Accelerating Intelligence

John D. Davis, Ph.D.

bigstream

Hardware Accelerators Break Through the Processing Wall

Microsoft Azure

WIRED

Microsoft Bets Its Future on a Reprogrammable Computer Chip

On track to millions of servers w/FPGAs

amazon web services

TC

AWS announces FPGA instances for its EC2 cloud computing service

Introduces FPGA powered server instances

Google Cloud Platform

WIRED

BUILDING AN AI CHIP SAVED GOOGLE FROM BUILDING A DOZEN NEW DATA CENTERS

TPUs help avoid cost of 12-15 data centers

intel FPGA

Forbes

Intel's Investment In FPGAs To Enable Reconfigurable Data Centers And AI Processing Of The Future

$16.7B acquisition of Altera

XILINX

BARRON'S

Xilinx May See Lift From Amazon, Microsoft

$1B in datacenter revenues

NVIDIA

BARRON'S

Nvidia Up Another 5%: Bulls Delighted Pondering 'Epic Disruption'

193% growth in datacenter segment

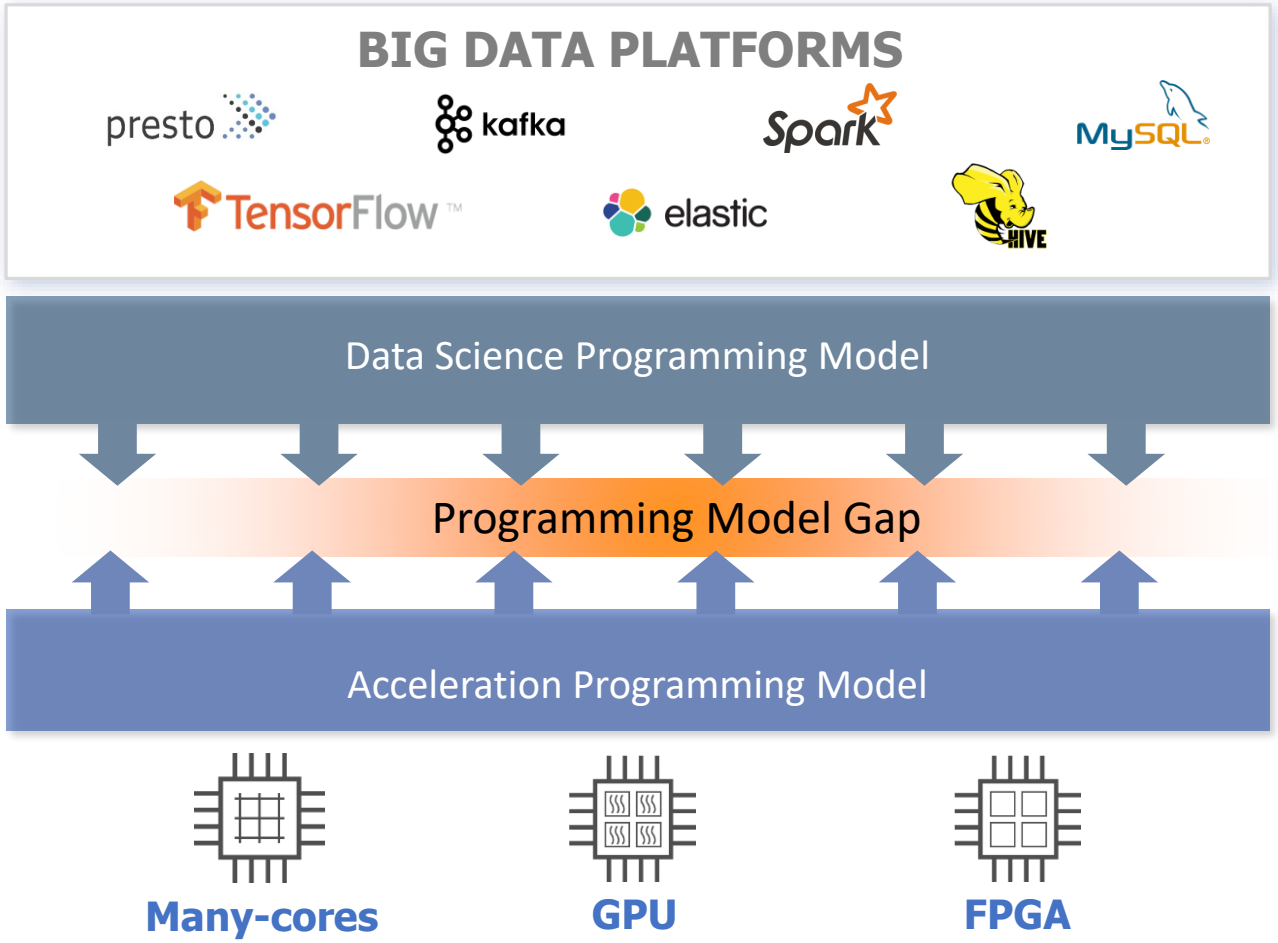
FPGA

ASIC

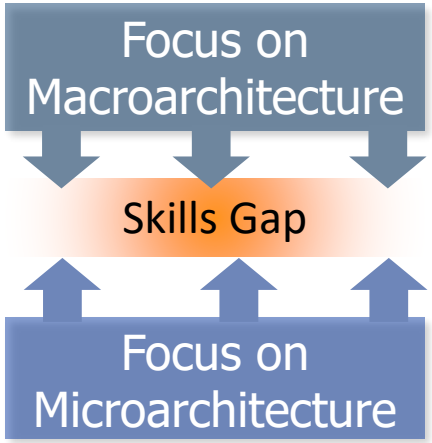
