ACCELERATING BUSINESS INTELLIGENCE

Overview

With every aspect of our living getting digitized, there is a deluge of data getting generated every moment. Enterprises too are creating massive amounts of business data. Data alone itself is not sufficient unless it can be converted into useful insights. Hive has emerged as a popular choice for data analysts due to its extensive support for SQL and distributed query processing over large clusters.

There is insatiable need for speed from analytical queries. At the same time, with Moore's law coming to an end, CPUs have hit a performance wall. Two popular ways of getting more performance out of CPUs are: Scale-up and Scale-out. Many studies have clearly established that RAM/disk contention and network I/O severely limit benefits of scaling up/out beyond a point.

FPGAs have emerged as a very popular choice for accelerating variety of algorithms. With ever increasing on-chip/DDR memory sizes, improvement in data transfer speeds and ease of programming, FPGAs are well suited to take on Big Data work-loads.

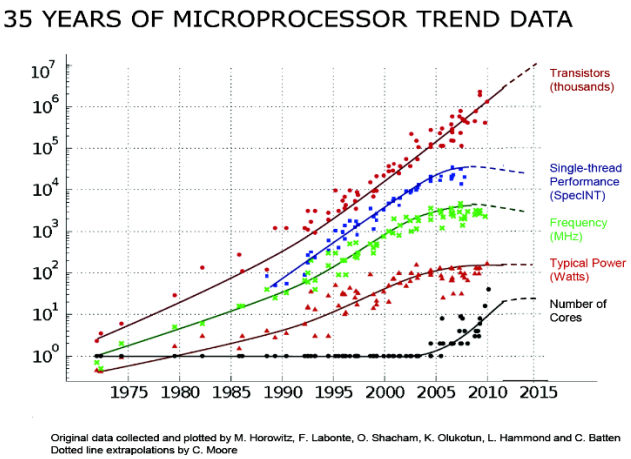
*Realizing the need for speeding up analytical queries, BigZetta has augmented Hive with FPGA based acceleration. BigZetta solution (**bzQAccel**) has demonstrated **speed-ups of 4x** (and more) over several TPCH benchmark queries.*

Data Deluge, End of Moore's Law and Way Forward

Personal assistants, sensors, cameras, IoT devices, social media, web crawlers etc. are generating and recording minutest of details about user interactions and movements. At the same time enterprises are creating massive amounts of business data. All this data needs to be converted to actionable insights. For decades, enterprises have relied on databases to store and extract knowledge. Bigger and powerful machines have been built and their clusters have been deployed to serve the needs of a variety of domains: finance, retail, health, manufacturing etc. Ever increasing compute power also meant sky-rocketing costs which only select few could afford.

Last decade or so saw emergence of open-source Big Data solutions whose primary objective was to democratize availability of enterprise grade solutions on commodity hardware. All these tools relied on sending computations closer to data which would be distributed across multiple nodes (sometimes spanning geographies). Fault-tolerance, error-recovery and resilience to software/hardware failure have been the key defining features of these tools. Primary use of Big Data solutions has been for batch-processing jobs where successful finishing of job is far more important than the time it takes to finish it. For this reason, use of Big Data tools for interactive analytics has been somewhat limited. Realizing this gap, Big Data community has been coming up with solutions which would enable Low Latency Analytical Processing over tera/petabytes of user data. Of these solutions, Hive has emerged as a popular choice for data analysts due to its extensive support for SQL like query processing.

There is an insatiable need for speed from analytical queries. At the same time, CPUs have hit a performance wall as the chart below shows



While it is clear from the chart that CPU frequency has nearly saturated, it is also evident that number of cores on a CPU has been continuously increasing to catch up with the performance requirement of modern computing.

Two very popular ways of reducing the turn-around time are: Scale-up and Scale-out. While Scale-up relies on increasing the power of every node (put more cores, RAM, disk etc.) in the cluster, Scale-out means putting more nodes in the cluster. Both the approaches have pros and cons and both tend to saturate rather quickly. In a nut-shell, more of something does not necessarily mean ever increasing performance benefits. Many studies have clearly established that limited RAM access, disk access and network I/O severely limit benefits of scaling up/out beyond a point.

The message is loud and clear that we **need to look beyond CPUs** for further performance improvements.

