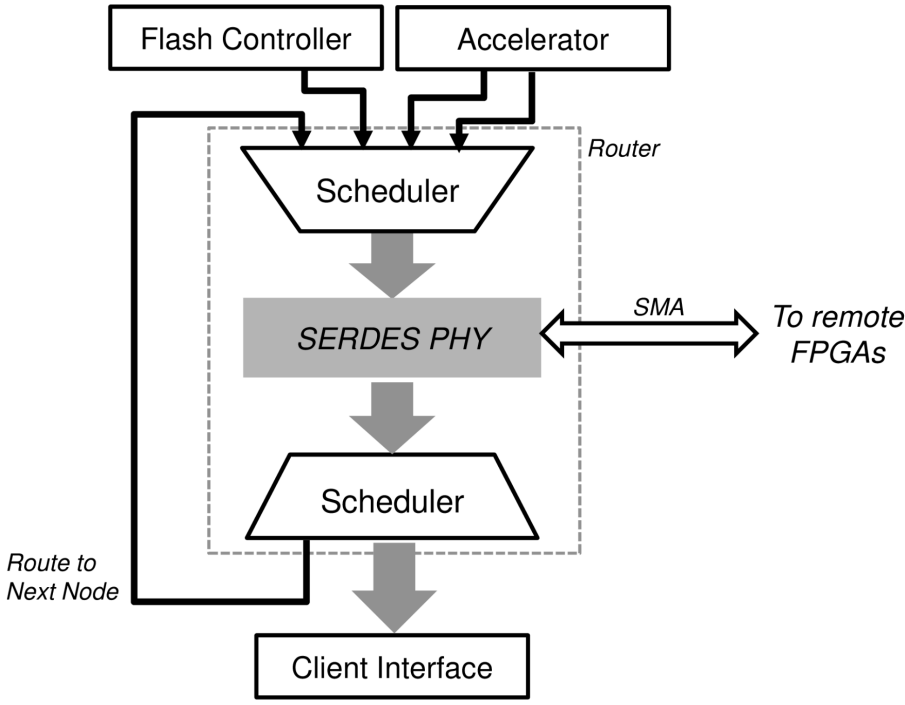


Figure 4: Inter-node network router.

Inter-Controller Network(Cont.)

- 2 model of usage of accelerator

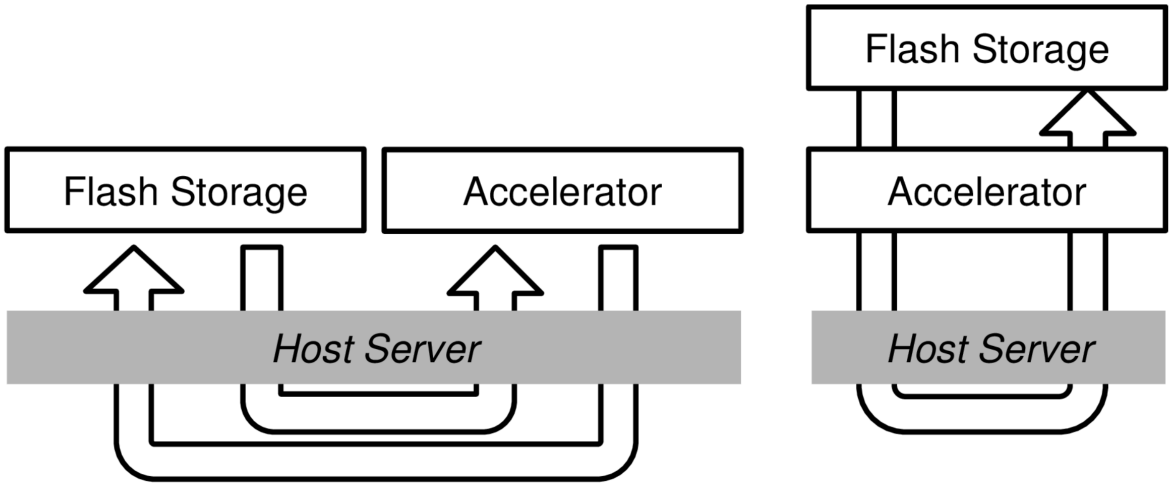


Figure 5: Flow of data when using an accelerator as a separate appliance (left) versus an accelerator in the data path of the storage device (right)

Inter-Controller Network(Cont.)