GPU

What about "Data Gravity"?
Push compute near the Data

Inhibitor: Programming Model Gap for Hardware Accelerators



Data Scientists & Developers



Performance Engineering

bigstream

Introducing: Bigstream Hyper-acceleration Layer

Goal: Provider of Open-ecosystem for Hyper-acceleration

BIG DATA PLATFORMS

presto, kafka, Spark, MySQL, TensorFlow, elastic, HIVE

Address the whole data market

Zero code change

bigstream HYPER-ACCELERATION

Dataflow Adaptation Layer

Bigstream Dataflow

Bigstream Hypervisor

Cross platform

Intelligent, automatic computation slicing

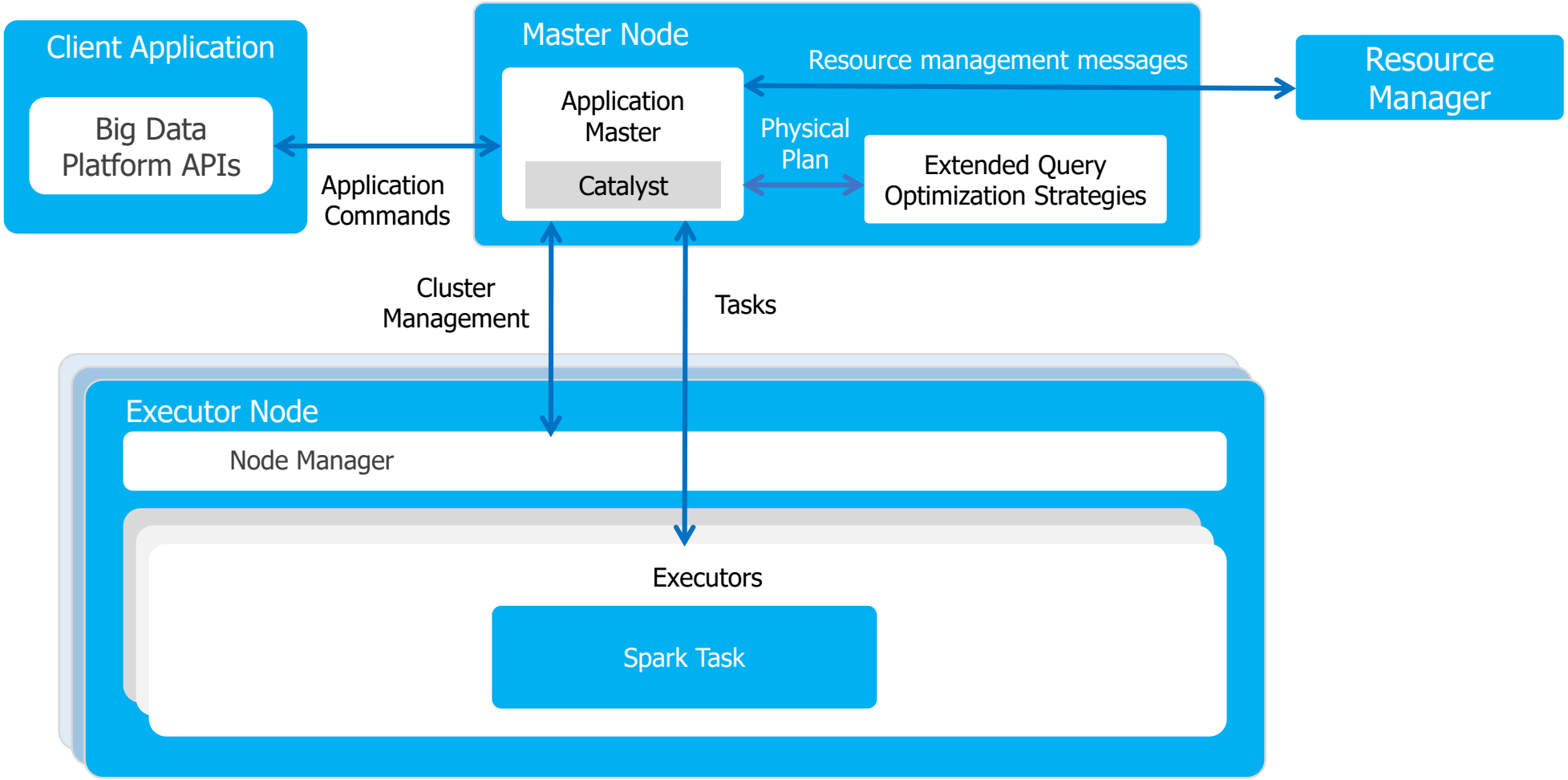
Cross acceleration hardware

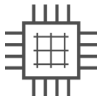
Many-cores | **GPU** | **FPGA**

2X to 30X acceleration

bigstream

Apache Spark

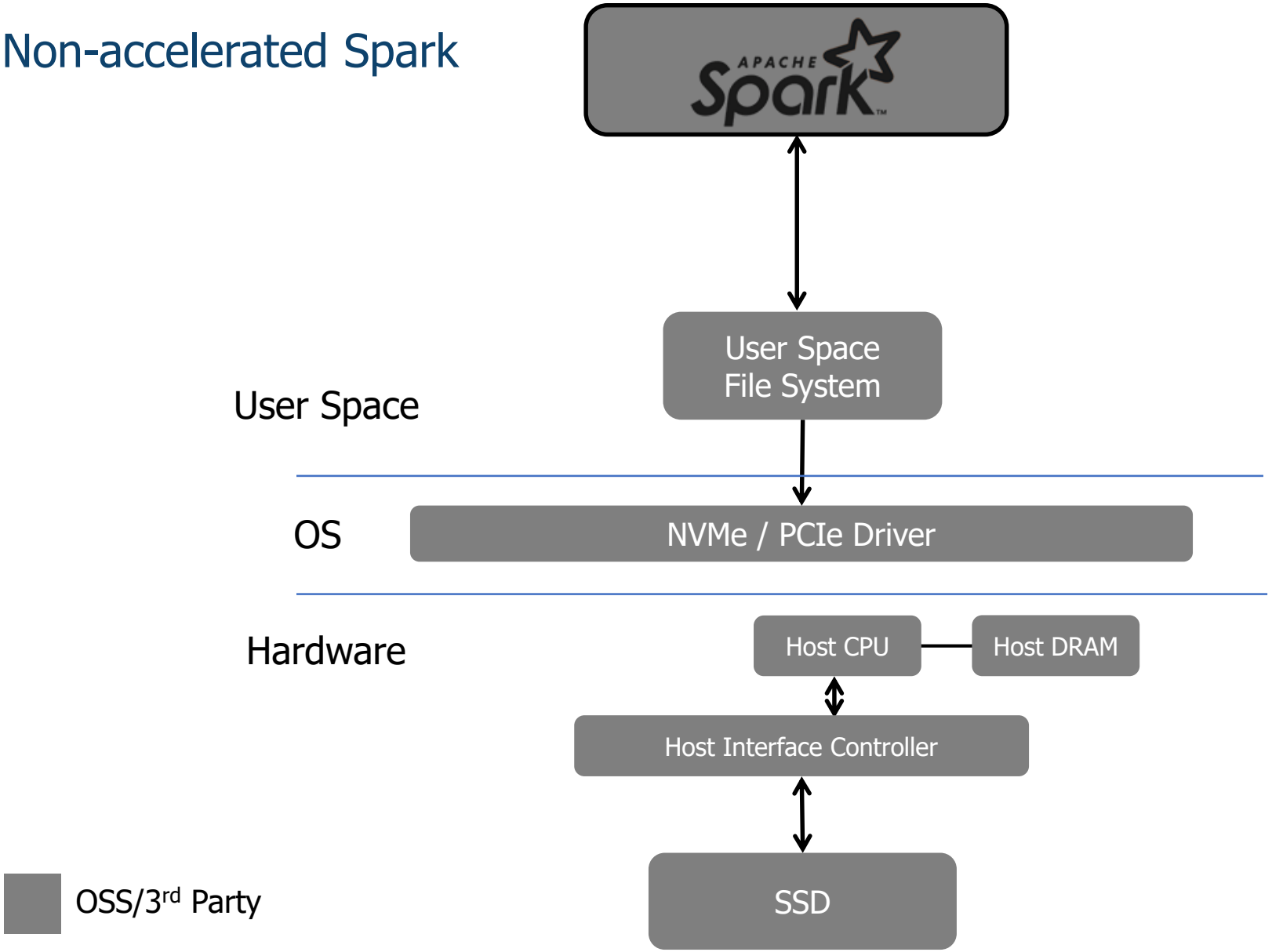




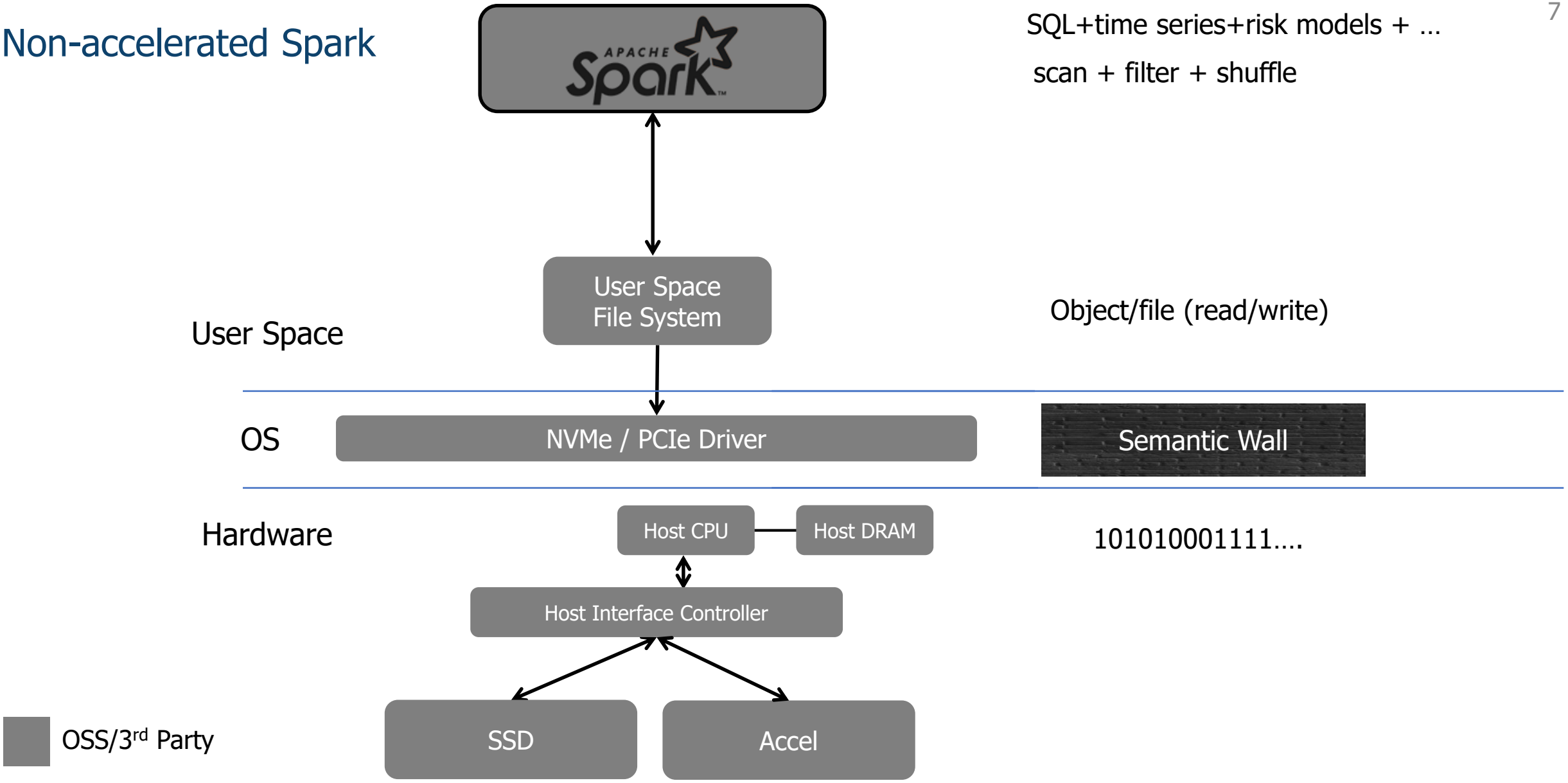
Many-cores

bigstream

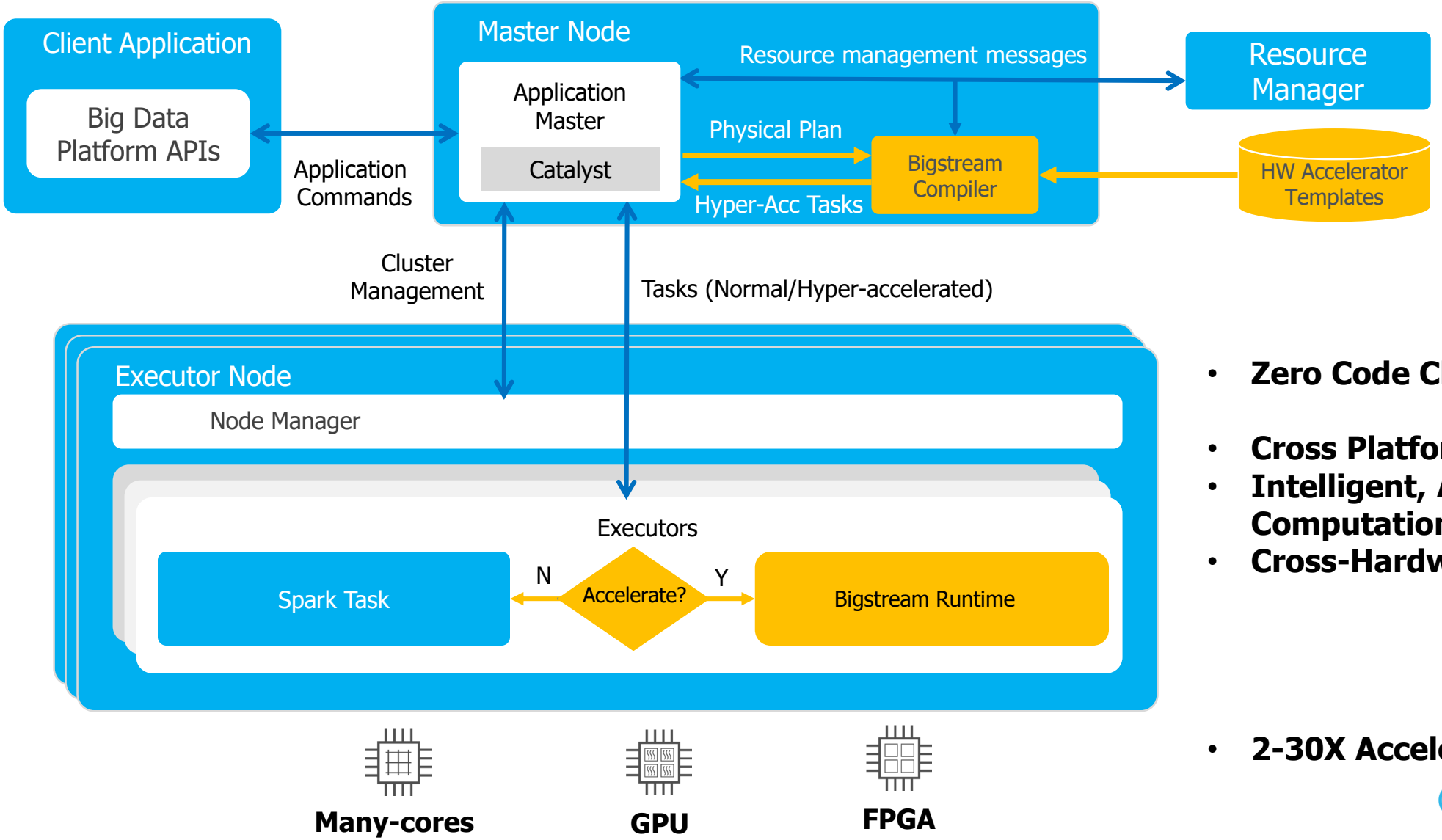
Non-accelerated Spark



