Speeding up Spark with Data Compression on Xeon+FPGA

David Ojika
University of Florida

Intel Collaborators: Piotr Majcher, Wojciech Neubauer, Suchit Subhaschandra, Ramesh Illikkal, Bhaskar Gowda, and PK Gupta

Motivation

- **Big data**
  - Growth in volume of data
  - Distributed processing: data shuffling across machines
- **Data compression**
  - Reduce data volume, optimize application performance
  - Forbes*: 60% organization using data compression
  - A CPU-intensive operation
- **Programmable accelerators (FPGAs)**
  - Core-scaling: CPU may be reaching performance limits
  - Rising demand for performance efficiency, cost in the datacenter

Compliment CPU cores with FPGAs for improved Spark performance

*https://www.altera.com/en_US/pdfs/literature/third-party/forbes_The_Coming_Data_Avalanche.pdf

About Me

- 4^{th} year PhD student
- Interests in distributed systems, FPGA acceleration of data intensive computing
- Research with CERN
- Past internship at Intel
- Current internship at Microsoft Research

What / Why FPGAs

- Field-programmable gate array (FPGA)
  - Custom circuit
  - Can accelerate specific tasks
- FPGAs offer:
  - Reconfigurable architecture
  - Low-power, energy efficiency

- FPGA attachment technology
  - Loosely-coupled
    - PCI-e attached FPGA
  - Tightly-coupled
    - Xeon+FPGA

Xeon+FPGA

- Xeon CPU and FPGA in a single processor socket
  - Cache coherent interface
- Supports "in-line" data via direct I/O
- Accelerator Function Unit (AFU)
  - Reconfigurable region (user logic)

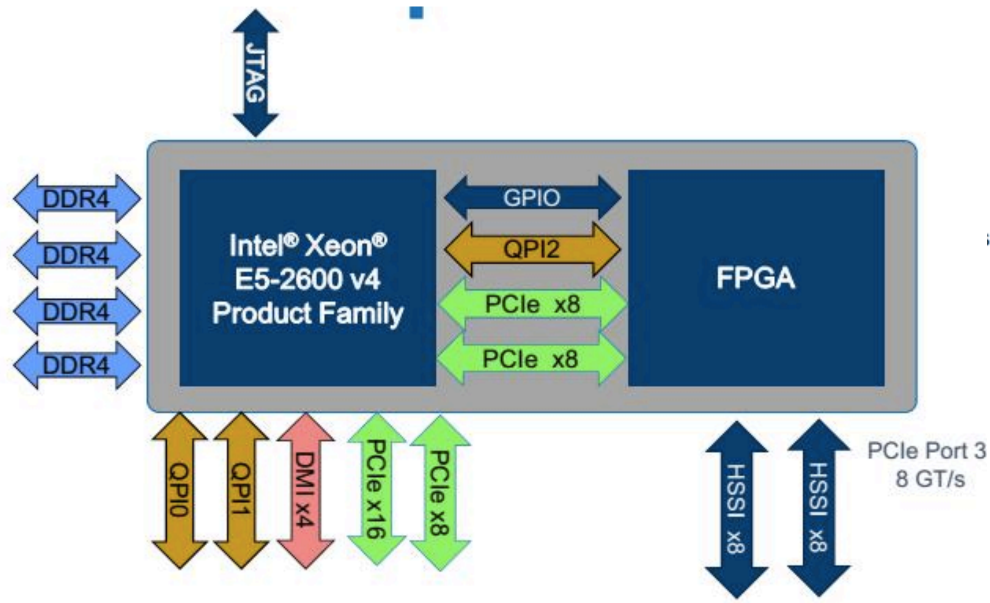


Image: Courtesy of Intel

Challenges of integrating FPGAs into big data systems

Challenging Programming Model

- Requirement on hardware-specific knowledge
- Long synthesis (compile) times
- Limited platform-portability

Complicated Software Interface

- JVM-to-FPGA interface
- Data transfer overheads

FPGA Sharing

- FPGA and CPU threads co-existence is non-trivial
- How to keep FPGA accelerator fully utilized

Reconfiguration

- FPGA reconfiguration can take milliseconds to a few seconds
- Certain workloads may be intolerable to downtime

…a gap between

FPGA accelerator developer

and

big data application developer

What did we do?