- accelerators can be implemented both be- fore and after the inter-FPGA router
- The global accelerator located between the router and the client interface implements higher-level functionalities that require a global view of data
  - Examples includes table join operations in a database accelerator, or the word counting example that will be described shortly
- The local accelerator, which is located between the flash storage and the router, is used to implement functions that only require parts of the data
  - For example, compressing pages before writing them to flash

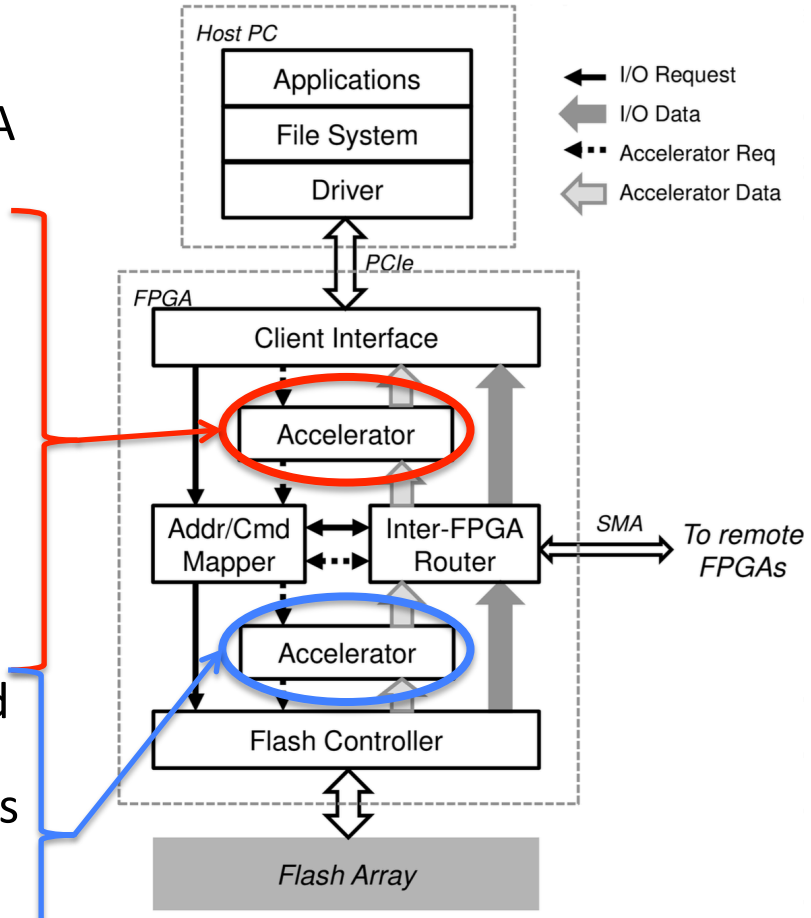


Figure 2: Hardware and software stack of a single node

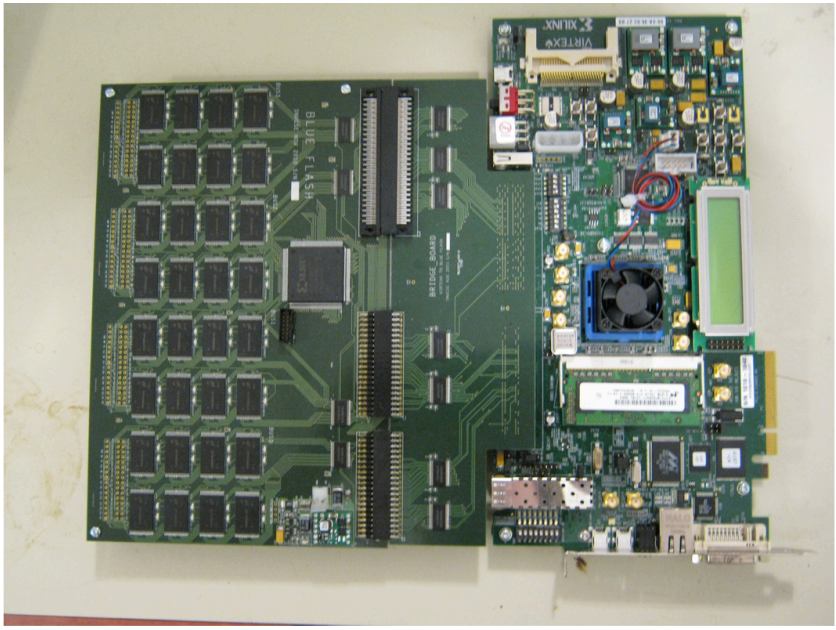
Outline

- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

Prototype System

- Figure 6(a), is based around the Xilinx ML605 board and our custom built flash board
- The ML605 board and the flash board is coupled using the FPGA Mezzanine Card (FMC) connector, as seen in Figure 6(b), and plugged into a PCIe slot on the host server.
- implemented a routing protocol in favor of dynamic reconfiguration of the network.

(a) Four-node prototype system



(b) ML605 and attached flash card

Figure 6: Prototype system

Prototype System(Cont.)

