

# Accelerating Intelligence: Bigstream Hyper-Acceleration

Roop Ganguly



# Hardware Accelerators Break Through the Processing Wall




On track to millions of servers w/FPGAs




Introduces FPGA powered server instances




TPUs help avoid cost of 12-15 data centers



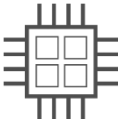

\$16.7B acquisition of Altera



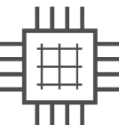

\$1B in datacenter revenues



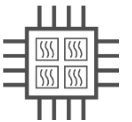

193% growth in datacenter segment



FPGA

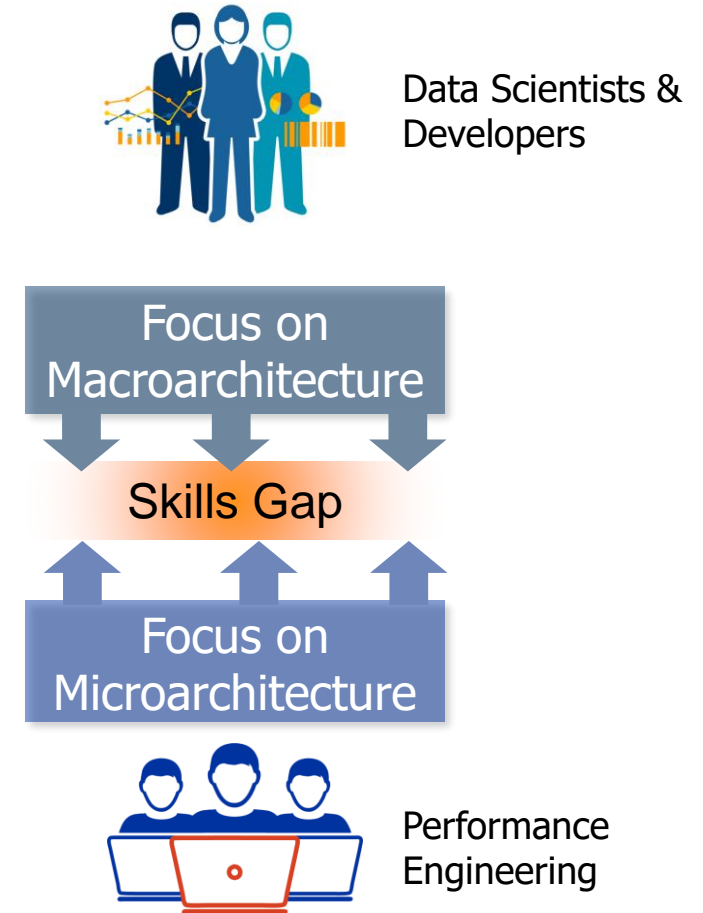
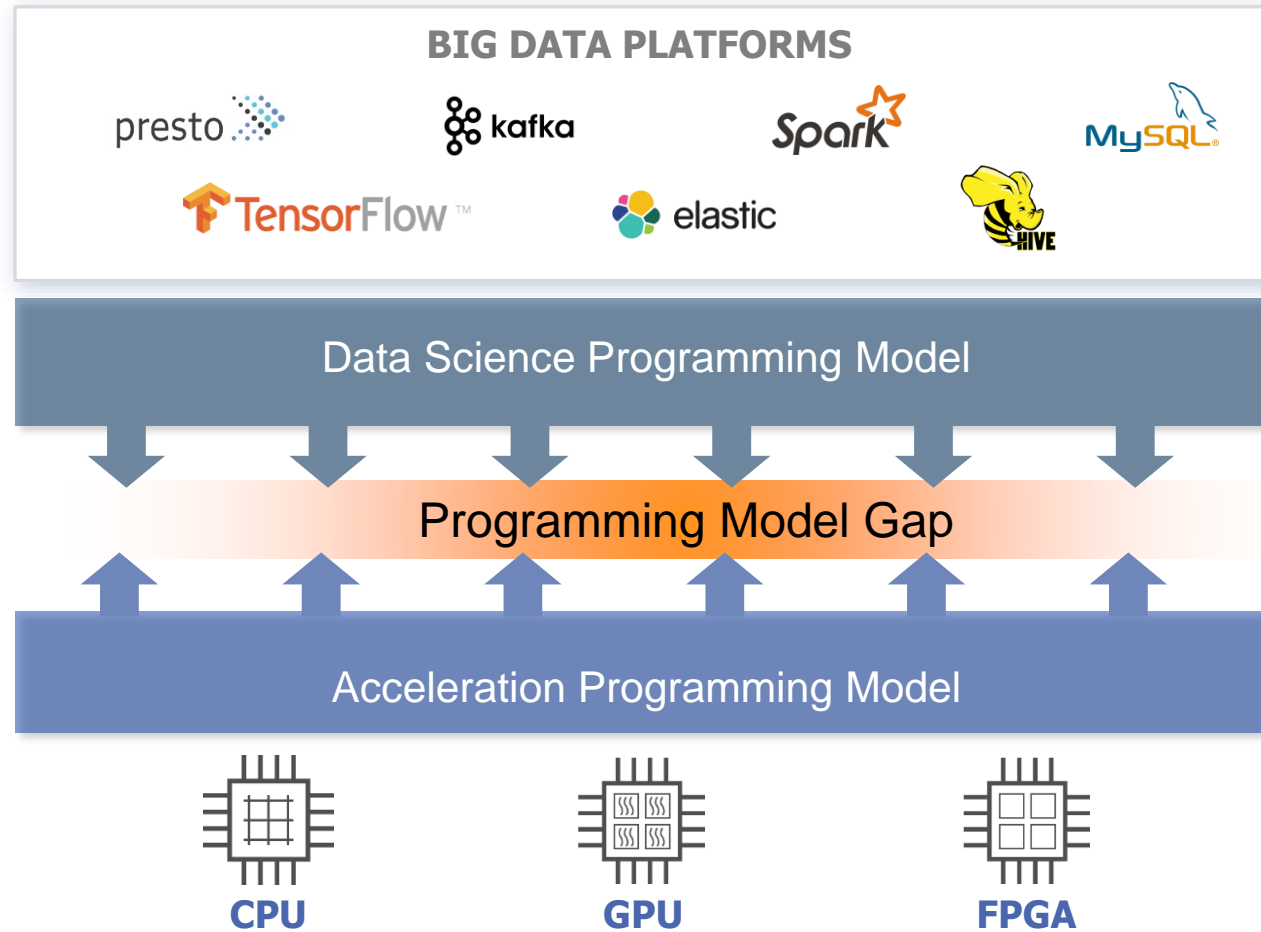