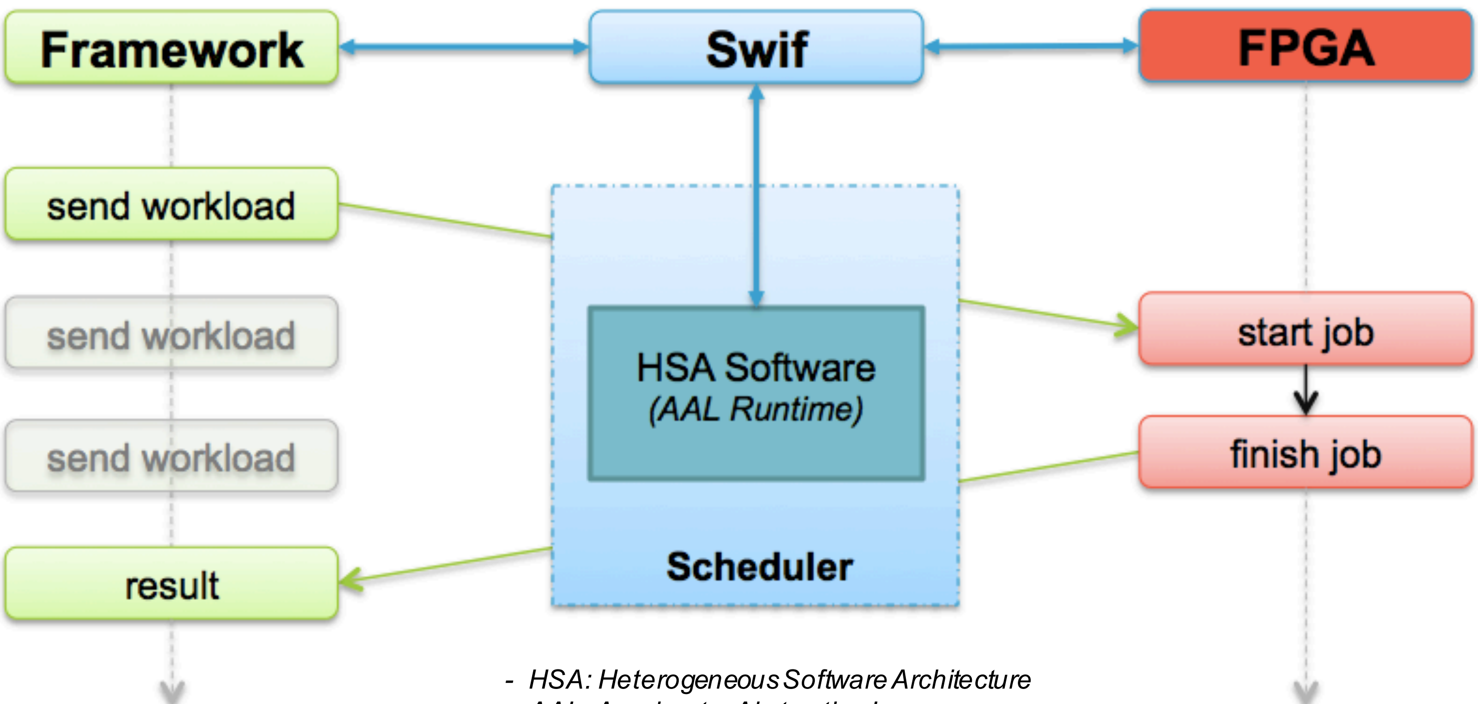
a. FPGA accelerator abstraction
   1. Java API for CPU offload to FPGA
   2. Manage JVM-to-FPGA data transfers
   3. Coordinate FPGA and CPU thread co-existence

b. FPGA-based compression plugin for Spark
   - No changes to existing application required
   - Compatible with existing Spark/Hadoop installations

***Swif* – "*simplified workload-intuitive framework*"**

- A flexible accelerator system with 'FPGA-accelerable' workloads as first-class citizens

Swif Overview

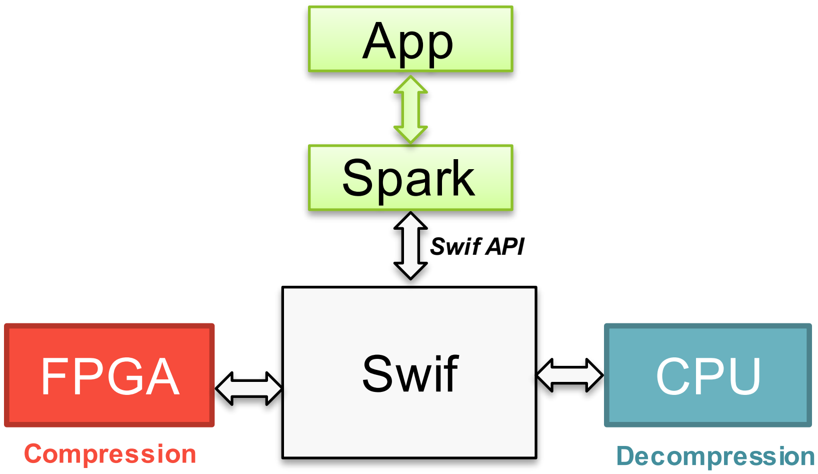


- *HSA: Heterogeneous Software Architecture*
- *AAL: Accelerator Abstraction Layer*

Swif API

```
//create accelerator (use FPGA by default)
Swif swif = new Swif();

//create workload
SwifWorkload workload = swif.create(WOKLOADS.Compression.zlib);
                        //other WORKLOADS: DNN, Transcoding, ...

//assign parameters to workload
workload.input  = inputArray;
workload.output = outputArray;
workload.size   = arraySize;

//start a task with the workload
swif.launch(workload);

/* SwifLauncher may be used instead, for a
   more advanced launch strategy  */
```

Compression use-case



Design Goals:

