Performance Update

Dr. Stefan Reimbold
10. Juni 2021

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

AIX*	DB2*	HiperSockets*	MQSeries*	PowerHA*	RMF	System z*	zEnterprise*	z/VM*
BladeCenter*	DFSMS	HyperSwap	NetView*	PR/SM	Smarter Planet*	System z10*	z10	z/VSE*
CICS*	EASY Tier	IMS	OMEGAMON*	PureSystems	Storwize*	Tivoli*	z10 EC	
Cognos*	FICON*	InfiniBand*	Parallel Sysplex*	Rational*	System Storage*	WebSphere*	z/OS*	
DataPower*	GDPS*	Lotus*	POWER7*	RACF*	System x*	XIV*		

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
OpenStack is a trademark of OpenStack LLC. The OpenStack trademark policy is available on the OpenStack website.
TEALEAF is a registered trademark of Tealeaf, an IBM Company.
Windows Server and the Windows logo are trademarks of the Microsoft group of countries.
Worklight is a trademark or registered trademark of Worklight, an IBM Company.
UNIX is a registered trademark of The Open Group in the United States and other countries.

* Other product and service names might be trademarks of IBM or other companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.
All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) („SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the "Authorized Use Table for IBM Machines" provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html ("AUT"). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Agenda

- Performance Work
- Integrated Accelerator for zEDC
- Elasticsearch
- Red Hat OpenShift on IBM Z

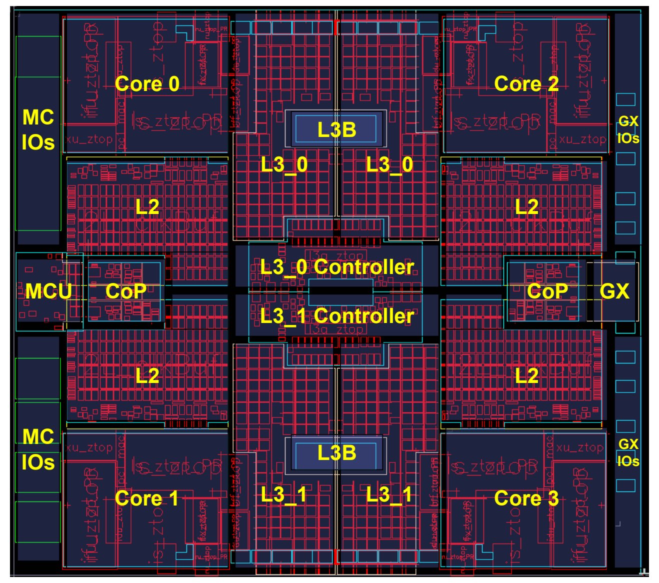
Performance Work

Dr. Stefan Reimbold

IBM Z - Processor Roadmap

45 nm

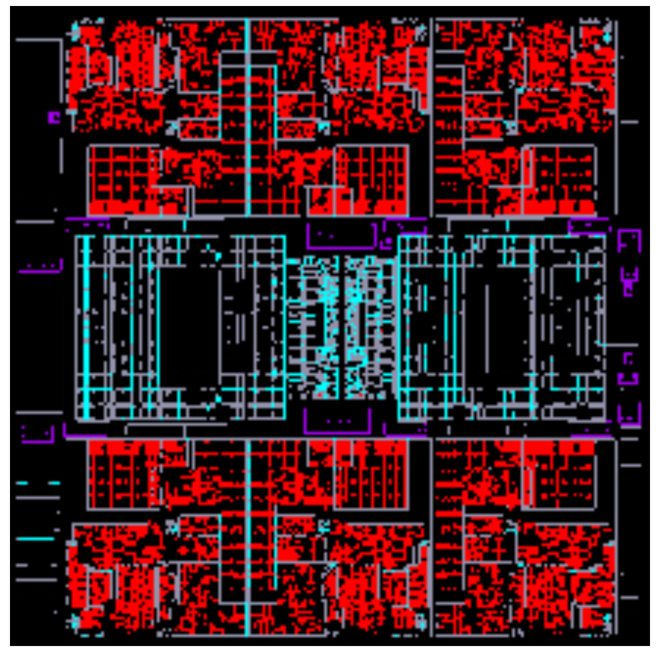
z196
9/2010



Top Tier Single Thread Performance,System Capacity
Accelerator Integration
Out of Order Execution
Water Cooling
PCIe I/O Fabric
RAIM
Enhanced Energy Management