ASIC

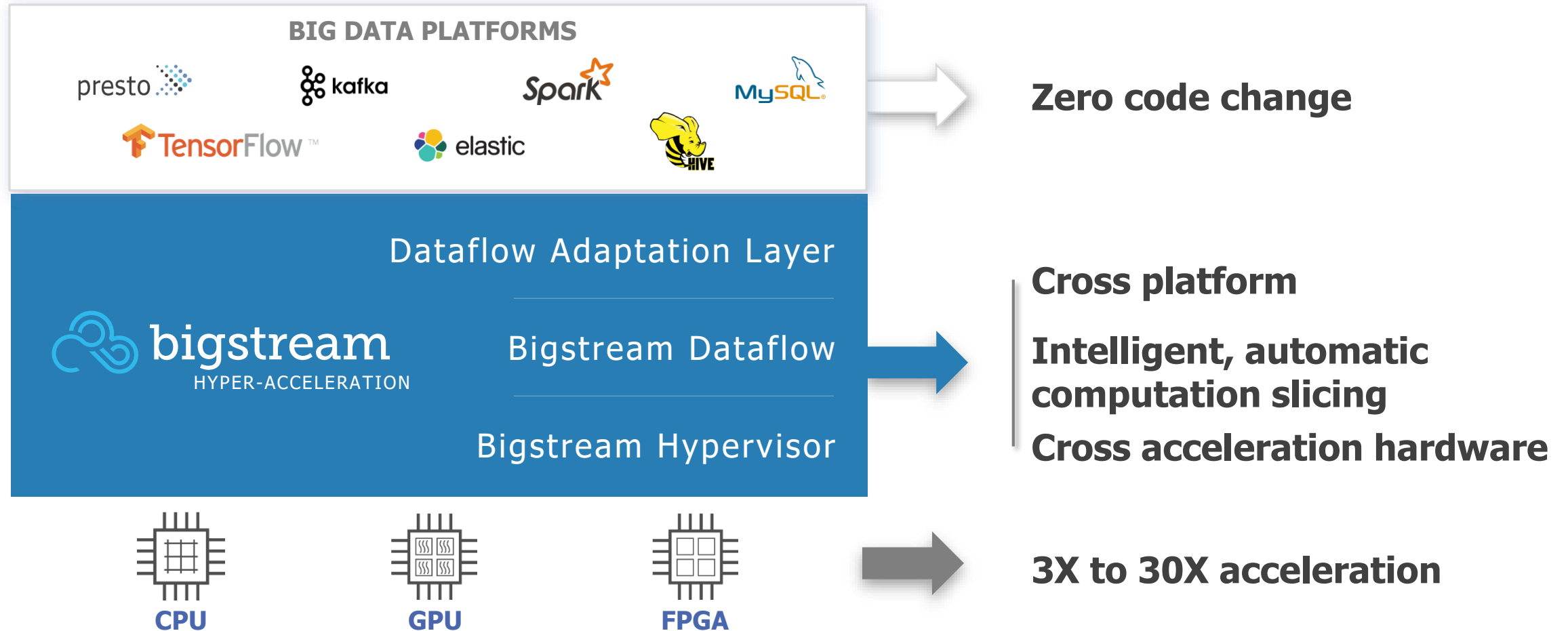


GPU

# Inhibitor: Programming Model Gap for Hardware Accelerators

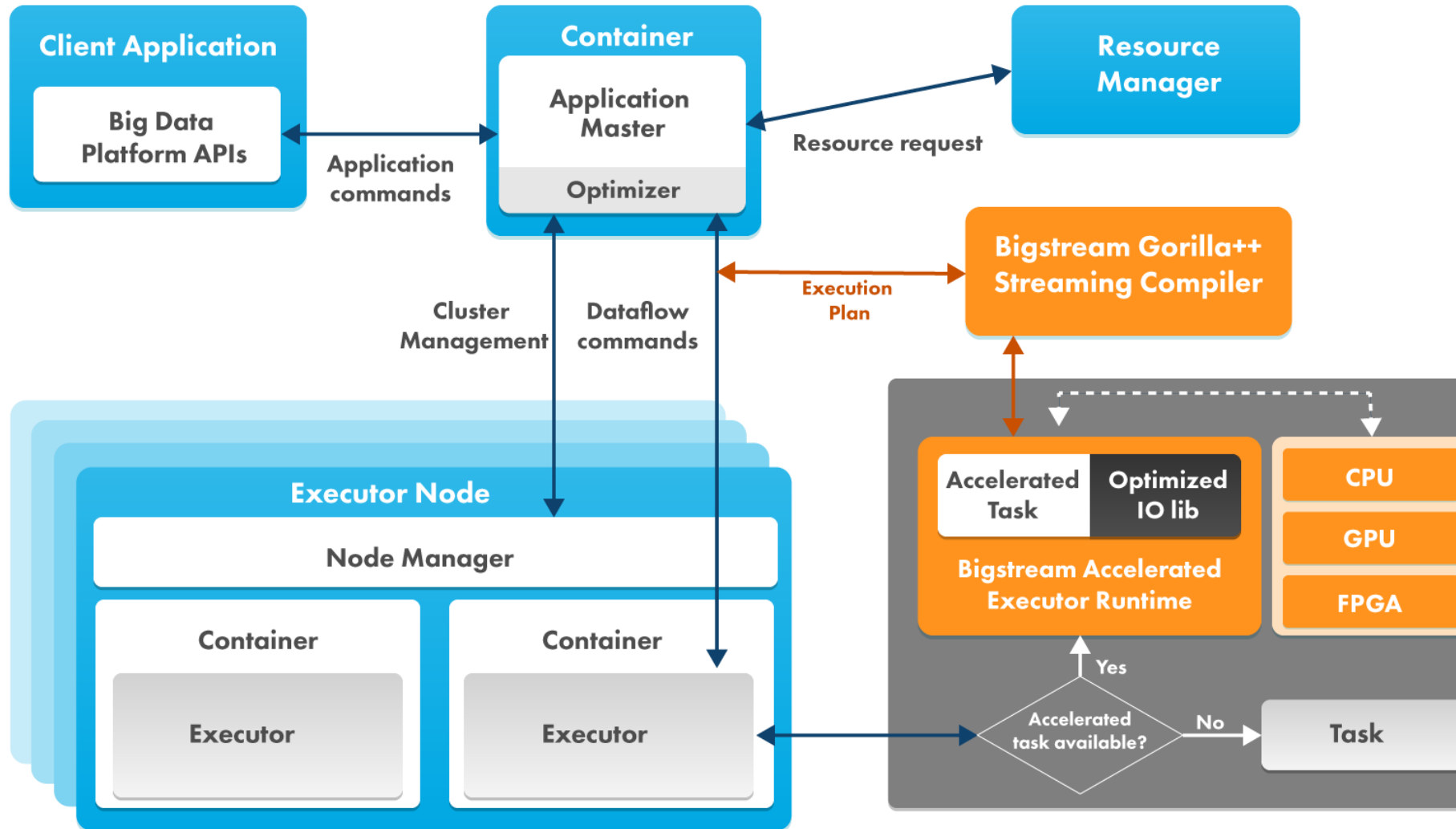


# Introducing: Bigstream Hyper-acceleration Layer

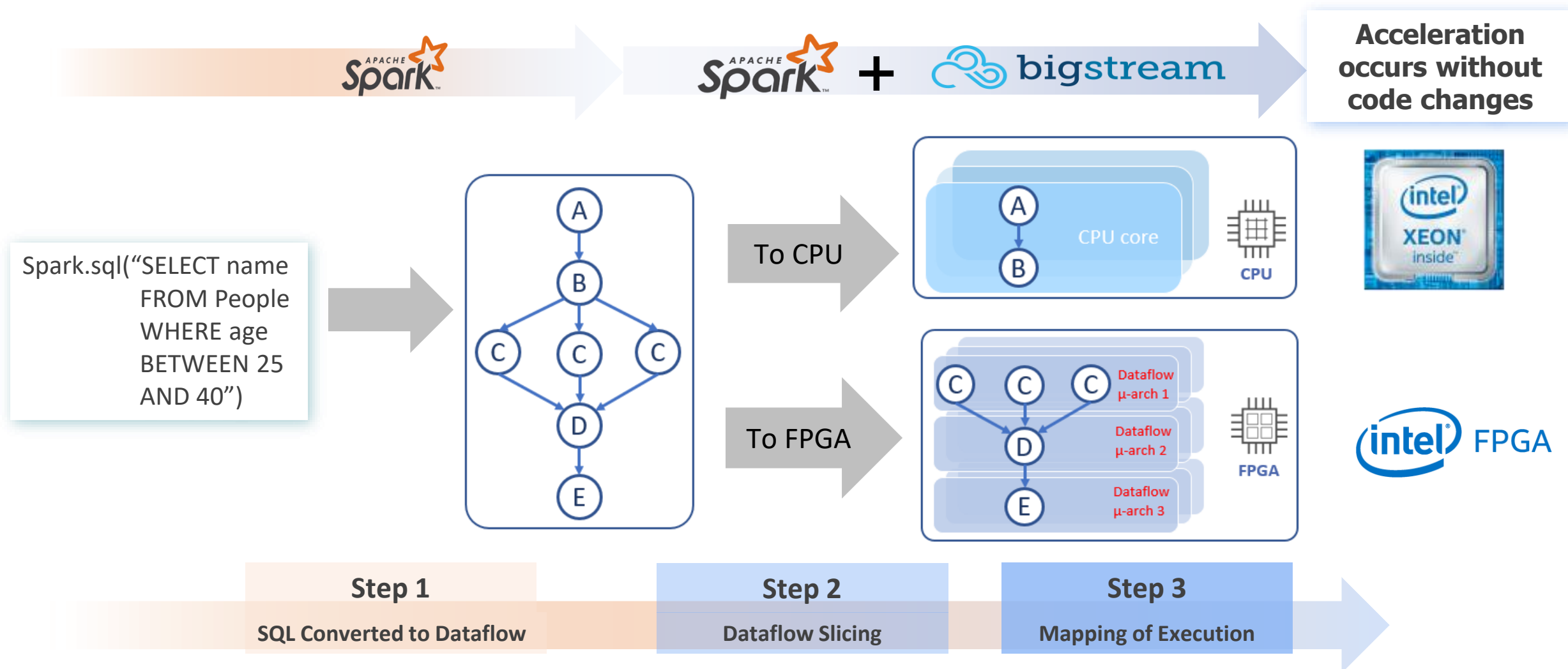


Not STAC Benchmarks

# Big Data Platform Architecture with Bigstream

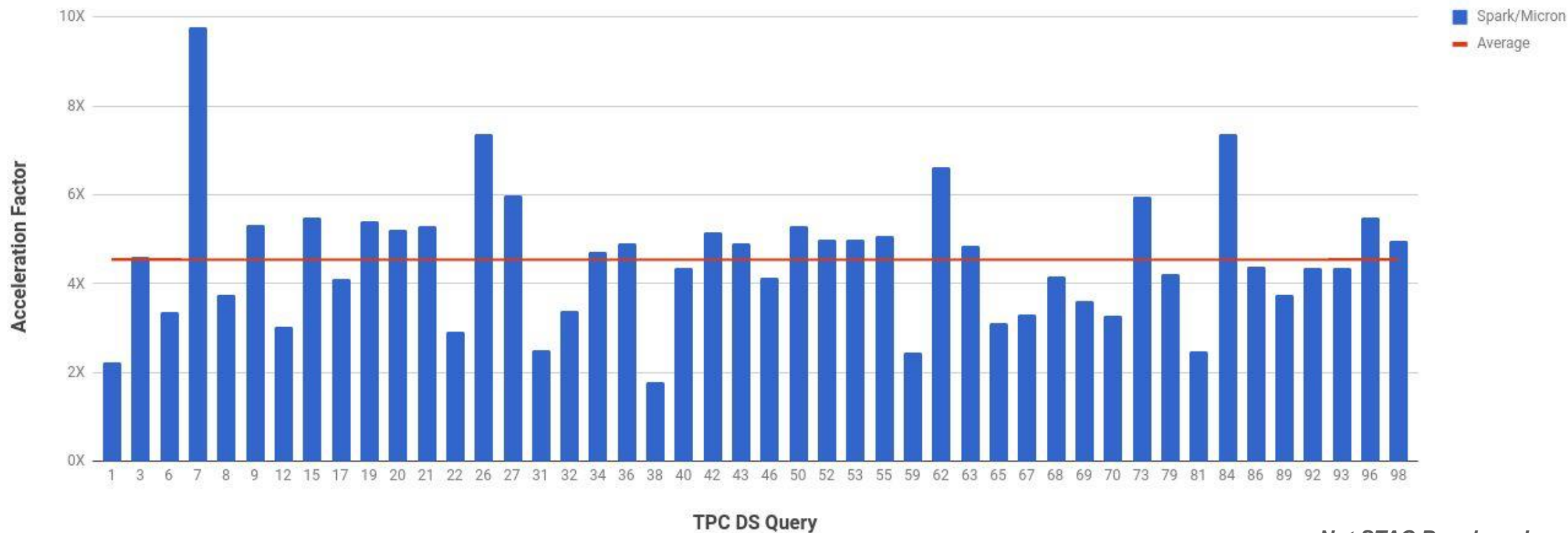


# Bigstream Hyper-acceleration with Intel Multi-core CPUs and FPGAs



# Bigstream + Micron AC-520: 4.5X Faster than Standard Spark

## Spark vs Spark/Micron AC520



*Not STAC Benchmarks*

Benchmark System: Xeon 6 core server, Apache Spark 2.1, results with and without Micron AC-520





