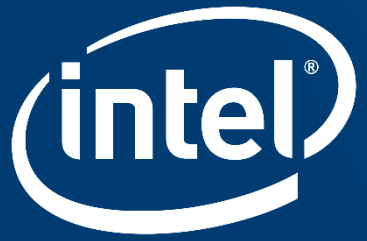


COMPRESSION CODEC IN SPARK*

SSG / DPD / BDT

Sophia Sun (sophia.sun@intel.com)

Qi Xie (qi.xie@intel.com)

Hao Cheng (hao.cheng@intel.com)

Outlines

Background

Micro-benchmark Result

TPC-DS* benchmark Result

HiBench* benchmark Result

Intel® Compression Codec Architecture

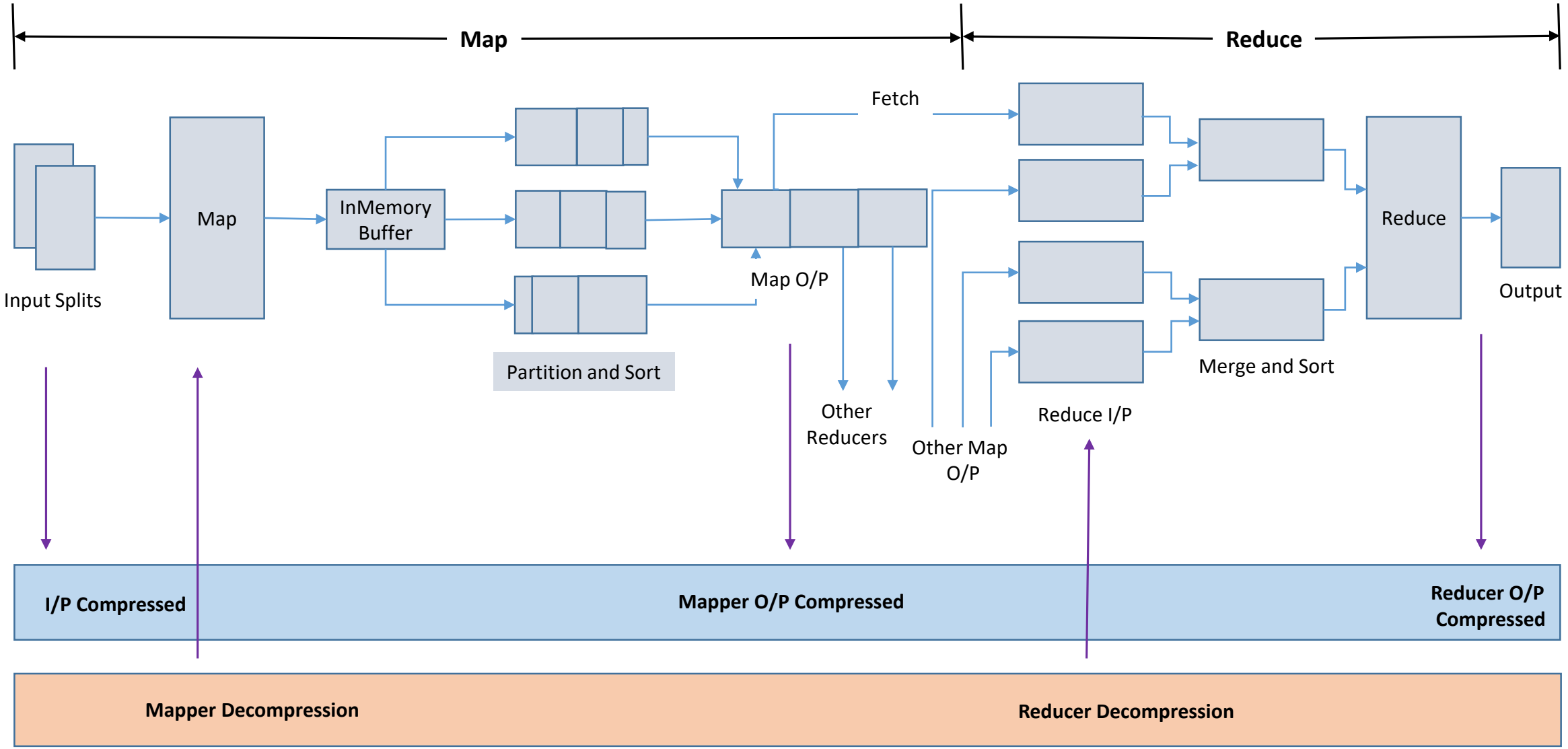
Compression Needs and Trade-offs in Spark*

- Why we need data compression?
  - Save storage space.
  - In data intensive workload, disk and network may as bottleneck. Compressing data can **speed up the I/O operations** and **data transfer** across network.
  - Spark Shuffle process has huge I/O pressure as it has to often "spill out" intermediate data to local disks before advancing from Map stage to Reduce stage.
- Data Compression trade-offs
  - Although compression could reduce I/O and network pressure, CPU utilization generally increases as data must be decompressed before the files can be processed.

Compression Codecs Info

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 optimization
Zlib-ipp	(1~9)	Best balance(near to 6)	High	Slow	High	Based on Intel® IPP library optimization
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
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

Data Compression Pipeline in Spark



Outlines

Micro-benchmark Result

TPC-DS benchmark Result

HiBench benchmark Result

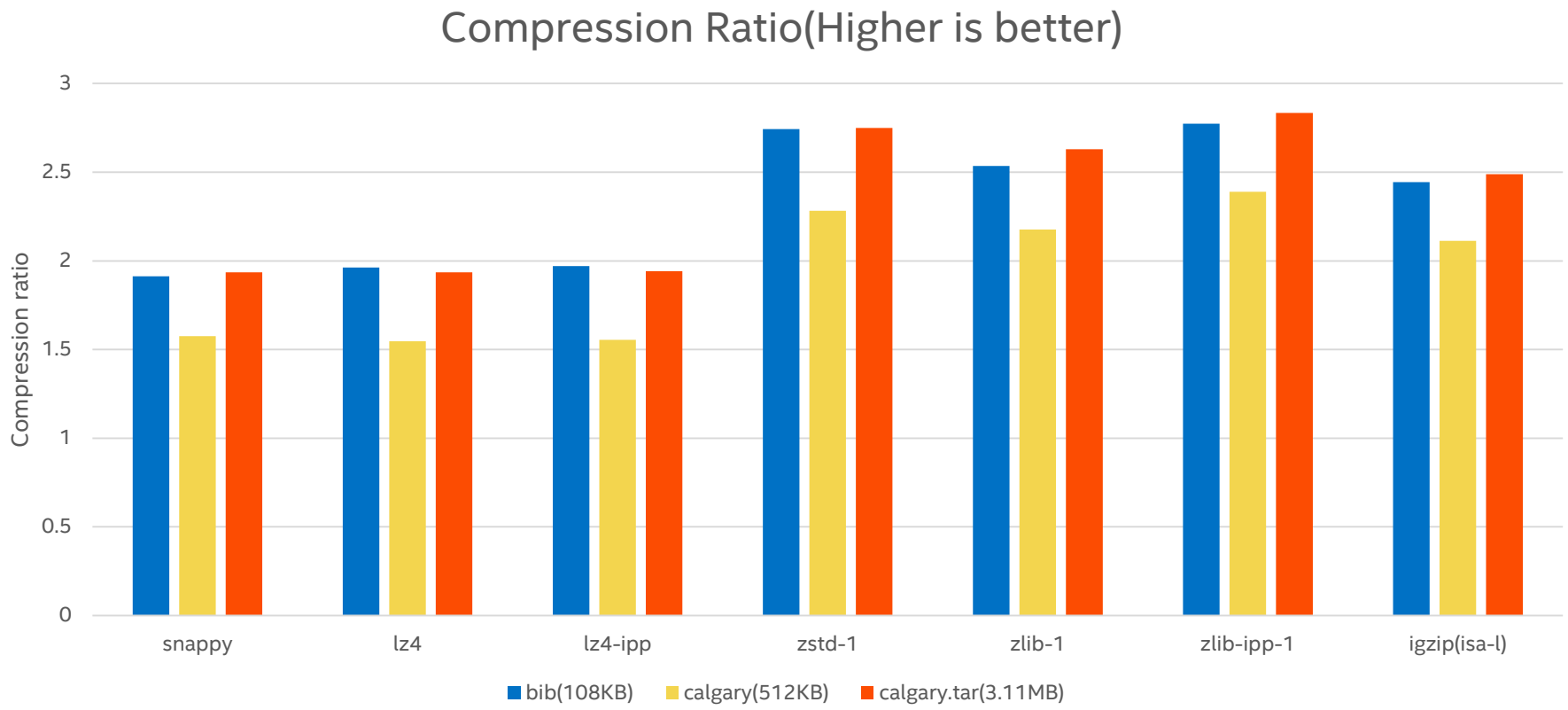
Intel Compression Codec Architecture

Hardware Configurations

Hardware	
Platform	**Intel ® Xeon® E5 V4 Platform**
CPU	Intel ®Xeon(R) CPU E5-2699 v4 @2.2 GHz 22 cores, 44 threads * 2 sockets = 88 vCores
Memory	DDR4 380GB
Storage	Intel® SSD S3700 Series ,400GB X 8 Disks
Network	1Gb
Software	
OS	CentOS* Linux release 7.2.1511 Kernel version: 3.10.0-327.el7.x86_64
Java	Java(TM)* SE Runtime Environment (1.8.0_40)

