bigstream

HYPER-ACCELERATION WITH BIGSTREAM TECHNOLOGY

Introduction

Those familiar with Moore's law now have the expectation that available compute power is an ever-expanding resource, ever diminishing in cost. Developers, BI/Analytics teams and IT operations naturally wish to use all available compute power to their advantage. Yet there is a growing number and variety of computing workloads for which compute capacity and application performance is increasingly critical.

Emerging big data technologies such as real-time and predictive analytics, machine learning, deep learning, natural language processing and artificial intelligence are becoming indispensable to tech-forward industries like digital media, healthcare, financial services, telecommunications and more. These new big data workloads have inspired a new generation of tools such as Apache Hive, Apache Spark, and TensorFlow that are pushing advanced analytics into the mainstream.

But achieving optimum performance on large clustered systems - even with the use of hardware acceleration such as GPUs, or FPGAs - has had limited success due to higher complexity and a lack of portability. These conditions lead to higher costs due to specialized support and skills

requirements, additional programming time, increased dev and test complexity, and the risk of operational instability. See Figure 1.

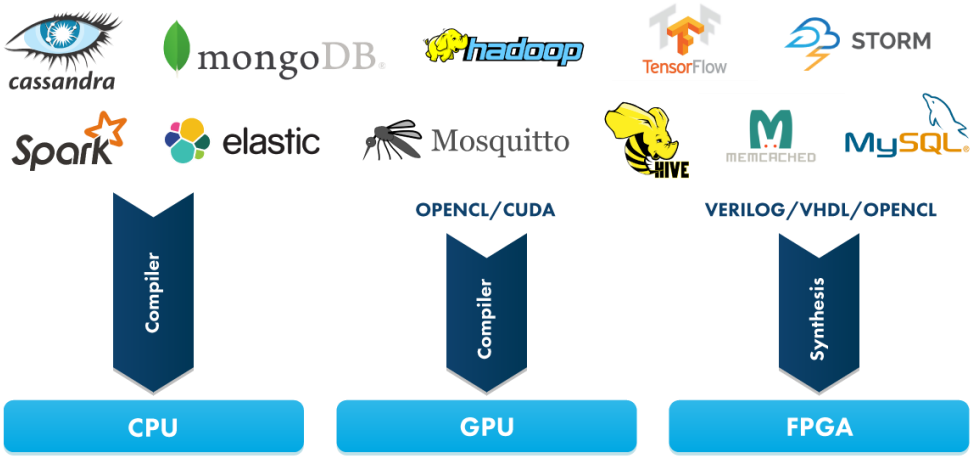


FIGURE 1. COMPLEXITY AND LACK OF PORTABILITY LIMIT ADOPTION OF HARDWARE ACCELERATION

Bigstream Solutions has pioneered an advanced data-flow computational architecture for the new generation of big data workloads. The ***Bigstream Hyper-acceleration Layer™*** *(HaL)* enables scaling of the analytics engines and accelerates time-to-insight for businesses. Besides orders of magnitude improvements in performance and scalability, the Bigstream HaL architectural model results in greater resiliency and predictability of these new applications. Most critically, it does this without the need to change or add a single line of code. Bigstream HaL can accelerate a variety of execution engines, with Apache Spark being the first supported platform. Others will follow.