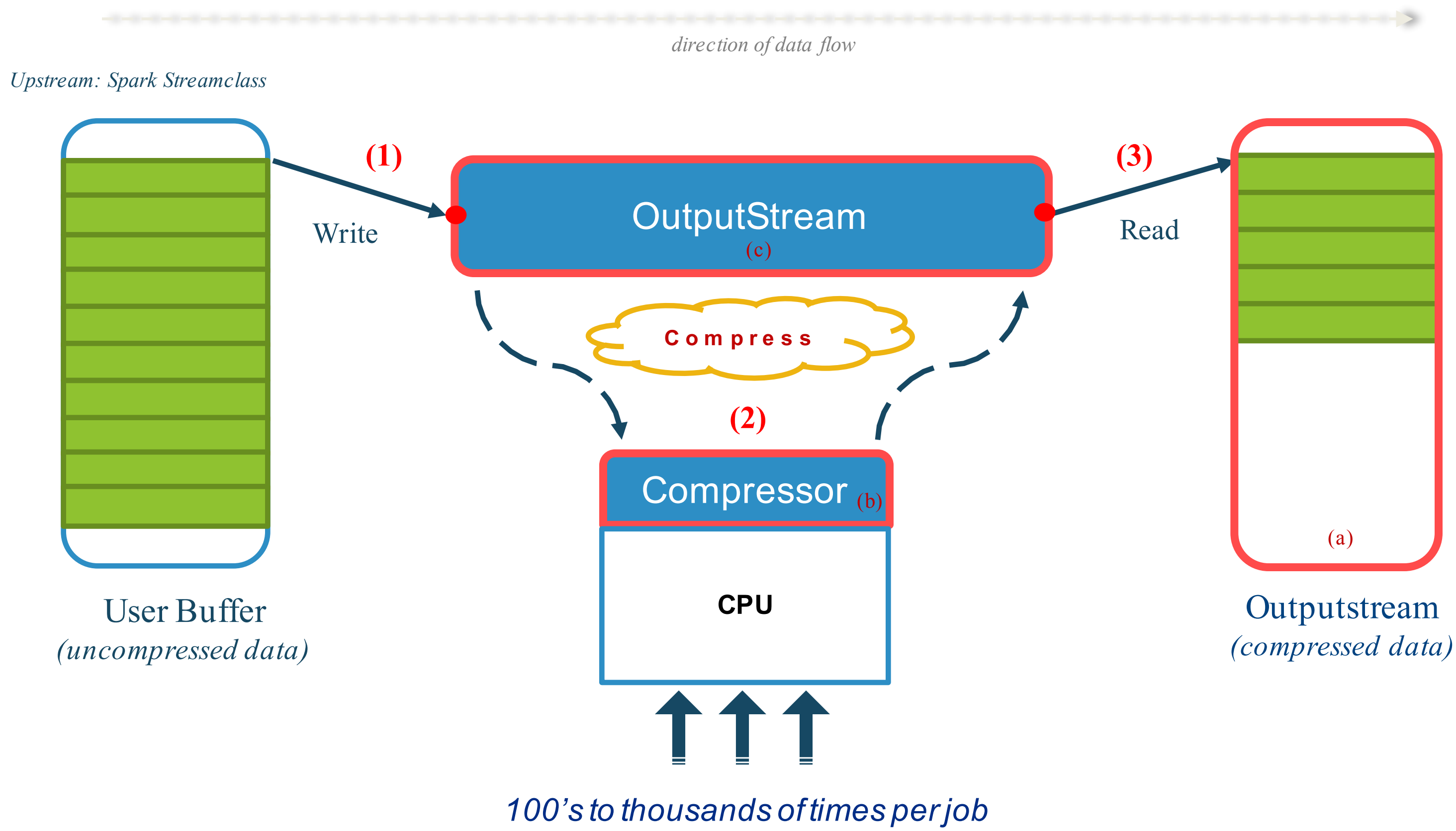
- Plugin model
- Failure resilience
- Heterogeneity