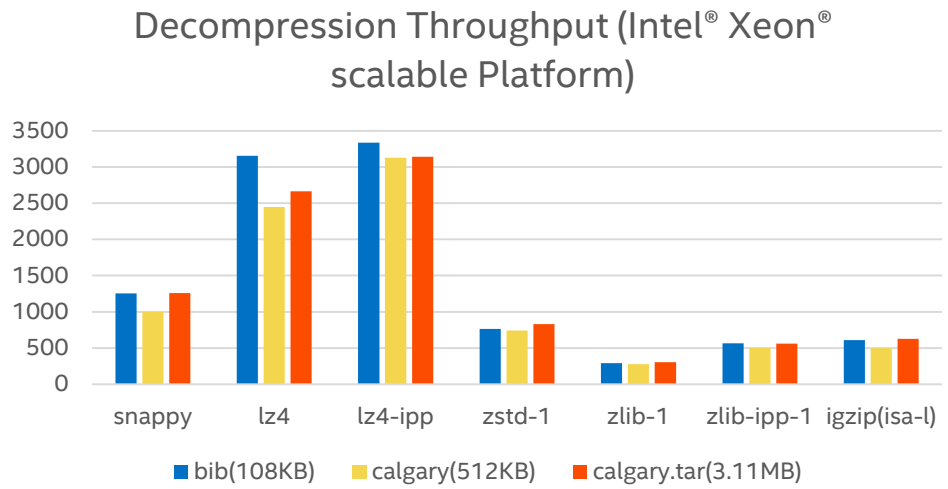
Hardware	
Platform	**Intel® Xeon® scalable Platform**
CPU	Intel® Xeon® 8180@2.5GHz 28 cores, 56 threads * 2 sockets = 112 vCores
Memory	DDR4 128GB
Storage	Intel® SSD S3700 Series , 400GB X 8 Disks
Network	1Gb
Software	
OS	CentOS* Linux release 7.2.1511 Kernel version: 3.10.0-327.el7.x86_64
Java	Java(TM)* SE Runtime Environment (1.8.0_40)

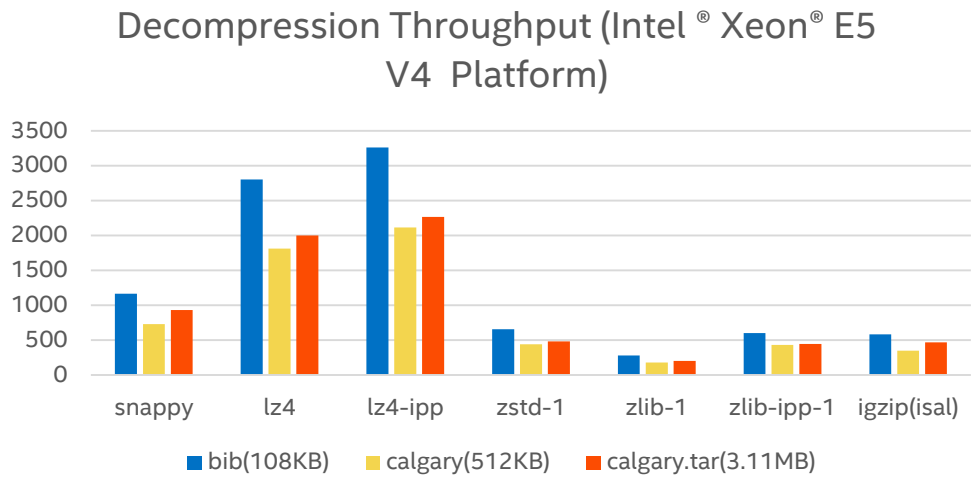
Compression Ratio



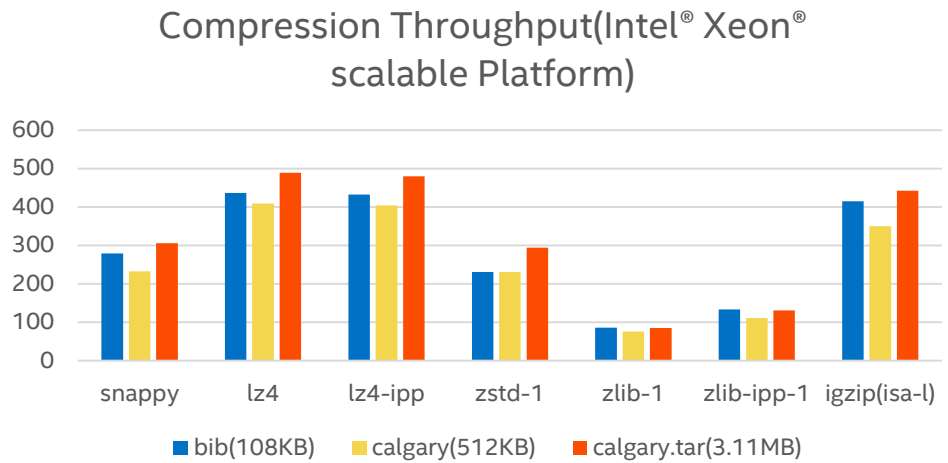
Codec zstd, zlib, zlib-ipp and igzip(isa-l) has higher compression ratio.

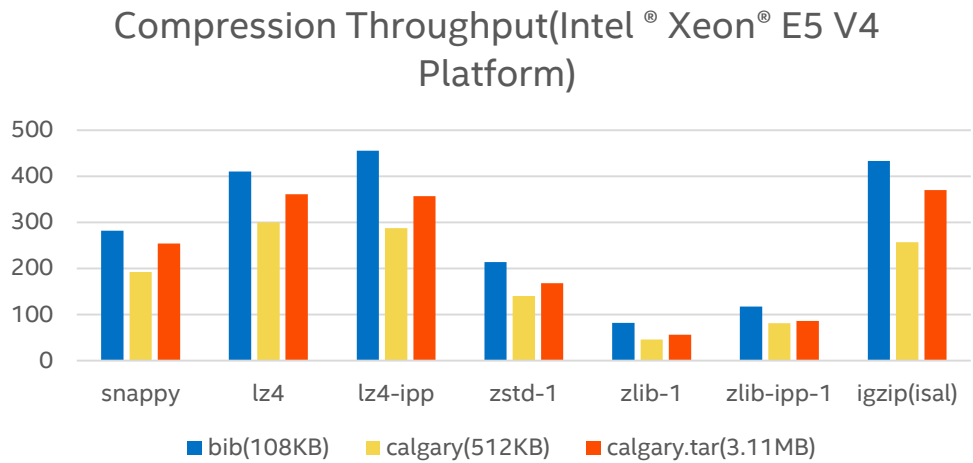
Native Codec Throughput





Codec LZ4, LZ4-IPP has high Decompression and Compression throughput!!





Codec igzip has high Compression ratio and relatively High compression throughput!