Thank You,  
Please tick the Bigstream box



# About Bigstream

Established: July 2015

First Product Shipped: March 2017

Headquarters: Mountain View, CA

Based on patented technology

Core Competencies:

- Advanced compiler technology
- FPGA/GPU programming and strategy
- Apache Spark and Big Data platforms

Key Insight:

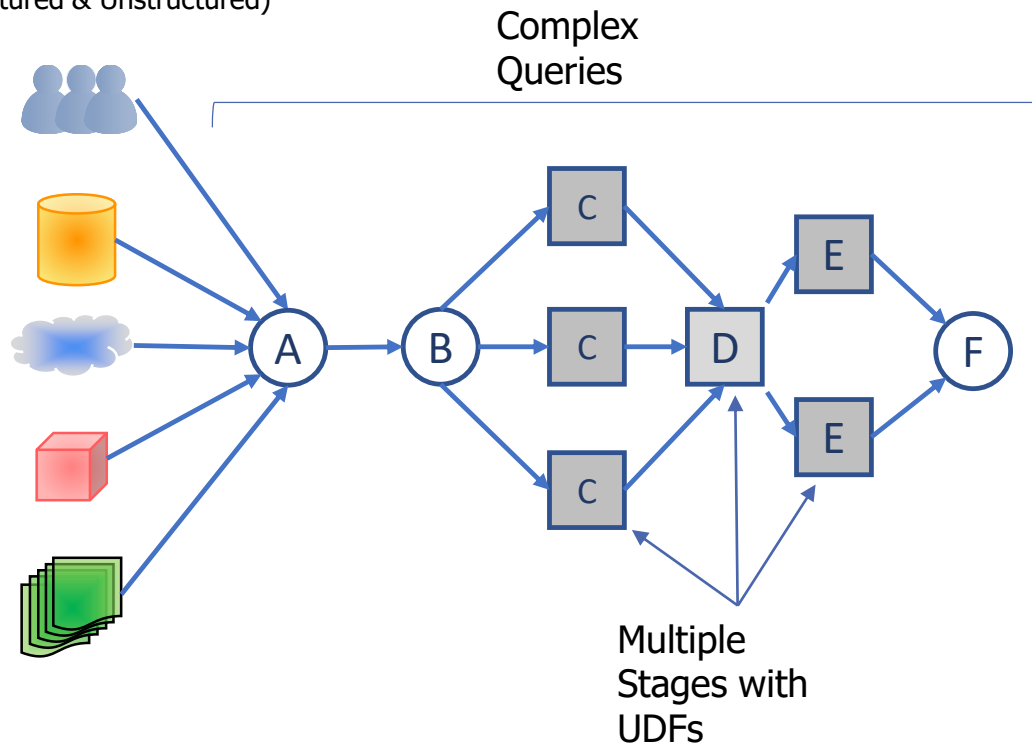
“Any big data/machine learning workload can be hyper-accelerated”

– M Lavasani, CEO

# Customer Case: Hedge Fund Pricing Models

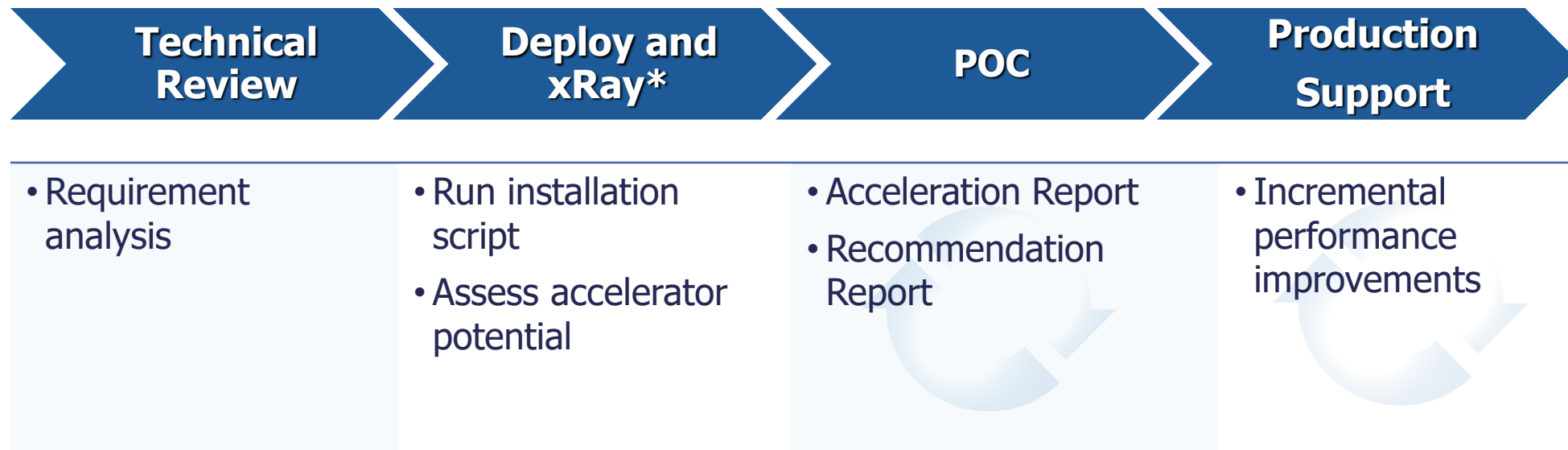
## Challenge: Iterate faster on investment pricing models

