Accelerator-centric operating systems

Rethinking the role of CPUs in modern computers

Mark Silberstein
EE, Technion
System design challenge: Programmability and Performance
System design challenge: Programmability and Performance
Computer hardware: circa ~2000

Network Adapter

Storage controller

Graphical Processing Units (GPUs)

CPU: Size = transistor count

Size = transistor count
Systems software stack circa ~2000

Applications

OS

CPU

I/O devices
Computer hardware: circa ~2015

- Accelerators for encryption, media, signal processing....
- Network I/O accelerator
- GPU parallel accelerator
- Storage I/O accelerator
Central Processing Units (CPUs) are no longer Central

- Storage I/O accelerator
- Network I/O accelerator
- GPU parallel accelerator

Power
Performance
Programmability

Accelerators for encryption, media, signal processing....

Mark Silberstein - EE, Technion
Systems software stack circa 2015

Accelerated applications

OS

CPU

I/O accelerators

FPGAs

DSPs

GPUs
Software-hardware gap is widening

Accelerated applications

Inadequate abstractions and management mechanisms

OS

CPU

I/O

I/O accelerators

FPGAs

DSPs

GPUs
THE problem:
CPU - centric software architecture
Breaking the CPU-centric system design

Hardware is here
We need OS support
Accelerator-centric OS architecture

- CPU Applications
- Accelerator applications
- Accelerator OS support
  (Interprocessor I/O, file system, networking APIs, memory management)
- Hardware support for OS
  - I/O accelerators
  - FPGAs
  - DSPs
  - GPUs

OS
- Accelerator I/O services (network, files)
- Accelerator abstractions and mechanisms
This talk

Accelerator applications

Accelerator OS support
(Interprocessor I/O,
file system, networking APIs

Accelerator I/O services (network, files)

Accelerator abstractions and mechanisms

Hardware support for OS

CPU

OS

I/O accelerators
FPGA
DSPs
GPUs

CPU Applications

Storage

Network

ASPLOS13, TOCS14, OSDI14, CACM14

Application

GPUs
• GPU 101 and motivation

• GPUnet: Network Stack for GPUs
• GPUfs: File access support for GPUs

• Recap: Accelerator-centric OS architecture
Hybrid GPU-CPU 101
Architecture
Co-processor model

CPU

Computation

Memory

GPU

Memory
Co-processor model
Co-processor model

CPU

Computation

Memory

GPU

kernel

Memory
Co-processor model
GPUs make a difference...

- Top 10 fastest supercomputers use GPUs
- GPUs enable order-of-magnitude speedups in ...
  - Physics
  - Vision
  - HCI
  - Meteorology
  - Graph Algorithms
  - Deep Nets
  - Bioinformatics
  - Linear Algebra
  - Finance
  - …
GPUs make a difference, but why only in HPC?

- Top 10 fastest supercomputers use GPUs
- GPUs enable order-of-magnitude speedups in:
  - Physics
  - Vision
  - HCI
  - Meteorology
  - Graph Algorithms
  - Deep Nets
  - Bioinformatics
  - Linear Algebra
  - Finance
  - ...

Web servers, Network services, Antivirus, File search, ????
Programming complexity exposed

Example: GPU-accelerated server
CPU server

NIC — CPU — Memory

recv() — compute() — send()
Inside a GPU-accelerated server

Theory

recv()
GPU_compute()
send()
Inside a GPU-accelerated server

**Theory**

```plaintext
recv()
GPU_compute()
send()
```

```plaintext
recv();
batch();
```
Inside a GPU-accelerated server

```
transfer();
```

```
CPU
```

```
GPU
```

```
Memory
```

```
 NIC
```

```
Memory
```

```
recv();
batch();
optimize();
transfer();
```

**Theory**

```
recv();
```

```
GPU_compute();
```

```
send();
```
Inside a GPU-accelerated server

CPU

Memory

NIC

invoke();

recv();

batch();

optimize();

transfer();

balance();

GPU_compute();

 recv();
 batch();
 optimize();
 transfer();
 balance();
 GPU_compute();

Theory

recv()

GPU_compute()

send()
Inside a GPU-accelerated server

```c
transfer();

CPU

Memory

GPU

Memory

NIC

recv();

batch();

optimize();

transfer();

balance();

GPU_compute();

scatter();

send();

recv();

GPU_compute();

transfer();

balance();

cleanup();

Theory
```
Inside a GPU-accelerated server

Theory

recv()
GPU_compute()
send()

recv();
batch();
optimize();
transfer();
balance();
GPU_compute();
transfer();
cleanup();
dispatch();
send();
Inside a GPU-accelerated server

- CPU
- NIC
- Memory
- Memory

Aggressive pipelining
Buffering, asynchrony, multithreading

recv();
batch();
optimize();
transfer();
balance();
GPU_compute();
transfer();
cleanup();
dispatch();
send();
recv();
batch();
optimize();
transfer();
balance();
GPU_compute();
transfer();
cleanup();
dispatch();
send();
unnecessary