Outlines

Micro-benchmark Result

TPC-DS benchmark Result

HiBench benchmark Result

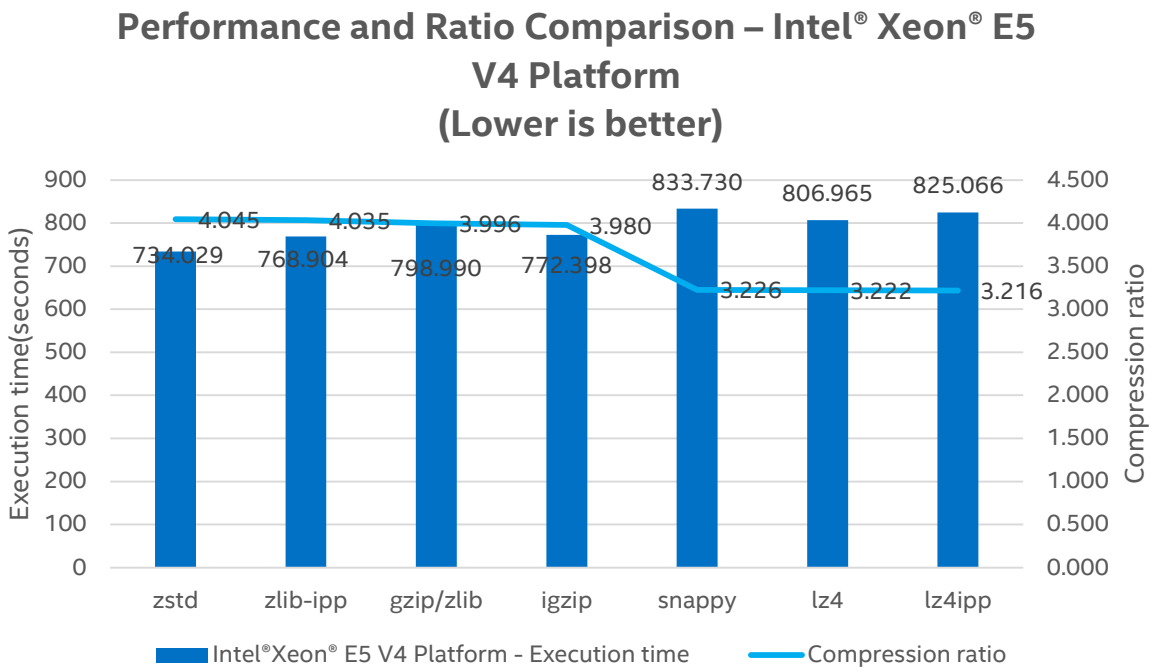
Intel Compression Codec Architecture

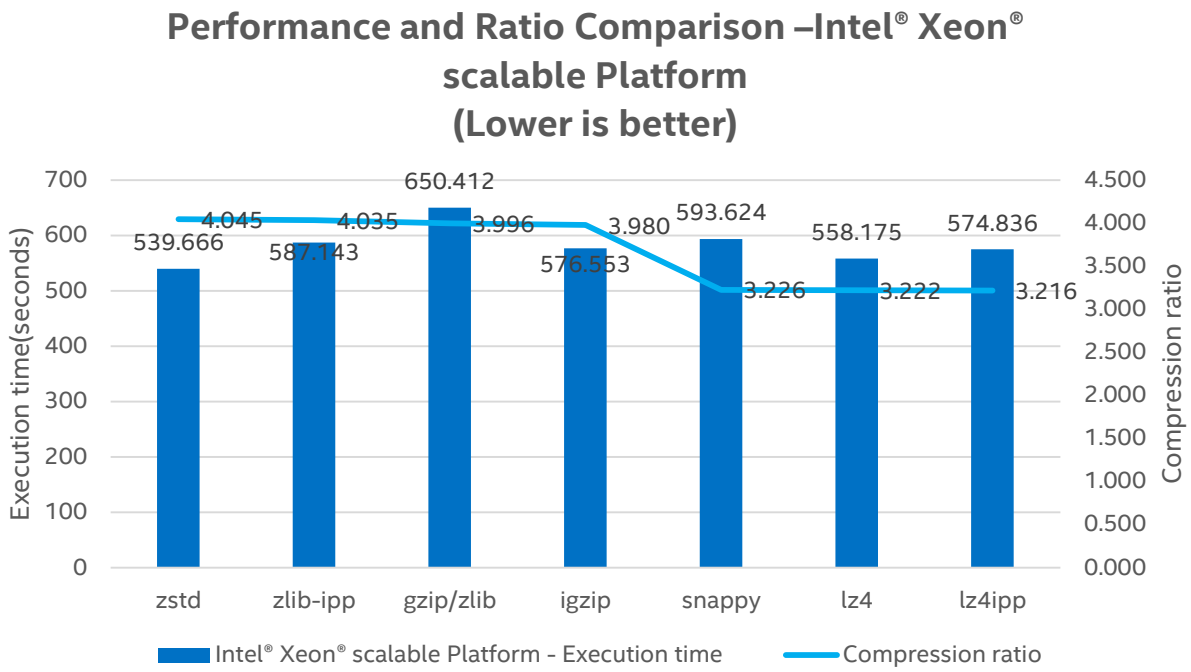
Hardware Configurations

Platform	Intel ® Xeon® E5 V4 Platform Cluster	Intel® Xeon® scalable Platform Cluster
# of Nodes	8	8
# of Master Nodes	1	1
# of Worker Nodes	7	7
Worker node CPU	Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz 14 cores, 28 threads * 2 sockets = 56 vCores	Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz 18 cores, 36 threads * 2 sockets = 72 vCores
Storage	Intel® SSD(S3520), 1.2TB X 8 Disks per node	Intel® SSD(S3520), 1.2TB X 8 Disks per node
Memory	384GB per node	384GB per node
Network	10Gb	10Gb
OS Version	CentOS* 7.3.1611	CentOS* 7.3.1611
Hadoop	Apache Hadoop* 2.7.3	Apache Hadoop* 2.7.3
Spark	Apache Spark* 2.1.0	Apache Spark* 2.1.0
JDK	OpenJDK 1.8.0_112	OpenJDK 1.8.0_112

Workload (TPC-DS)	
Queries	19,42,43,52,55,63,68,72,98
Data Scale (Raw Data)	10 TB
Data Format	Parquet
Compression Codec	gzip, zlib-ipp, igzip(ISA-L),zstd, Snappy, lz4, lz4-ipp

TPC-DS(Subset) Throughput Benchmark Result





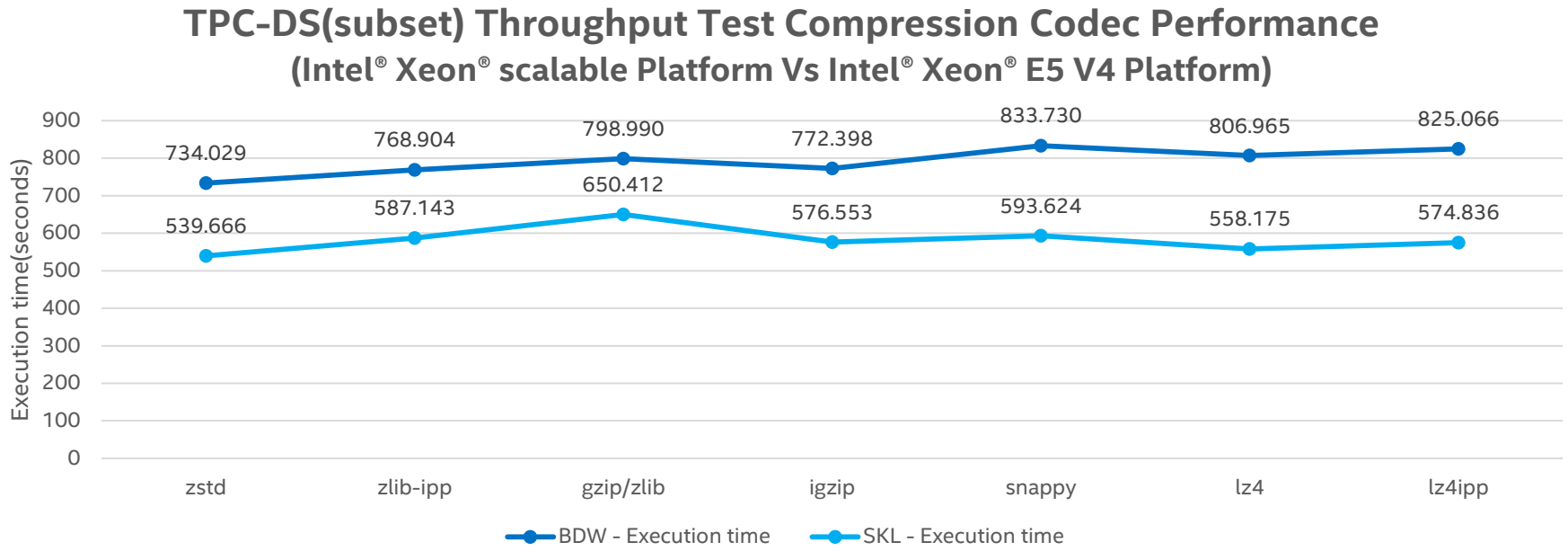
High compression ratio codec has better performance on Intel(R) Xeon(R) CPU E5 V4 platform .

- **Zstd > zlib-ipp, igzip > gzip > lz4 > lz4-ipp > snappy**

On Intel® Xeon® scalable Platform

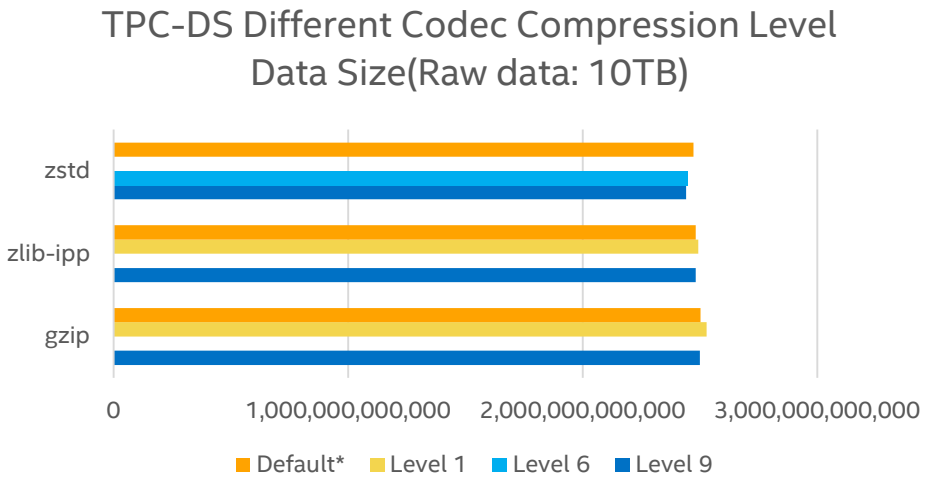
- ***Zstd > lz4 > lz4-ipp, igzip > zlib-ipp > snappy > gzip***

TPC-DS(Subset) Benchmark Summary



- Intel® Xeon® scalable Platform has average ~36.05% performance boost compared to Intel® Xeon® E5 V4 Platform with all compression codecs.
  - Performance boost on Intel® Xeon® scalable Platform benefit from more CPU cores.

TPC-DS(Subset) Benchmark Result – 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%**

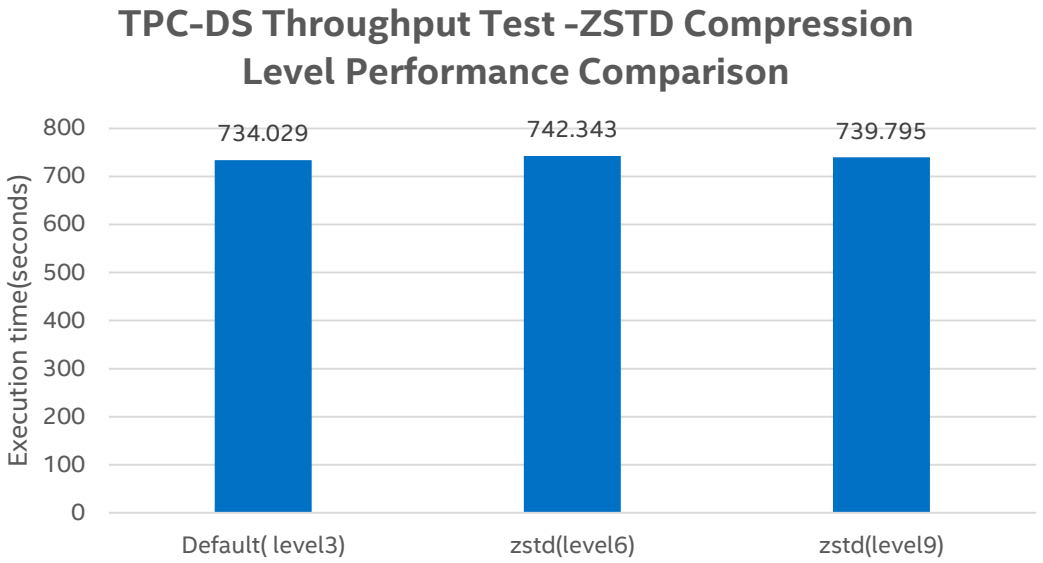
Compression codec	Level9 Data Size	Level1 Data Size	*Default level Data Size	Default Vs Level9	Level1 Vs Level9
gzip	2,500,252,836,007	2,528,269,315,543	2,502,656,222,082	**0.096%**	**1.12%**
zlib-ipp	2,482,050,449,516	2,492,687,484,854	2,482,595,509,721	**0.022%**	**0.429%**

- Codec zstd, gzip/zlib, zlib-ipp and igzip support compression level adjustment, codec lz4 and snappy do not support compression level adjustment.
- No big data size difference among different compression level in TPC-DS data generation. For instance, the gip/zlib, zlib-ipp only has 0.01% data between default level(6) and Higher compression level 9.

*In gzip/zlib and zib-Ipp codec, the default compression level is close to level 6
*In zstd codec, the default compression is Level 3
*In igzip codec, the default compression level is 1, we didn't try to test level 0 since Level 1 has higher compression ratio.