Diverse Data Sources  
(Structured & Unstructured)



- Runtime Environment
  - 500+ node cluster
  - Amazon EMR
  - R3.2XLarge instances
  - Spark 2.X
- Business Goals
  - Intra-day price modeling
  - 3+ iterations/day

# Customer Engagement



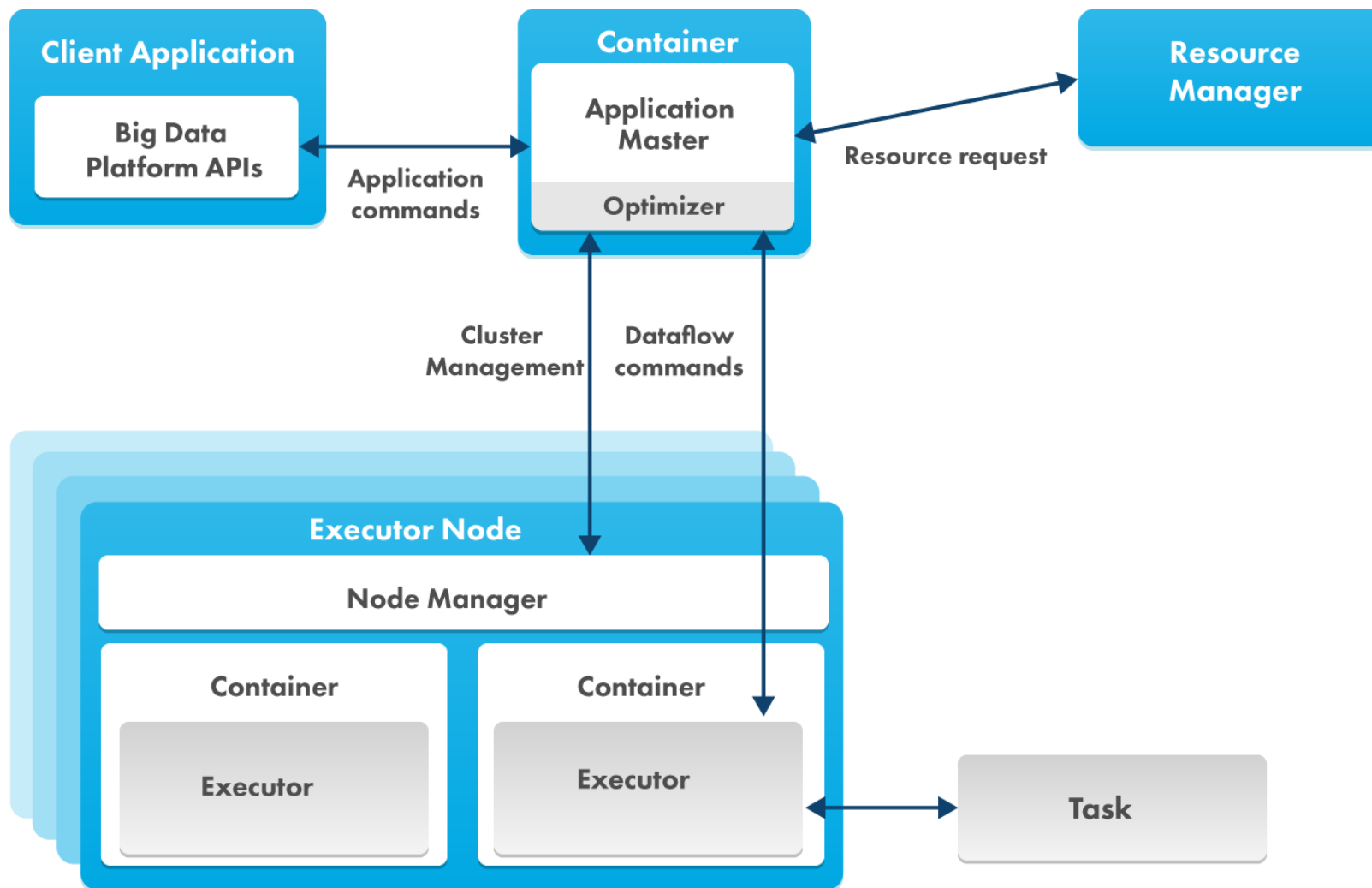
\* xRay is a performance profiling tool built on the Bigstream Hyper-acceleration platform