FPGAs for Big Data Acceleration

Both GPUs and FPGAs have been widely used to accelerate specific computations and have enjoyed varying degree of success in doing so. While GPUs use massive parallelism (through thousands of cores running SIMD instructions in parallel) to achieve speed-ups, FPGAs rely on building dedicated hardware for a computation and use techniques like pipelining/unrolling etc. to speed-up general computations. GPUs and FPGAs exist as co-processor to CPU and have very similar steps for interacting with the CPU. Figure 1 shows steps involved in offloading a computation to FPGA:

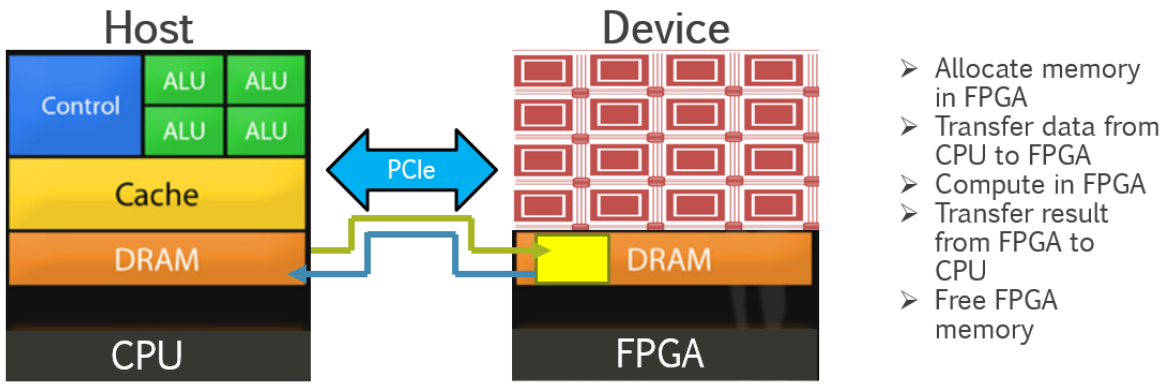


Figure 1: CPU <-> FPGA interaction

FPGAs provide better performance/Watt than CPUs or GPUs. Clever use of pipelining and unrolling can let an application developer explore many solutions for a given application and pick the one which suits performance/area/power consideration. Traditional drawbacks of FPGAs have been lack of availability as well as requirement of hardware design knowledge to get the best performance. Of late, FPGAs have been made available in all major clouds and are also coming pre-installed in many commercial servers. Major advancements in High Level Synthesis technology have ensured that application developers (with no or little background in hardware design) can write their algorithms in C/C++ and get the benefits of FPGA acceleration.

FPGAs are available on all major commercial clouds (AWS, Azure, Alibaba, Nimbix etc.) and can be rented by application developers in pay-per-use model.

Three big care-abouts in Big Data space are: Volume, Velocity and Variety. As we will see later that modern FPGAs are well suited in catering to these requirements and can offer very superior performance for mission critical Big Data applications like interactive query processing.

Distributed Query Processing Using 

Hive has emerged as one of the most popular query processing tools from Apache Open Source foundation. Its extensive support for SQL, integration with MapReduce/Spark/Tez and support for various input formats (ORC, Parquet, Text etc.) make it a very popular choice for data-warehousing. As of today, there are more than ten thousand (according to published reports) companies using Hive to solve their data processing needs. Hive caters to a variety of use modes: data warehousing, ETL, analytics etc. Besides being available as Open Source technology, Hive is being offered by all major cloud vendors (Amazon, Microsoft, Google, Alibaba, IBM etc.) and also from commercial Big Data vendors like HortonWorks/Cloudera, MapR/HPE, Qubole etc.

Hive processes input query by taking it through a number of stages: parsing, logical/physical plan creation, DAG creation, job execution etc. These stages are shown for a sample query in Figure 2:

