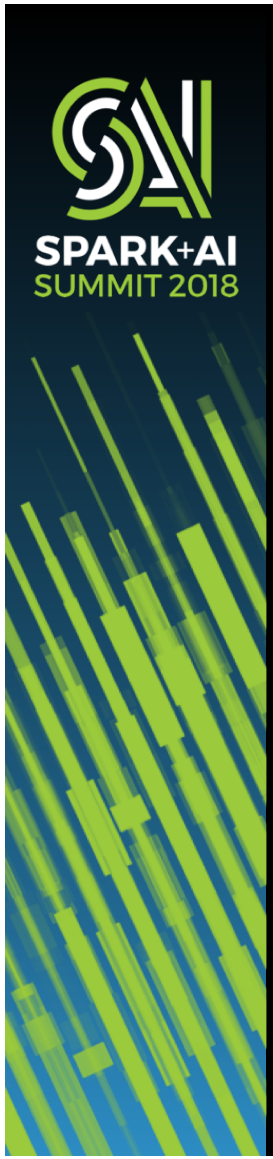


Best Practice of Compression Codecs in Spark

Sophia Sun (sophia.sun@intel.com)
Qi Xie (qi.xie@intel.com)
Hao Cheng (hao.cheng@intel.com)

Legal Disclaimer

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

For Performance Claims and Optimization Notice

Software and workloads used in performance tests may have been optimized for performance only on Intel® microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804.

Performance estimates were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

For more information go to http://www.intel.com/performance.

About me

- Big data software engineer from Intel.
- Focus on Spark performance profiling and optimization for Intel Architecture.

Outlines

- Compression Needs & Motivations
- Data Compression Pipelines in Spark
- Experiment Compression Codecs Intros
- Intel® Codec Accelerator Architecture Overview
- Takeaways
- Future Works