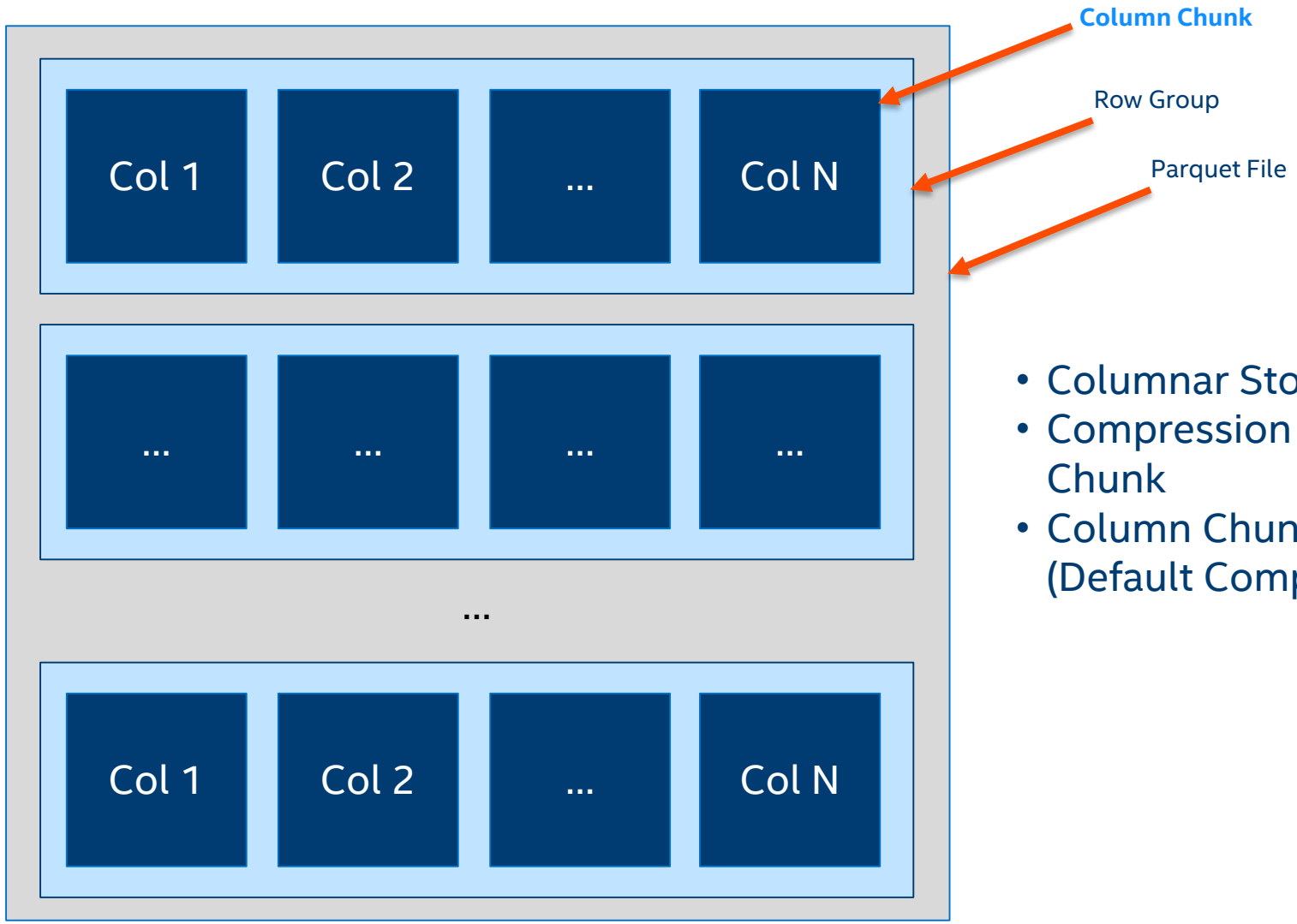
TPC-DS*(Subset) Benchmark Result – Compression level

- No obvious data size reduction or performance gain by setting the higher compression level on codec zstd, the root reason probably due to Parquet(Spark SQL) codec is very effective in size for the structured data



*In gzip/zlib and zib-Ipp codec, the default compression level is close to level 6
*In zstd codec, the default compression is Level 3

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)

TPC-DS(Subset) Benchmark Summary

- Performance results related with compression ratio - High compression ratio codec has higher performance on Intel® Xeon® E5 V4 Platform.
- More CPU cores brings more benefits for high throughput compression codecs(lz4, lz4-ipp) on Intel® Xeon® scalable Platform.
- Intel® Xeon® scalable Platform has average ~36.05% performance boost compared to Intel Xeon® E5 V4 Platform with all tested compression codecs.