Non-accelerated Spark



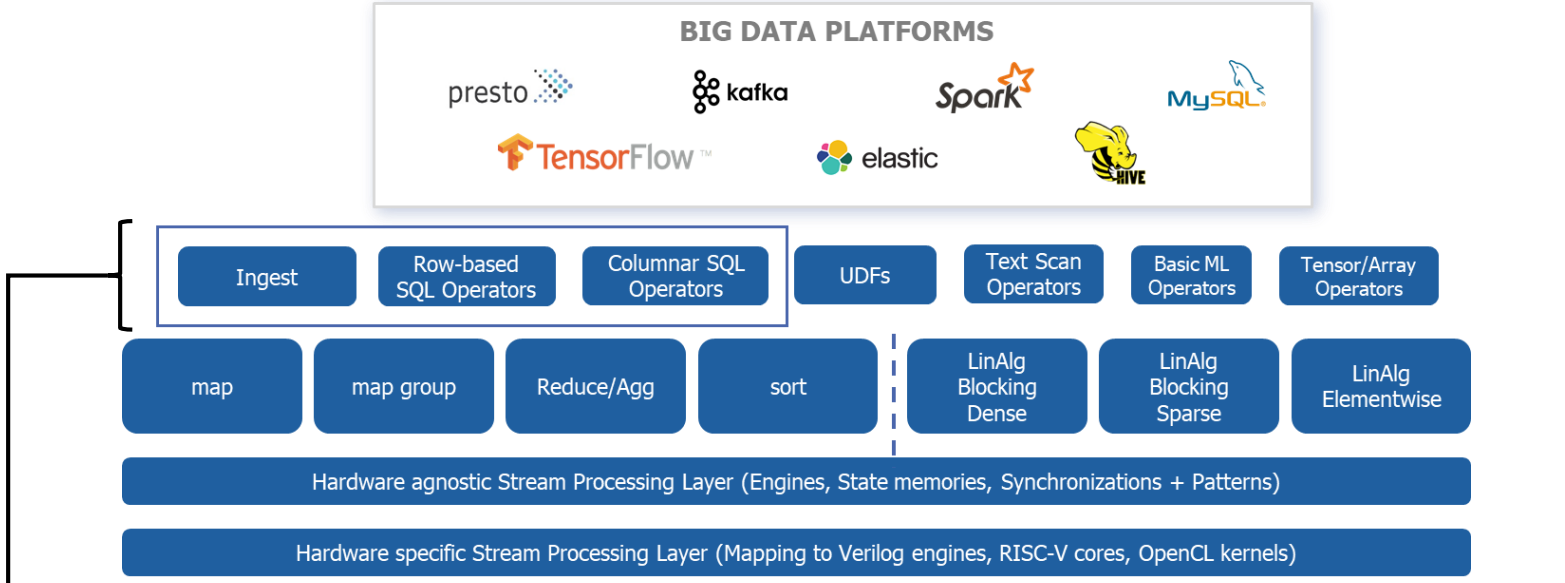
Bigstream Seamless Acceleration of Apache Spark



- **Zero Code Change**
- **Cross Platform**
- **Intelligent, Automatic Computation Slicing**
- **Cross-Hardware Acceleration**
- **2-30X Acceleration**

bigstream

Hyper-Acceleration Layered Compilation Approach



Ingest	SQL
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.)

bigstream

HW Accelerator Engines

Deserialization	JSON
	CSV
	Parquet (under development)
	FIX (under development)
Decompression	Snappy
	GZIP (3rd Party)
Encryption/ Decryption	AES
SQL	Project
	Filter
	Sort
	Hash Aggregate

Search/Regex	PCRE (3rd Party)
CPU Cores	RISC-V (3rd Party)
Machine Learning	Linear/Logistic Regression
	K-means
Deep Learning	CNN (3rd Party)
	RNN (3rd Party)
Networking	IP/UDP
	IP/TCP (3rd Party)

bigstream

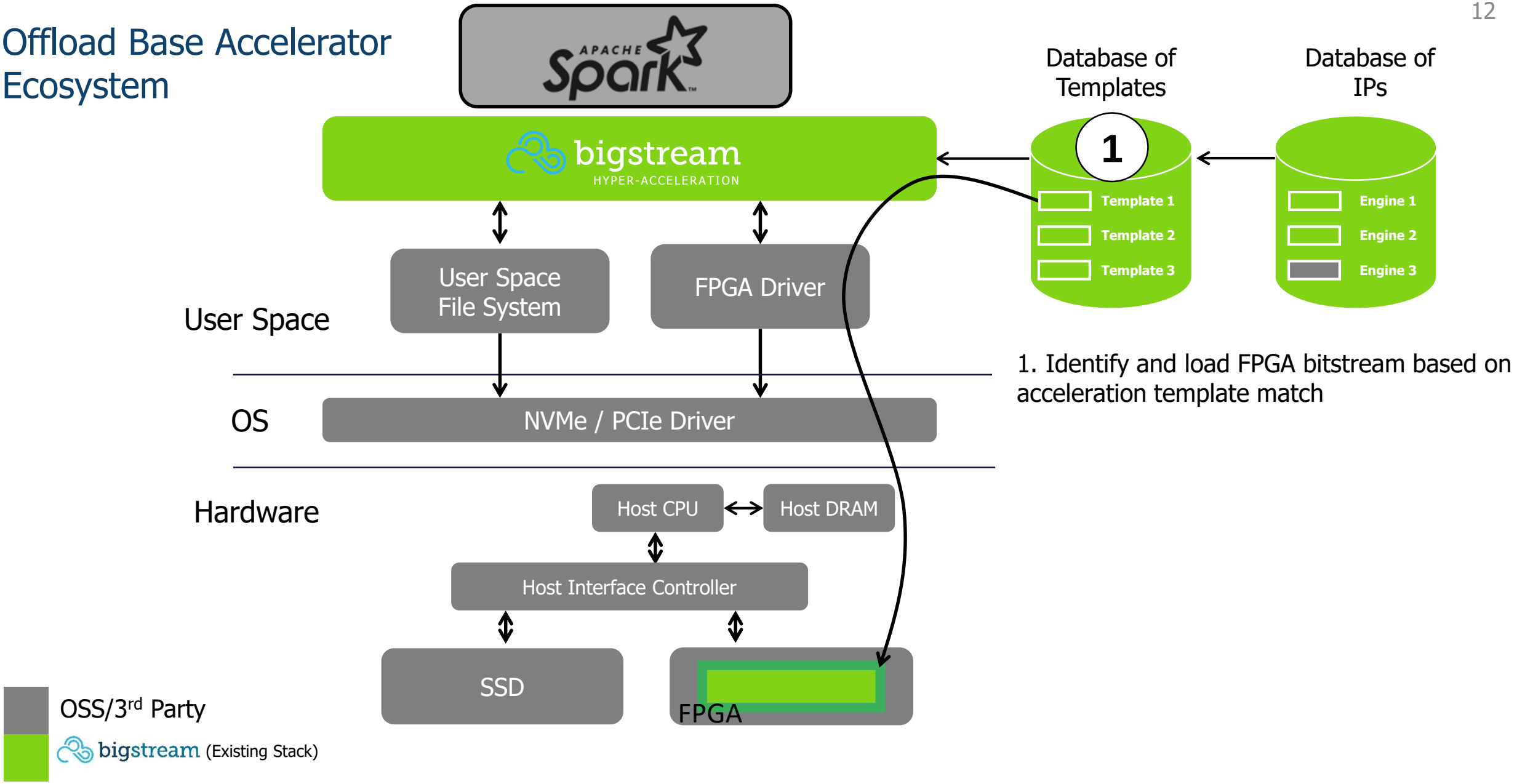
General Application and HW Characteristics

- 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.