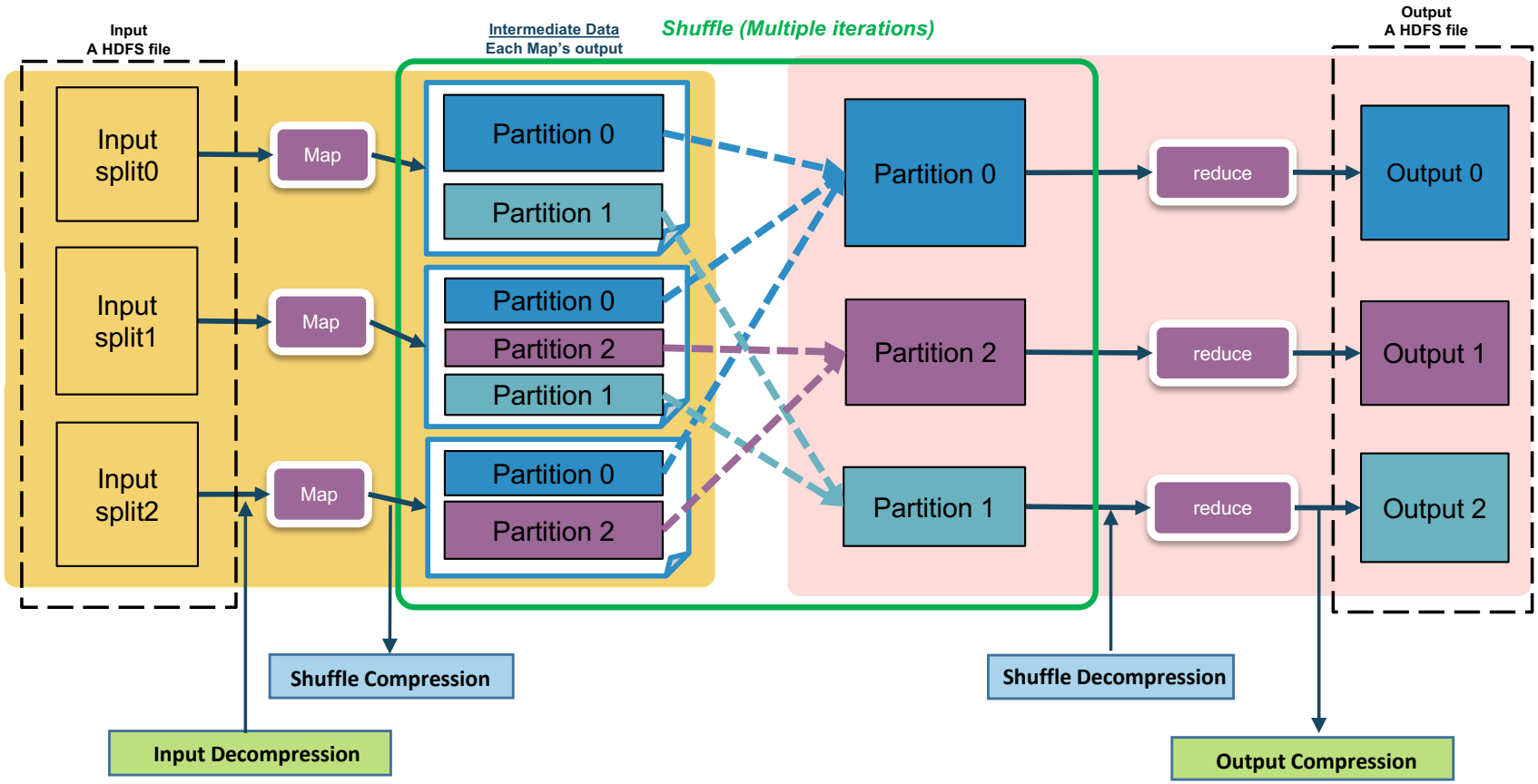
Compression Needs

- **Compression Needs**
  - Reduce data volume and save storage space.
  - **Speed up the disk I/O operations** and **data transfer** across network, optimize workload performance.
- **Trade-off**
  - Computation overhead for high compression ratio codecs.

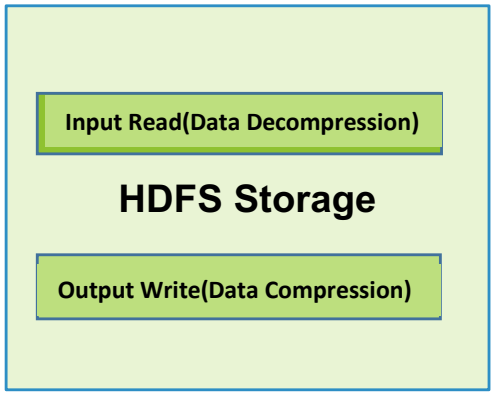
Motivations

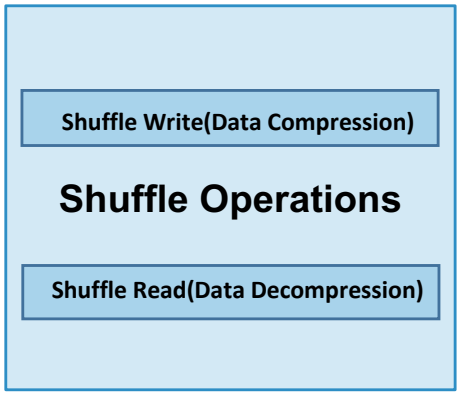
- Understanding popular compression codecs in Spark.
- Take advantage of **Intel® optimized libraries** or **accelerate hardware** for data compression/decompression.