- Each flash board hosts 16GB of flash storage arranged in four parallel buses comprised of 8 512MB Micron SLC flash chips.
- We network the processing nodes of our system by way of the Virtex-6 GTX high speed transceivers.
- Each transceiver is capable of transporting up to 5Gbps
- each node can only connect to one other node via the only remaining SMA port on the ml605 board. Therefore, the prototype uses a tree topology shown in Figure 7

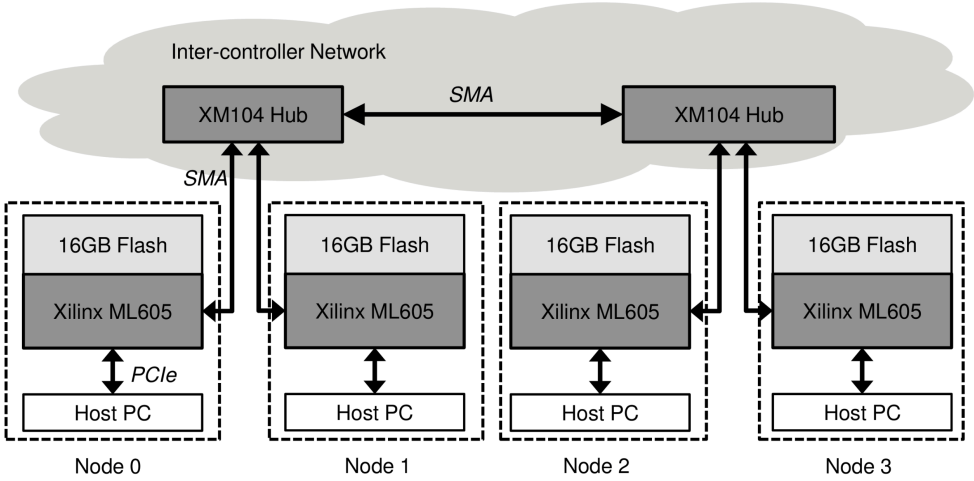


Figure 7: Prototype physical implementation with 4 storage nodes and 2 hubs

Outline

- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

Result

- The approximate area breakdown of each node in our flash system is shown in Table 1
- This area corresponds to approximately 35% of the resources of the medium sized Virtex-6 chip.
- The rest of the area is free to be used for accelerator implementation
- even with a thousand- node system, we can easily fit the routing table within a few BRAMs on the FPGA given that each entry is merely 128 bits
- Packet headers will require 10 bits to identify the source/destination node in a thousand-node system, which means a corresponding increase in FIFO sizes
- However, this area increase remains insignificant compared to the rest of the design on the FPGA
- Thus we are able to scale to thousands of nodes without significant impact on area

	LUTS	**Registers**	**BRAM**
Client Interface	17387	17312	51
Flash Controller	10972	8542	151
Networking	24725	27530	16
Total	53084 (35%)	53384 (17%)	218 (52%)

Table 1: Synthesis metrics for controller components at 100MHz.

Result(Cont.)

- Considering that the typical latency of a flash read is several tens of microseconds, requests in our network can, in theory, traverse dozens of nodes before the network latency becomes a significant portion of the storage read latency, potentially enabling the addressing of multiple terabytes worth of data across many nodes.