32 nm

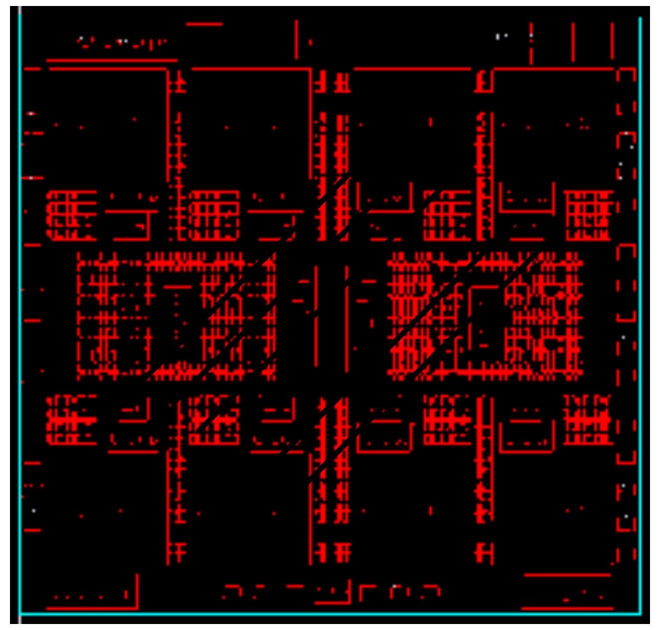
zEC12
8/2012



Leadership Single Thread, Enhanced Throughput
Improved out-of-order
Transactional Memory
Dynamic Optimization
2 GB page support
Step Function in System Capacity

22 nm

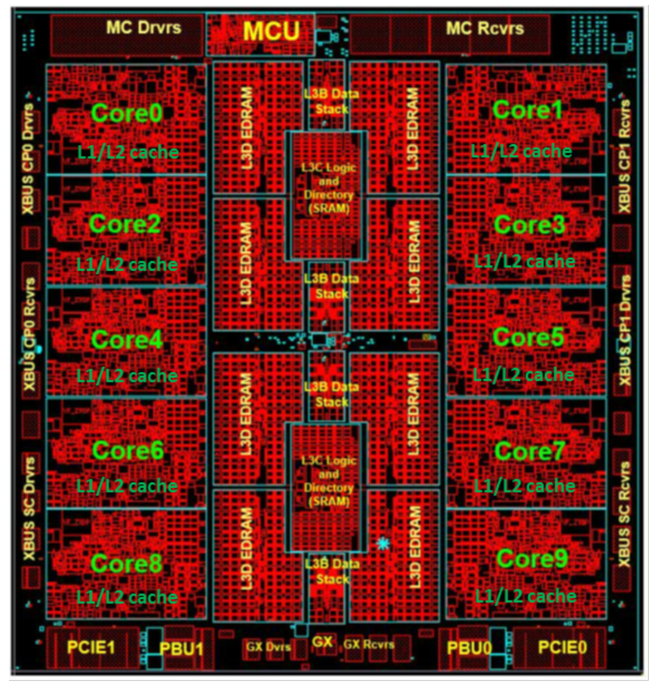
z13
1/2015



Leadership System Capacity and Performance
Modularity & Scalability
Dynamic SMT
Supports two instruction threads
SIMD
PCIe attached accelerators
Business Analytics Optimized