Data Compression Pipeline in Spark



Data Compression Pipeline in Spark - I/O Characteristics





- HDFS Storage
  - Generally sequence read/write
  - Generally one time read/write
- Shuffle Operations
  - Random read/write
  - Multiple times read/write

Experiment Compression Codecs Intros

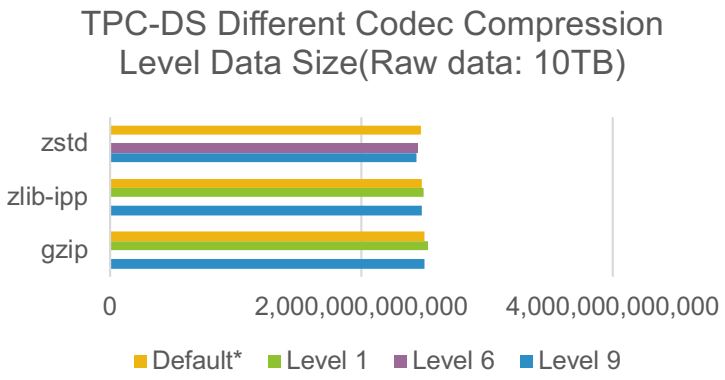
Codecs	Supported levels	Default level	Degree of Compression	Compression speed	CPU Usage	Comments
ISA-L(igzip)	(0~1)	1	Medium	Medium	Medium~High	Based on Intel® ISA-L ver 2.0.19 optimization
Zlib-ipp	(1~9)	Best balance(near to 6)	High	Slow	High	Based on Intel® IPP library optimization
Zlib/gzip	(1~9)	Best balance(near to 6)	High	Slow	High	Open source codec
zstd	1~22	3	High	Medium	Medium~High	Open source codec
Lz4-ipp	N/A	N/A	Medium	Fast	Low	Based on Intel® IPP library optimization
Lz4	Lz4 fast Lz4 hc	Lz4 fast	Low Medium	Fast Low	Low Medium	Open source codec
snappy	N/A	N/A	Low	Fast	Low	Open source codec

High compression ratio codecs

High throughput codecs

Intel® ISA-L reference: https://software.intel.com/en-us/storage/ISA-L ; Intel® IPP reference: https://software.intel.com/en-us/intel-ipp