Outlines

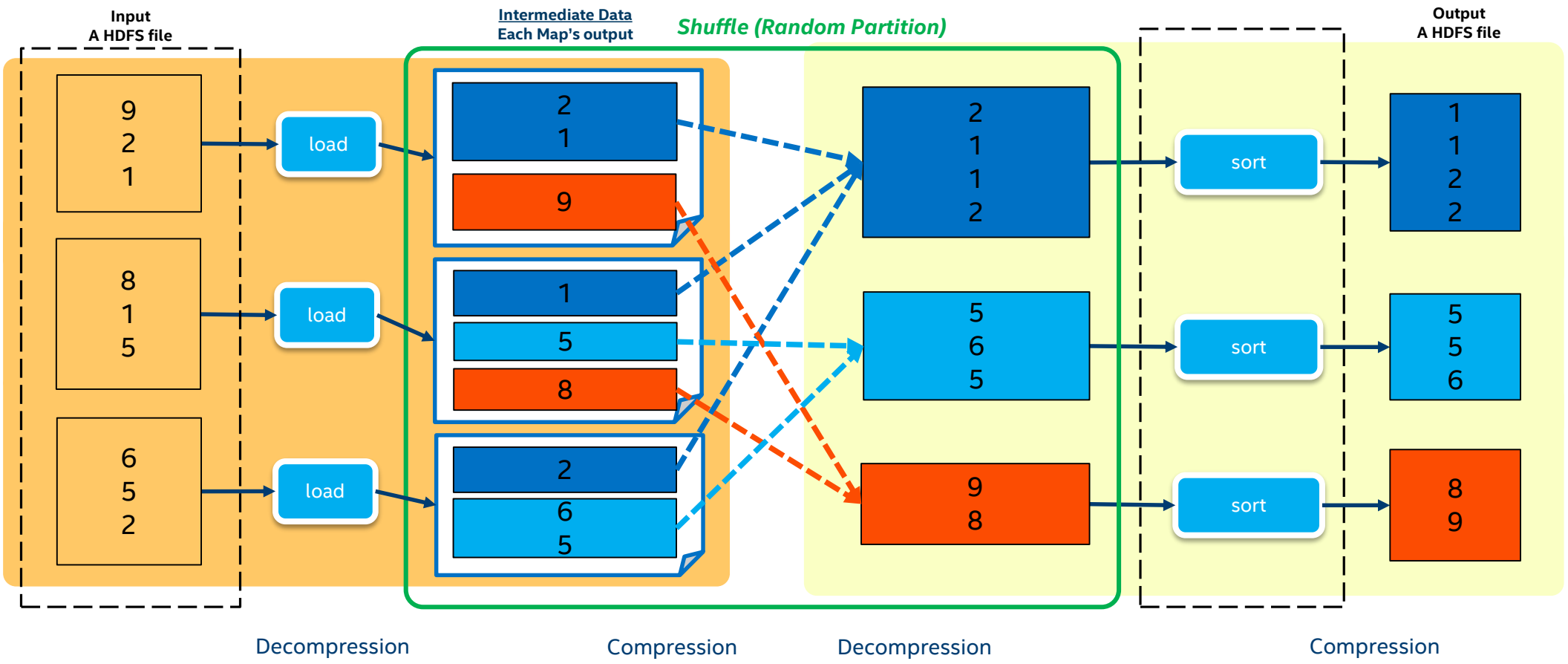
Micro-benchmark Result

TPC-DS benchmark Result

HiBench Sort benchmark Result

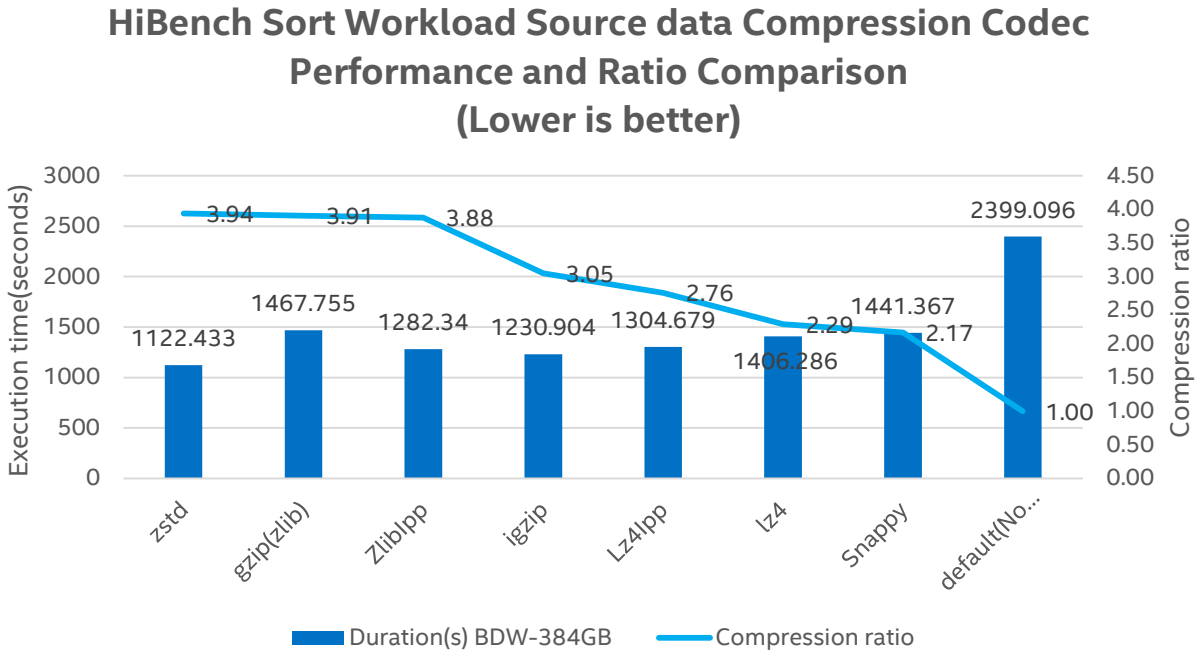
Intel Compression Codec Architecture

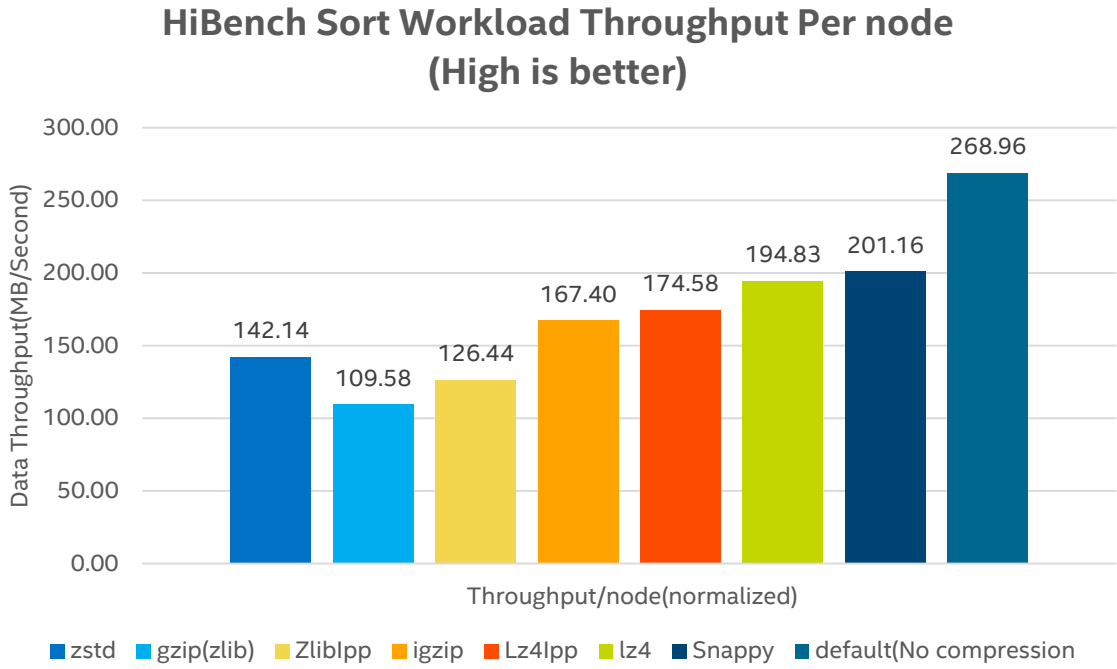
HiBench* Sort Workload Intro



https://github.com/intel-hadoop/HiBench/blob/master/sparkbench/micro/src/main/scala/com/intel/sparkbench/micro/ScalaSort.scala

HiBench Sort Workload Benchmark Result – Intel® Xeon® E5 V4 Platform



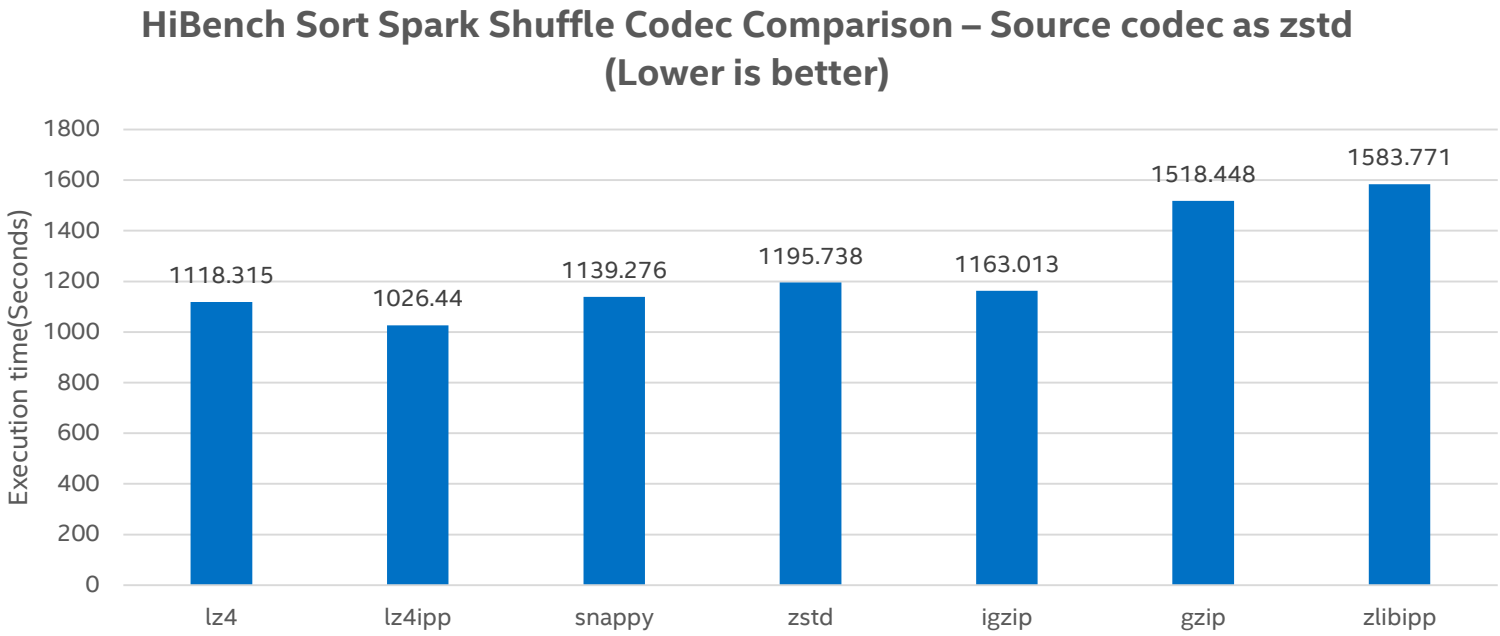


- Zstd has the best compression ratio with acceptable compression throughput. zstd > gzip(zlib), zlibIpp > igzip > lz4Ipp > lz4 > snappy > default (No compression).

- Zstd has best performance in HiBench Sort workload, it has ~81.7% performance gain compared with no compression mode. (zstd > igzip > zlibIpp > lz4ipp > lz4 > snappy> gzip > No compression)