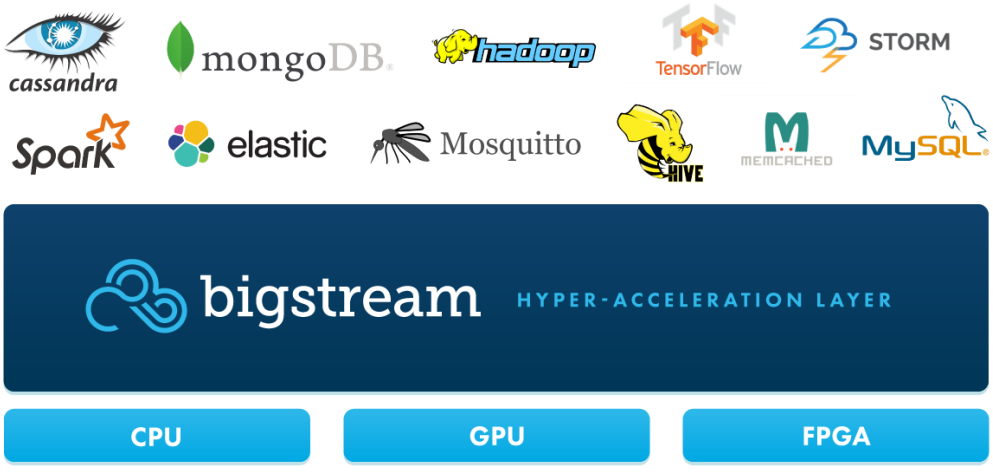


FIGURE 2. BIGSTREAM HYPER-ACCELERATION LAYER REDUCES COMPLEXITY AND REQUIRES NO CODE CHANGES

The high-level Bigstream HaL architecture illustrating the difference from current solutions is shown in Figure 2. Bigstream HaL address the two issues with current acceleration because it is transparent from a user perspective, and also agnostic from a hardware adaptation perspective. That is, it presents the user with an unchanged programming/scripting interface, while automatically adapting to the underlying hardware to extract high performance.

Bigstream hyper-acceleration is achieved by translating the dataflow representation of a computation which is specific to a platform such as Spark into an optimized, platform independent dataflow. Bigstream HaL is composed of two main approaches to Hyper-acceleration:

Bigstream software acceleration performs runtime generation of C++ code from the dataflow representation of the computation, which is then natively compiled. Performance testing shows *2x to 5x* performance improvement over native Spark in well-known benchmarks

Bigstream hardware acceleration provides a runtime adaptation layer and a portable runtime system. It generates FPGA-based accelerators from the dataflow representation of the computation, yielding very high performance. Performance testing shows from *10x to 30x* performance improvement in benchmarks.

Both are appropriate for cloud and on-premise deployment, and are transparent to Spark end users. Note that big data technologies in addition to Spark are listed in the figure as they are all being considered for the Bigstream product roadmap.

Use Cases and Results

We have developed and performance benchmarked this technology. The initial tech target is Spark/SQL and Dataframes, which are widely used in the data analytics community. In this section, we describe the evaluation setup and results. The benchmarks use techniques that apply to a wide set of currently relevant analytics areas. Bigstream HaL displays best-in-breed performance, while maintaining the simplicity of existing Spark and Spark/SQL interfaces. The latter aspect suggests it can readily be deployed in a production environment.

Kafka Use Case

Use of Kafka data streaming in conjunction with Spark Streaming is a very important use case for a number of big data application areas. Areas include online or near-real-time applications such as at-watch network security, digital media bidding/auctioning, financial decision systems and many more. The basic setup of such a data processing system is shown below.