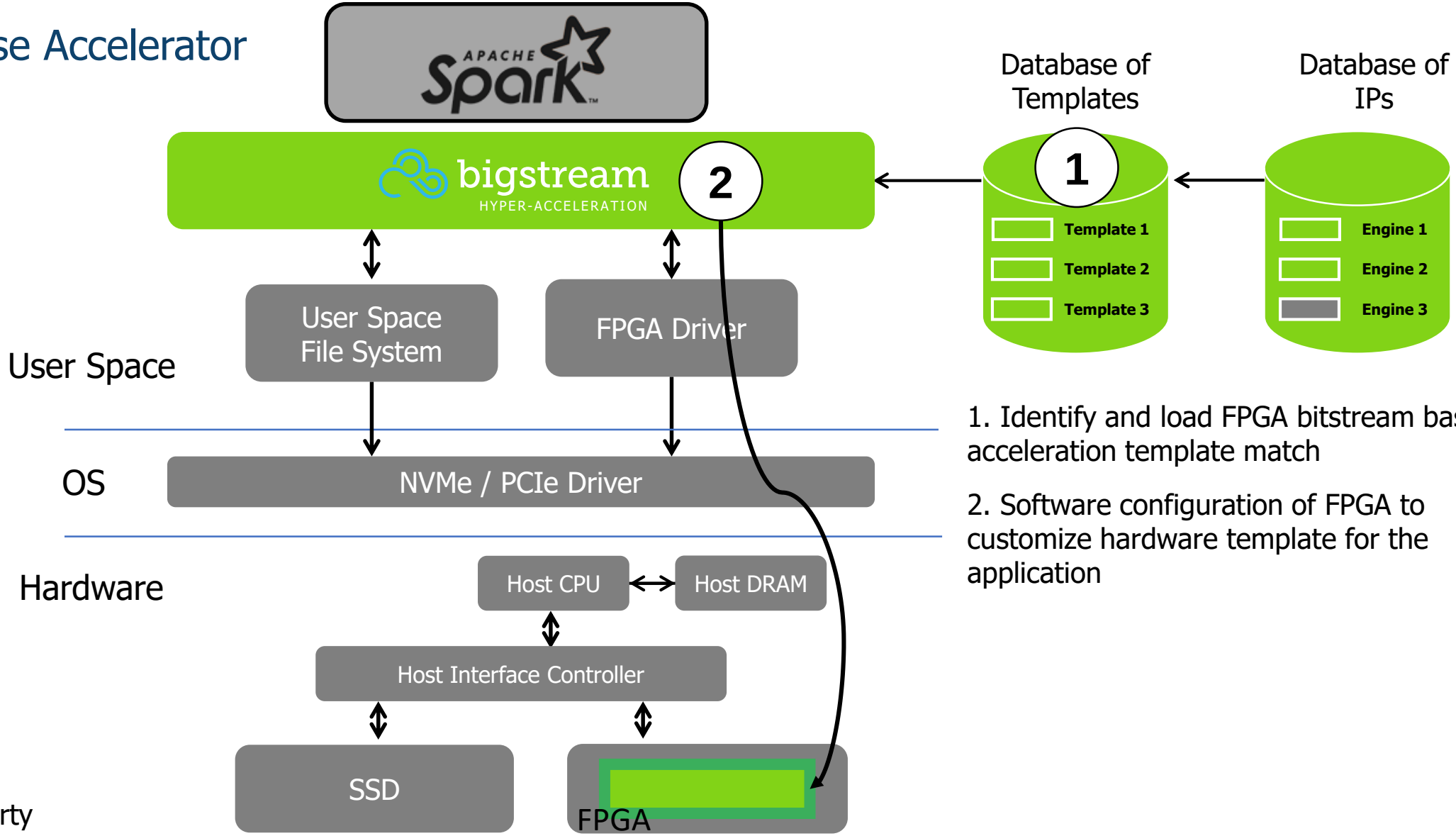
bigstream

Offload Base Accelerator Ecosystem



1. Identify and load FPGA bitstream based on acceleration template match

Offload Base Accelerator Ecosystem



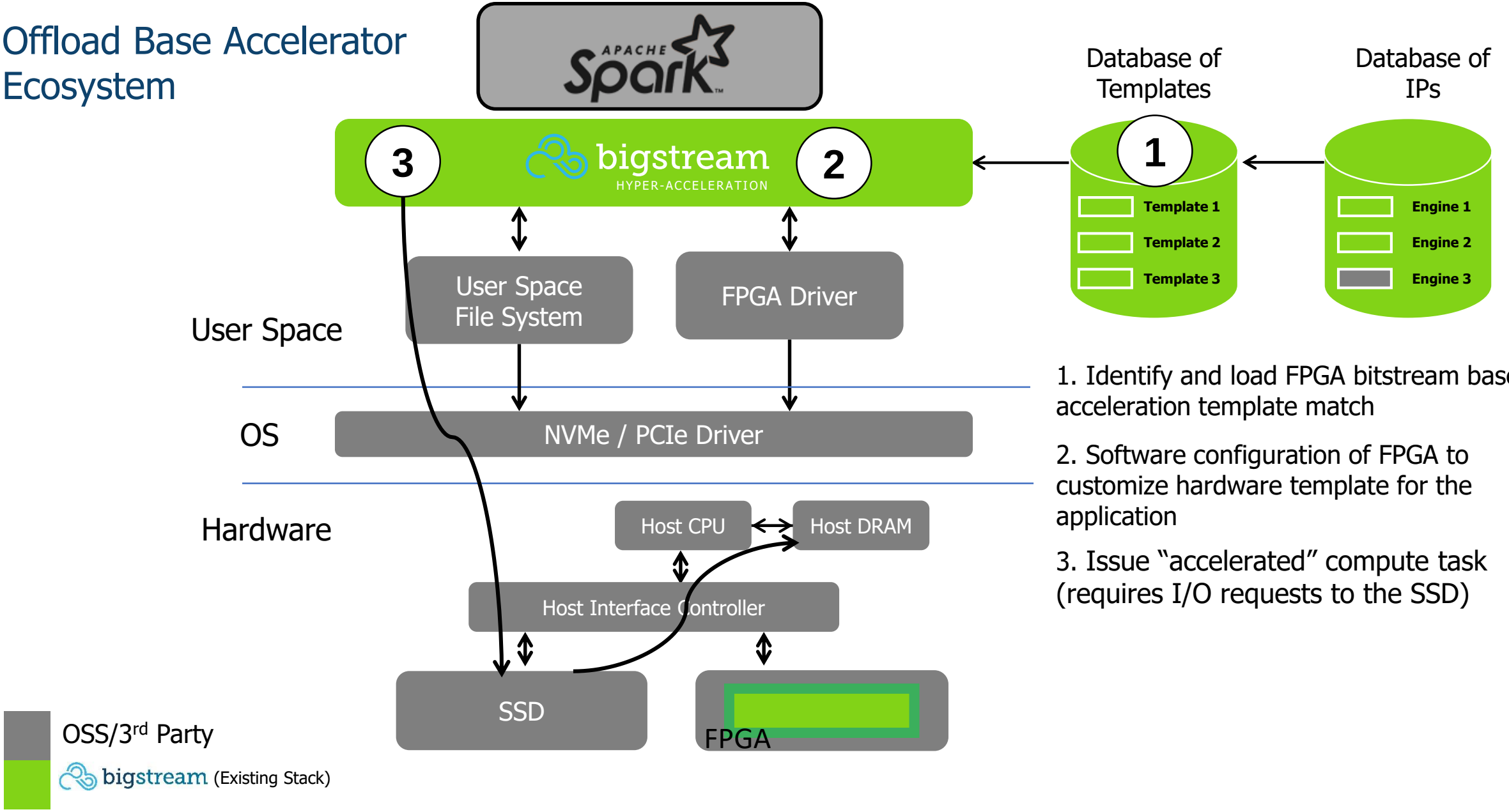
1. Identify and load FPGA bitstream based on acceleration template match

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

OSS/3rd Party

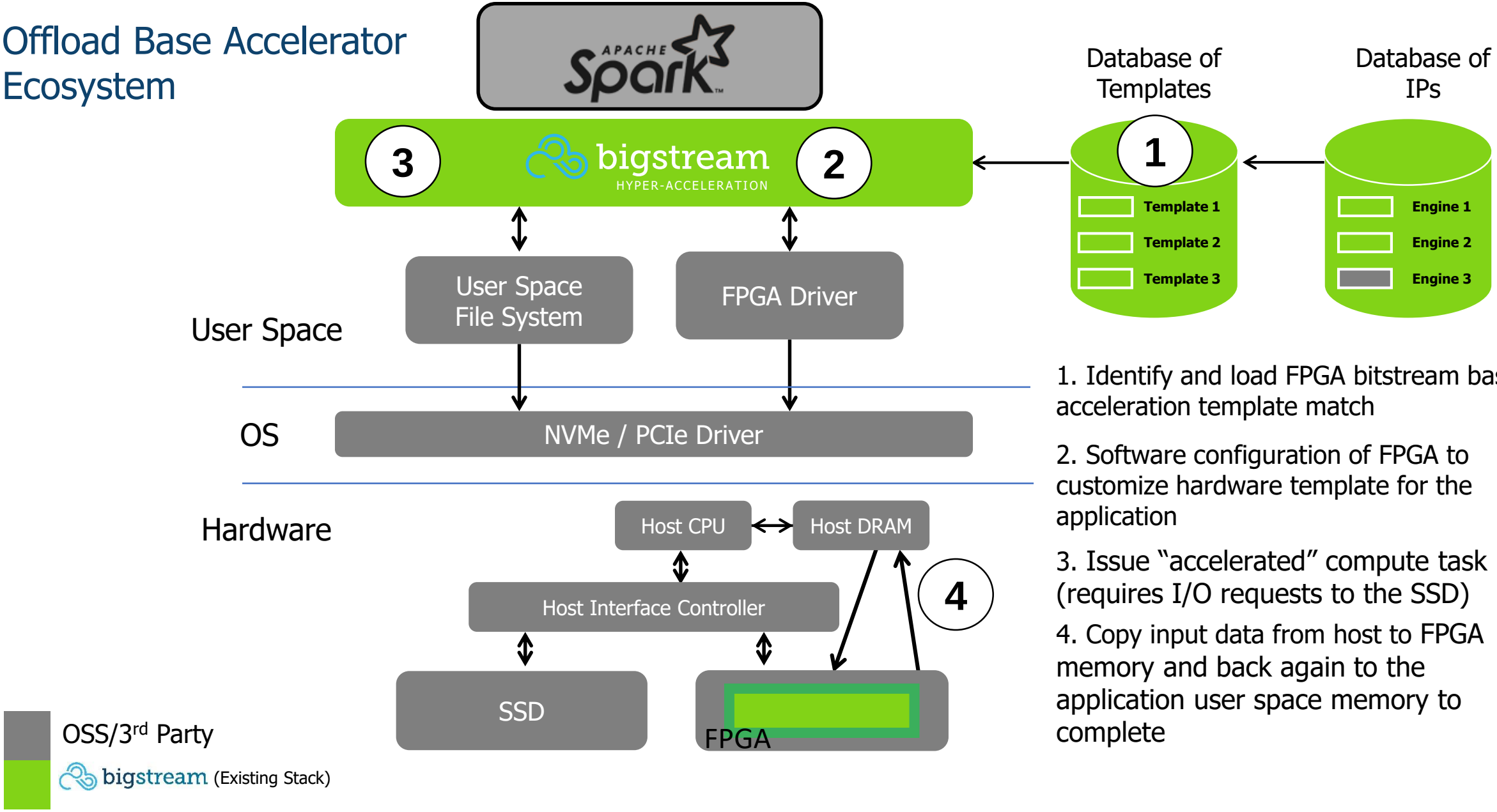
bigstream (Existing Stack)