Compression Level

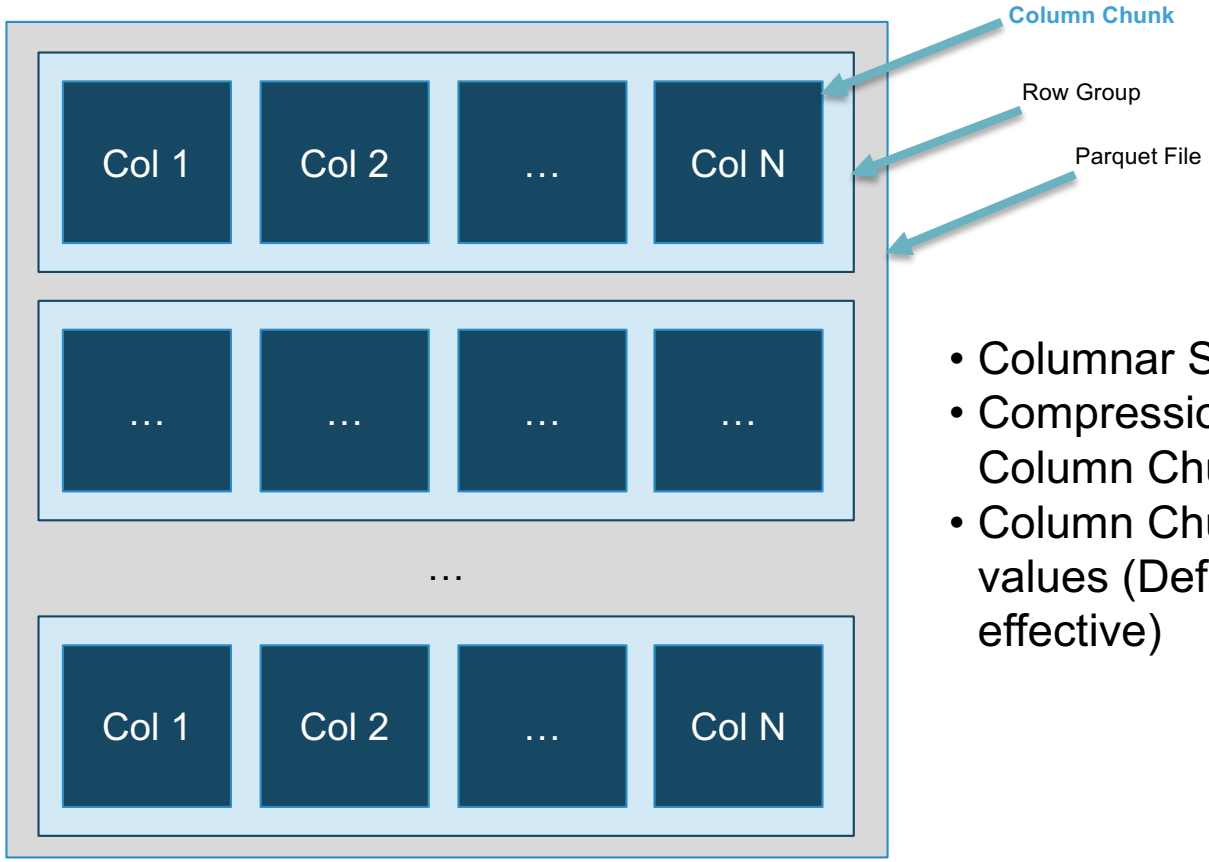


Compression codec	*Default level Data Size	Level6 Data Size	Level9 Data Size	Default Vs Level6	Default Vs Level 9
zstd	2,472,315,429,619	2,446,857,474,146	2,440,389,051,782	**1.04%**	**1.31%**

Compression codec	Level9 Data Size	Level1 Data Size	*Default level Data Size	Default Vs Level9	Level1 Vs Level9
gzip/zlib	2,500,252,836,007	2,528,269,315,543	2,502,656,222,082	**0.096%**	**1.12%**
zlib-ipp	2,482,050,449,516	2,492,687,484,854	2,482,595,509,721	**0.022%**	**0.429%**

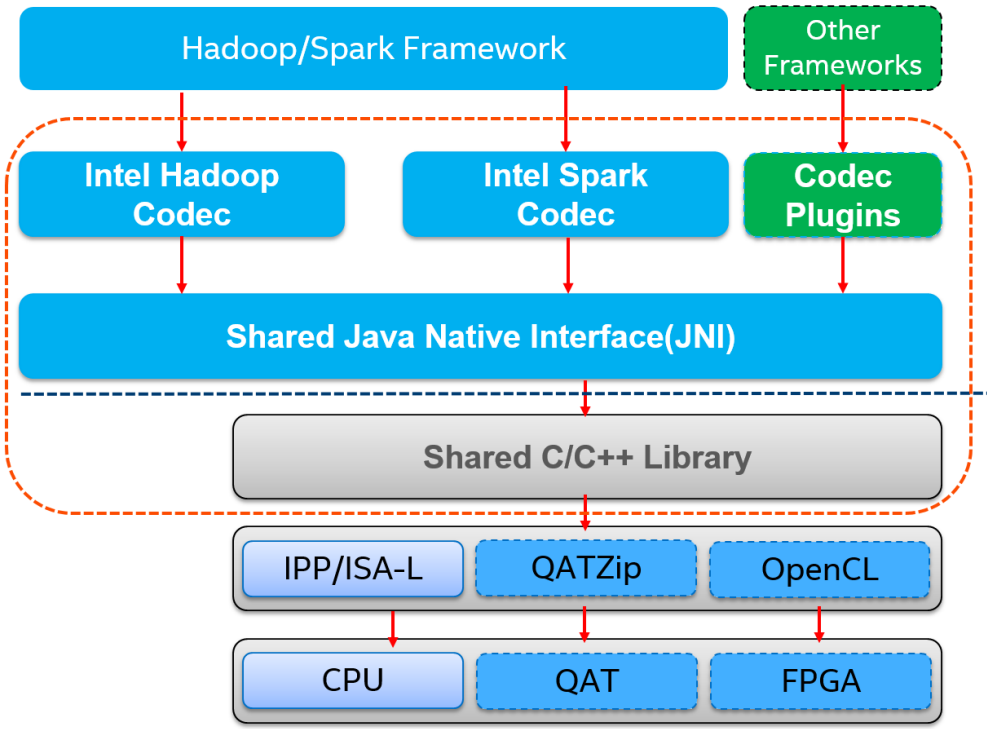
- zstd, gzip, zlib-ipp and igzip support compression level adjustment, while codec lz4 and snappy does not support.
- No big data size difference among different compression level in TPC-DS parquet format data generation test.