14 nm

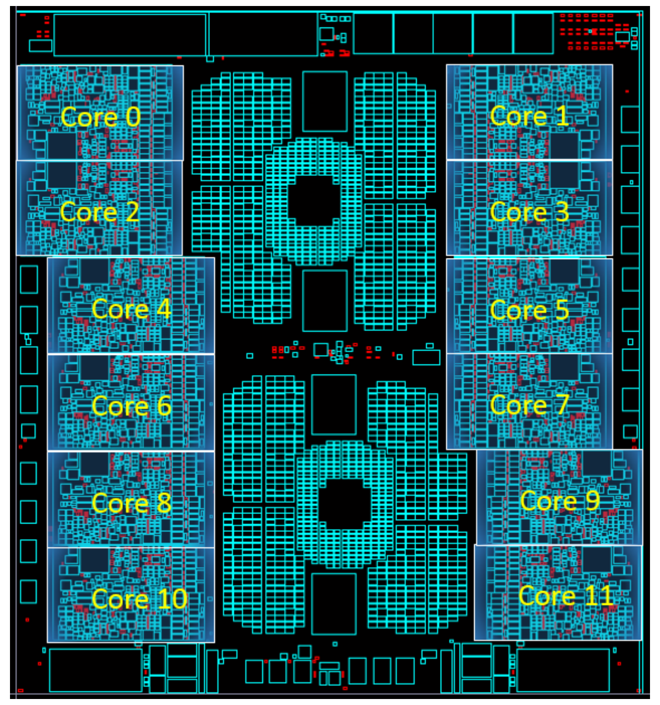
z14
7/2017



Pervasive encryption
Low latency I/O for acceleration of transaction processing for DB2 on z/OS
Pause-less garbage collection for enterprise scale JAVA applications
New SIMD instructions
Optimized pipeline and enhanced SMT
Virtual Flash Memory

14 nm

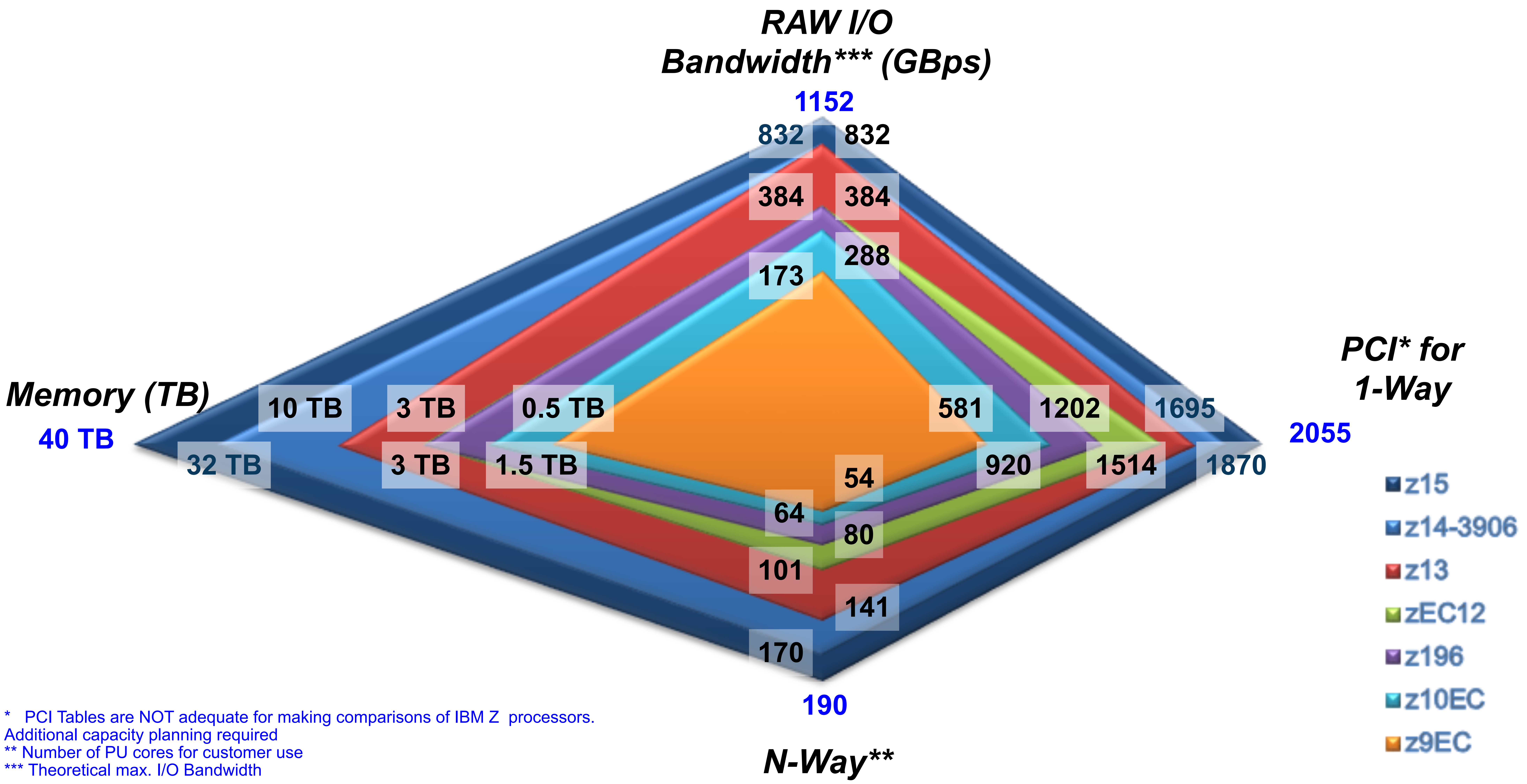
z15
9/2019



Focus on power efficiency and new on-chip architectures
Improved and enlarged caches
Optimized Out-of-Order architecture
Binary Floating point enhancements
IBM Integrated Accelerator for zEDC (On-chip compression support (DEFLATE))
Enhanced Cryptographic Coprocessor (CPACF)

