Accelerating Intelligence

John D. Davis, Ph.D.
Hardware Accelerators Break Through the Processing Wall

What about “Data Gravity”?
Push compute near the Data
Inhibitor: Programming Model Gap for Hardware Accelerators

**BIG DATA PLATFORMS**

- Presto
- Kafka
- Spark
- MySQL
- TensorFlow™
- Elastic
- Hive

**Data Science Programming Model**

**Acceleration Programming Model**

**Programming Model Gap**

**Focus on Macroarchitecture**

**Focus on Microarchitecture**

**Skills Gap**

**Data Scientists & Developers**

- Many-cores
- GPU
- FPGA

Performance Engineering

---

Bigstream Confidential © 2018
Introducing: Bigstream Hyper-acceleration Layer

Goal: Provider of Open-ecosystem for Hyper-acceleration

Address the whole data market
Zero code change

Cross platform
Intelligent, automatic computation slicing
Cross acceleration hardware

2X to 30X acceleration
Apache Spark

Client Application

Big Data Platform APIs

Master Node

Application Master

Catalyst

Resource management messages

Extended Query Optimization Strategies

Cluster Management

Tasks

Executors

Node Manager

Spark Task

Bigstream Confidential © 2018
Non-accelerated Spark

User Space

File System

OS

NVMe / PCIe Driver

Hardware

Host CPU

Host DRAM

Host Interface Controller

SSD

OSS/3rd Party
Non-accelerated Spark

User Space

OS

NVMe / PCIe Driver

Semantic Wall

Hardware

Host CPU

Host DRAM

Host Interface Controller

SSD

Accel

OSS/3rd Party

User Space File System

SQL+time series+risk models + ... scan + filter + shuffle

Object/file (read/write)

101010001111…..

Non-accelerated Spark

User Space

OS

NVMe / PCIe Driver

Semantic Wall

Hardware

Host CPU

Host DRAM

Host Interface Controller

SSD

Accel

OSS/3rd Party

User Space File System

SQL+time series+risk models + ... scan + filter + shuffle

Object/file (read/write)

101010001111…..

Non-accelerated Spark

User Space

OS

NVMe / PCIe Driver

Semantic Wall

Hardware

Host CPU

Host DRAM

Host Interface Controller

SSD

Accel

OSS/3rd Party

User Space File System

SQL+time series+risk models + ... scan + filter + shuffle

Object/file (read/write)

101010001111…..
Bigstream Seamless Acceleration of Apache Spark

- Zero Code Change
- Cross Platform
- Intelligent, Automatic Computation Slicing
- Cross-Hardware Acceleration
- 2-30X Acceleration
Hyper-Acceleration Layered Compilation Approach

BIG DATA PLATFORMS
presto
kafka
Spark
MySQL
TensorFlow
elastic

Ingest
Row-based SQL Operators
Columnar SQL Operators
UDFs
Text Scan Operators
Basic ML Operators
Tensor/Array Operators

map
map group
Reduce/Agg
sort
LinAlg Blocking Dense
LinAlg Blocking Sparse
LinAlg Elementwise

Hardware agnostic Stream Processing Layer (Engines, State memories, Synchronizations + Patterns)

Hardware specific Stream Processing Layer (Mapping to Verilog engines, RISC-V cores, OpenGL kernels)

<table>
<thead>
<tr>
<th>Ingest</th>
<th>SQL</th>
</tr>
</thead>
</table>
| Compression (5 var.)
  JSON
  CSV
  AVRO
  Parquet
  ORC
  Kafka
  FIX
  PCRE | Map (12 var.)
  Project/Filter (40 var.)
  Sort
  Hashaggregate/
  Sortaggregate (16 var.)
  Window (15 var.)
  Join/Shuffle (12 var.)
  SQL UDF (7 var.) |
# HW Accelerator Engines

<table>
<thead>
<tr>
<th>Deserialization</th>
<th>JSON</th>
<th>CSV</th>
<th>Parquet (under development)</th>
<th>FIX (under development)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decompression</td>
<td>Snappy</td>
<td>GZIP (3rd Party)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encryption/Decryption</td>
<td>AES</td>
<td>Project</td>
<td>Filter</td>
<td>Sort</td>
</tr>
<tr>
<td>SQL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(.* ) Search/Regex</td>
<td>PCRE (3rd Party)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU Cores</td>
<td>RISC-V (3rd Party)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine Learning</td>
<td>Linear/Logistic Regression</td>
<td>K-means</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deep Learning</td>
<td>CNN (3rd Party)</td>
<td>RNN (3rd Party)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Networking</td>
<td>IP/UDP</td>
<td>IP/TCP (3rd Party)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
General Application and HW Characteristics

• Application characteristics:
  • Batch and streaming analytics applications
  • Collection of computation/transformation stages
  • Terabytes to Petabytes or more of data
  • Large clusters: 10’s – 10,000’s servers
  • I/O (network and/or storage) and compute bound

• How we use the FPGA:
  • Build out pipelines and import as kernels
  • Computational overlays (Time Division Multiplexing)
  • Offload and Inline (Storage and Network) acceleration
  • Moving between Streaming, Memory Mapped, and P2P DMA with Software
  • Bringing Computation to the Data.
1. Identify and load FPGA bitstream based on acceleration template match.