Offload Base Accelerator Ecosystem

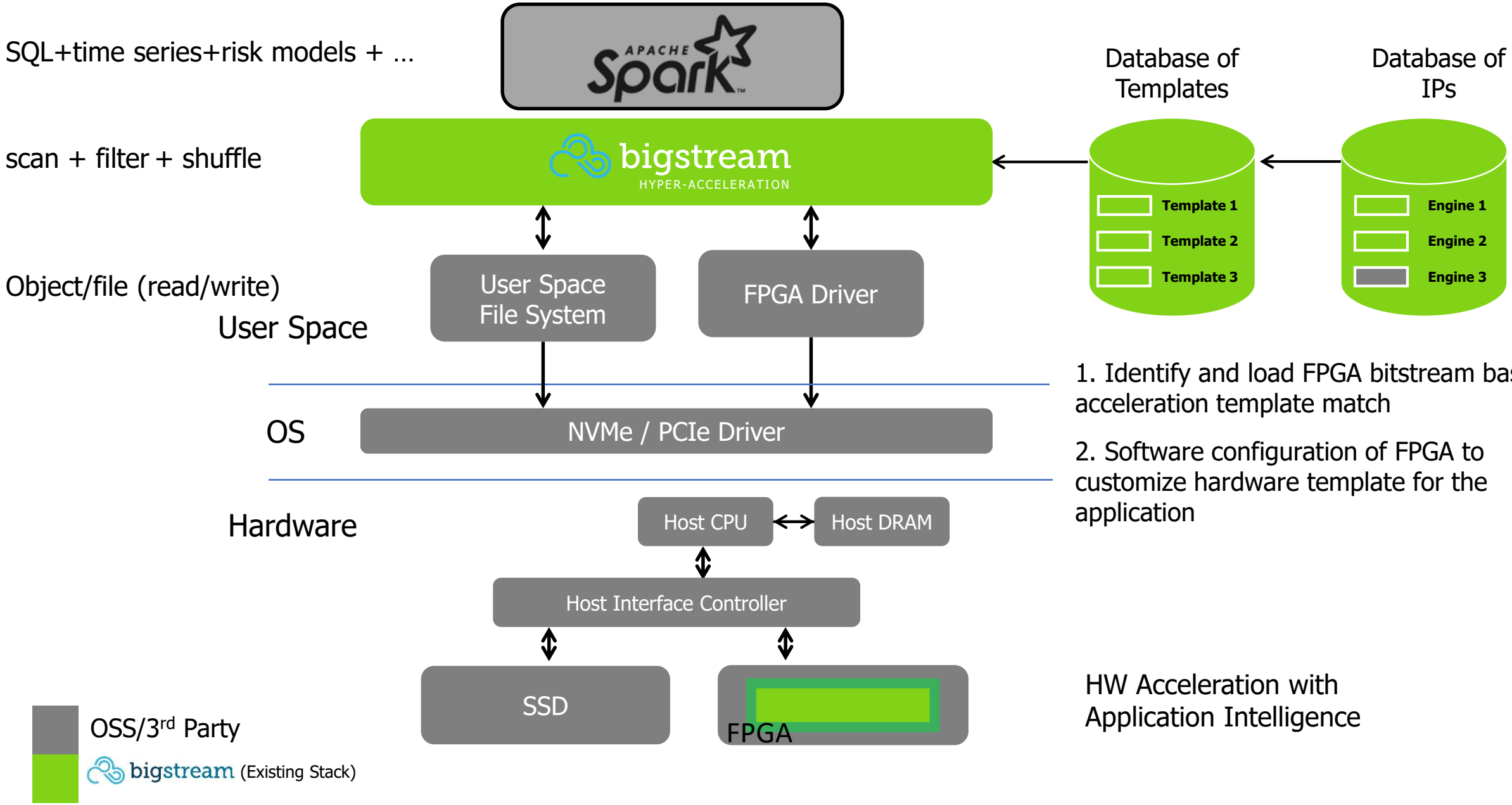


1. Identify and load FPGA bitstream based on acceleration template match
2. Software configuration of FPGA to customize hardware template for the application
3. Issue "accelerated" compute task (requires I/O requests to the SSD)

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
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

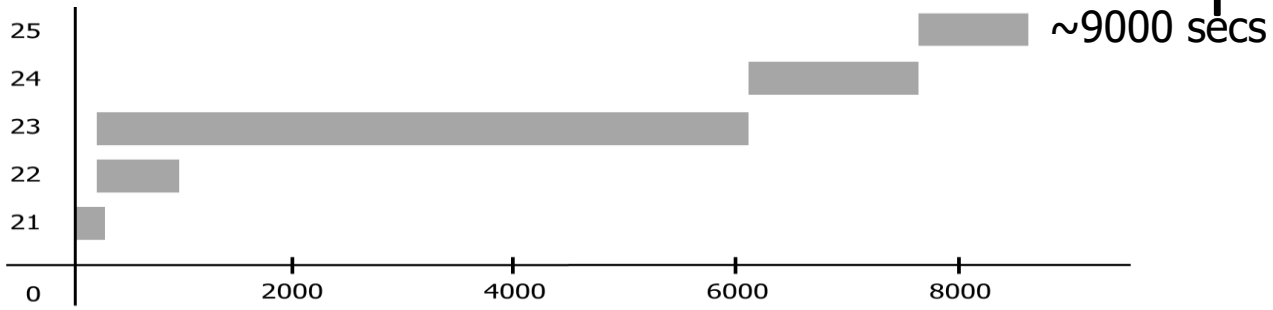


1. Identify and load FPGA bitstream based on acceleration template match

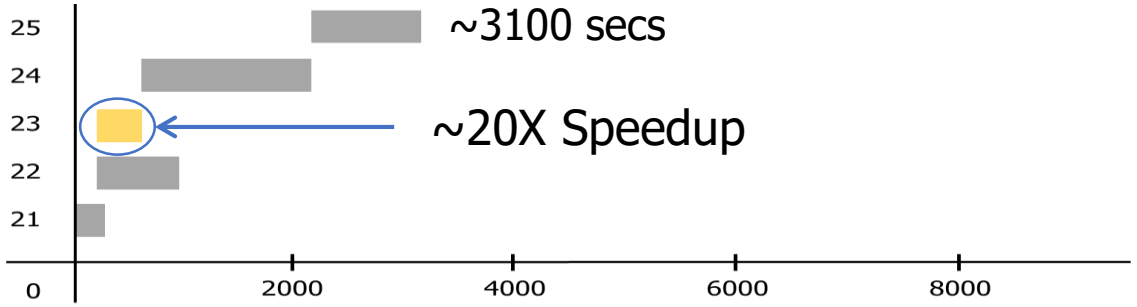
2. Software configuration of FPGA to customize hardware template for the application