This code is for a **CPU** to manage a **GPU**

```c
batch();
optimize();
transfer();
balance();
```

unnecessary

```c
GPU_compute();
transfer();
cleanup();
dispatch();
```

Mark Silberstein - EE, Technion
GPUs are not co-processors

GPUs are peer-processors

They need I/O abstractions
GPUnet: socket API for GPUs

Application view

```
socket(AF_INET,SOCK_STREAM);
connect("node0:2340");
```

**GPU native server**

```
socket(AF_INET,SOCK_STREAM);
listen(:2340)
```

**GPU native client**

```
socket(AF_INET,SOCK_STREAM);
connect("node0:2340");
```

**CPU client**

```
socket(AF_INET,SOCK_STREAM);
connect("node0:2340")
```
GPU-accelerated server with GPUnet

**CPU not involved**

```
recv()
GPU_compute()
send()
```
GPU-accelerated server with GPUnet

```
recv()
GPU_compute()
send()
```
GPU-accelerated server with GPUnet

No request batching

recv()  send()

NIC  Memory

recv()  GPU_compute()  recv()  GPU_compute()  recv()  GPU_compute()

send()  send()  send()  send()  send()  send()

Transparent pipelining
GPU-accelerated server with GPUnet

Seamless buffer management
GPUnet design

- GPU Socket API
- Reliable in-order streaming
- Reliable channel

RDMA Transports
- *Infiniband*

Non-RDMA Transports
- *UNIX Domain Sockets, TCP/IP*

Simplicity

Performance
GPUs: file access for GPUs

Application view

System-wide shared namespace

open("shared_file")

host_file_system

CPU

GPU 1

GPU 2

GPU 3

write()

mmap()

POSIX (CPU)-like API

Persistent storage

Mark Silberstein - EE, Technion
Face verification server

CPU client (unmodified) via rsocket

GPU server (GPUnet)

Infiniband

memcached (unmodified) via rsocket

recv()

GPU_features()

query_DB()

GPU_compare()

send()
Face verification: Different implementations

Throughput (KReq/sec) vs Latency (μsec)

1 GPU (no GPUnet)

1 GPU GPUnet

1.9x throughput
1/3x latency (500usec)
½ LOC

CPU
6 cores

Mark Silberstein - EE, Technion
Recap: Accelerator-centric OS design
Why OS layer on accelerators?
To abstract away...

• …Hardware interaction overhead
• …Programming model gap
• …I/O and memory performance gap
• …I/O topology
Challenges

- Hardware
  - Memory consistency, NUMA, limitations

- Systems software
  - No OS hardware support, physical device sharing, state sharing

- Applications
  - Data layout reorganization, resource management
Central Processing Units (CPUs) are no longer Central

Power
Performance
Programmability

Accelerators for encryption, media, signal processing....

Accelerator-centric OS architecture

CPU Applications
Accelerator applications

Accelerator OS support
(Interprocessor I/O, file system, networking APIs, memory management)

OS
Accelerator abstractions and mechanisms

Hardware support for OS
CPU I/O

I/O accelerators FPGAs DSPs GPUs
Central Processing Units (CPUs) are no longer central.

Accelerator-centric OS architecture

- CPU Applications
- Accelerator applications
  - Accelerator I/O services (network, files)
  - Accelerator OS support (interprocessor I/O, file system, networking APIs, memory management)

- OS
  - Accelerator abstractions and mechanisms

- Hardware support for OS
  - CPU accelerators
  - I/O accelerators
  - FPGAs
  - DSPs
  - GPUs

Power, Performance, Programmability

Accelerators for encryption, media, signal processing...
Central Processing Units (CPUs) are no longer Central

Power
Performance
Programmability

Accelerators for encryption, media, signal processing...

Accelerator-centric OS architecture

CPU

I/O

Accelerator applications

Accelerator I/O services (network, files)

OS

Accelerator abstractions and mechanisms

Hardware support for OS

I/O accelerators

FPGAs

DSPs

GPUs

Mark Silberstein - EE, Technion
Coming up next...

**Distributed accelerator** applications
High concurrency servers

**Multi-accelerator** OS support
Interprocessor I/O,
file system, networking APIs,
VM, memory consistency, isolation, security

I/O accelerators ➔ FPGAs ➔ DSPs ➔ GPUs
Team

• Accelerated systems group, Technion
  • Amir Wated, Sagi Shachar, Feras Daud, Pavel Lifshitz

Collaborators

• Operating System Architecture group, UT Austin
  • Sangman Kim, Yige Hu, Emmett Witchel
Accelerator-centric OS design

GPUsfs

GPUnet

Looking for a graduate degree in systems? We're hiring!

mark@ee.technion.ac.il