HYPER-ACCELERATION WITH BIGSTREAM TECHNOLOGY
Bridging the Gap

After holding for 50 years, Moore’s Law is coming to an end. The law that predicted a doubling in processor transistor count, and hence compute power, every 18 months has ceased to hold largely because of fundamental, physics-based reasons.

At the same time, big data and machine learning are being adopted by enterprises as a means of creating a competitive advantage. With input data needs ever-growing in size, 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.

Large clustered systems have been employed to address these large computations, but cluster scaling alone has its limitations in providing high performance. Scale up and scale out strategies can work effectively for smaller workloads, but they run into diminishing returns when cluster sizes (Scale out), or server capability (Scale up) grow larger.

Hardware acceleration such as GPUs, or FPGAs provide a vehicle to provide high performance and, in fact, enhance the gains of scaling as well. To date, however, acceleration has had limited success due to a key gap, illustrated by Figure 1.

![FIGURE 1. Programming Model Gap Inhibiting Hardware Acceleration]
Today, there is no automated way for big data platforms such as Spark to leverage advanced field programmable hardware. Consequently, data scientists, analysts and quants must work with performance engineers to fill the programming model gap illustrated in Figure 1. Though feasible, this process is typically inefficient and time consuming.

The gap stems from the fact that data scientists, developers and quants are accustomed to programming using big data platforms in a high-level language. Performance engineers, on the other hand, are focused on programming at a low level, including field programmable hardware. Thus, the scarcity of resources, along with additional implementation time can significantly lengthen time to value of analytics when accelerating. In addition, the resultant solutions are typically difficult to change/update as analytics evolve.

Bigstream has developed technology to address this gap. The architecture is illustrated in Figure 2.

At a high level, Bigstream Hyper-acceleration automates the process of acceleration for users of big data platforms. It is comprised of compiler technology for both software acceleration via native C++, and FPGA acceleration via bitfile templates. As shown in Figure 2, this technology yields between 2x-30x end-to-end factor in performance for analytics, but with zero code change.

The rest of this paper discusses performance results, use cases and technical details of Bigstream Hyper-acceleration.
Performance Analysis

Cluster Scaling with Acceleration

The most common method of increasing cluster performance when data needs grow is scaling. Scale up increases the capability of cluster nodes, keeping their number the same. Scale out refers to increasing the number of nodes in the cluster, keeping their type the same. Obviously mixed scaling approaches that apply both scale up, scale out also exist.

Figure 3 illustrates the two approaches to scaling. In this example, both approaches increase the number of virtual CPUs (vCPUs) as the cluster scales, increasing the compute power. It is also possible to scale in other ways, such as network connections, memory, disk and other resources. In both scale up and scale out, however, the idea is to increase performance by adding resources.

Scaling, in almost all cases, yields sub-linear performance increase as resources are added. That is, as the cluster is scaled by a factor of N, performance increase is almost always less than N. As scales become large and very large, this diminishing return becomes very severe. The technical reasons for this are listed below:

- **Scale Up**
  - Increased I/O overhead/throttling
  - Shared resource contention (memory, L2 Cache)
  - Scheduling complexity increase
• **Scale Out**
  - Increased network overhead
  - Straggler effect exacerbated
  - Failure rate increase

Acceleration such as that provided by Bigstream has the opportunity to improve the outlook for scaling in two ways: 1. It provides the ability to reduce the size of the cluster needed to yield a given performance level, and 2. It reduces the overhead of some of the above factors (i.e. network, I/O), thus reducing their impact.

![vCPU vs Speedup](image)

**FIGURE 4. Results of Scale up Experiment with Spark**

Figure 4 shows the results of some experimentation conducted that illustrates the scaling issue, in this example for scale up. We ran two TPC-DS (http://www.tpc.org/tpcds/default.asp) benchmark queries on Amazon EMR using Spark, in various cluster scenarios. Moving from left to right in the figure, each point represents the performance seen with the given number of vCPU (16, 32, 64, 128, 256). Thus, we scale up the cluster by 2x at each step. Speedup is calculated with respect to the datapoint labeled “Base”. The speedup performance of both benchmarks fall off from the blue linear line as the cluster scales, likely due to reasons listed above.
Figure 5 shows the same results as Figure 4 (dashed lines), but with results for clusters equipped with Bigstream software-based acceleration in use (solid lines) added. The accelerated curve displays a much more gentle falloff with scaling than the Spark curve. In addition, comparing datapoints horizontally, we see that acceleration has the potential to allow a smaller cluster to actually outperform a larger cluster.