Swif in Spark: How to use

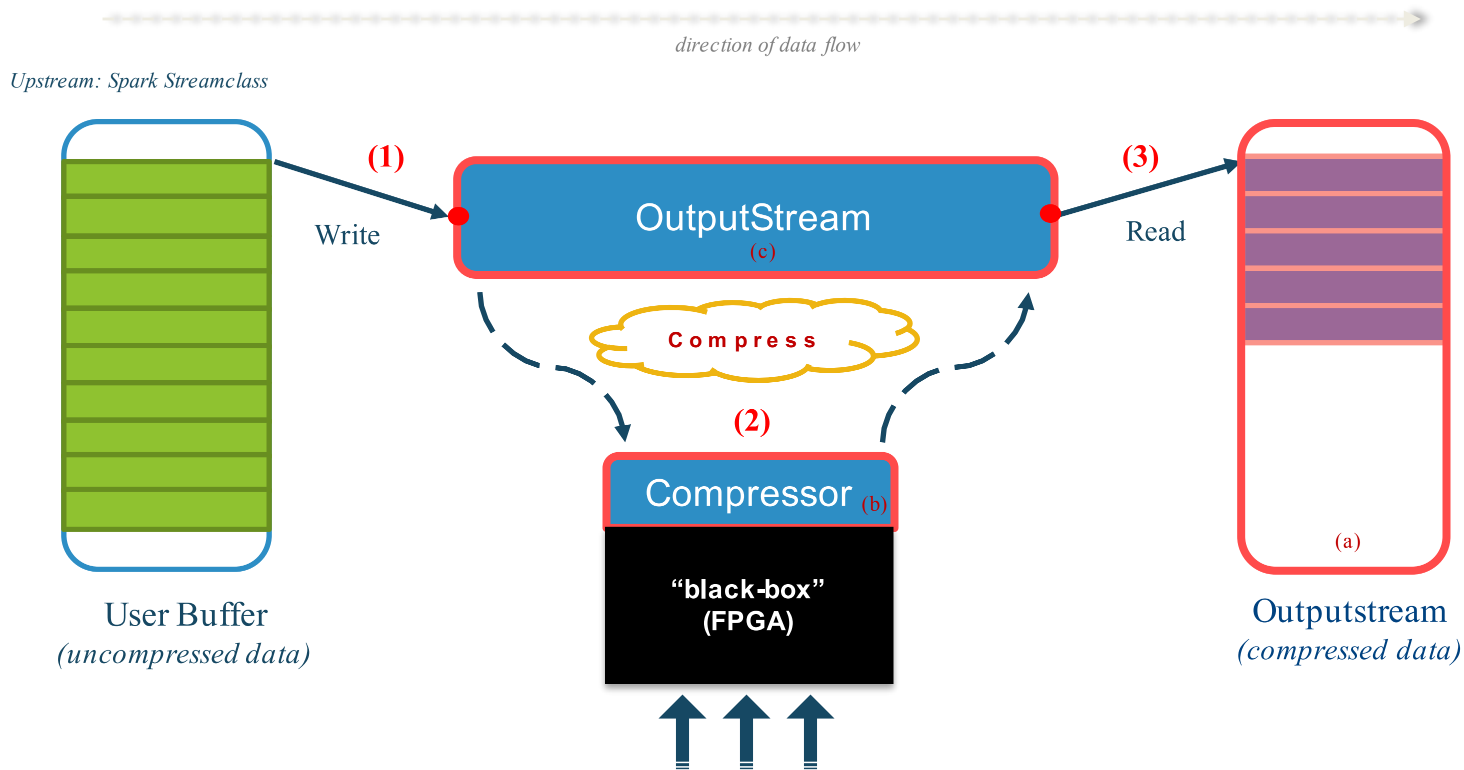
1. Export
   - **LD_LIBRARY_PATH** *= FPGAnatives.so*
2. Set
   - **CLASSPATH** *= FPGA.JAR*
3. Configure
   - **spark-defaults.xml** → *compression.codec = FPGACompressorCodec*
4. Run
   - **spark-submit --class myApp**

Implementation Details

Compression in Spark

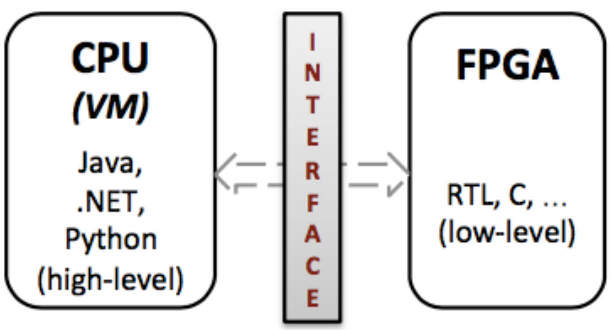


Compression in Spark – *with FPGA?*



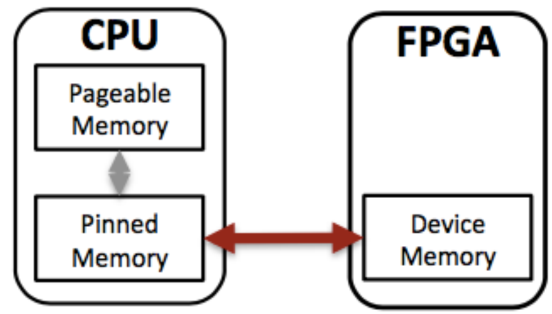
100's to thousands of times per job

FPGA-to-JVM Interface



JNI, other language wrappers

Expose FPGA accelerator functions



Low-latency, high-bandwidth link

Manage buffer allocation/movement

Interface with Spark

- *FPGACompressor, FPGACompressorCodec*
  - *Extendable* classes
  - *Implements* compression interfaces of Spark

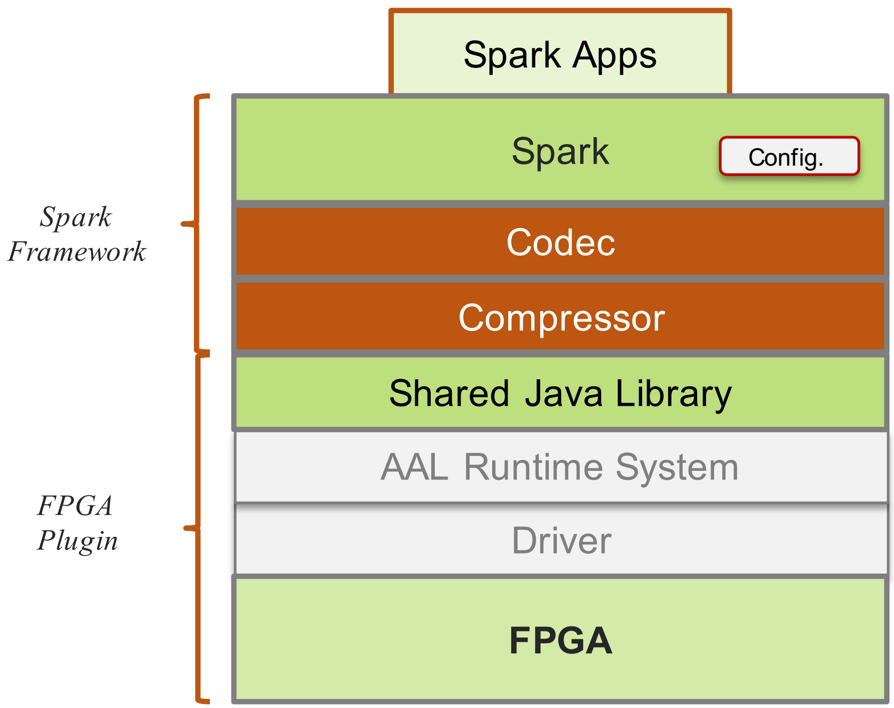
```
class FPGACompressor {

 //FPGA compress
 Compress(byte[] b, int off, int len) {

  //Swif API here...
  //...

 }
}
```