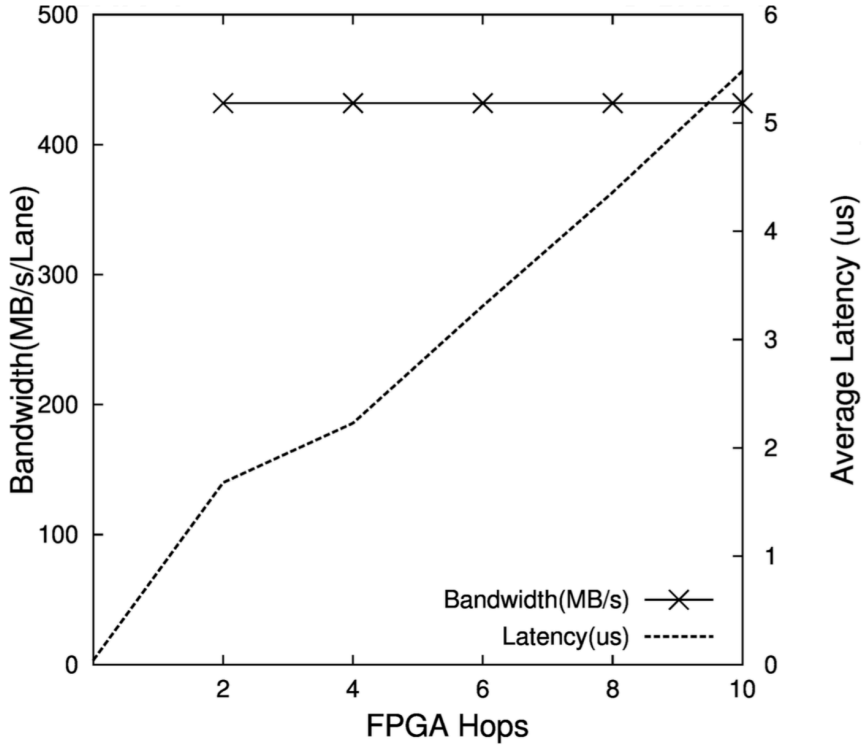
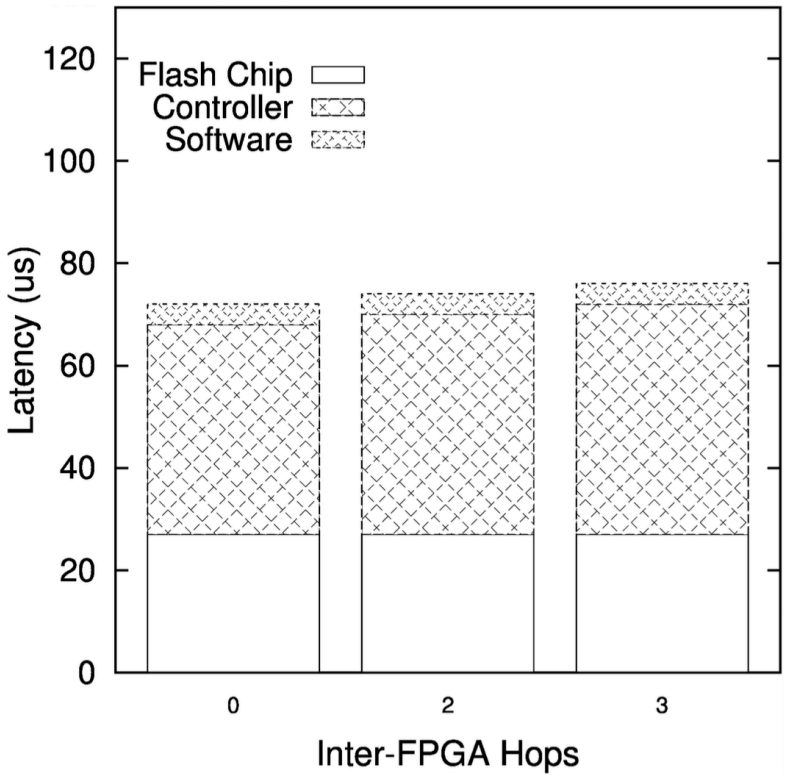
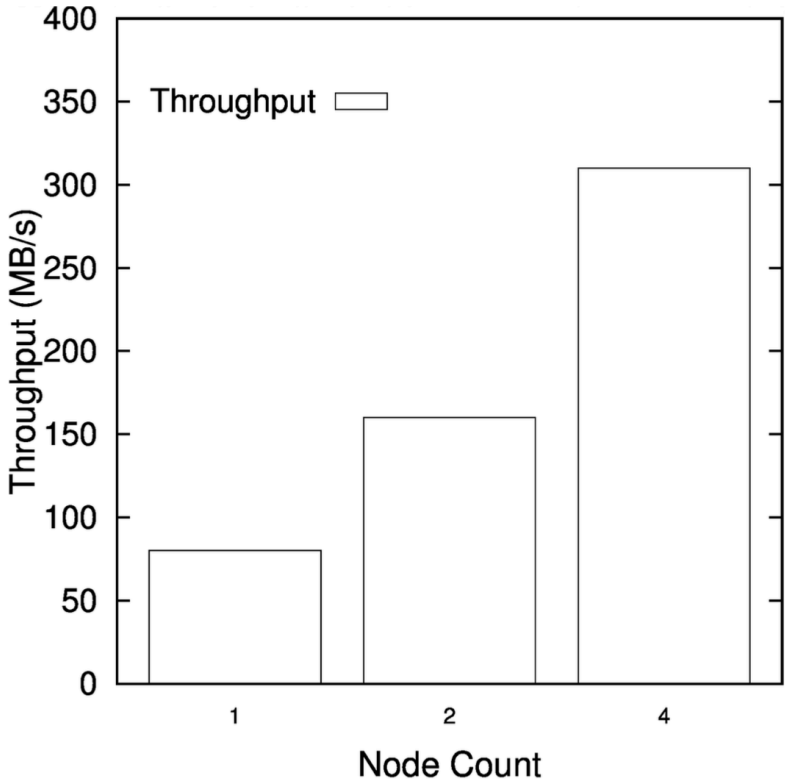


Figure 8: Throughput and latency of our inter-FPGA network using a 5Gbps SERDES connection on the Virtex-6 ML605.

Result(Cont.)