# The End of Moore's Law

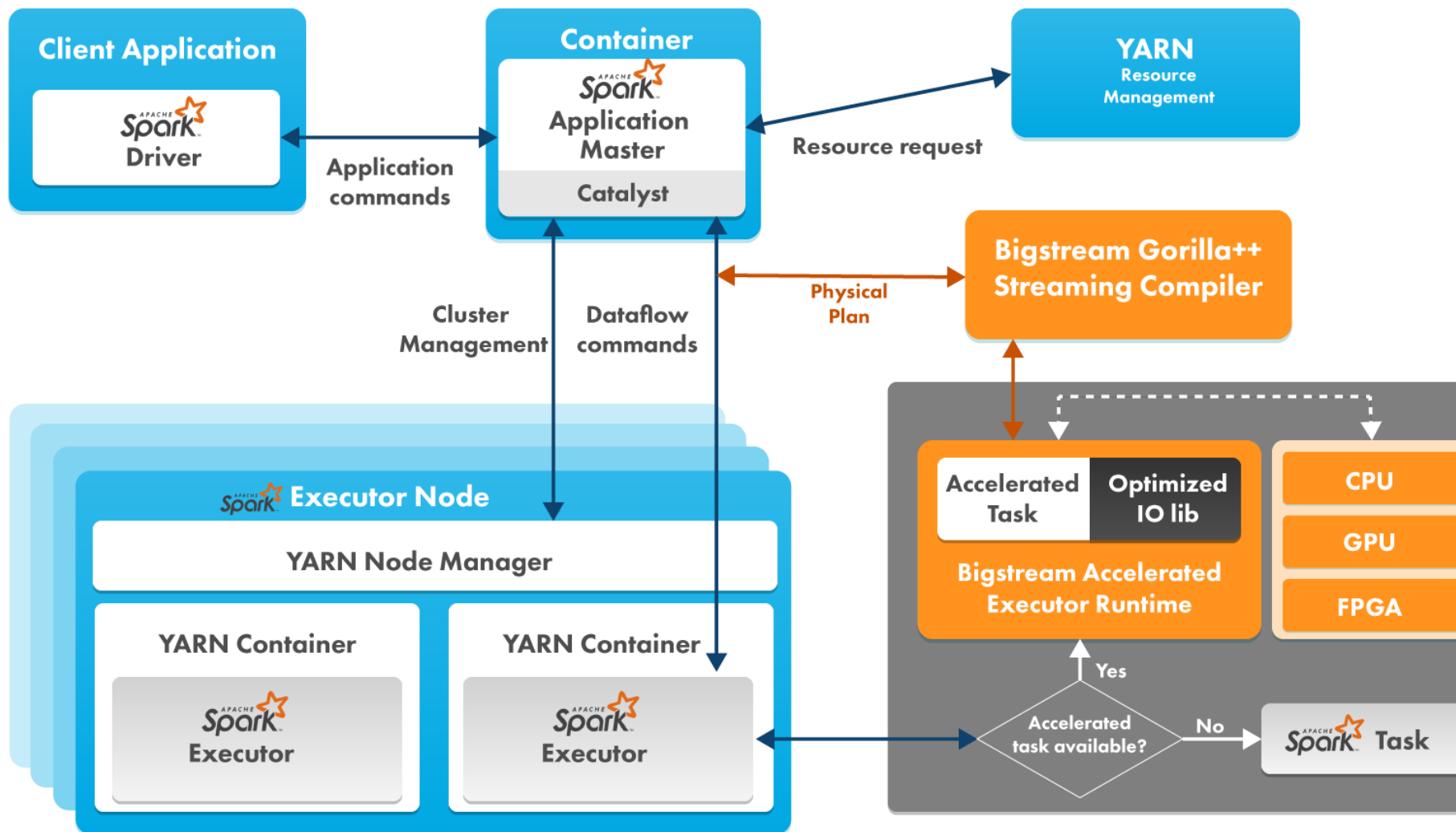
---



# Big Data Platform Architecture

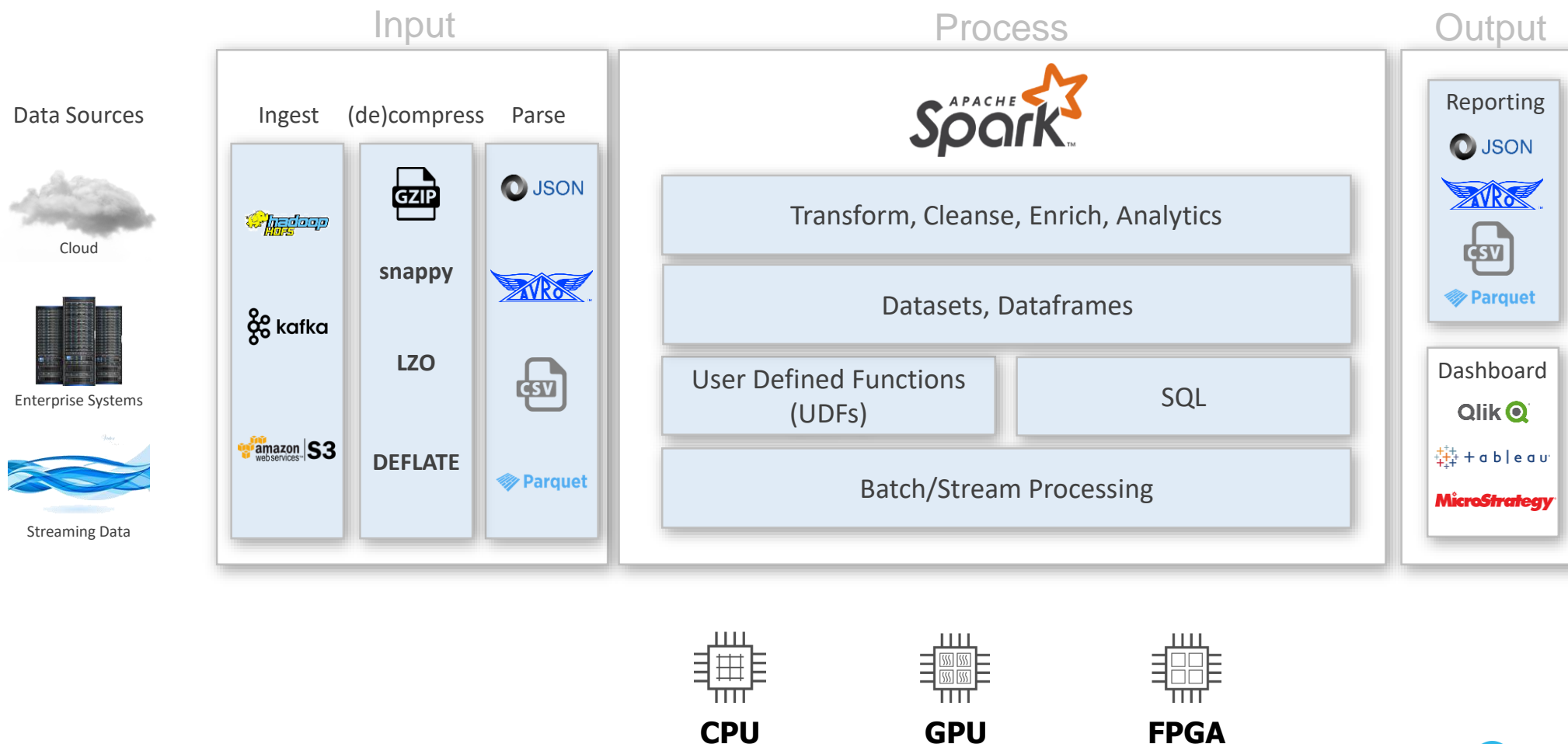


# Bigstream Spark Architecture



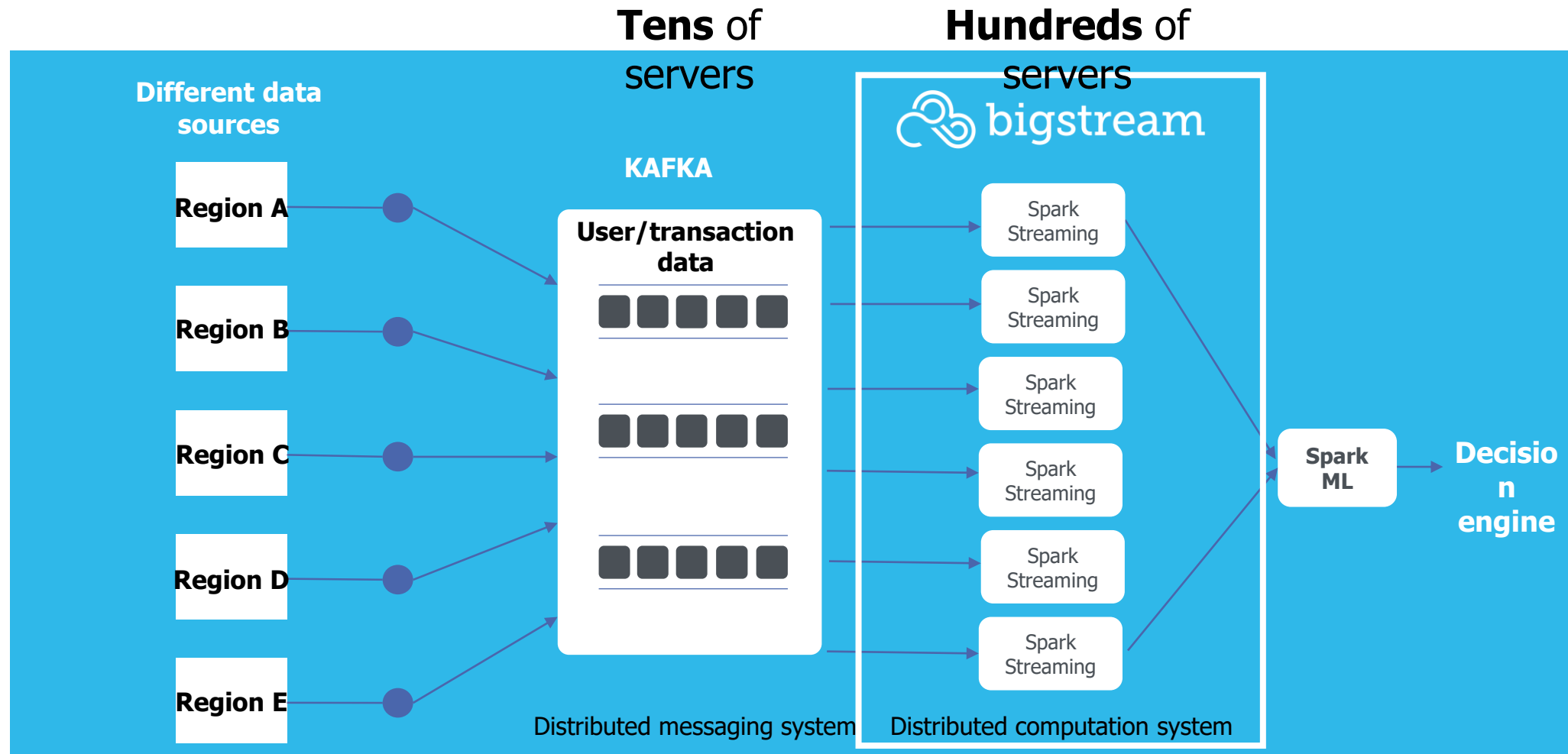
# Spark + Bigstream Hyper-acceleration

15





# Sentiment Analysis Pipeline



# Acceleration Value Propositions



## **Faster Intelligence, Time To Insight**

- Improved latencies for algorithmic trading
- Real-time risk assessment at big data scale