FPGACompressor: *base class*

```
class ZlibFPGACompressor: FPGACompressor
 implements ZlibCompressor

 //called by a Spark task
 Compress(byte[] b, int off, int len) {
  base.compress( in , off, len);
 }
}

```

ZlibFPGACompressor*: Compressor class for Spark*

Putting it all together → *Swif Stack*

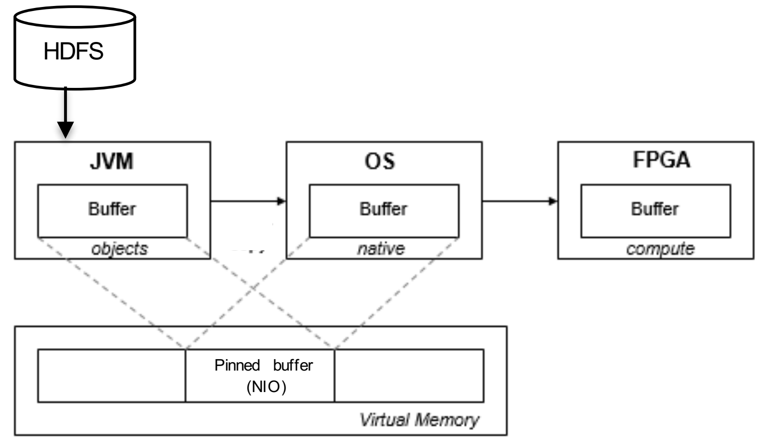


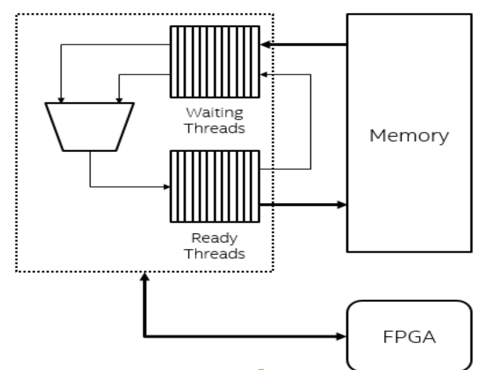
- ***Config:*** *Enable/Disable codec in Spark configuration settings*
- ***Standard interfaces for Spark:*** *compressor, streams, etc.*
- ***Compressor:*** *FPGA Compressor (ZLIB)*
- ***Library & Commons:*** *Generic access to FPGA from Java*

Optimizations

System Optimizations

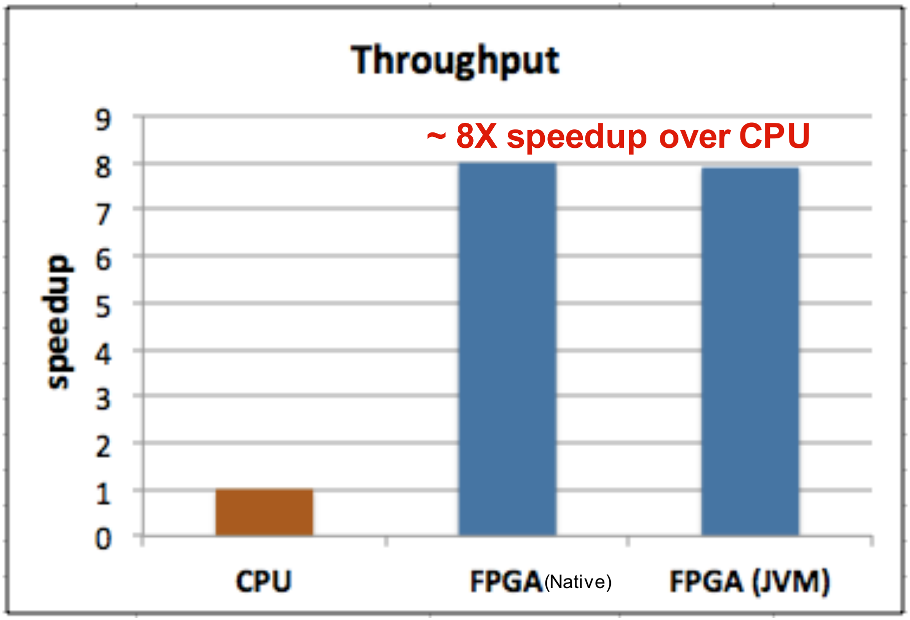
- HDFS block size
  - Apache: 64 MB, Cloudera:128 MB
- NIO buffer
  - Buffer size = *block size*
- Accelerator sharing among threads
  - Granularity of task parallelism effectively controlled by block size
  - Buffer reuse
- RDD caching
  - Faster FPGA access to data