z15

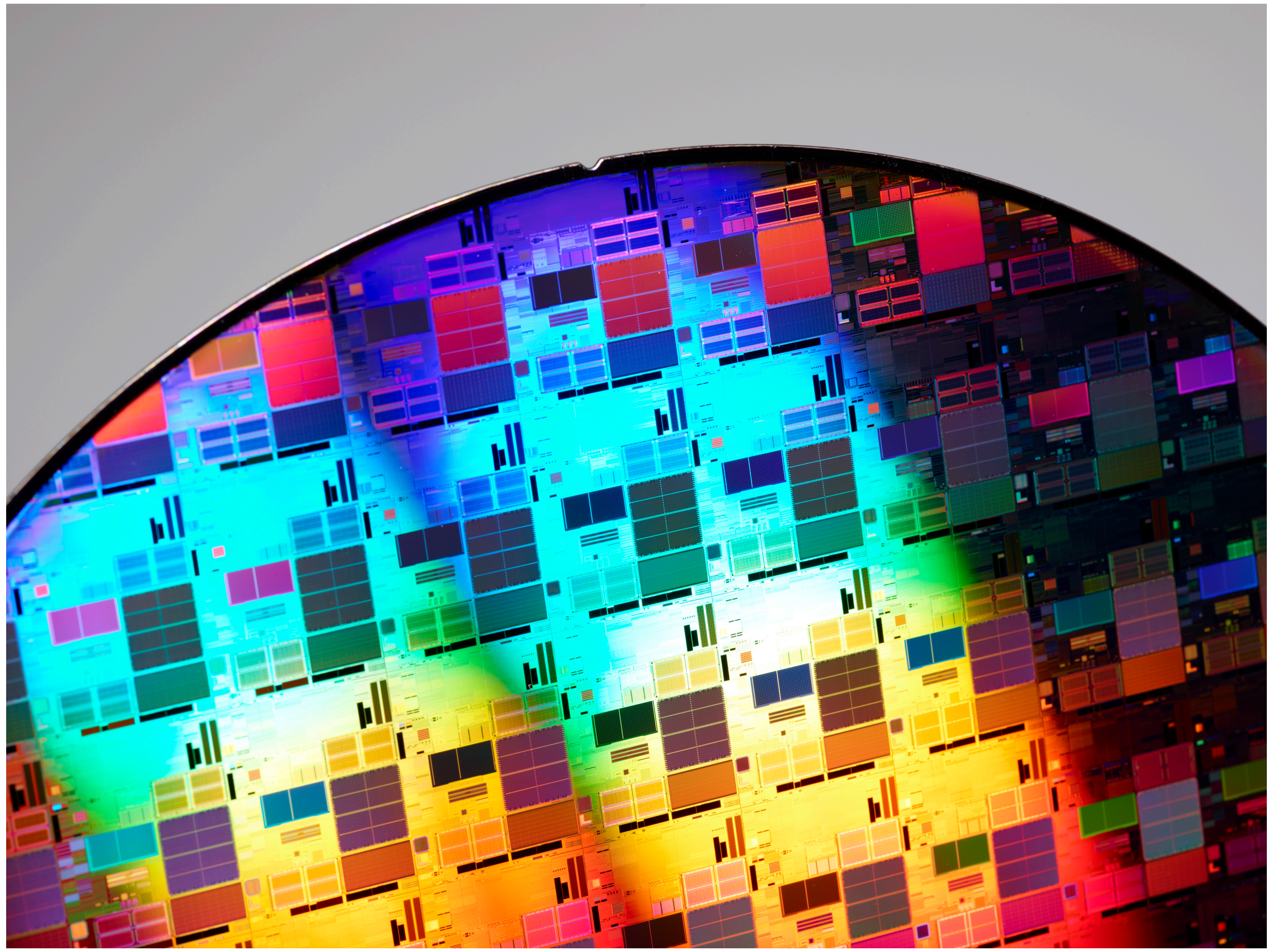
Balanced System Design



* PCI Tables are NOT adequate for making comparisons of IBM Z processors. Additional capacity planning required
** Number of PU cores for customer use
*** Theoretical max. I/O Bandwidth