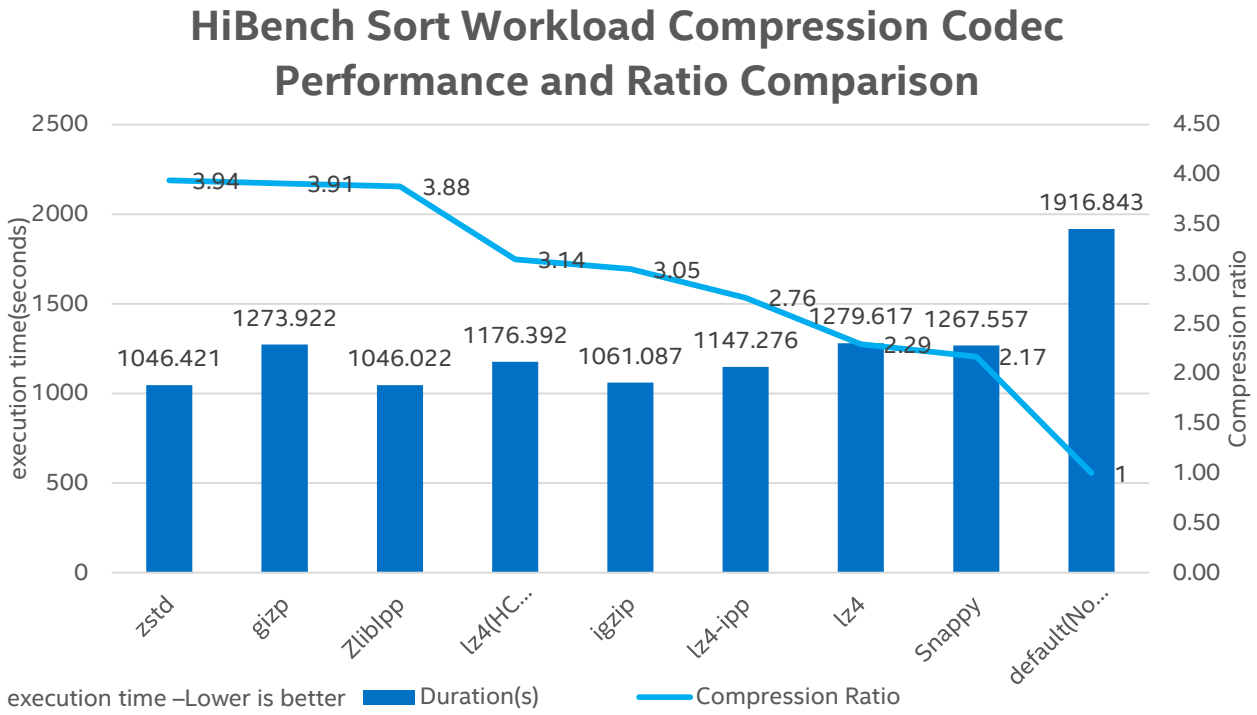
use lz4 as Spark Shuffle codec

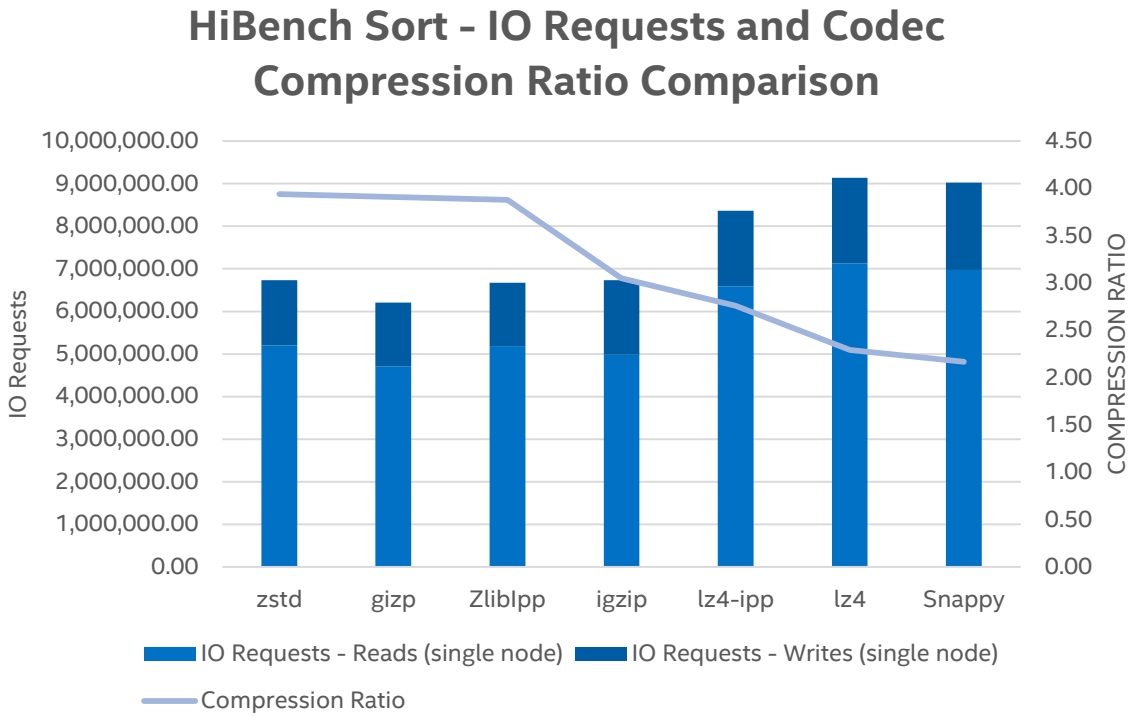
HiBench Sort – Shuffle Codec Comparison



- Codec lz4-ipp/lz4 has relatively better performance than other codecs when worked as spark shuffle codec.
- Codec lz4-ipp has 8.95% performance increase compared with default shuffle codec lz4.

HiBench Sort Workload Benchmark Result - Intel® Xeon® scalable Platform

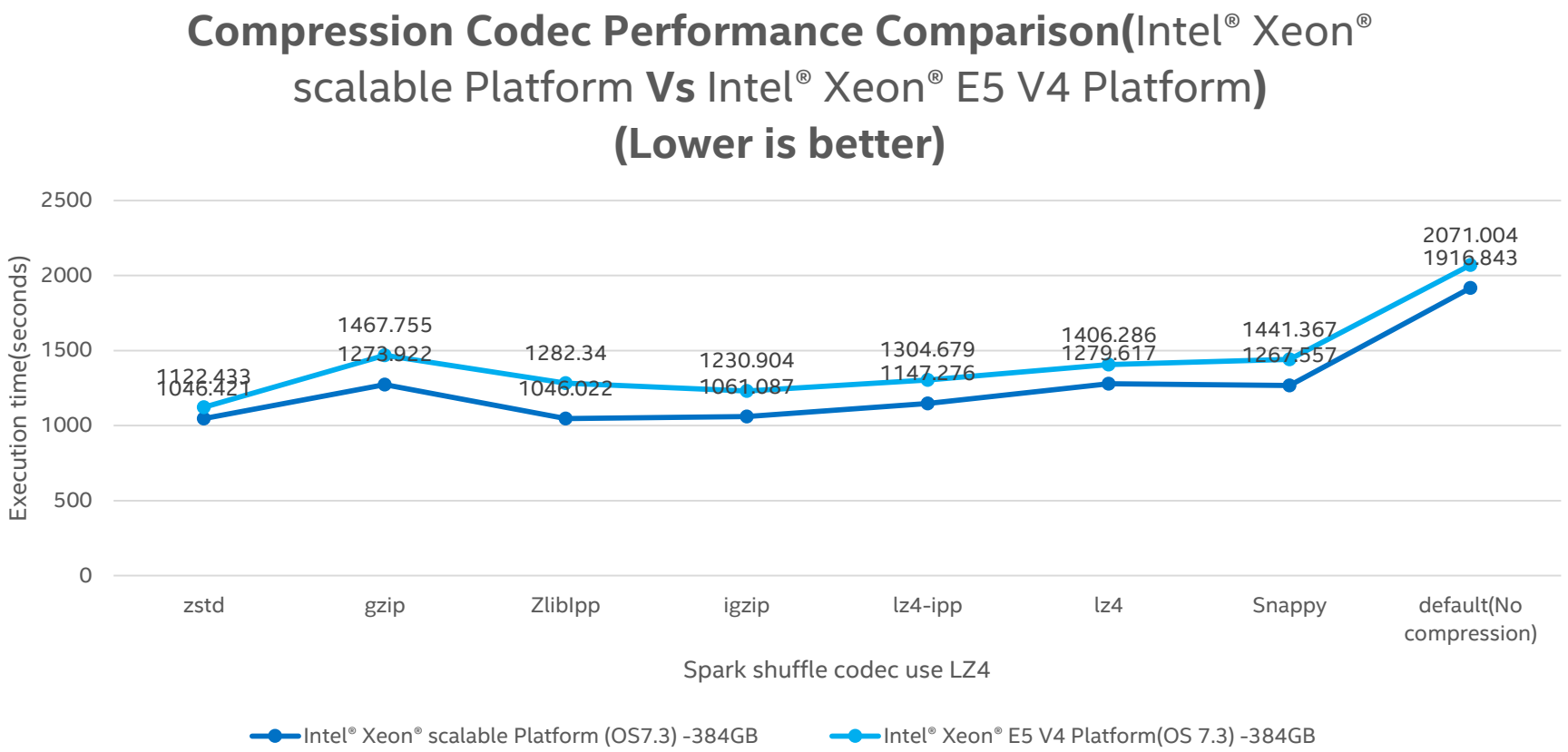




- zlib-ipp, zstd and ISA-L gzip has best performance in HiBench Sort workload on Intel® Xeon® scalable Platform, it has ~83.25% performance gain compared with no compression mode. (zlib-ipp, zstd, igzip > lz4-ipp > snappy, gzip/zlib, lz4 > default-No compression)

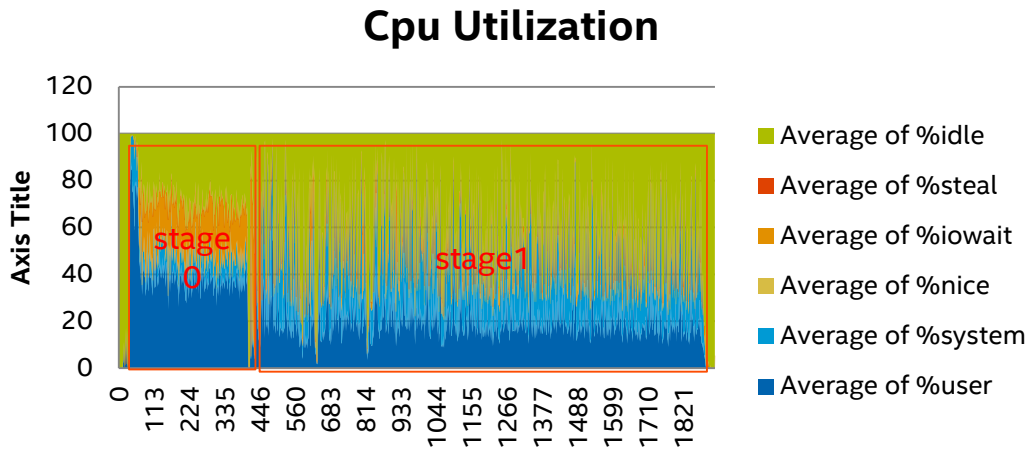
- High compression ratio codec has less IO. Better compression also reduces the bandwidth required to read the input.

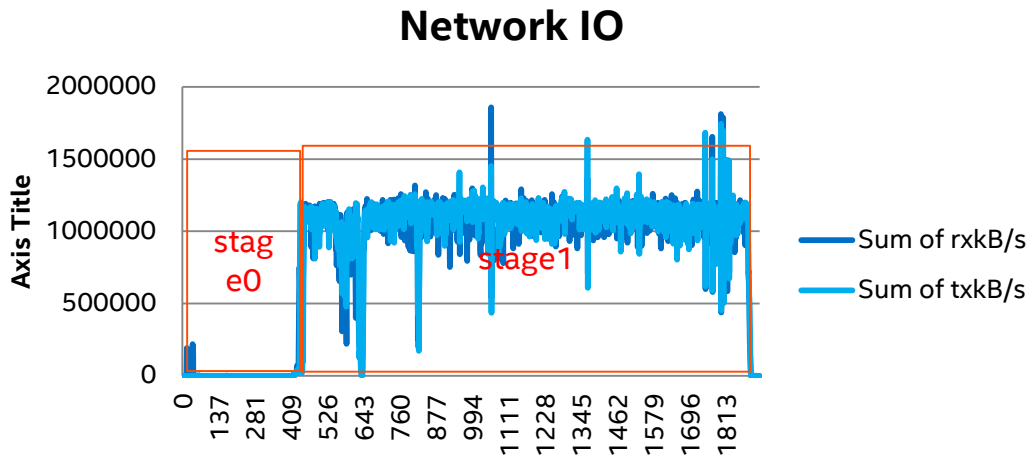
HiBench Sort Workload Benchmark Result



- Intel® Xeon® scalable Platform has average 13.1% performance gain. Since Network as bottleneck in shuffle stage1, there is no benefit on Xeon® scalable Platform. Take zstd codec as example, Intel Xeon® scalable Platform has 48.8% performance gain in stage0.