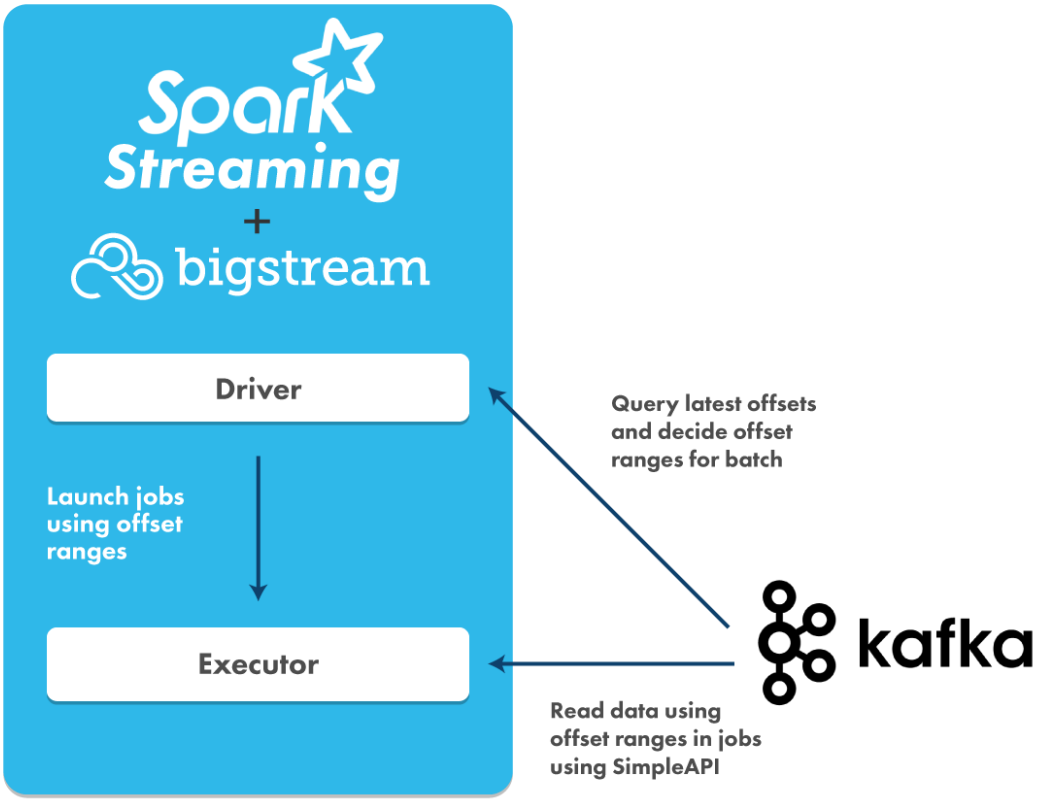


FIGURE 3. BIGSTREAM APPLIED TO A SPARK STREAMING + KAFKA APPLICATION ARCHITECTURE

Figure 3 is essentially a schematic of our Spark Streaming/Kafka test setup, where a set of Kafka servers provides streaming data to Spark for processing. For efficient data consumption, the key parameter is the *request frequency* with which Spark requests data from the Kafka server. Data is requested and delivered to Spark in chunks called *micro-batches*. Micro-batches are then processed by executors as jobs comprised of RDD operations.

For real-time or online operation, it is important that the request frequency be less than the processing time for each micro-batch to prevent data backlog. Of course, if this relationship is violated, the backlog will be constantly-expanding and unbounded, an untenable situation.

We have written a prototype Spark Streaming application for the setup above to compare standard vs. Bigstream HaL accelerated Spark. We emphasize that all three measurements were performed with no change to the user-level processing code, and were run using the same Kafka broker configuration. In these tests, micro-batches of JSON data were almost 1GB each in size.

Spark Performance Gains Using Bigstream HaL		
RUNTIME	PROCESSING TIME /MICROBATCH	SPEEDUP (OVER SPARK)
Spark (no acceleration)	20 seconds	-
Bigstream software acceleration	8 seconds	3x
Bigstream FPGA acceleration	2 seconds	13x

TABLE 1: BIGSTREAM HYPER-ACCELERATION RESULTS OVER SPARK

Table 1 shows the results for micro-batch processing times. Bigstream HaL shows a 3x and 13x speedup for software and hardware acceleration, respectively. This implies a potential for 3x and 10x improvement for online processing throughput. We proffer that this level of performance improvement will widen the scope of problem that can be solved using a given configuration. Viewed another way, it enables cost savings by producing equal throughput for a smaller configuration.