## **Maximize Infrastructure ROI**

- Free up computing resources
- Reduce project backlog



## **Quant/Data Science Productivity**

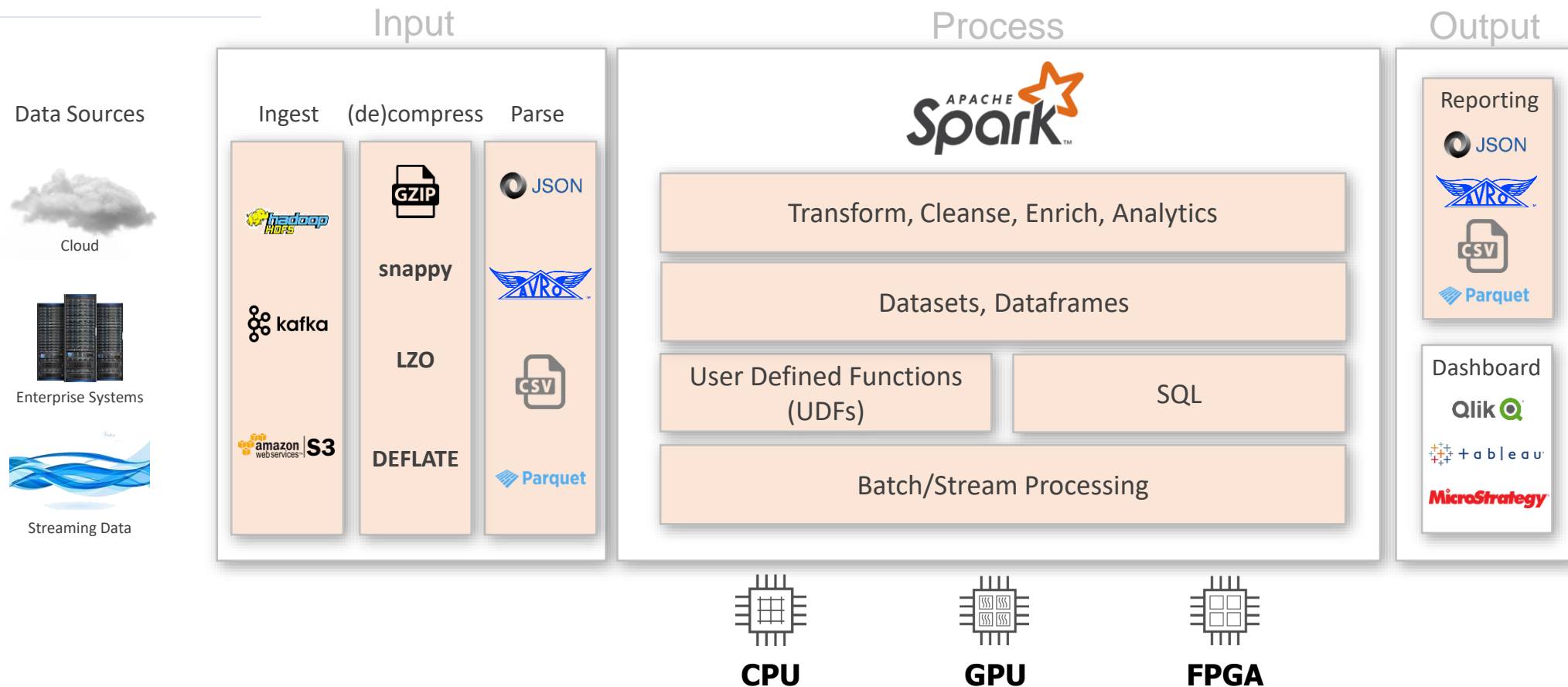
- Iterate faster on simulations and models
- Include fast data (eg tick data) in models



## **Simplify Scalability**

- Avoid forklift upgrades
- Minimize floor space, power, HVAC

# Spark + Bigstream Hyper-acceleration



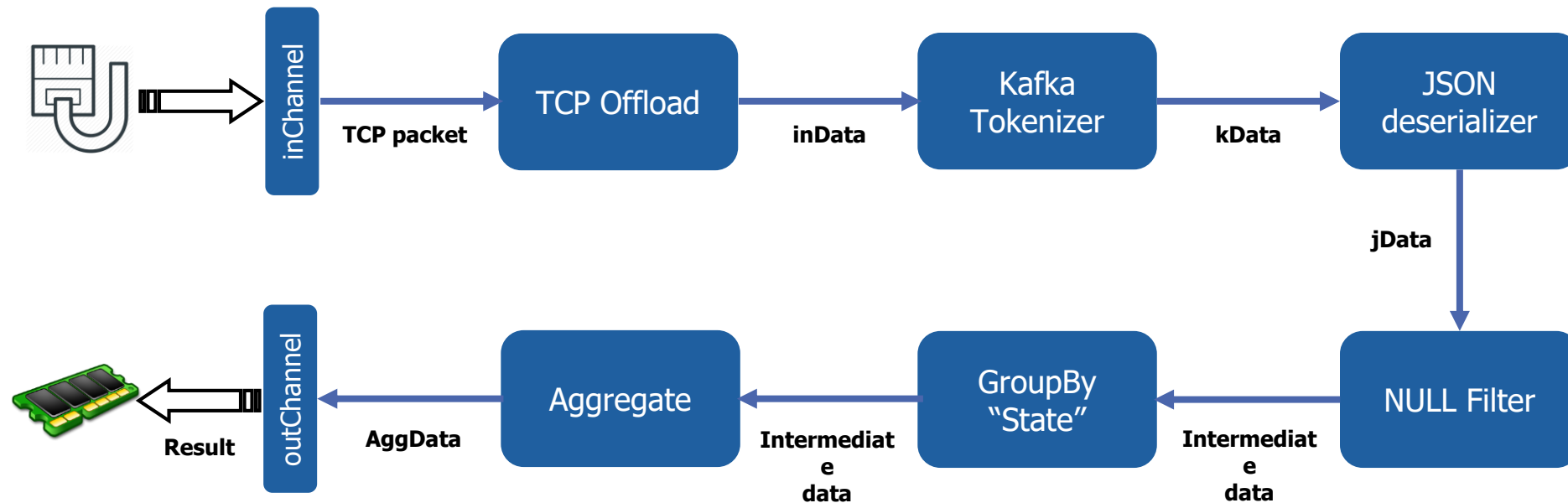
## -Compare to “point solutions”:

- New languages/APIs
- Single hardware/software platform

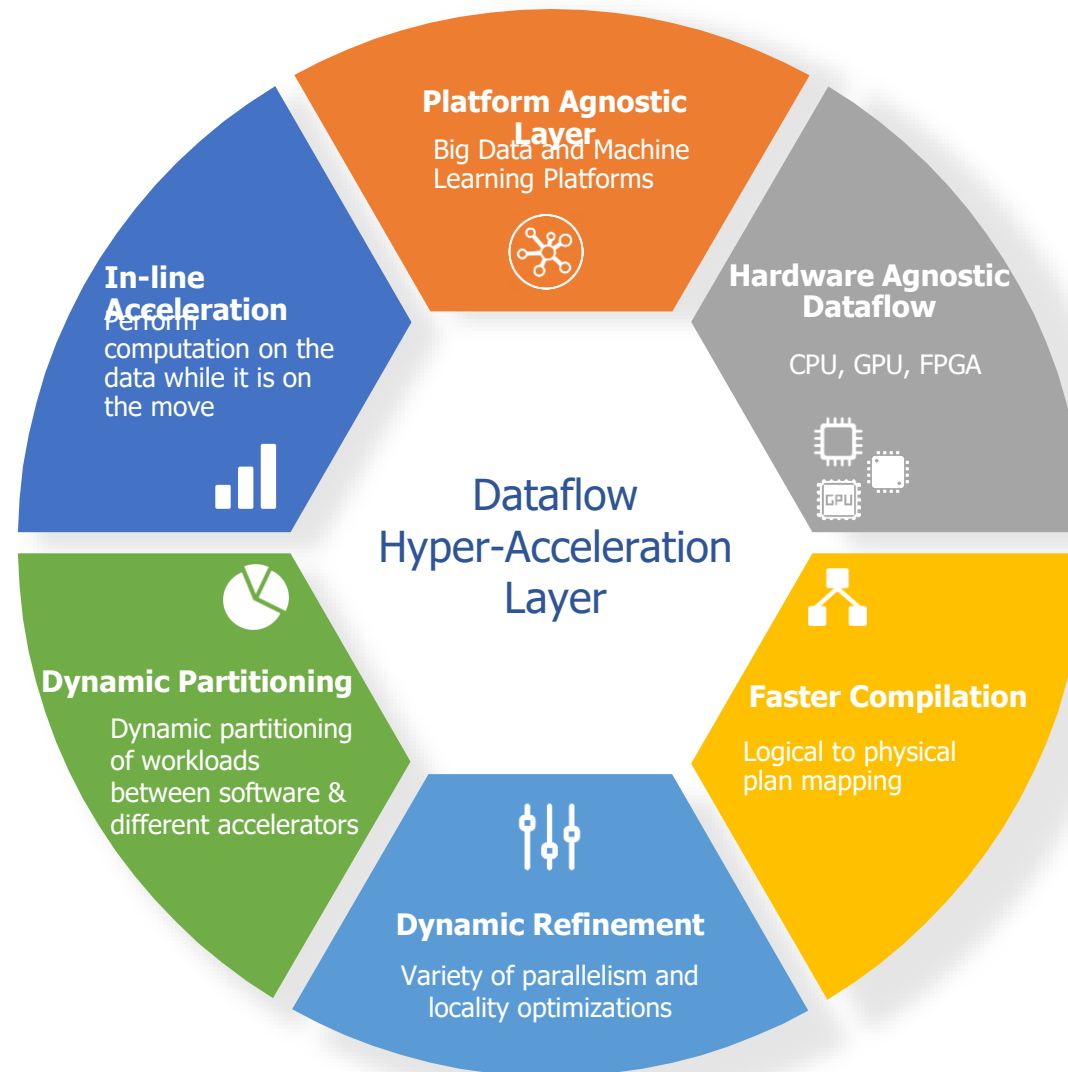
- Query plan-only optimizations
- I/O-only optimizations

- Hand-optimized UDFs
- Library calls

# Architecture for an User Behavior Pipeline In-line Accelerator



# Bigstream Technology



# Spark Code for an User Behavior Pipeline

---

```
val lines =  
  KafkaUtils.createDirectStream  
    [String, String, StringDecoder, StringDecoder]  
    (ssc, kafkaParams.toMap, topicsSet).map(_._2)  
  
lines.foreachRDD((jsonRDD: RDD[String]) => {  
  val sparkSession =  
    SparkSessionSingleton.getInstance(sparkConf)  
  val df = sparkSession.read.schema(schema).json(jsonRDD)  
  
  df.filter("country is not null and price is not null")  
    .filter(df("country").equalTo(30))  
    .groupBy("country").agg(count("price"), min("price"),  
max("price")).show()})
```

# Gorilla DAG for a User Behavior Pipeline

## DAG Code for ETL Pipeline

```
val context    = new bstContext
val kafkaOff   = context.variable()
val kafkaSize  = context.variable()
val kafkaServerIP = context.variable()
val kafkaServerPort = context.variable()
val inCh       = context.inChannel(bstTCP(CLIENT, kafkaServerIP, kafkaServerPort))
val kData      = inCh.rcv(bstKafka(topic="ETL", partition = 0, offset= kafkaOff, size = kafkaSize))
val jData      = kData.deserialize(bstJSON(schema="clickStreamSchema.xml"))
val aggData    = jData.filter("state != NULL").groupby("state").aggregate(min("price"), max("price"))

case class outSchema(state: Int, min: Float, max: Float)
result = aggData.outChannel(bstMem[outSchema])

val size = 1024
var beginOffset = 0
val endOffset = 65536

do {
  kafkaOff.set(beginOffset)
  kafkaSize.set(size)
  beginOffset += size
  context.run()
  result.foreach {i => println("minPrice:" + i.min + " state:" + i.state)}
} while (beginOffset < endOffset);
```

**Expose simplistic, composable acceleration primitives to developers**



# Gorilla-Generated Code for a User Behavior Pipeline

```
val depack = Engine("depacketizer.c")
val kafka_token = Engine("kafka_tokenizer.c")
val json_deserializer = Engine("json_deserializer.c")
```

Packet processing and de-serialization parts

```
val tkCntrl = Engine("tokenMemController.c")
val data = spMem(height = 64, width = 256)
val membuff = Offload(tkCntrl, data, "data")
```

Token memory controller for passing variable length tokens

```
val flit2mem = Engine("FM.c")
val mem2flit = Engine("MF.c")
```

Two additional engines for storing streaming data into memory and fetch streaming data from memory

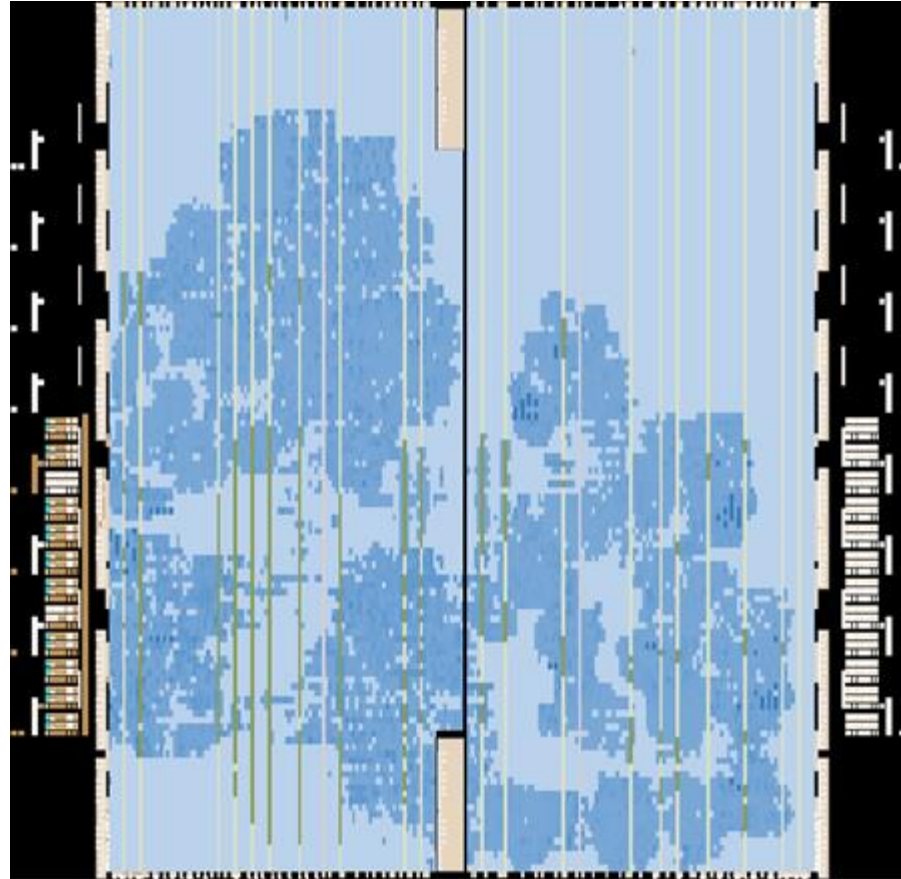
```
val group_by = Engine("group_by_state.c")
val aggEngine = Engine("aggregate.c")
val aggMem = spMem(height=20, width = 192)
val aggregate = Offload(aggEngine, aggMem, "aggMem")
val array_serializer = Engine("array_serializer.c")
```