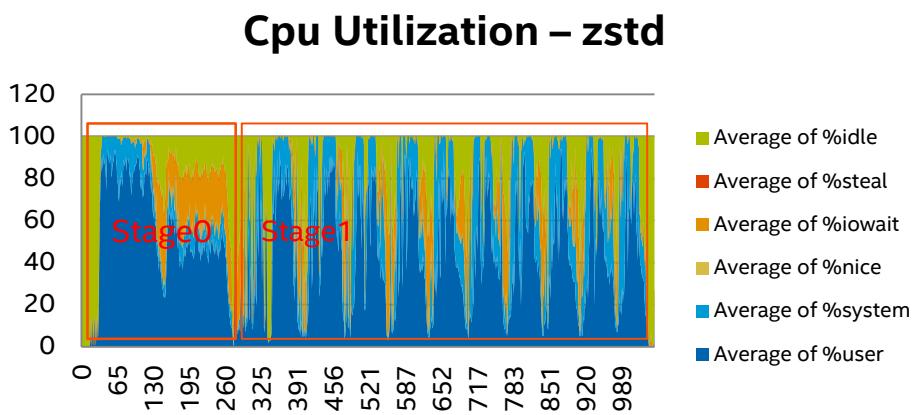
HiBench Sort Workload Resource Utilization – No compression data

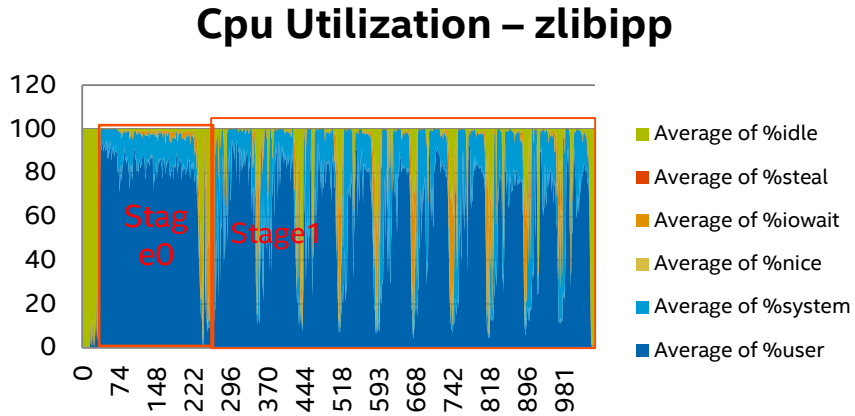


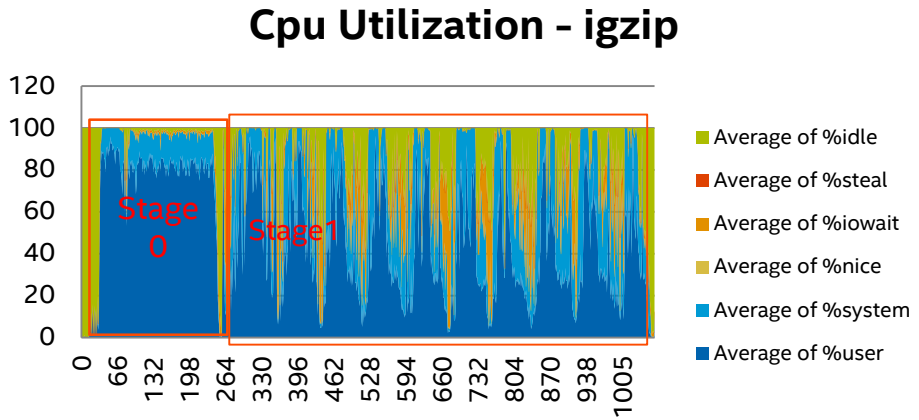


- No compression data has big data size, mapping data make the IO disk as bottleneck in stage0
- No compression data cause big pressure in stage1- shuffle stage. 10Gb(~1.2GB) network as bottleneck in experiment environment. While the average of CPU idle time account for ~42.66%.

HiBench Sort Workload Resource Utilization – High Compression ratio codec characteristics

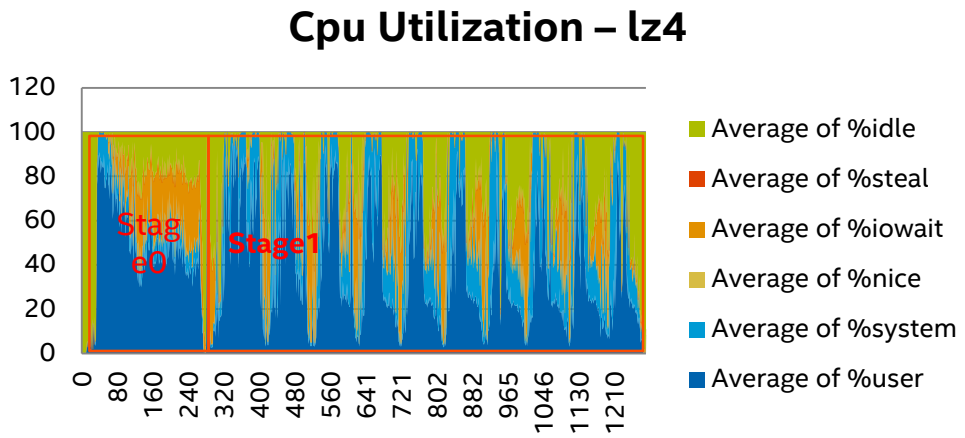


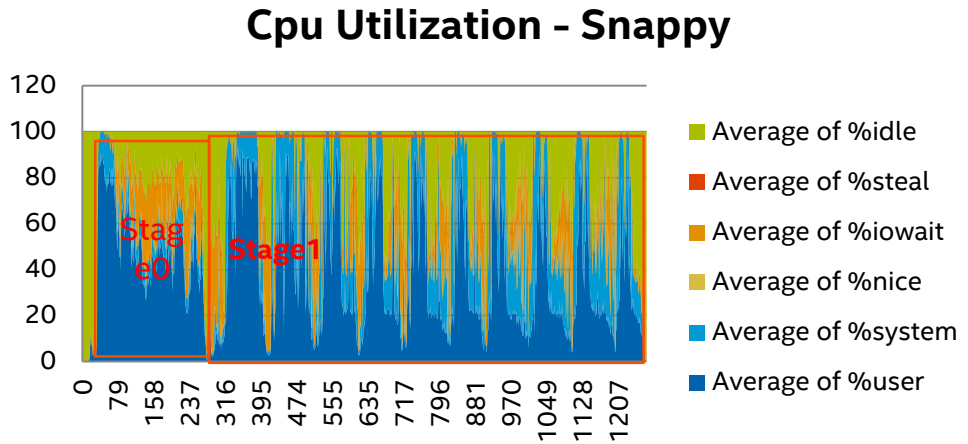


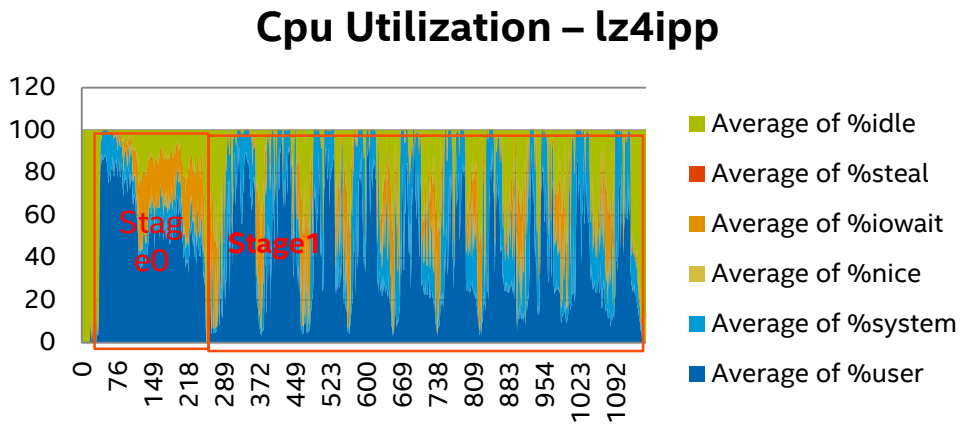


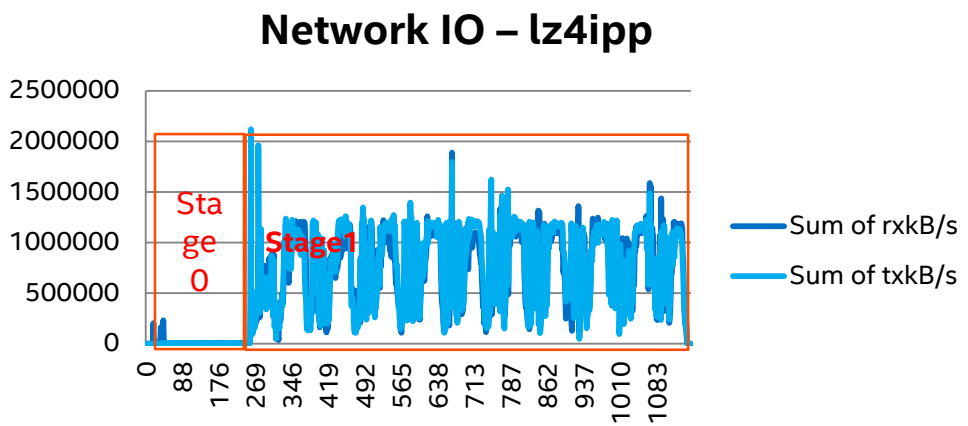
- CPU as bottleneck on High compression ratio codecs (like zstd, zlibipp and igzip)
- zlibipp, zstd and igzip has similar performance in Sort workload, but zstd consumes less CPU resources (**average CPU usage zlib-ipp > igzip > zstd**)

HiBench Sort Workload Resource Utilization – Low Compression ratio codec characteristics