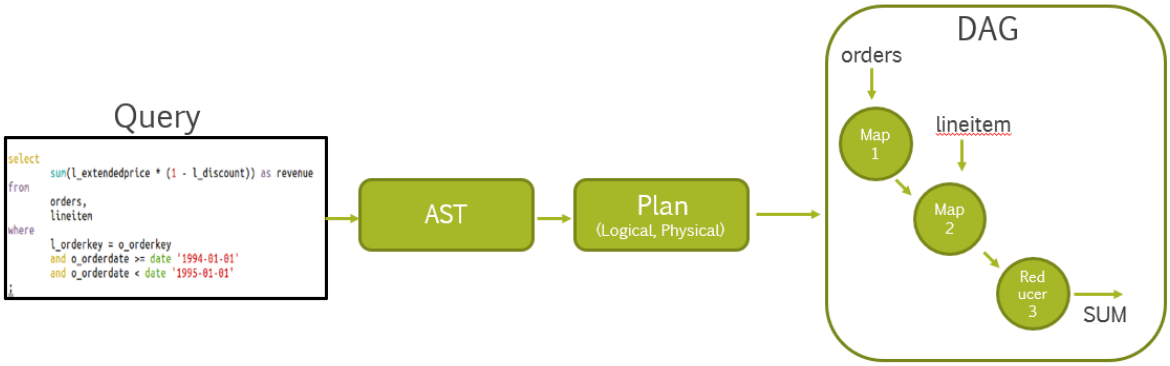


Figure 2: Query processing in Hive

DAG is then submitted to a cluster of machines (Hadoop cluster) to run in a distributed fashion. Each machine contains a portion of table data and runs assigned node(s) of DAG on that data. This distributed processing framework is shown in the Figure 3:

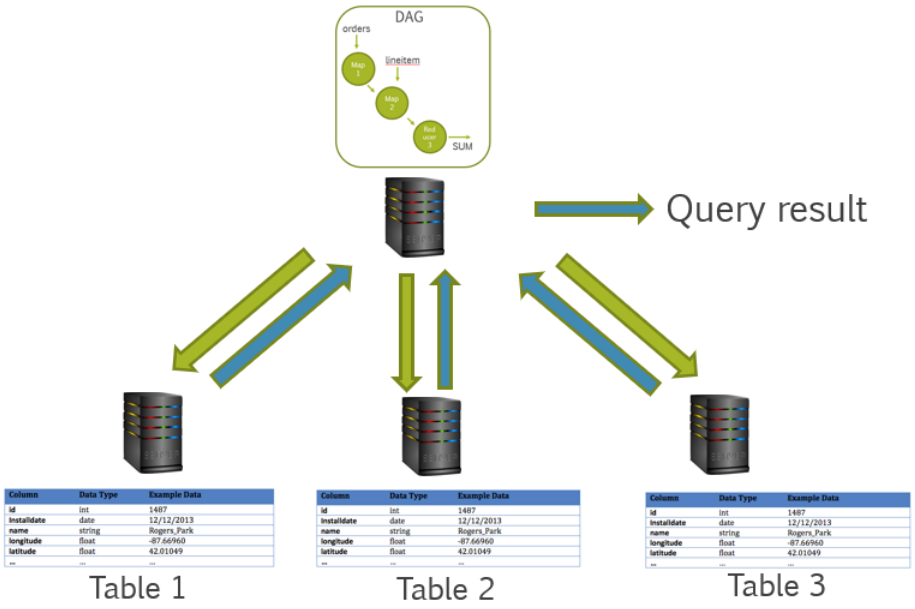


Figure 3: Distributed query processing

While initial use of Hive was as a batch-processing tool, there have been numerous improvements to make it suited for low-latency analytical processing. Introduction of ORC (Optimized Row Columnar) format, Tez engine, vectorization of operations and LLAP (Long Live and Process) have made it very suited for Business Intelligence queries where speed is a major requirement.

So far, Hive developers have been relying on CPU-only techniques to optimize Hive for runtime. New data formats (like ORC, Parquet) cut down data ingestion time. Introduction of Tez removes overheads associated with MapReduce processing engine. Use of vectorization makes use of super-scalar technology to improve operator processing time. LLAP enables in-memory processing so that expensive disk accesses can be eliminated. A positive outcome of these changes has been that Hive has transformed from an IO heavy engine to more of a compute heavy engine where majority of time is now spent in computations rather than file reading and data processing. This also means that Hive is ready to benefit from advances made in heterogeneous computing whose primary target is to cut down the computation time of an application.

BigZetta Query Accelerator: Accelerating Using FPGAs

Hive (like most of other Big Data technologies) has been developed with CPU in consideration. All the stages of query processing: plan generation, DAG creation, DAG scheduling and execution are oblivious of heterogeneous devices (like FPGAs). Forcefully fitting a plan, generated to work with CPUs, onto FPGAs can lead to sub-optimal results. BigZetta has identified this problem and has changed major stages of Hive processing: plan generation and optimization, DAG creation, job execution etc. to make optimal use of FPGAs to provide best possible speed.