HW Acceleration with Application Intelligence

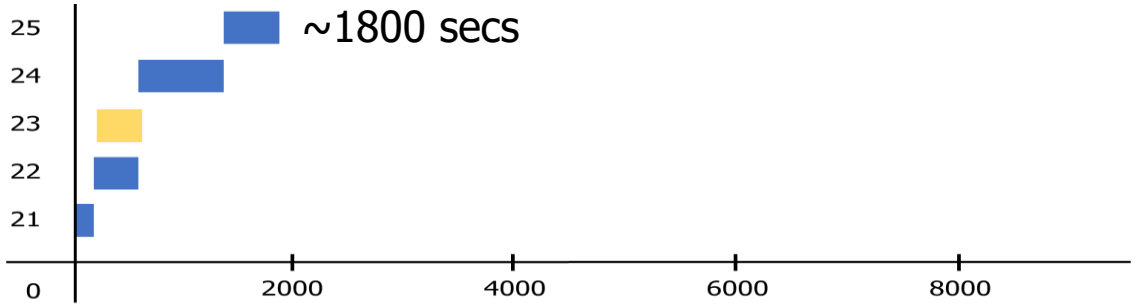
Overall Acceleration Comparison



Baseline – No Acceleration

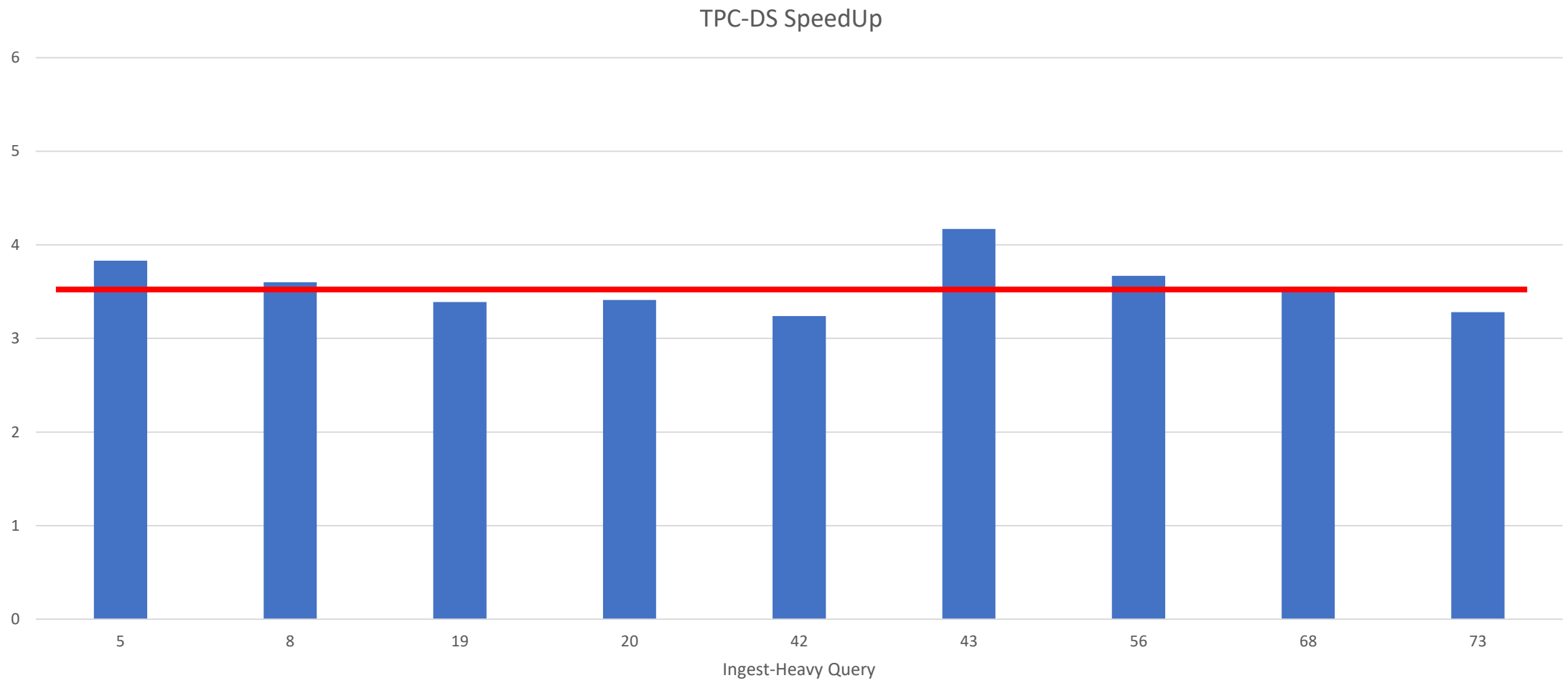


Hardware Acceleration Only: **~3x**



Software + Hardware Acceleration: **~5x**

Offload Performance: 5 Node Cluster



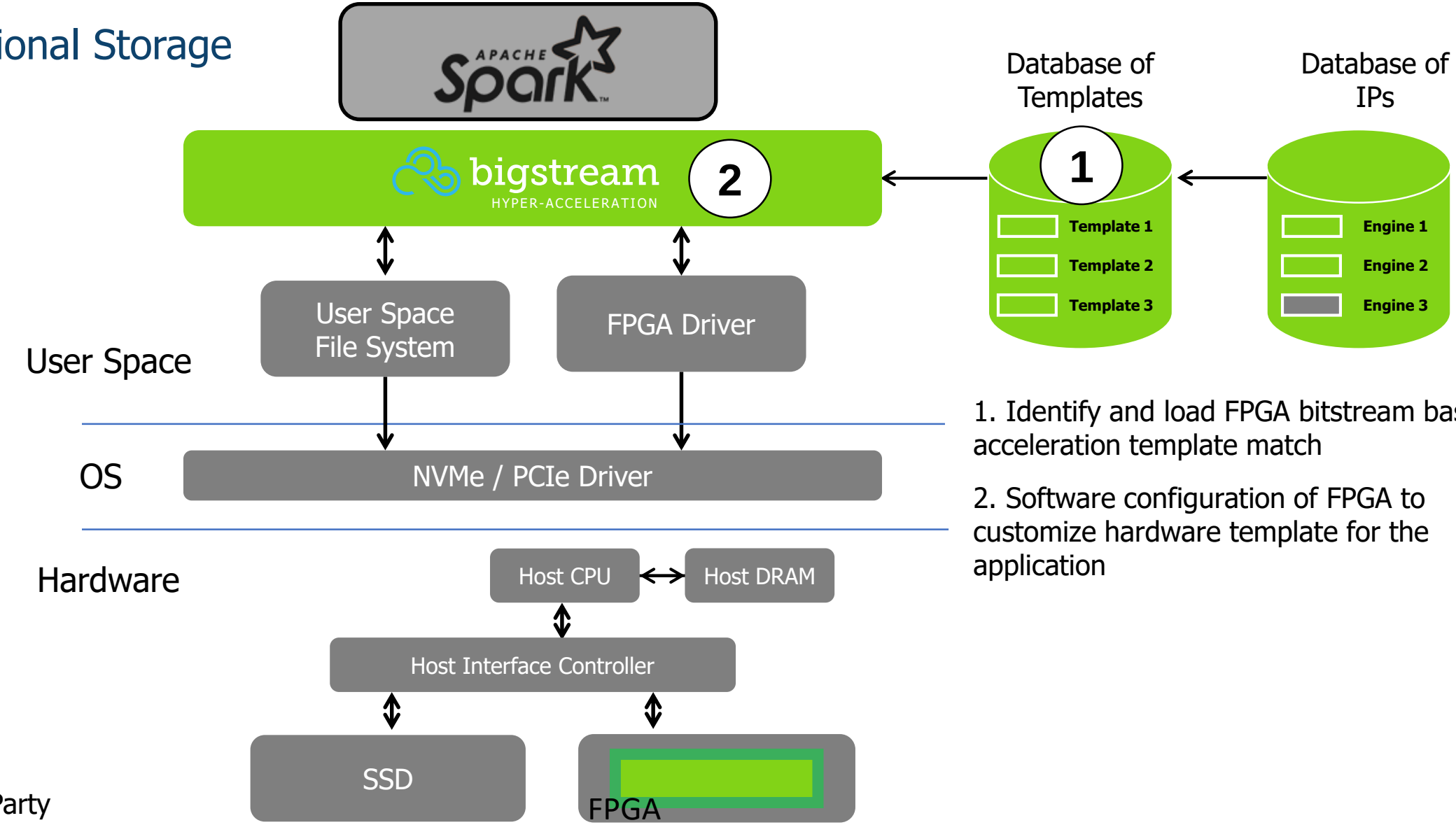
1 Master Node + 4 Worker Nodes

Spark baseline vs FPGA acceleration with Zero Code Change

- Rewind for Computational Storage

bigstream

Computational Storage



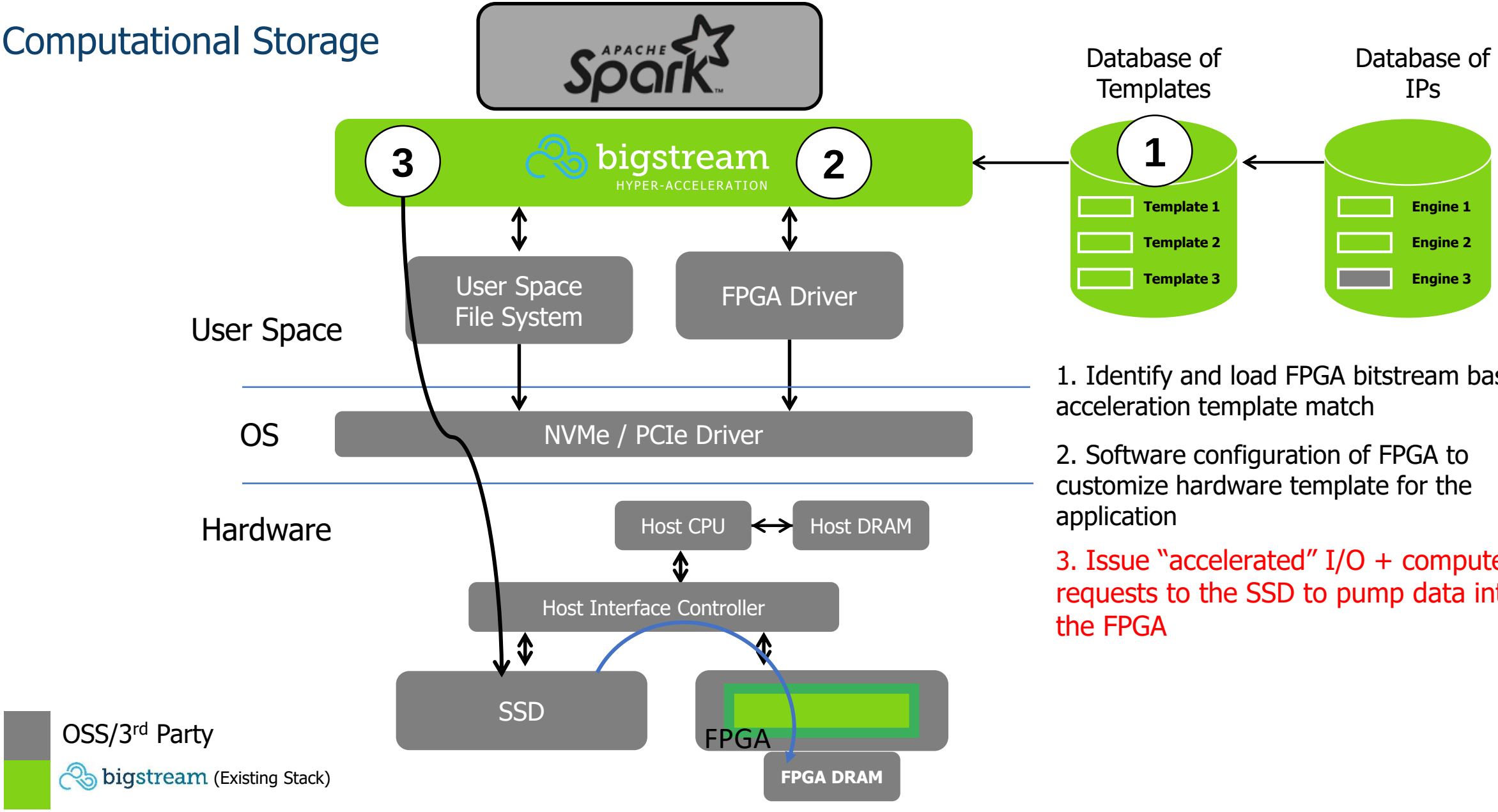
1. Identify and load FPGA bitstream based on acceleration template match