Performance Work

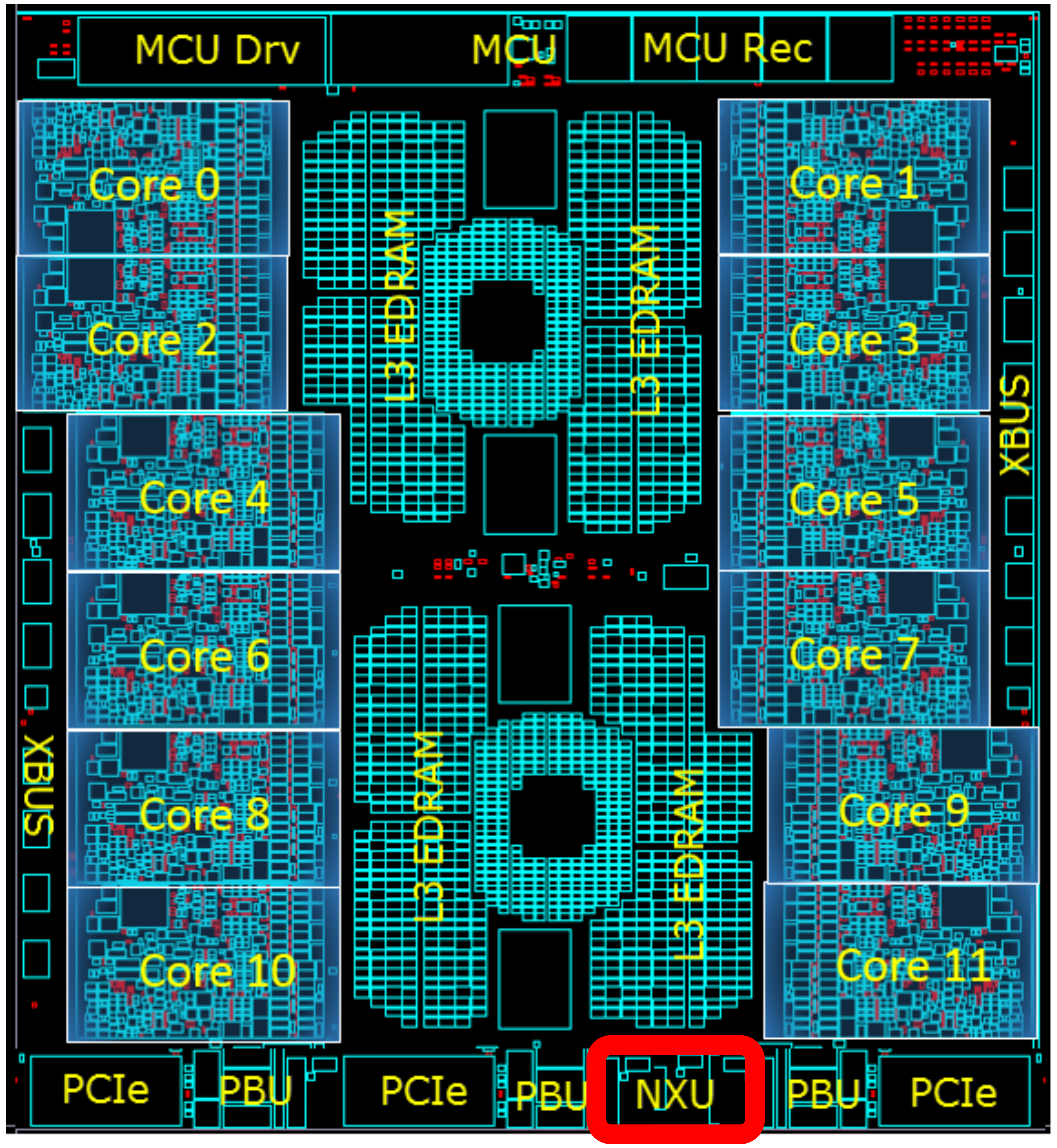
- Simulation
- Early bring-up
- Competitive analysis
- Tuning Hints & Tips
- Customer scenarios on OS level