Data filtering, aggregation, and serialization into an array in program memory

```
val fm_groupby = Chain(flit2mem, group_by)
val fm_groupby_mem = Offload(fm_groupby, membuff, "gBTMem0")
val fm_groupby_agg = Chain(fm_groupby_mem, aggregate)
val fm_groupby_agg_mem = Offload(fm_groupby_agg, membuff, "gBTMem1")
val aggregate_mf = Chain(fm_groupby_agg_mem, mem2flit)
val aggregate_part = Offload(aggregate_mf, membuff, "gBTMem2")
val prepare_data = Chain(depack, kafka_token, json_deserializer)
val result = Chain(prepare_data, aggregate_part, array_serializer)
```


Connecting the above components by "Chain" and "Offload" operations

# User Behavior Pipeline Accelerator after Place & Route



# Announcement – Bigstream on AWS EMR

AMI & SaaS



## Bigstream Hyper-Acceleration Layer

Sold by: [Bigstream](#)

14-DAY FREE TRIAL - Bigstream hyper-acceleration software provides a performance boost of up to 5x for your Spark SQL applications without changing a single line of code. Using native compilation, vectorization, locality optimization and custom data connectors, Bigstream provides faster time-to-insight and cost avoidance of up to 50% of your EMR infrastructure. The installation of our software is incredibly easy. After you have registered your Bigstream service we will send you a link that you can paste into the bootstrap field of your EMR cluster provisioning screen. And that's it!... [Read more](#)

**Customer Rating** ★★★★★ [\(0 Customer Reviews\)](#)

**Delivery Method** Software as a Service (SaaS) Subscriptions ([Read more](#))

**Support** [See details below](#)

**Highlights**

- Seamless hyper-acceleration: automatic performance gains without changing a single line of code
- Seamless installation: subscribe to the Bigstream service once, and anytime you want to accelerate your EMR cluster, simply add the Bigstream bootstrap URL to turn on hyper-acceleration
- If after the Free Trial Period, you want to continue using Bigstream, you are all set. Your usage fees will be added to your monthly AWS bill.

### Product Description

14-DAY FREE TRIAL - Bigstream hyper-acceleration software provides a performance boost of up to 5x for your Spark SQL applications without changing a single line of code. Using native compilation, vectorization, locality optimization and custom data connectors, Bigstream provides faster time-to-insight and cost avoidance of up to 50% of your EMR infrastructure.

The installation of our software is incredibly easy. After you have registered your Bigstream service we will send you a link that you can paste into the bootstrap field of your EMR cluster provisioning screen. And that's it! You can now run your job with state of the art hyper-acceleration.

Continue

You will have an opportunity to review your order before subscribing or being charged.

### Pricing Details

**Software Fees**

UNITS	COST
All small, medium, large, and xlarge instances	<b>\$0.100 / host / hour</b>
all 2xlarge instances	<b>\$0.300 / host / hour</b>
all 4xlarge instances	<b>\$0.600 / host / hour</b>
all 8xlarge and 10xlarge instances	<b>\$1.200 / host / hour</b>
all 16xlarge instances	<b>\$2.400 / host / hour</b>
all 32xlarge instances	<b>\$4.800 / host / hour</b>

**Note:** This software is priced along a consumption dimension. Your bill will be determined by the number of hosts you use per hour.

### Recent Product Reviews

Create Your Own Review

# Bigstream ON EMR

Add the Bigstream bootstrap URL  
and your cluster has hyper-acceleration

AWS Services Edit

## Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps  
Step 2: Hardware  
**Step 3: General Cluster Settings**  
Step 4: Security

### General Options

Cluster name

☒ Logging ⓘ  
S3 folder

☒ Debugging ⓘ  
☒ Termination protection ⓘ

### Tags ⓘ

Key	Value (optional)
<input type="text" value="Add a key to create a tag"/>	<input type="text"/>

### Additional Options

☐ EMRFS consistent view ⓘ

▼ Bootstrap Actions

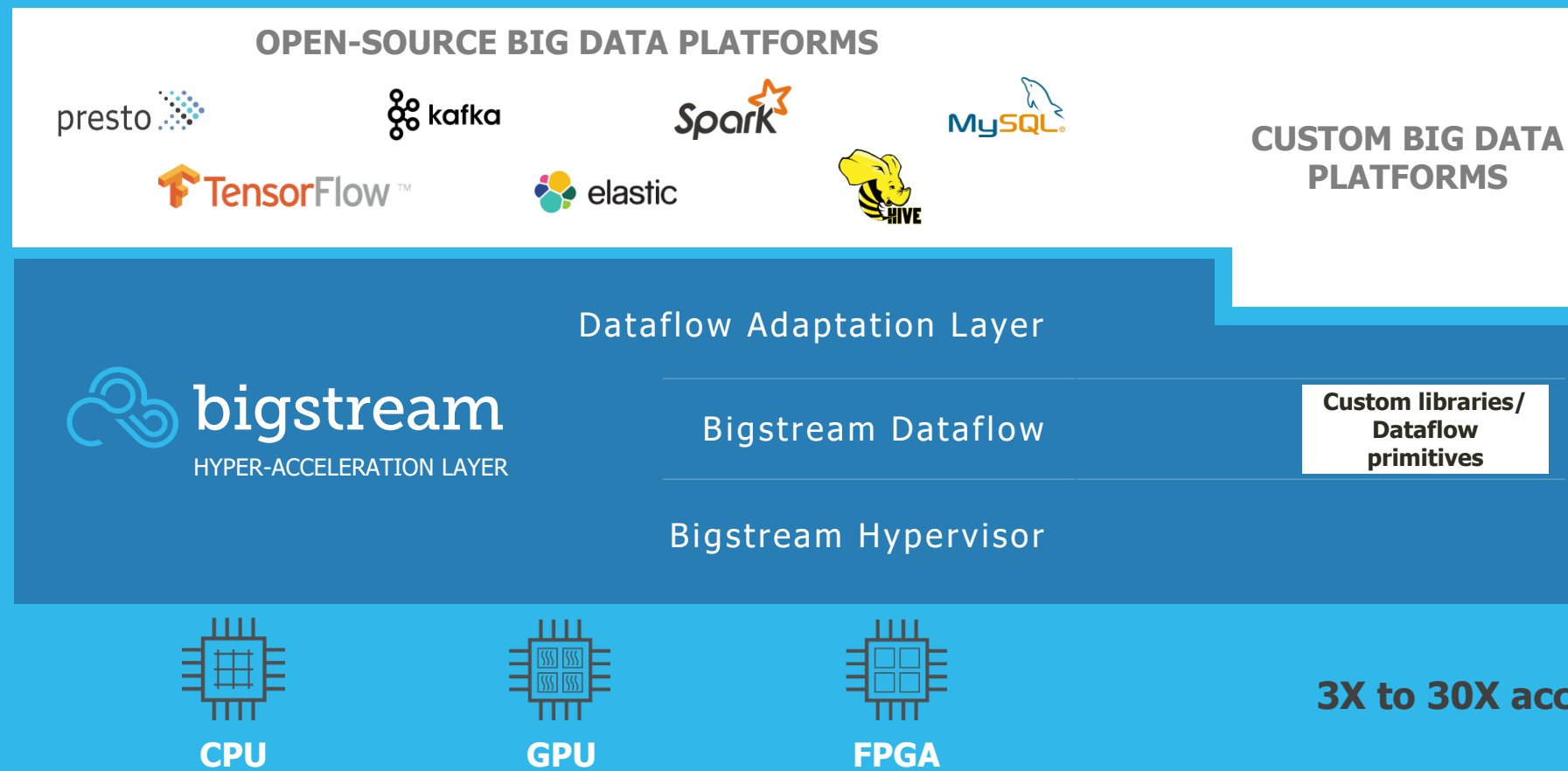
Bootstrap actions are scripts that are executed during setup before Hadoop starts on every cluster node. You can use them to install additional software and customize your applications. [Learn more](#)

Bootstrap action type	Name	JAR location	Optional arguments	
Custom action	Custom action	s3://bigstream-emr/bootstrap.sh		

Add bootstrap action

**Setting the bootstrap script**

# Future of Big Data: Hyper-Acceleration AP



Automatic computation routing