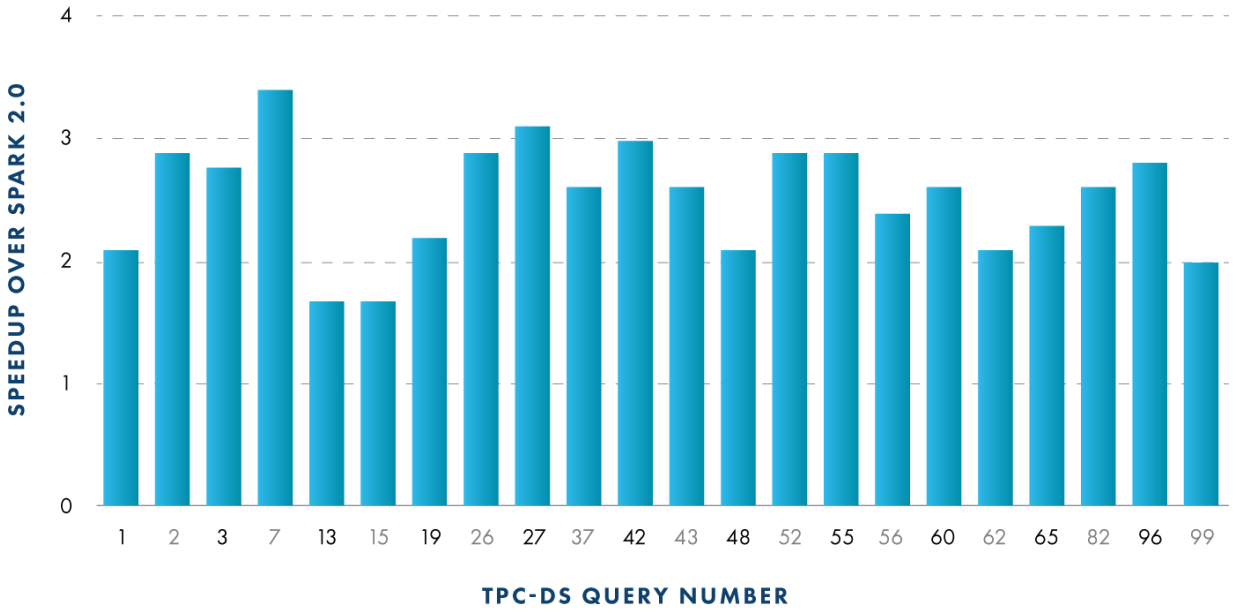
Transaction Processing Performance Council – Decision Support (TPC-DS) Benchmarks

The TPC-DS benchmarks (http://www.tpc.org/tpcds/) are probably the best known for testing performance of decision system support solutions, e.g. big data systems. They are a set of SQL queries, along with associated test data generation tools, designed to evaluate performance of big data systems that support SQL.

For our purposes, we have refactored these benchmarks in Spark/SQL for which Bigstream HaL has support. The tests were run on the familiar Amazon Web Services (AWS) Elastic Map Reduce (EMR) platform, with EMR clusters being created using the standard interface. Installing Bigstream HaL on EMR is a matter of configuring a single bootstrap action in the EMR interface, a single-click operation. This makes running and comparing Bigstream HaL to standard Spark on EMR very easy to do.

The graph below shows the speedup Bigstream software acceleration provides over EMRs Spark for a selected set of TPC-DS queries. The dataset used is the standard TPC-DS dataset with a scaling factor of 100, and has a size of 100GB. The benchmarks are composed of standard, common SQL operations such as SELECT, GROUP BY, ORDER BY, LIMIT, WHERE (i.e. filters), and implicit JOIN operations via the use of multi-table SELECT,WHERE clauses. Please see the TPC website for more details on the benchmarks.

FIGURE 4. RESULTS FOR THE TPC-DS BENCHMARKS WITH BIGSTREAM SOFTWARE ACCELERATION



The results in Figure 4 show that Bigstream software acceleration provides an average of 3x speedup for the 22 benchmarks tested. This performance improvement can be leveraged in either of two ways: 1) reducing the size of the cluster for a given analysis (hence, reducing cost), 2) increasing the compute power of a given cluster, reducing the time needed for analyses. Importantly, speedup was achieved on EMR cloud by a one-step operation with zero changes to the benchmark queries.

Amazon is planning on offering "F" (i.e. FPGA-based) instances in the near future, and we are in the process of testing Bigstream hardware acceleration on these instances. Our expectation is that Bigstream hardware acceleration when benchmarked, will display an order of magnitude speedup similar to the streaming performance presented in Section 2.1.

Technical Overview

This section presents a technical overview of Bigstream HaL and the role of its components. We focus on its relationship to the standard Spark architecture and how it enables acceleration transparently.