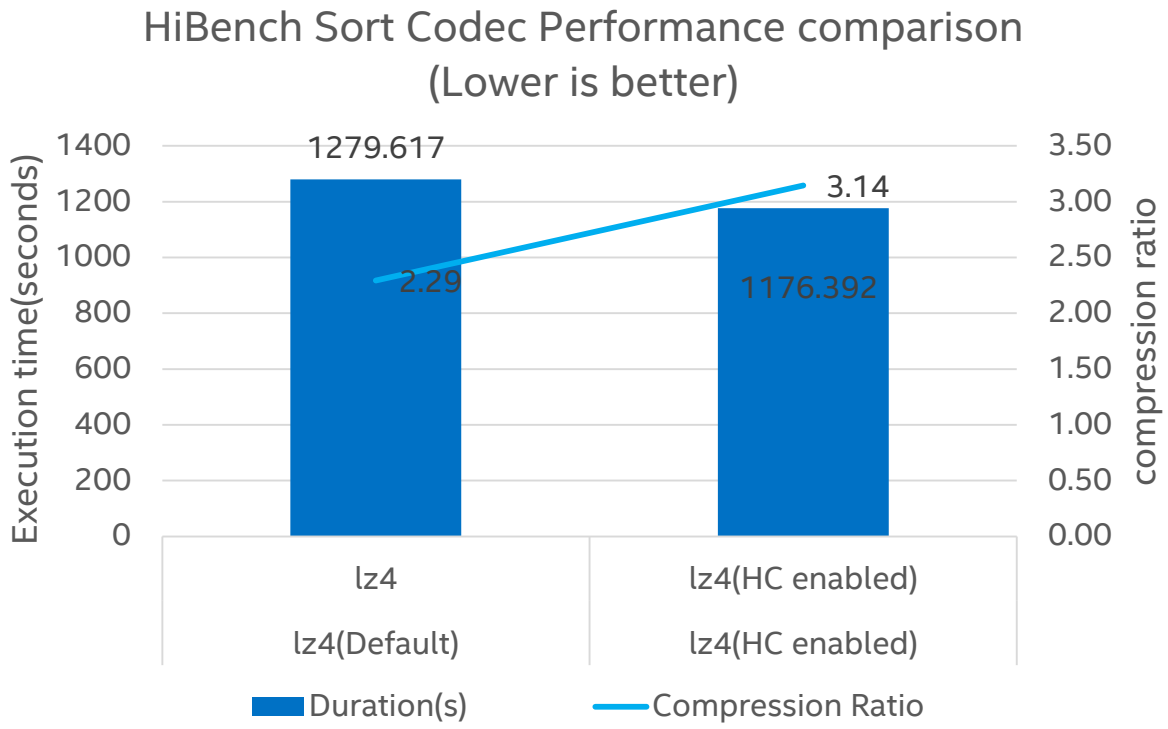






- Codec lz4, lz4ipp and snappy has lower compression ratio, large size of data read/write caused the disk as the bottleneck in stage0 and large shuffle data caused network as bottleneck in stage1.

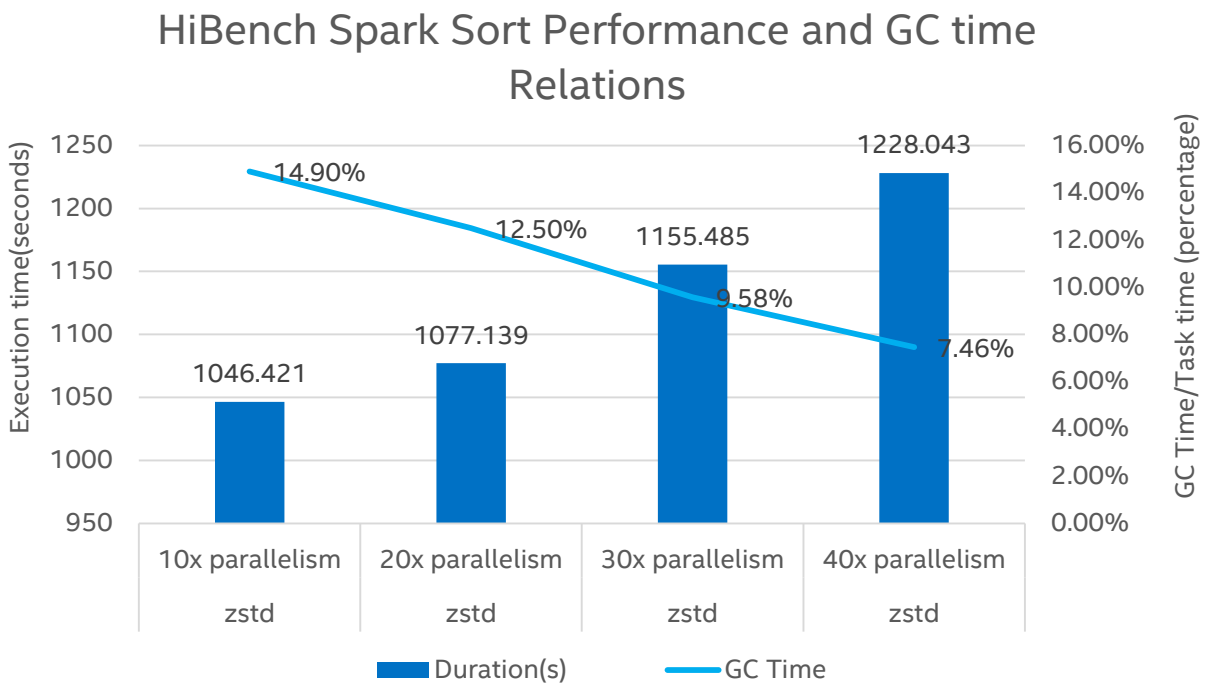
HiBench Sort – lz4 default Vs lz4 hc enabled



- lz4 hc(High compression) mode has higher compression ratio and has 8.77% performance increase compared to lz4 default fast mode.

HiBench Sort - balance the performance and GC time

- Parameter spark.sql.shuffle.partitions impact the performance result
  - Reduce shuffle partition number from 40X cluster parallelism to 10X cluster parallelism, the performance improved ~17.35%, but since more data in one tasks would cause the GC time increase.



HiBench Sort Workload Summary

- Zstd can qualify as both a reasonably strong compressor and a fast one(high compression ratio and acceptable compress throughput).
- higher Compression codecs like zlibipp/gzip/igzip/zstd consumes more CPU.
- Intel®-based Codec lz4-ipp got ~11.53% performance gain compared to default lz4 in HiBench Sort workload on Intel® Xeon® scalable Platform.
- Intel® based zlib-ipp costs similar CPU utility with zlib codec, but overall disk bandwidth higher than zlib, and zlibipp has ~21.8% performance increase compared with default gzip on Intel® Xeon® scalable Platform.

Conclusion

- Generally if you need more speed and less compression, lz4ipp/lz4 is the good choice.
- if you need more compression, zstd, zlib-ipp would be as good choice.
- Better to use faster codecs for spark shuffle compression codec, such as lz4-ipp, lz4.
- Higher compression reduce I/O and network pressure, but consumes CPU resource.
- Best balance of compression codec depends on cluster characteristics and workloads.

Outlines

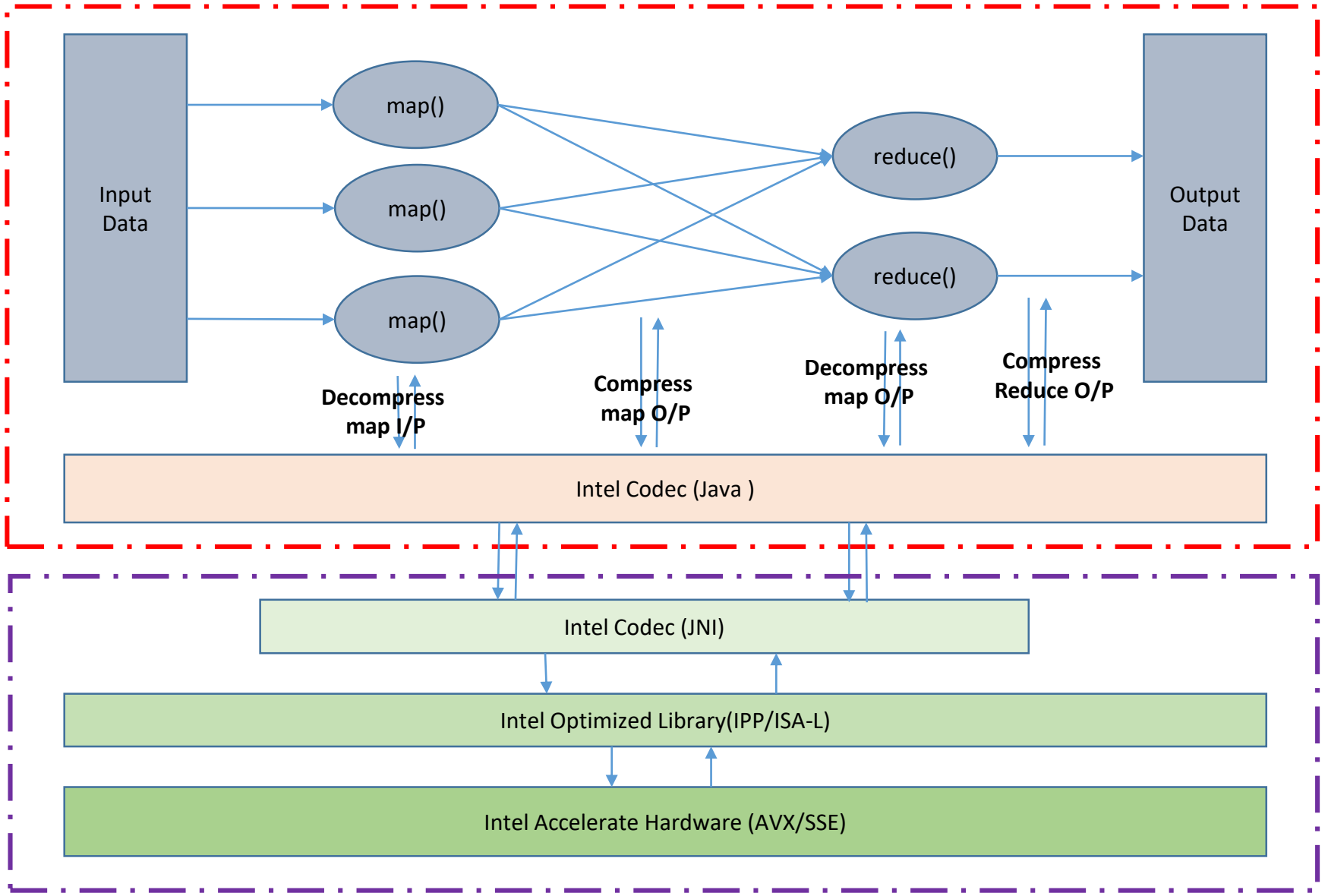
Micro-benchmark Result

TPC-DS benchmark Result

HiBench benchmark Result

Intel® Compression Codec Architecture

Intel Compression Codec Architecture



Source Code in OAP

https://github.com/Intel-bigdata/OAP/pull/396
ZSTD Added, and more codec open sourced soon

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

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.

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

THANK YOU