Integrated Accelerator for zEDC

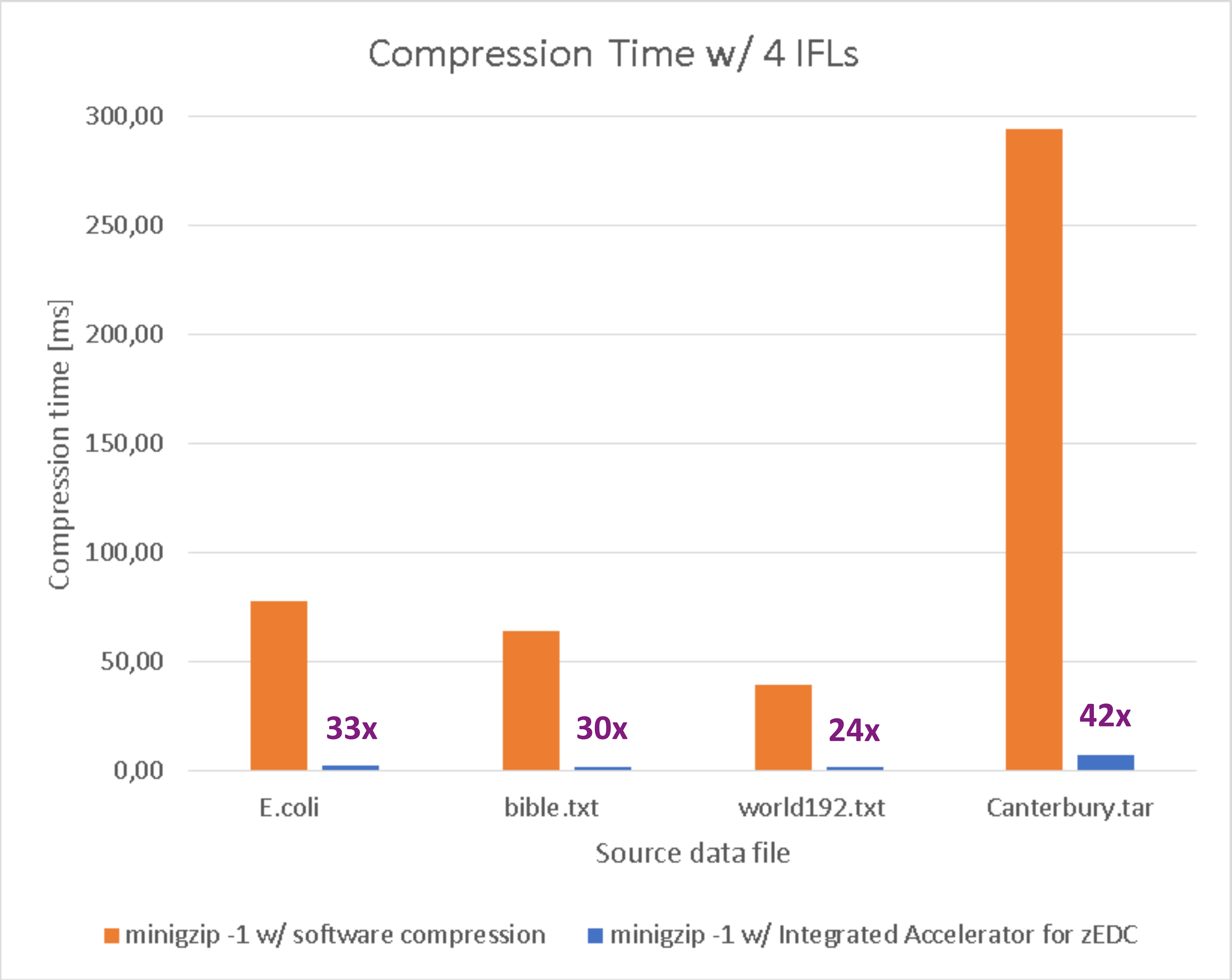
Dr. Stefan Reimbold

Integrated Accelerator for zEDC



Compression Time with Integrated Accelerator for zEDC versus Software Compression on z15