Changed FPGA aware Hive flow is shown in the Figure 4:

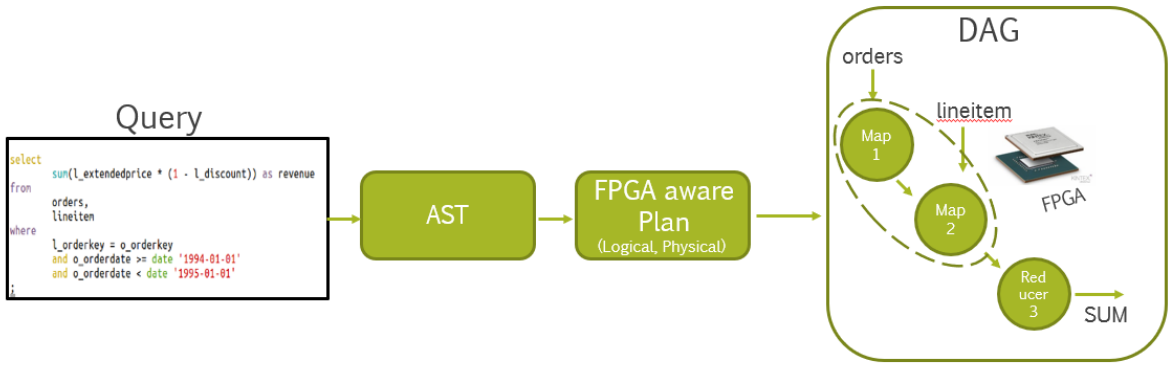


Figure 4: FPGA aware query processing in Hive

For this example query, FPGA aware plan generation has identified that it is best to run Map1 and Map2 on FPGA(s) to get fastest execution. Whereas, node Reducer3 can run entirely on CPU. DAG is then submitted to a cluster of machines (having FPGAs in addition to CPUs) to run in a distributed fashion. Figure 5 shows query getting processed on a cluster of machines with CPUs and FPGAs:

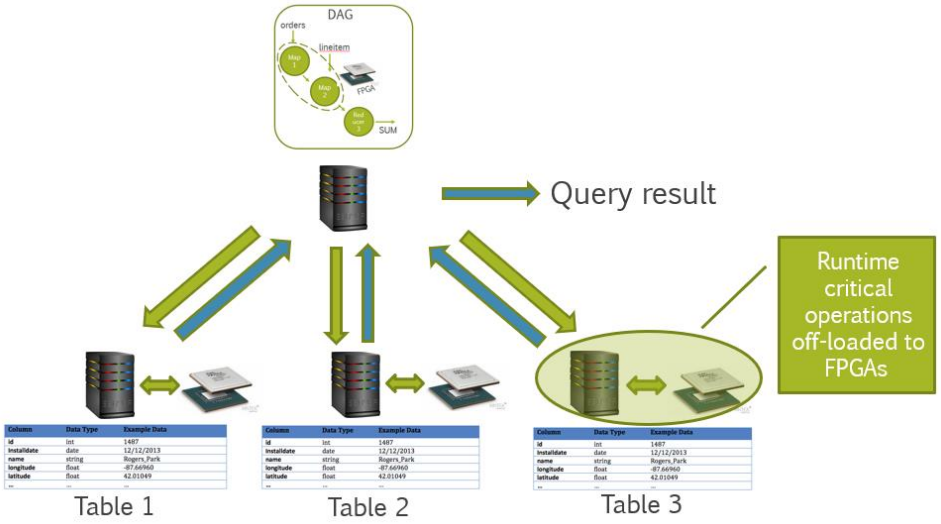


Figure 5: Distributed query processing on cluster of CPU+FPGA machines

The key aspect of FPGA based Hive acceleration is the interaction between the CPU (Host) and the FPGA (Device). BigZetta's Query Accelerator (bzQAccel), has been designed to manage the interaction between Hive code running on CPU and **offloading of runtime critical operations onto FPGA**. Figure 6 shows the interaction between CPU and FPGA through bzQAccel layer:

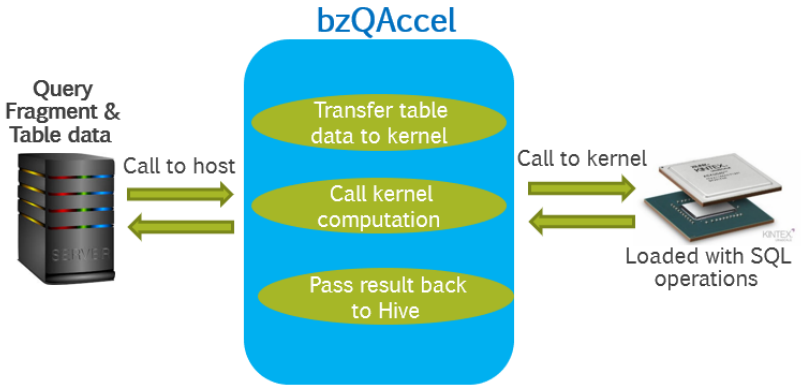


Figure 6: bzQAccel layer between CPU and FPGA

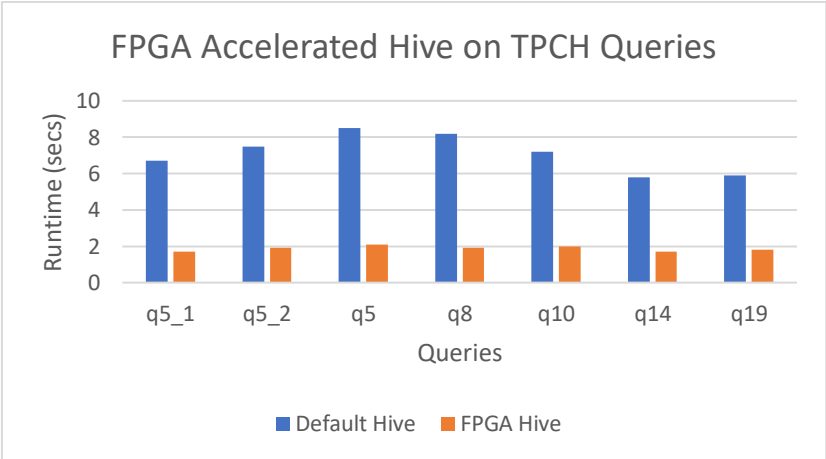
Key features of bzQAccel are:

- Acts as middleware between Hive and underlying hardware
- Optimizes query execution plan suited for FPGAs
- Provides fastest execution of the plan on FPGAs
- For different queries, no need to recompile either the host code or the FPGA kernel
- Minimal penalty of data movement (from CPU to FPGA and back)

bzQAccel works in conjunction with standard Hive deployment (either Open Source version or provided by Big Data vendors).

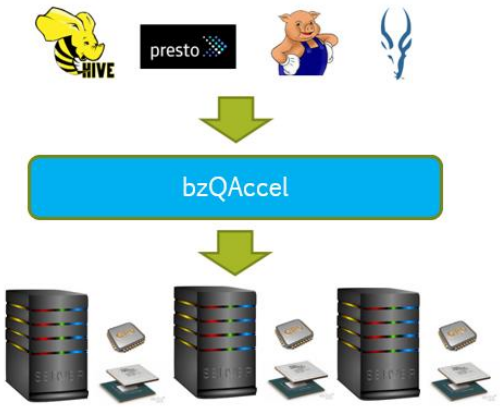
bzQAccel Results on TPCH Benchmark Queries

We compared performance of Hive augmented with bzQAccel to that of standard Hive for several TPCH benchmark queries. Chart below shows the runtime performance analysis.



Gains of FPGA based acceleration are very evident. For variety of queries from TPCH suite, there is 3-4x speedup over default Hive. Neither the host code nor the FPGA kernel required any changes to support different queries.

Benefits of bzQAccel



- Improves runtime of analytical queries
- Provides up to 4x speed-up over CPU
- No need to build separate hardware for each query
- Can be deployed on top of an existing Hive installation (either open source or from a commercial vendor)
- Technology can easily fit into any other query processing engine (Spark, Presto etc.)

Availability

BigZetta is engaging with early customers who want to do a pilot with our technology. Early adopters will get a chance to influence the product development suited to their feature requirements. To engage, please contact us at info@bigzetta.com OR sales@bigzetta.com .

BigZetta Systems

Reimagining Big Data