Compression in Parquet Format



- Columnar Storage (For Column Pruning)
- Compression / Decompression for each Column Chunk
- Column Chunk has same data type even same values (Default Compression Level is usually effective)

Intel® Codec Accelerator Architecture(1/2)



JAVA Application:
TeraSort, WordCount, PageRank etc.

Extend class from Hadoop and Spark CompressionCodec, configurable, implements compression codec interfaces of Hadoop/Spark And extend class from Hadoop Compressor, Decompressor, implements interface of Hadoop

Forward Hadoop/Spark Compress/decompress request from JAVA to under layer C/C++

Codec Adapter Layer, Self adaption to codec according to configuration from JAVA

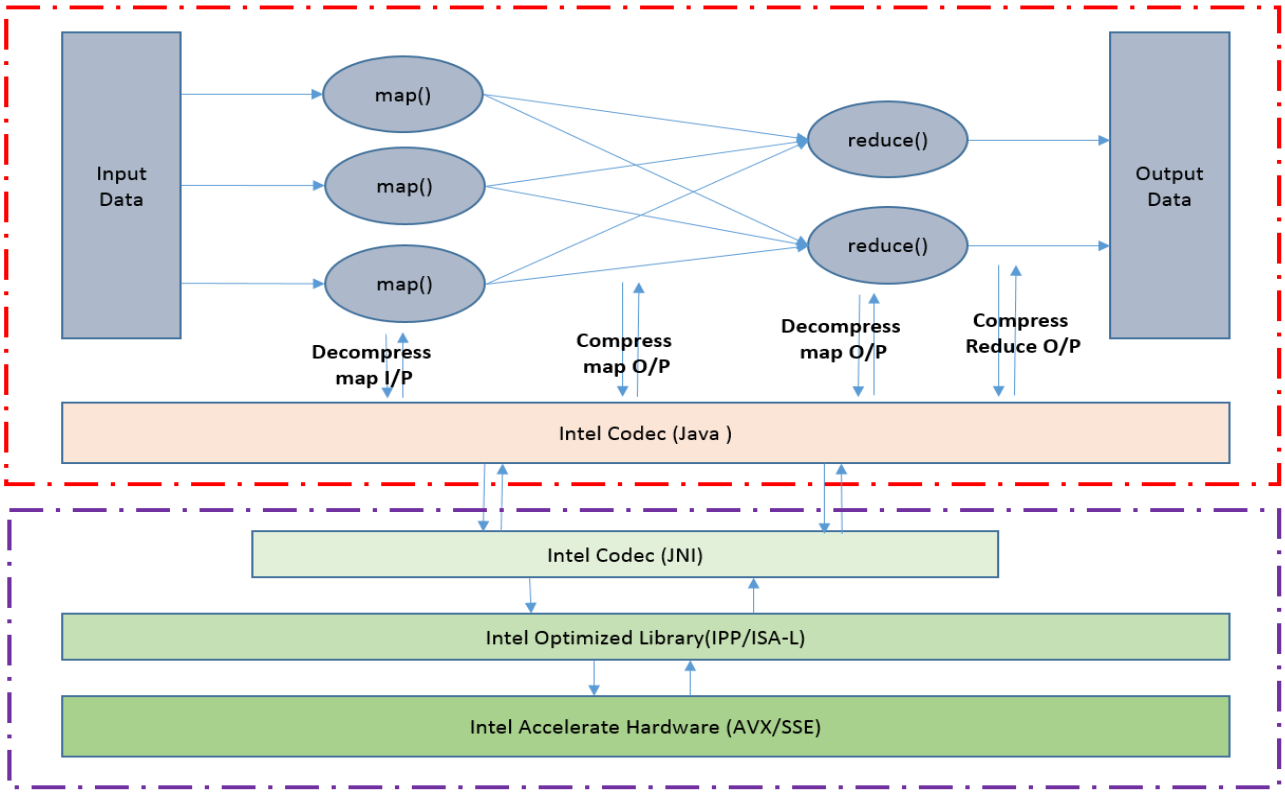
Intel Optimized Accelerate Library Using AVX-512, QAT, FPGA