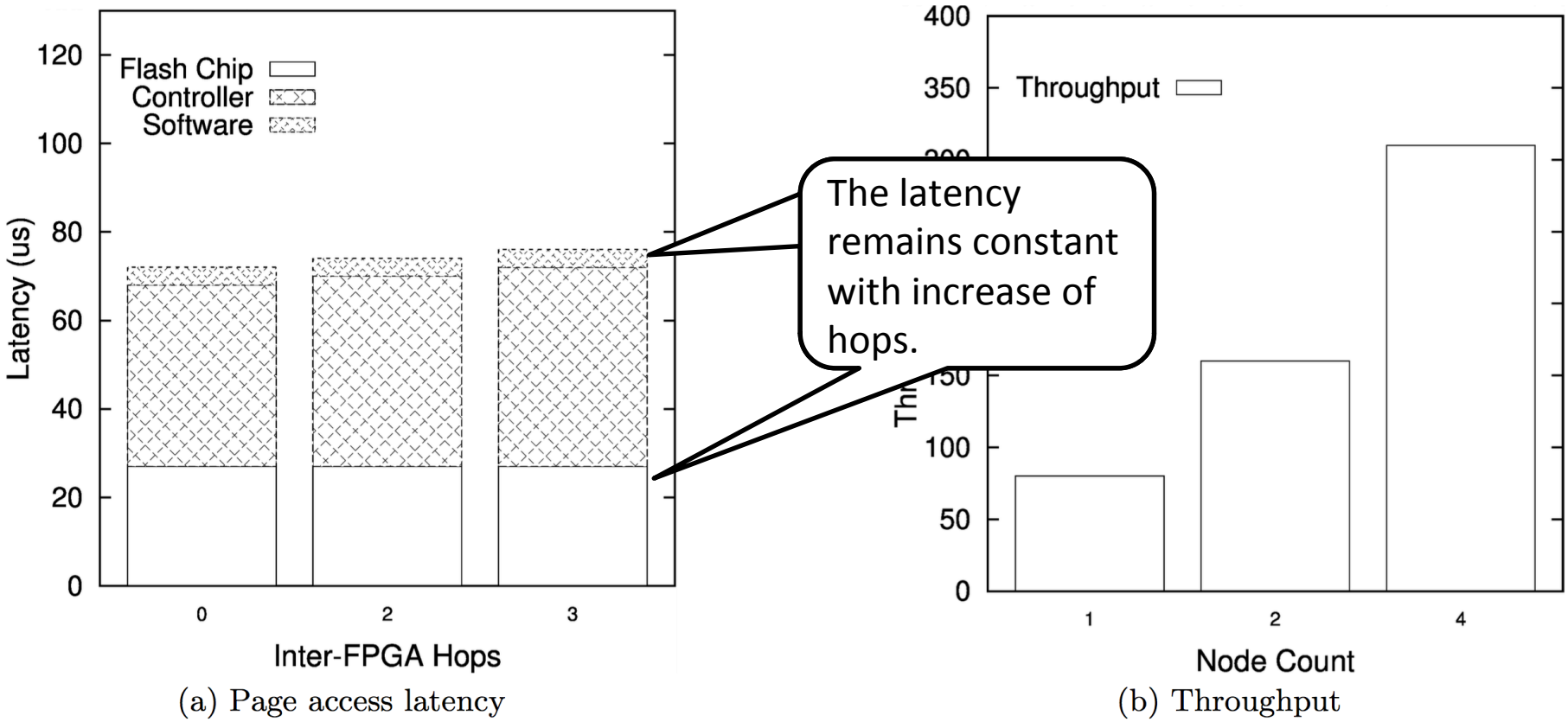
(a) Page access latency



(b) Throughput

Figure 9: Raw latency and throughput measurements of our 4-node prototype

Result(Cont.)



(a) Page access latency

(b) Throughput

Figure 9: Raw latency and throughput measurements of our 4-node prototype

Result(Cont.)