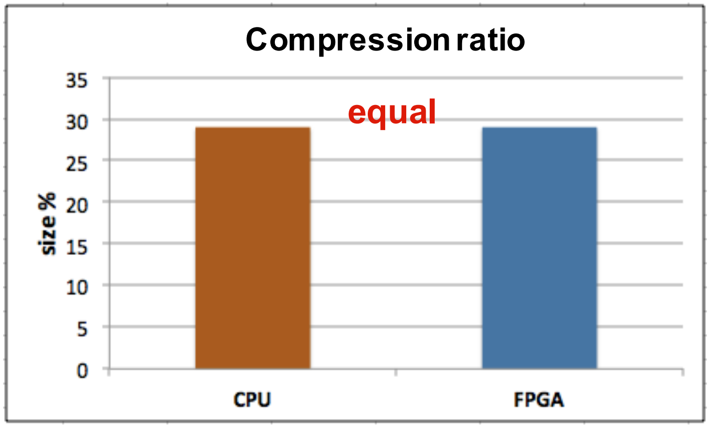




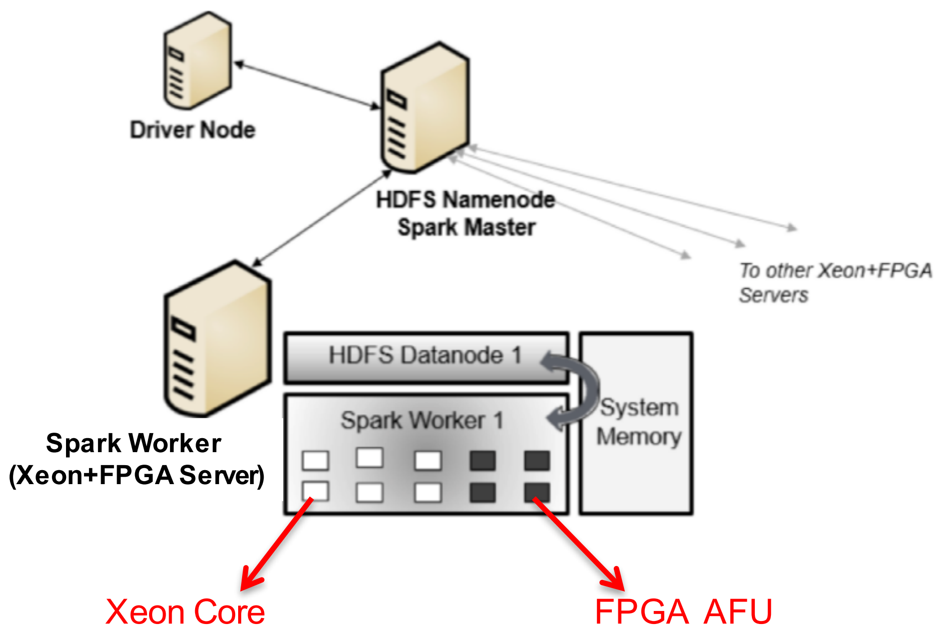
Results

Raw Compression Performance





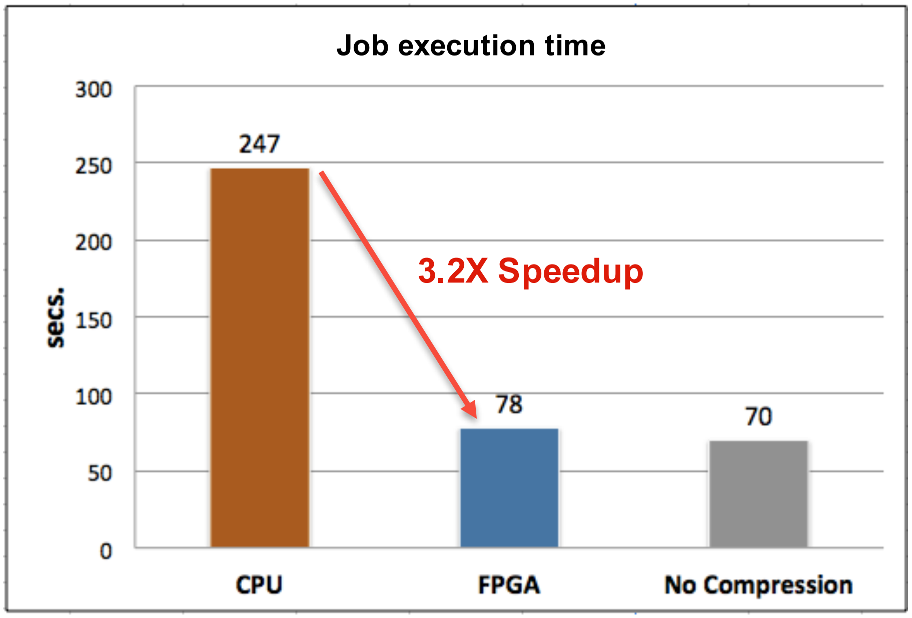
Application Profile

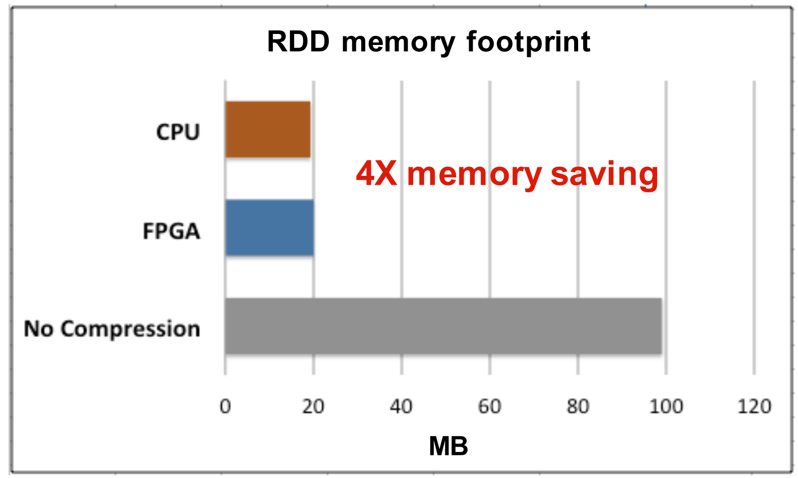