Accelerate Hardware

Notes:

- QAT and ISA-L AVX512 is available on Intel® Skylake-X platform.
- Open Source codec zstd also can build with Intel® ISA-L AVX512 support to accelerate data compression/decompression.

Intel® Codec Accelerator Architecture(2/2)



User Layer

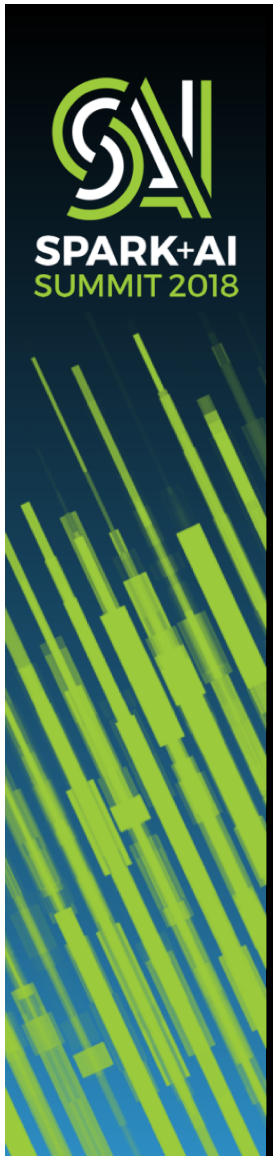
Native Layer

Takeaways

- Better to choose high compression codecs for source data* for IO intensive workload, such as zstd, zlib-ipp, zlib, igzip.
- Better to use high throughput codecs for spark shuffle compression codec, such as lz4-ipp, lz4.
- Higher compression codec reduce I/O and network pressure, but consumes CPU resource, use accelerate hardware such as QAT and FPGA can help to offload CPU resources.
- Zstd can qualify as both a reasonably strong compressor and a fast one.
- Best balance of compression codec depends on cluster characteristics and workloads.