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

OSS/3rd Party

bigstream (Existing Stack)

Computational Storage

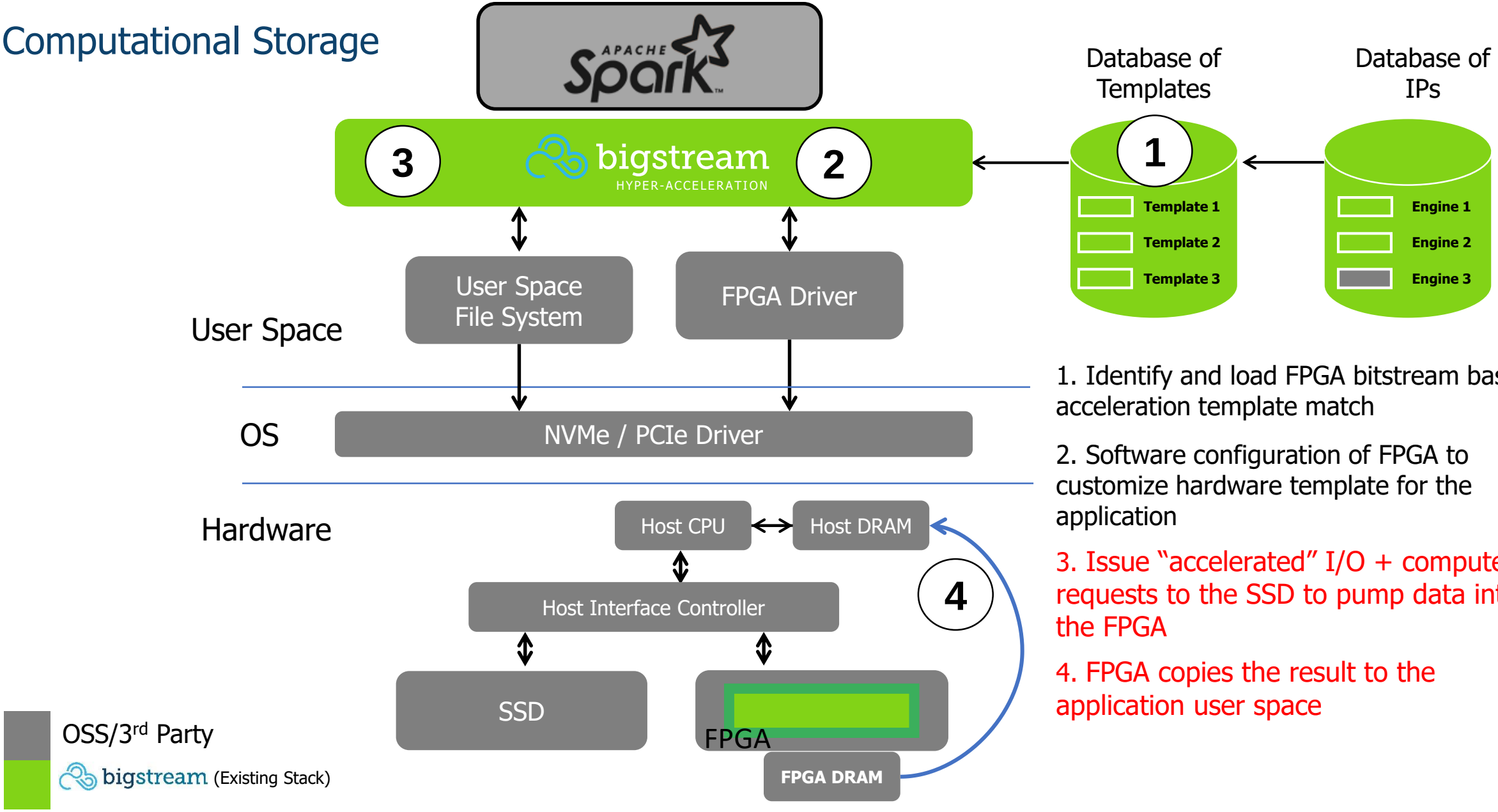


1. Identify and load FPGA bitstream based on acceleration template match

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

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);
```

> **P2P DMA**

```
// Read from SSD & write to fpga

clCreateBuffer(context[I], CL_MEM_READ_ONLY | CL_MEM_EXT_PTR_XILINX,
(aligned_size), & xmem_flags, & err);

_mapped_virtual_addr = clEnqueueMapBuffer(commands, d_axi00_ptr0_wr,

    CL_TRUE, CL_MAP_READ | CL_MAP_WRITE, 0, (aligned_size), 0, NULL, & map_event, & err);

clWaitForEvents(1, & map_event);

bytes_read = read(_fd, ((char * ) _mapped_virtual_addr), size);

//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

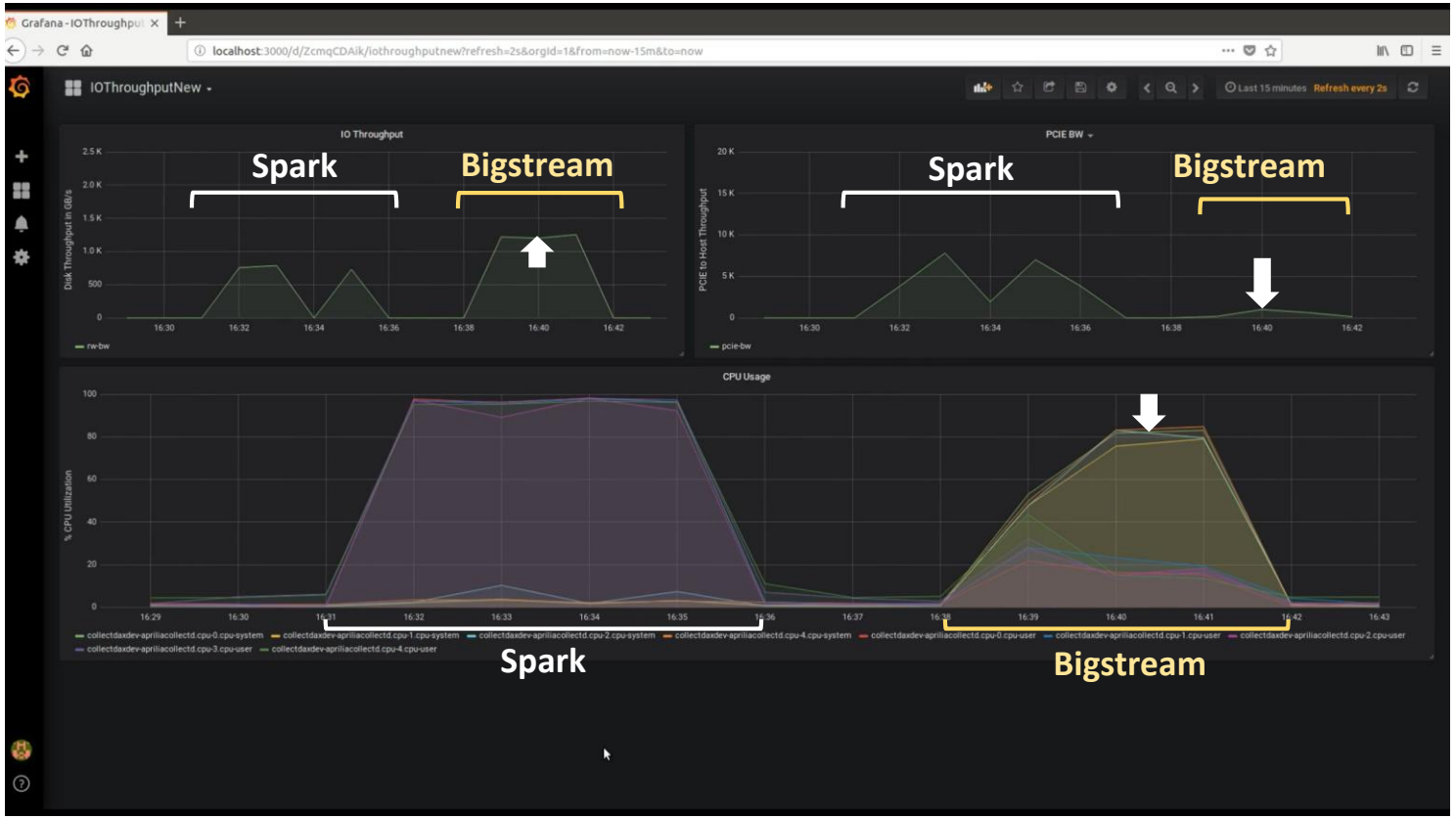
P2P Comparison

- 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)