We see similar results in experiments with scale out. These results indicate that acceleration can work synergistically with scaling, to provide maximum performance and a wide variety of performant configuration choices for the user. This, in turn, can result in total cost of ownership (TCO) savings. For cloud users, it enables the use of smaller clusters, or use of the same cluster for a shorter amount of time, to achieve a given analysis. For on-premise clusters, it allows for more analyses to be accomplished per unit of operation time.
FPGA Acceleration

As stated earlier, hardware-based acceleration has the highest performance potential. Adding an FPGA to a server can be a cost-effective way to speed up big data platforms, if the introduced hardware can be easily leveraged. These chips are typically a fraction of a cost of a full CPU-based server.

![Speedup FPGA over CPU-only](image)

**FIGURE 6. F1 (FPGA Instance) Speedup Results for TPC-DS Suite**

The performance results in Figure 6 represent a demonstration of Bigstream FPGA-based acceleration on (i.e. running on an FPGA instance). The results were obtained using a commodity FPGA platform, with Bigstream Hyper-acceleration software installed. 104 TPC-DS Spark benchmarks were run on the platform CPU-only (baseline), and using the FPGA (accelerated). Speedup was calculated per-benchmark by dividing the baseline runtime by the accelerated runtime.

A maximum and average of 5x and 3.3x speedup, respectively, were observed, with zero code change to the benchmarks. As the Bigstream FPGA product evolves, we expect to use both multiple-FPGA configurations, and a larger footprint on each chip. Therefore, we expect this number, and hence performance, to increase. This result demonstrates not only the performance advantage that hardware-based acceleration can provide, but also the usability Bigstream can provide for FPGA platforms.
Technical Overview

This section presents a technical overview of Bigstream technology as applied to Spark, 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

![Baseline Spark Architecture Diagram]

Many-cores

**FIGURE 7. Baseline Spark Architecture**

Figure 7 shows the basic components of standard Spark using YARN for resource management. 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 Resource Manager with requests for resources as per the application needs. The Resource Manager 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**

![Bigstream Hyper-acceleration Layer Architecture Diagram](image)

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

• 2-30X Acceleration

Figure 8 above shows Spark architecture with Bigstream acceleration integrated. Note that this illustration applies equally to software and hardware (many-core, GPU and FPGA) acceleration. 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, Resource Manager components, and the structure of the Master and Executors all remain 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 (software acceleration), or bitfile templates (FPGA acceleration) and their associated APIs that implement accelerated versions of Spark operations.

• **Streaming Compiler** - The Bigstream Gorilla++ Streaming Compiler 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 possess 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. It is important to note that this check is transparent to the programmer; she is unaware whether a stage is running accelerated, except for the difference in performance.

Thus, stages are accelerated optimistically, defaulting to being run as in standard Spark. This approach ensures that users of Bigstream are presented an identical interface to standard Spark. This also allows Bigstream acceleration software to be updated incrementally as features become available, making it easily extensible.

### Use Cases

Bigstream’s current product focuses on acceleration of Spark Platform in the following use cases:

• ETL
• Streaming analytics: Spark Streaming, Kafka
• SQL Analytics: Hive or Spark SQL
Engage with Bigstream

We are offering the Bigstream acceleration layer to solve real-world big data problems. Engaged organizations can expect state of the art acceleration technology and the expertise to realize significant, measurable ROI, and staggering performance gains. Working with Bigstream will result in successful accelerated production deployments, as well as a better understanding of computing workloads. The optimal performance architecture will make big data and advanced analytics part of your competitive edge, in any business area.