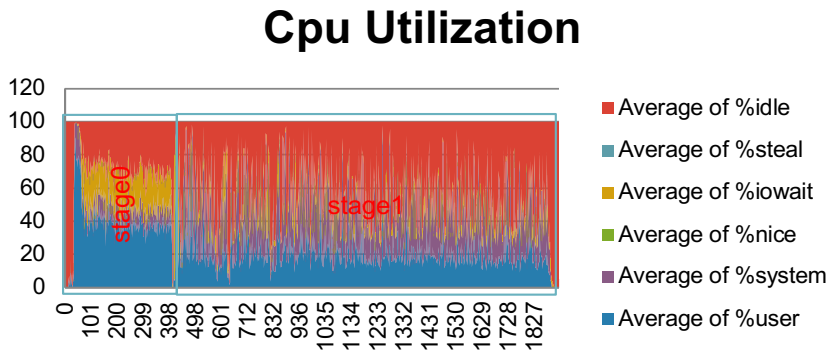
Future Plan

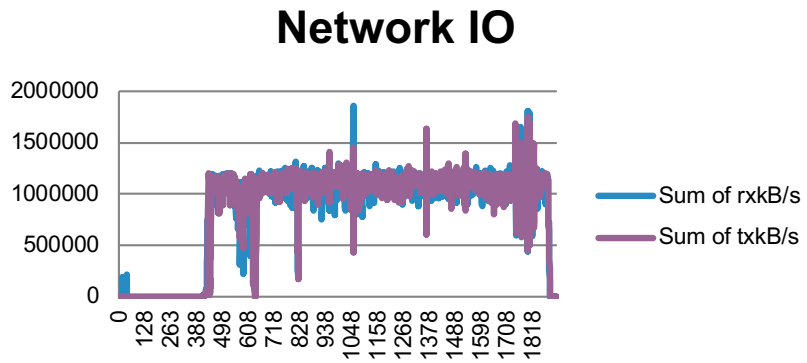
- Open source Intel® Codec Accelerator project and make it as well supported library.
- Add codec compatibility support.
- Integrate with more IA optimized codecs along with the acceleration library releases under different platform.
- Introduce more big data frameworks (Cassandra / HBase etc.)
- Besides compression / decompression, we will support more types of codec like the encryption / decryption etc.
- Keep release new version along with new Intel® Platform release or new acceleration libraries released.



Thanks!

HiBench Sort Workload bottleneck – No data compression

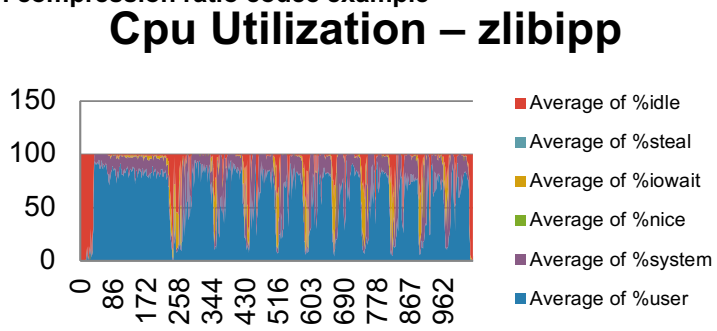




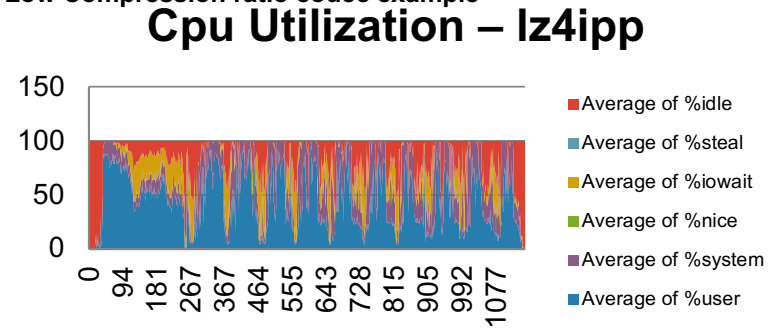
- No compression data has big data size, mapping data make the IO disk as bottleneck in stage0
- No compression data cause big pressure in shuffle stage(Stage1). 10Gb(~1.2GB) network as bottleneck in experiment environment. While CPU still has much idle resource.

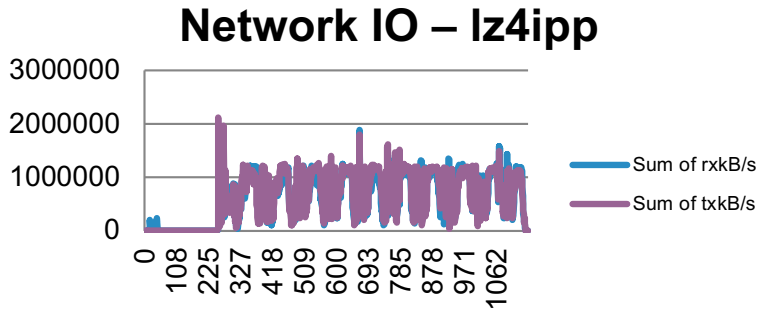
HiBench Sort Workload Resource Utilization Examples

High compression ratio codec example



Low Compression ratio codec example





- CPU as bottleneck on High compression ratio codecs (like zstd, zlibipp and igzip)
- Codec lz4, lz4ipp and snappy has lower compression ratio, large size of data read/write caused the disk as the bottleneck in stage0 and large shuffle data caused network as bottleneck in stage1