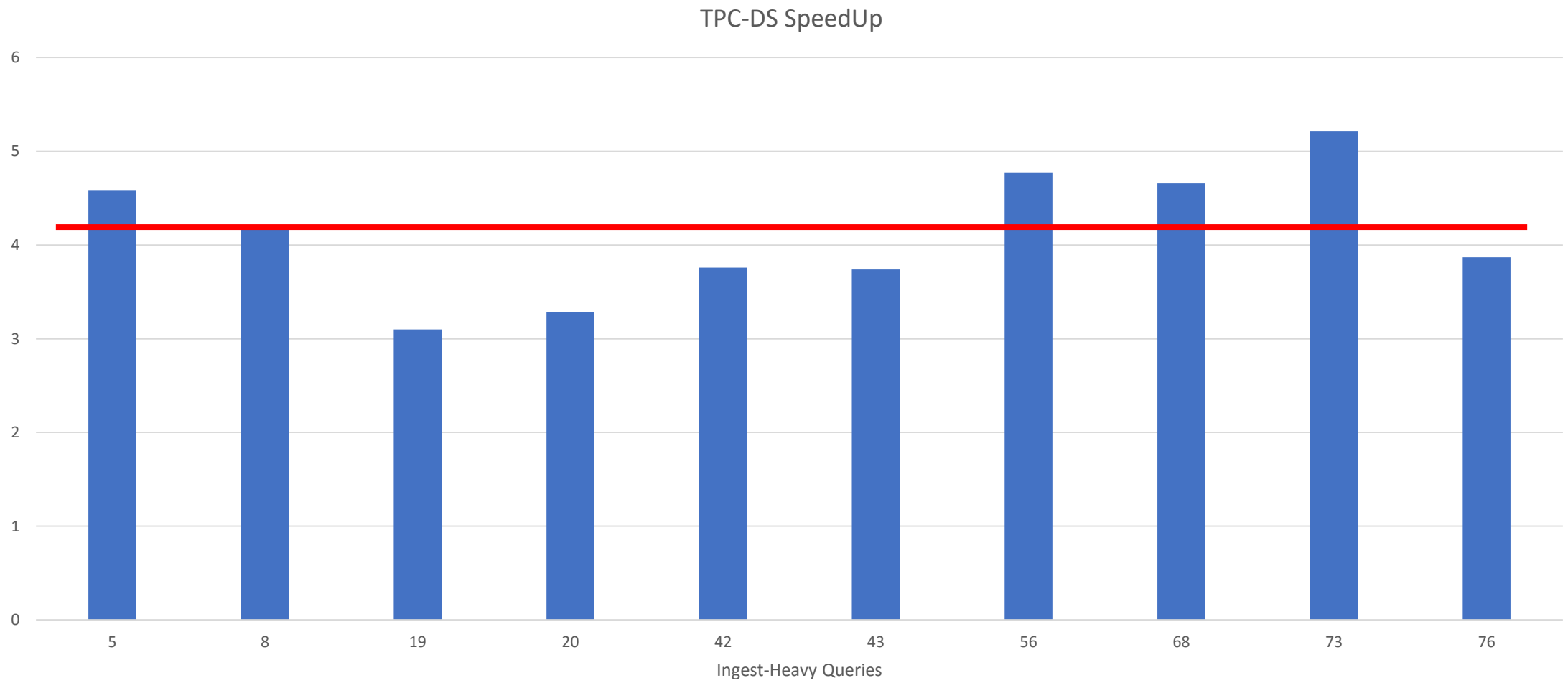
System-Level Results

- 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?



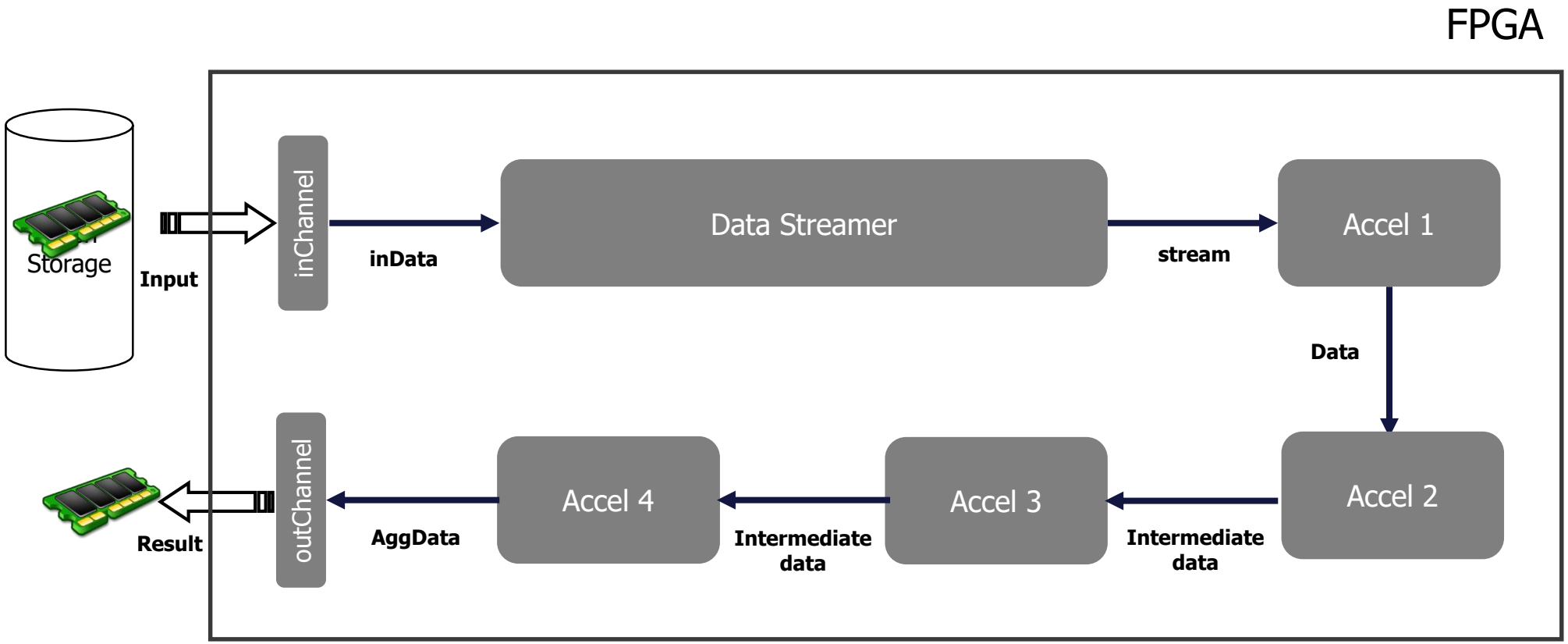
P2P Performance: 5 Node Cluster



1 Master Node + 4 Worker Nodes

Spark baseline vs FPGA acceleration with Zero Code Change

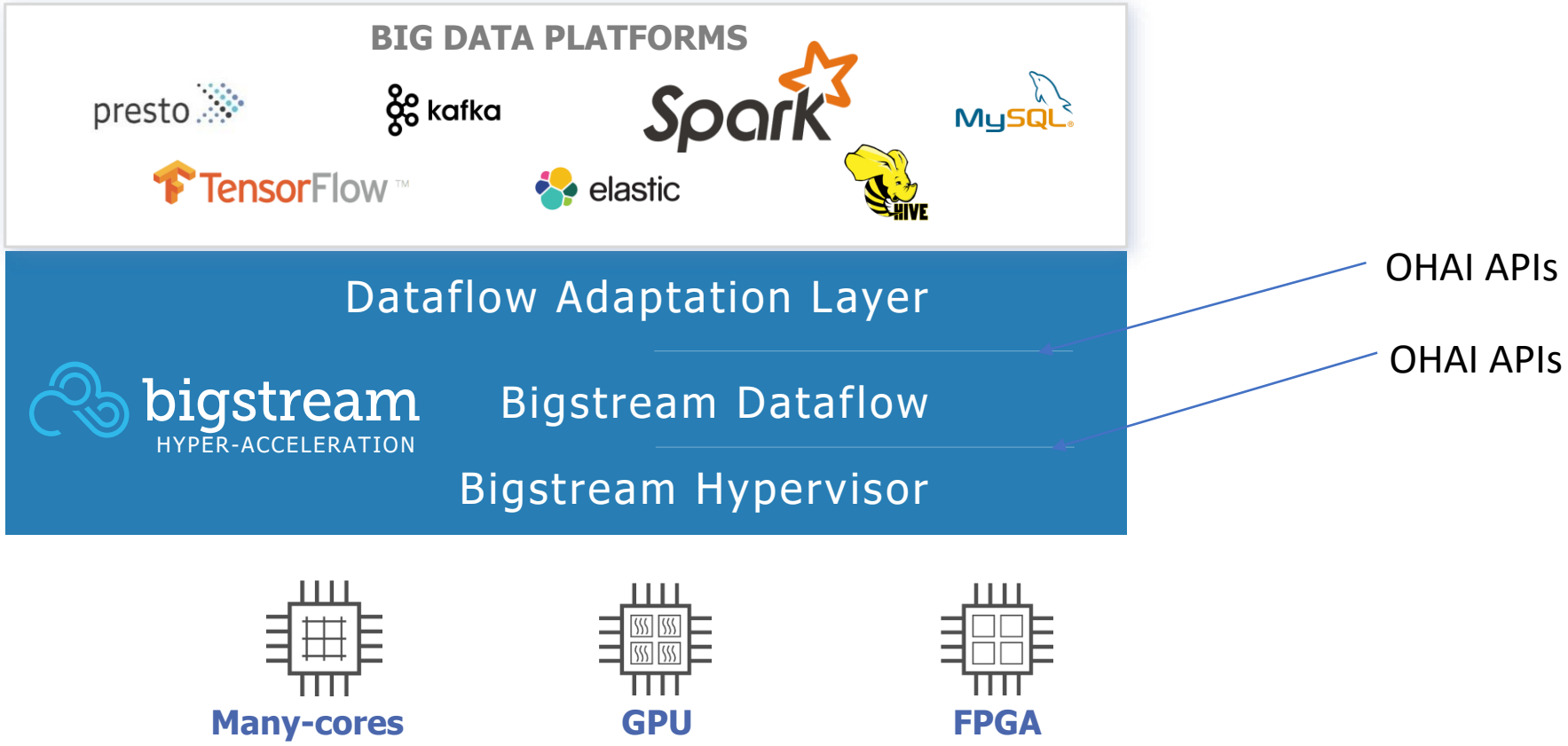
Accelerators Everywhere!



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



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

bigstream

Thank You

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