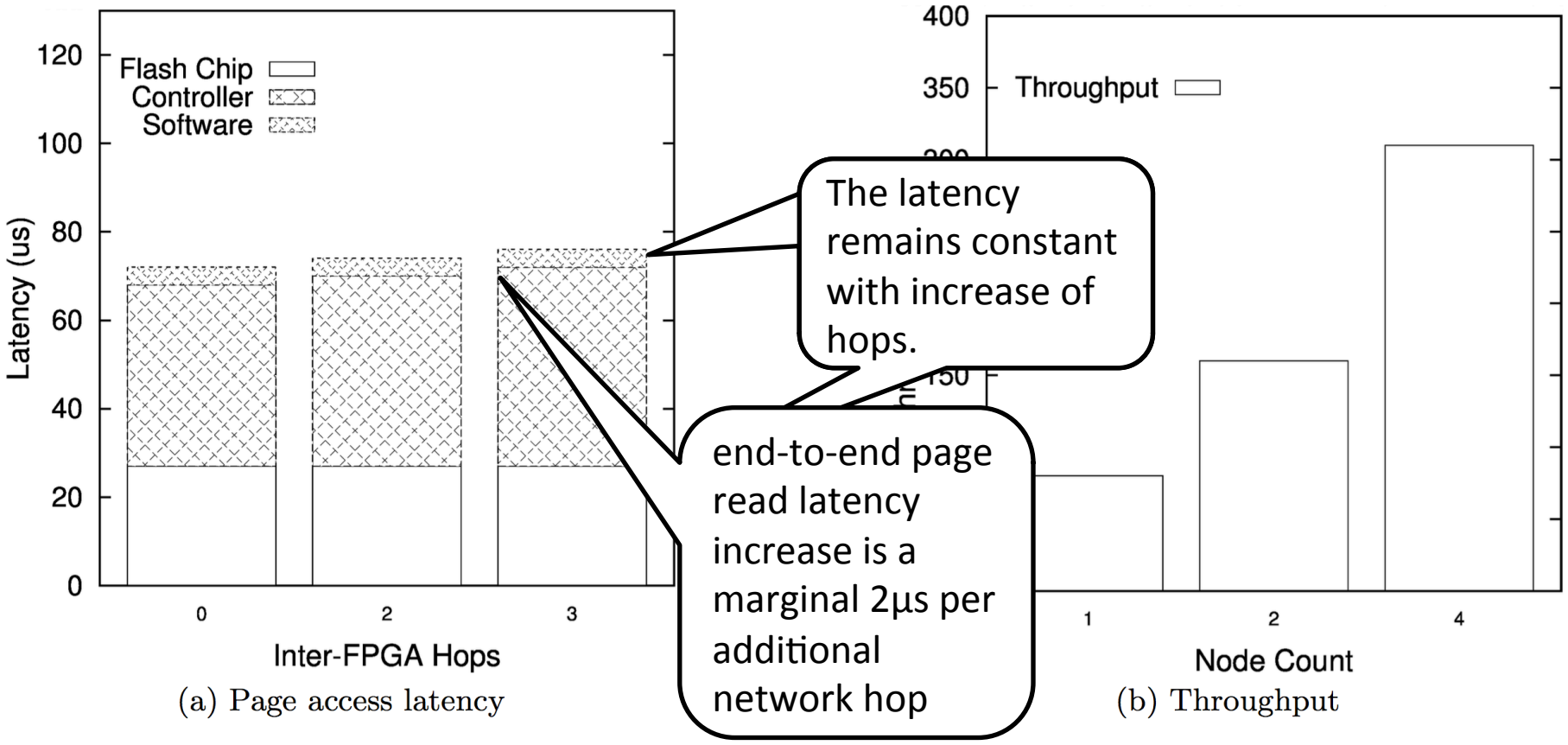
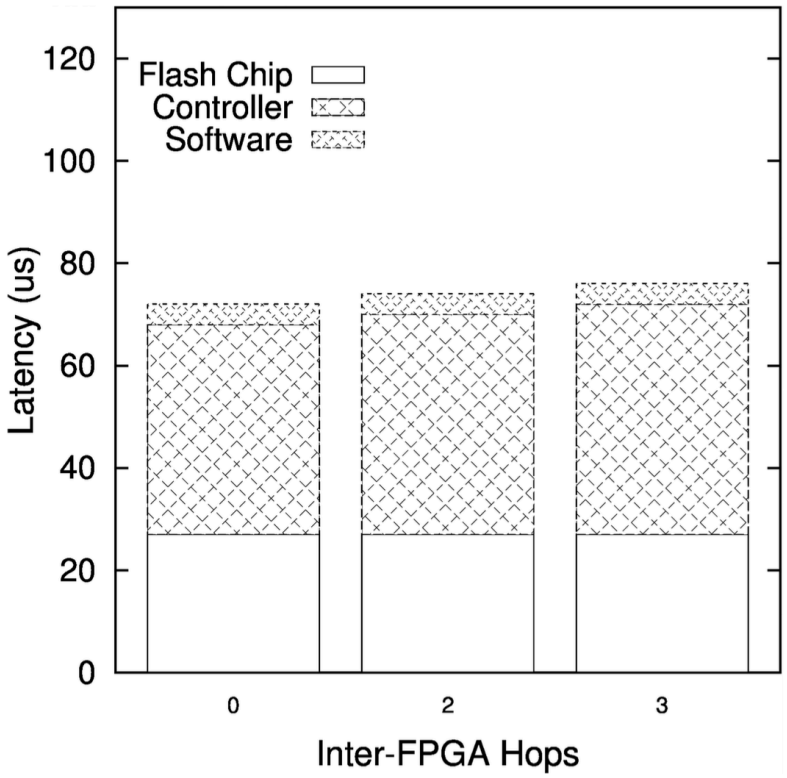