Baseline Spark Architecture

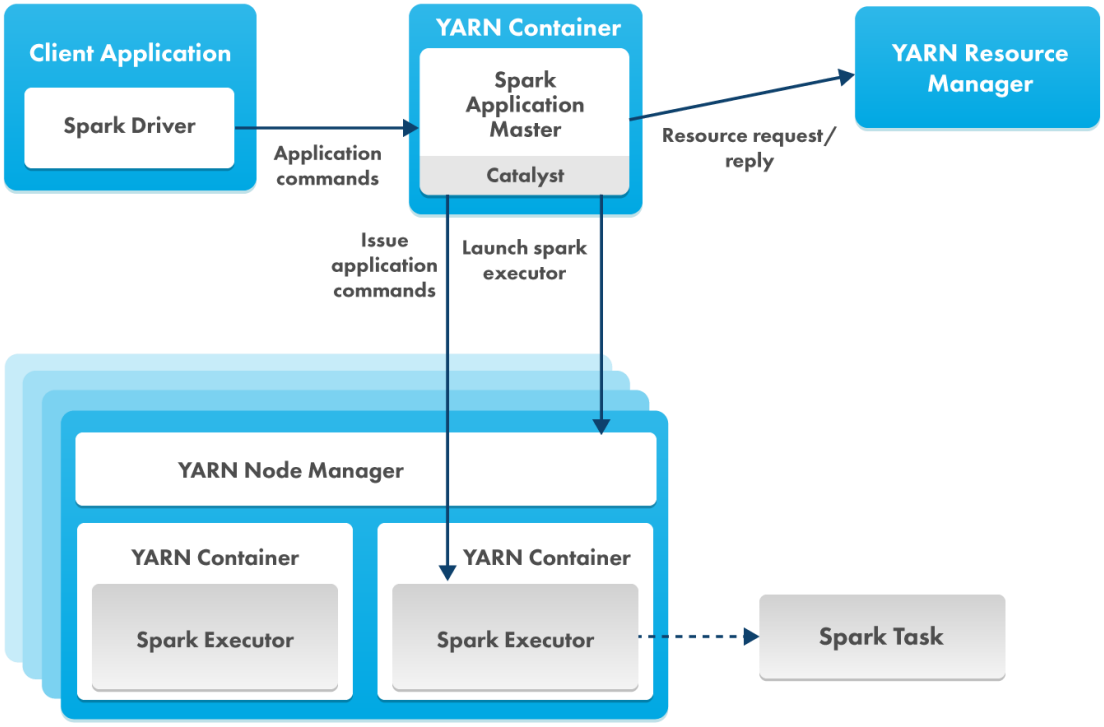


FIGURE 5: BASELINE SPARK ARCHITECTURE

Figure 5 shows the basic components of standard Spark when using YARN for resource management. We do not discuss YARN except to say that it is responsible for allocating resources. The Spark components and associated roles are as follows:

- **Spark Driver -** Runs the client application and communicates with the Master to install the application to be run and configurations for the cluster. The configurations include number of Masters and Core nodes as well as memory size choices for these.

- **Spark Master -** Instantiates the Spark Executors, also known as the Core nodes. The Master must communicate with the YARN Resource Manager with requests for resources as per the application needs. The YARN system, in turn, allocates resources for Executor creation. The Master creates the stages of the application and distributes tasks to the Executors.

- **Spark Executor -** Runs individual Spark tasks, reporting back to the Master when stages are completed.
  The computation proceeds in stages, generating parallelism among the Executor nodes. It's clear that the faster that the Executors can execute their individual task sets, the faster stages can finish, and therefore the faster the application finishes. In standard Spark, tasks are created as Java bytecode at runtime and downloaded to the Executors for execution.