- Single-node Spark Cluster
- Focus on RDD Output compression on FPGA
- Multi-executor Spark Job
  - TeraSort

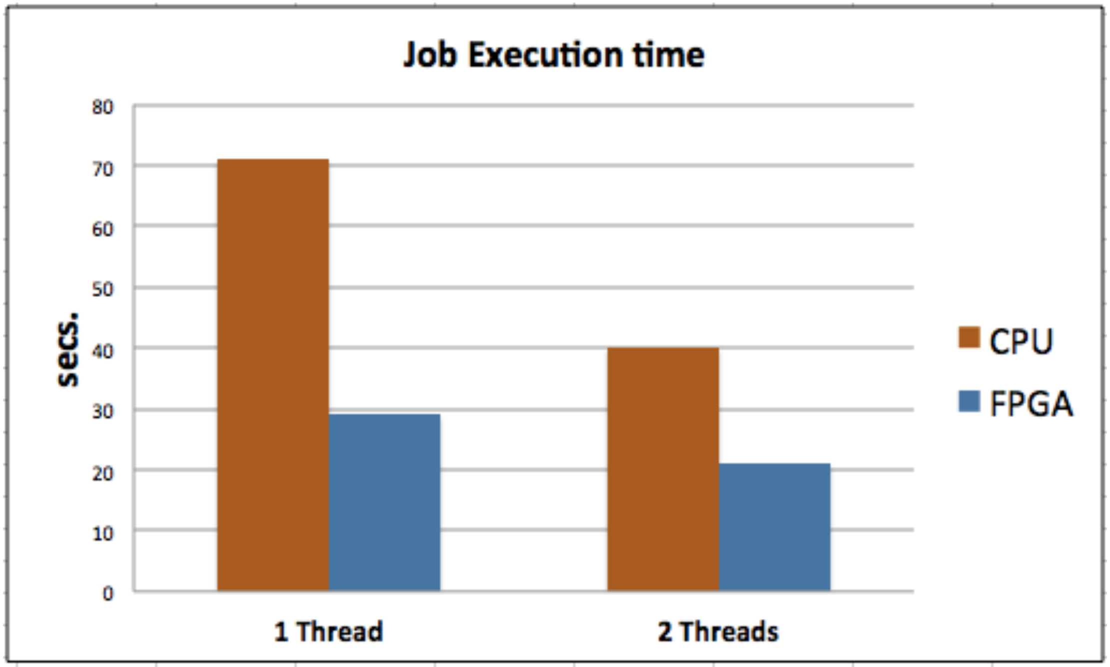
"Swif: A Simplified Workload-centric Framework for FPGA-Based Computing" D. Ojika, et. al., FCCM 17