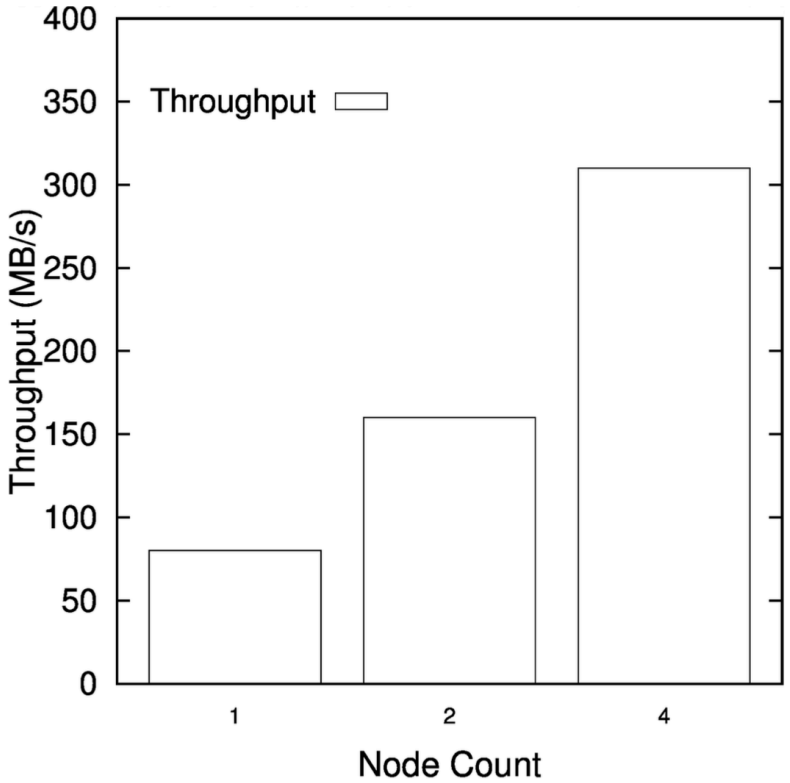