Offload Base Accelerator Ecosystem
1. Identify and load FPGA bitstream based on acceleration template match

2. Software configuration of FPGA to customize hardware template for the application
1. Identify and load FPGA bitstream based on acceleration template match
2. Software configuration of FPGA to customize hardware template for the application
3. Issue “accelerated” compute task (requires I/O requests to the SSD)
1. Identify and load FPGA bitstream based on acceleration template match
2. Software configuration of FPGA to customize hardware template for the application
3. Issue “accelerated” compute task (requires I/O requests to the SSD)
4. Copy input data from host to FPGA memory and back again to the application user space memory to complete
1. Identify and load FPGA bitstream based on acceleration template match.

2. Software configuration of FPGA to customize hardware template for the application.

HW Acceleration with Application Intelligence.

SQL+time series+risk models + ...

scan + filter + shuffle

Object/file (read/write)

User Space

User Space File System

FPGA Driver

NVMe / PCIe Driver

OS

Host CPU

Host DRAM

Host Interface Controller

SSD

FPGA

OSS/3rd Party

bigstream (Existing Stack)
Overall Acceleration Comparison

Baseline – No Acceleration

Hardware Acceleration Only: \(~3x\)

Software + Hardware Acceleration: \(~5x\)
Offload Performance: 5 Node Cluster

TPC-DS SpeedUp

1 Master Node + 4 Worker Nodes
Spark baseline vs FPGA acceleration with Zero Code Change
• Rewind for Computational Storage
1. Identify and load FPGA bitstream based on acceleration template match
2. Software configuration of FPGA to customize hardware template for the application
1. Identify and load FPGA bitstream based on acceleration template match

2. Software configuration of FPGA to customize hardware template for the application

3. Issue “accelerated” I/O + compute requests to the SSD to pump data into the FPGA
Computational Storage

1. Identify and load FPGA bitstream based on acceleration template match
2. Software configuration of FPGA to customize hardware template for the application
3. Issue "accelerated" I/O + compute requests to the SSD to pump data into the FPGA
4. FPGA copies the result to the application user space
OpenCL Code

• Memory-mapped DMA

// Read from SSD part

buffer = (char*) malloc(size);

bytes_read = fread(buffer, 1, size, _fd);

// WRITE PART- from host to fpga

err = clEnqueueWriteBuffer(commands, buffer, CL_TRUE, 0, (data_actual_size + 96), json_write_host_mem, 0, NULL, &write_event);

clWaitForEvents(1, &write_event);

//Actual compute task

err = clEnqueueTask(commands, kernel, 0, NULL, &kernel_event);

clWaitForEvents(1, &kernel_event);

//READ PART - from fpga to host

err = clEnqueueReadBuffer(commands, json_data_ptr, CL_TRUE, s_read_offset, bytes_to_read + 32, son_read_host_mem, 0, NULL, &readevent2);

Same code can run in VU9P+ and ZU19, just change the target device
P2P Comparison

• SmartSSD (SSD + FPGA)
  • 45% Utilization (BRAM, LUTs, DSPs)
  • System level results 6c Xeon Workstation

• Spark vs P2P DMA

• Single FPGA, FPGA Pooling, & Clusters of FPGAs
  • TPC-DS with 340 GB to 2.7 TB
  • Up to 2 SmartSSDs/server
  • 5 node cluster (1 master node and 4 worker nodes)
System-Level Results

• Row-based engine with TPC-DS queries
  • Lower CPU utilization and fewer cores
  • Higher device I/O bandwidth
  • Lower host PCIe bandwidth
  • Reduced CPU DRAM Utilization

• Opportunities
  • Not all DMAs are the same.
  • Flexibility to mix and match DMA
  • When to use P2P?
P2P Performance: 5 Node Cluster

1 Master Node + 4 Worker Nodes
Spark baseline vs FPGA acceleration with Zero Code Change
Accelerators Everywhere!
Open Hyper-Acceleration Initiative

• Goal: Extend the development community around hyper-acceleration

• Enable different developers from completely differently skill set
  • HW Template and engine developers
  • HW platform developers (add new FPGAs and FPGA cards)
  • Add new Platforms (i.e., Presto, Hive, TF, etc.)

• Invitation only contributions at this time
  • If interested, please come talk to me
OHAI Open APIs

- OHAI APIs
- OHAI APIs

**BIG DATA PLATFORMS**
- presto
- kafka
- Spark
- MySQL
- TensorFlow
- elastic
- Hive

**Dataflow Adaptation Layer**
- Bigstream Dataflow
- Bigstream Hypervisor

**HYPER-ACCELERATION**
- Many-cores
- GPU
- FPGA
Conclusions

• **Flexibility**
  • Accelerating more than just compute: moving compute to the data vs. data to compute

• **Programmability**
  • Time division multiplexing/hardware overlays
  • Software and Hardware configuration

• **Generality**
  • Combining HW and SW acceleration transparently embedded in Big Data Platforms

• **Automatically**
  • Bigstream Hyper-Acceleration with Zero Code Change
Thank You

John D. Davis, Ph.D.
john@bigstream.co