System Performance





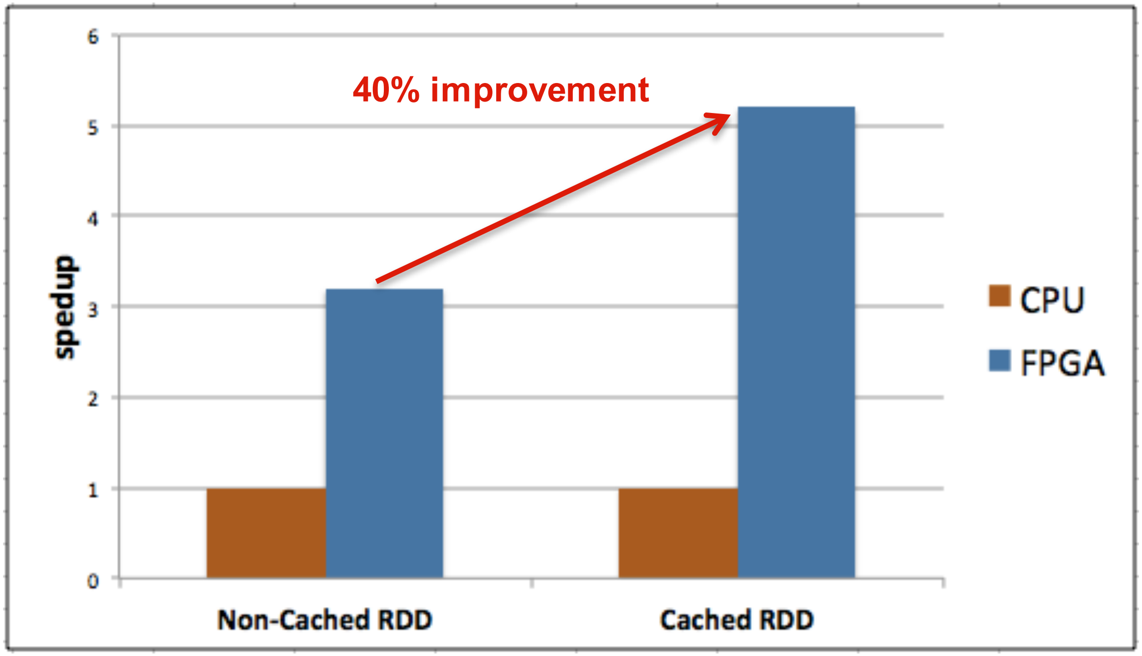
System Performance (Multicore)



Offload of multiple CPU threads (Spark Executors) to FPGA

- Increased FPGA utilization
- Still 2X faster than CPU run

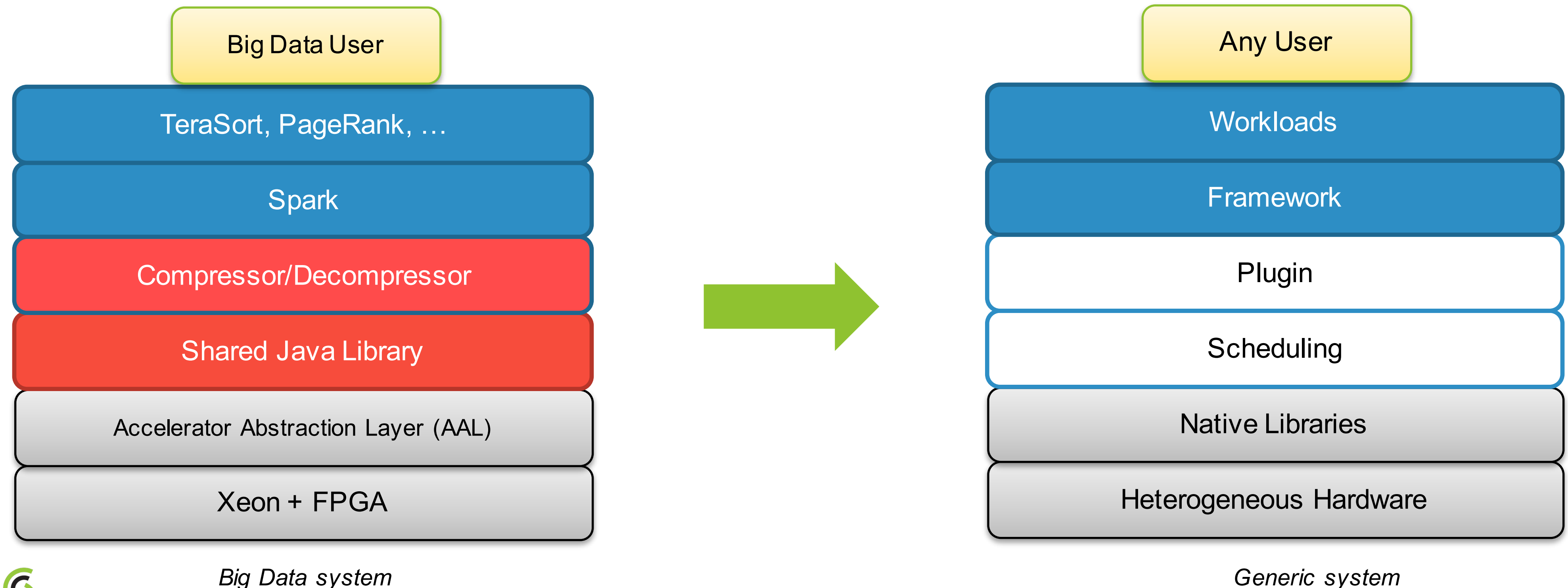
System Performance (with Data caching)



Conclusion

- JVM-based frameworks can efficiently leverage FPGA accelerators
  - Key to efficiency is software to FPGA interfacing
  - Treat workloads as first-class citizens
- Case-study on compression offload in Spark:
  - 3.2X job speedup, 4X reduction in RDD footprint
  - Potential for larger savings in a multi-node cluster environment
    - Storage, network bandwidth, etc.
- Swif is an ongoing effort
  - More work still to be done

Swif: The Big Picture



More Details

- "Towards FPGA as a Microservice"
  - Invited talk: *12th Workshop on Virtualization in High Performance Cloud Computing (VHPC)* at ISC '17

Acknowledgments

- Intel for internship opportunity
- University of Florida / Intel collaboration (HARP)
- Intel for PhD fellowship

Thank You.

David Ojika, davido@ufl.edu