Figure 9: Raw latency and throughput measurements of our 4-node prototype

Result(Cont.)



(a) Page access latency



(b) Throughput

Figure 9: Raw latency and throughput measurements of our 4-node prototype

Result(Cont.)

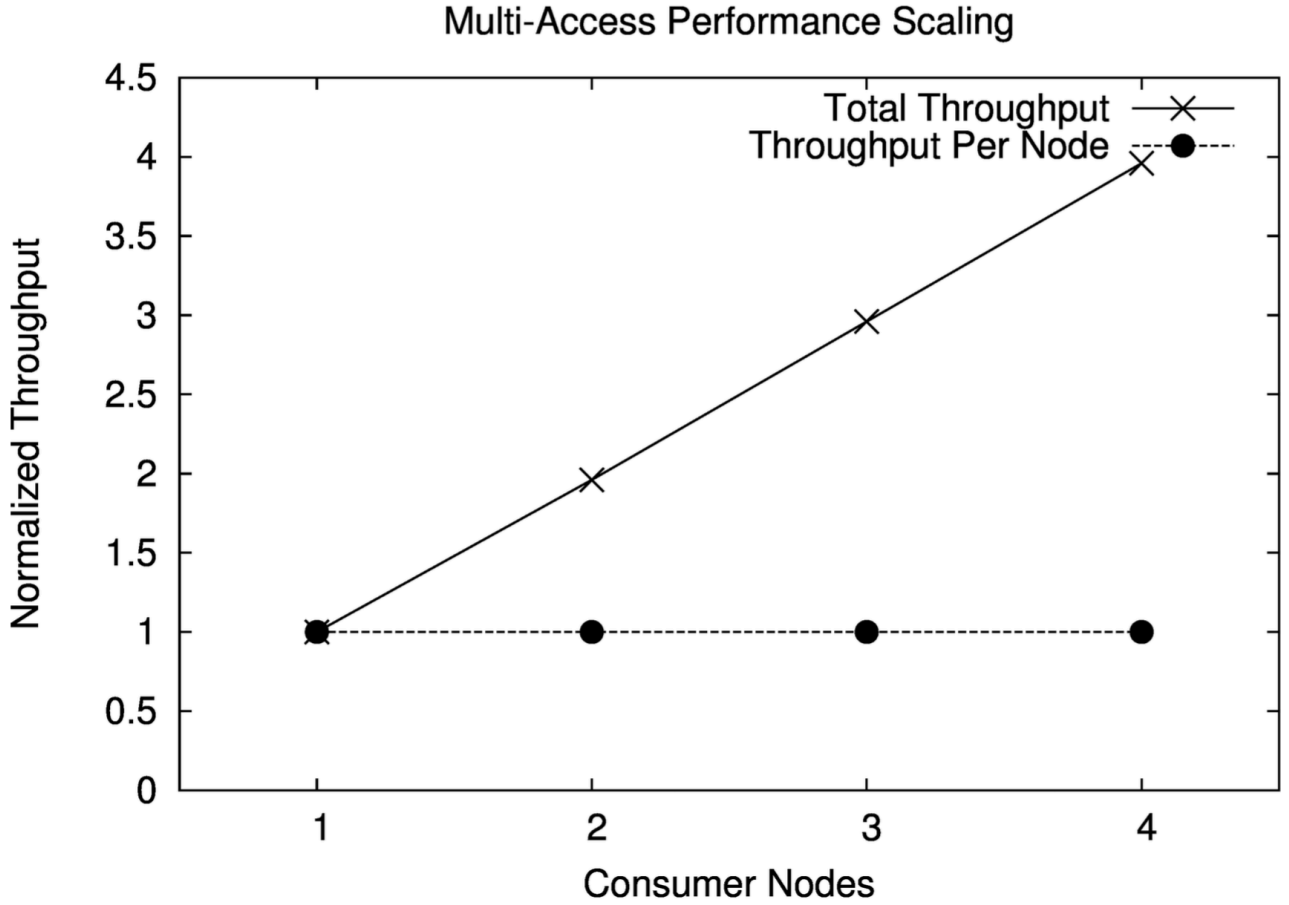
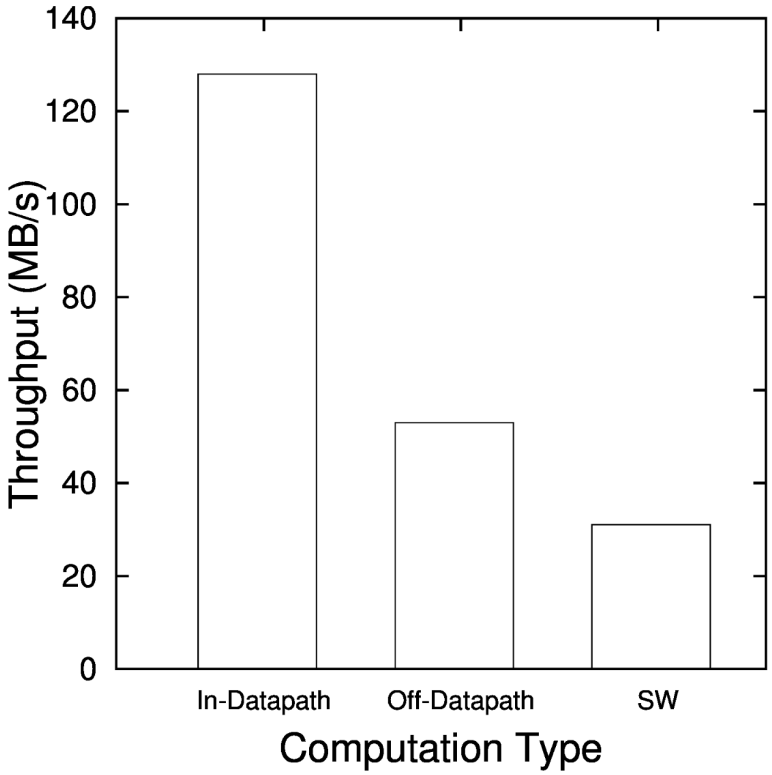


Figure 10: Performance scaling in multi-access scenario

Result(Cont.)



run on 2 nodes
maximum bandwidth is 140MB/s

Figure 11: Word counting accelerator performance scaling

Outline

- Paper
- Abstract
- Introduction
- Related Work
- System Architecture
- Inter-Controller Network
- Prototype System
- Result
- Conclusion & Future Work

Conclusion & Future Work

- demonstrated
  - by having the inter-FPGA network connecting the controllers directly, each node is able to access re- mote storage with negligible performance degradation
  - offloading computation into the storage controller as an accelerator provides performance benefits against implementing acceleration as a separate appliance
- Next BlueDBM planned improvements
  - Improved FTL
    - to optimize writes, design wear leveling, garbage collection, write amplification reduction algorithms
  - DRAM Caching
    - We can cache reads and writes to the SSD in DRAM on the FPGA board. because of our low latency inter-FPGA network, we could create a shared global DRAM cache from DRAM of all the nodes and dynamically partition them ac- cording to the workload characteristics.
  - Database Acceleration
    - we aim to further accelerate database management systems such as Postgres or SciDB by offloading database operations to the FPGA.