Bigstream Hyper-acceleration Layer Architecture

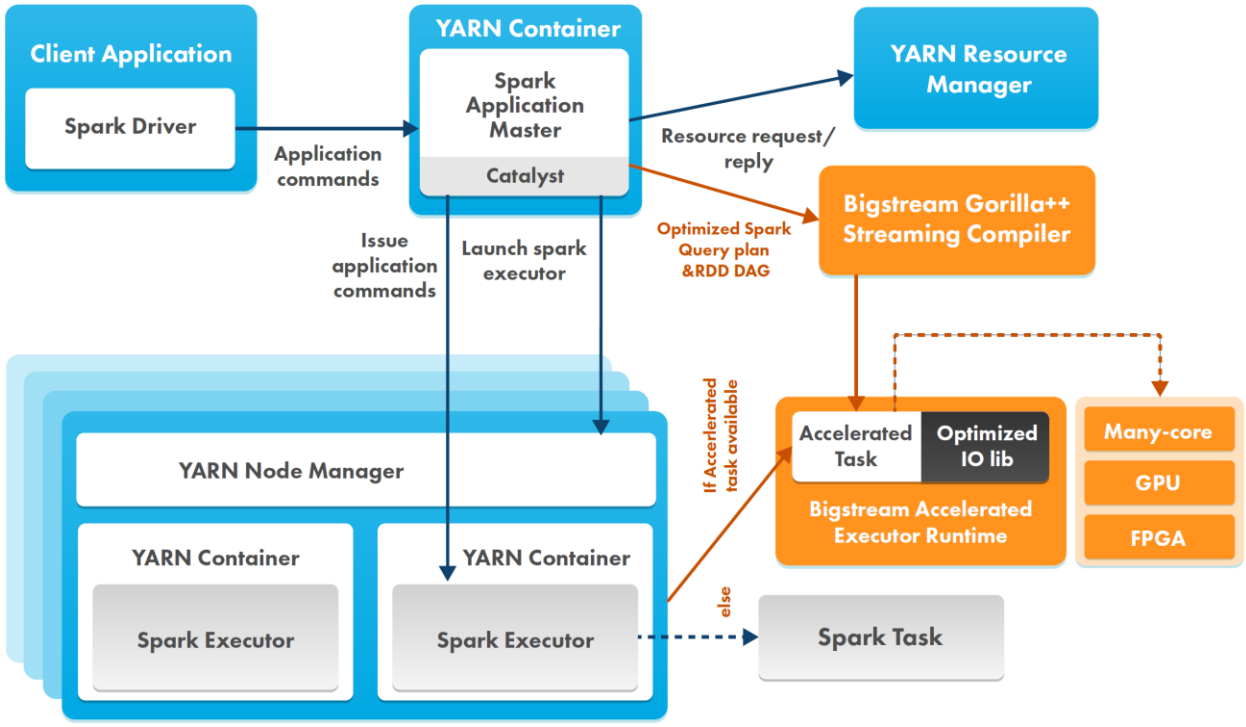


FIGURE 6: BIGSTREAM HYPER-ACCELERATION ARCHITECTURE

Figure 6 above shows Spark architecture with Bigstream HaL acceleration integrated. Note that this illustration applies equally equally to software and hardware (many-core, GPU and FPGA) accelerators. The red arrows and red outlined items indicate HaL components that are added at bootstrap time and can then provide acceleration throughout the course of multiple application executions. The Client Application, Driver, YARN components, and the structure of the Master and Executors remains unchanged. Bigstream HaL does not require changes to anything in the system related to fault tolerance, storage management and resource management. It has been carefully designed only to provide an alternative execution substrate

at a node level that is transparent to the rest of Spark. We describe the functions and interfaces of the components:

- **Spark Master -** Generates the physical plan exactly as in standard Spark through the execution of the Catalyst optimizer. Note that the standard byte-code for Spark tasks are generated by the Master as normal.

- **Bigstream Runtime -** The Bigstream runtime is a set of natively compiled C++ modules and their associated APIs that implement accelerated versions of Spark operations.

- **Streaming Compiler -** The Bigstream Streaming Compiler (Gorilla++) examines the physical plan and inspects/evaluates individual stages for potential optimized execution. Details of the evaluation are omitted here, but the output of the process is a set of calls into the Bigstream Runtime API, implementing each stage in the plan if deemed possible.

- **Spark Executor -** Via a hook inserted at cluster bootstrap time, all Executors posses a pre-execution check that determines if a stage has been accelerated. If so, the associated compiled module is called. Otherwise, the standard java byte-code version is executed. Thus, stages are accelerated optimistically, defaulting to being run as in standard Spark. As can be seen from this description, users of Bigstream HaL are presented an identical interface to standard Spark. This also allows Bigstream HaL to be updated incrementally as features become available, making it easily extensible.

Engage with Bigstream

When you collaborate with Bigstream to solve your real-world big data problems, your organization can expect state of the art acceleration technology and the expertise to realize significant, measurable ROI, and staggering performance gains. Engaging with Bigstream will result in successful production deployments, a better understanding of your computing workloads, and the optimal performance architecture to make big data and advanced analytics part of your competitive edge.

ACCELERATING INTELLIGENCE

Bigstream Solutions, Inc. 2570 W. El Camino Real, Ste 510, Mountain View, CA 94040