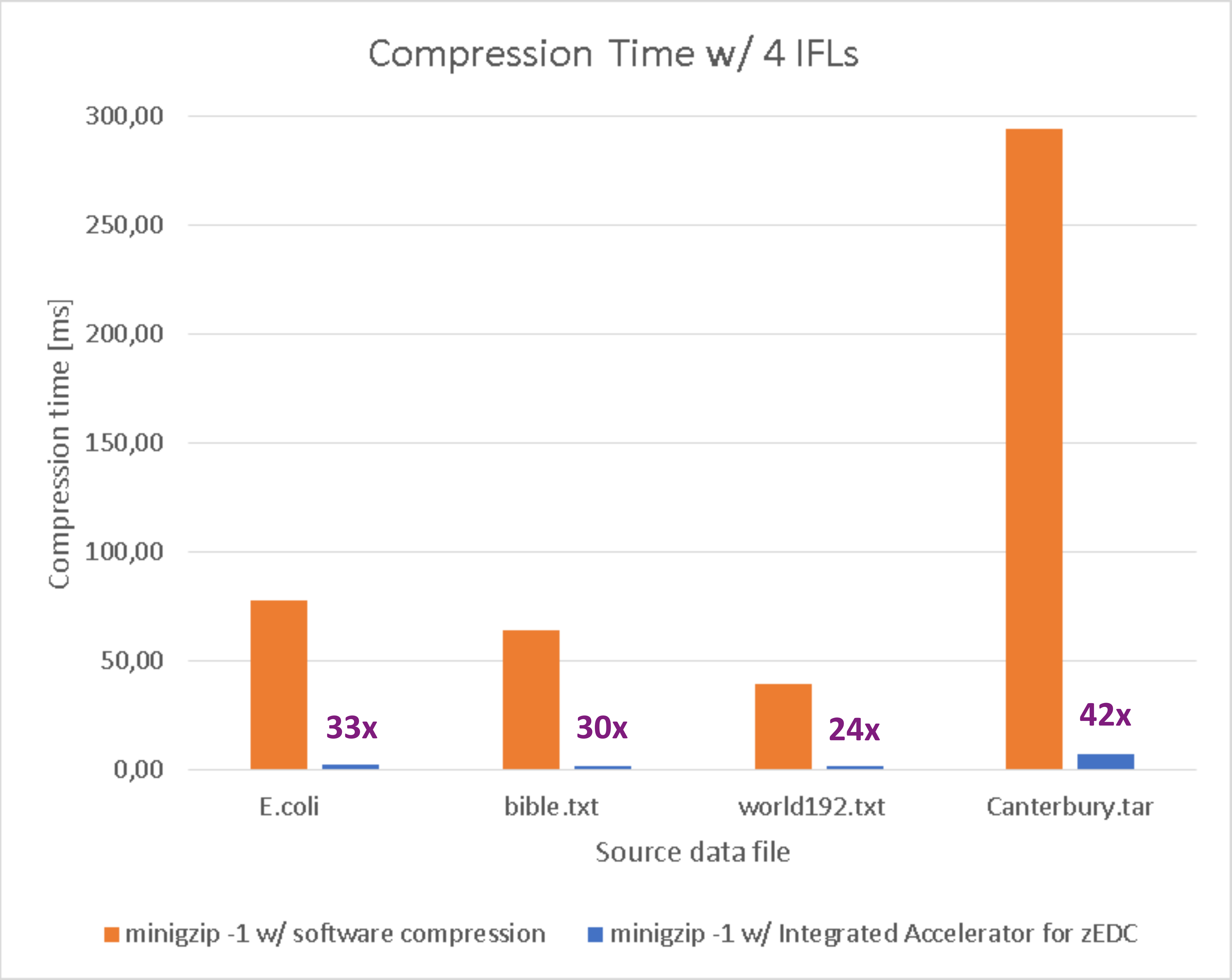
Compress data with zlib on z15 with 4 IFLs up to 42x faster with Integrated Accelerator for zEDC compared to using software compression



DISCLAIMER: Performance results based on IBM internal tests running the minigzip benchmark with compression level -1 from the dfl tcc branch of zlib (downloaded from https://github.com/iii-i/zlib/tree/dfltcc-20190708). Source data files were taken from the Large Corpus (downloaded from http://corpus.canterbury.ac.nz/descriptions). Canterbury.tar contained all files from all corpora. Results may vary. z15 configuration: LPAR with 4 dedicated IFLs, 64 GB memory, 40 GB DASD storage, SLES 12 SP4 (SMT mode).

Integrated Accelerator for zEDC

- Data compress and uncompress using new hardware instruction
- Reported with new feature flag in `/proc/cpuinfo: dflt`
- Compression equivalent to **gzip -1**
  - -1 is fastest
  - -9 slowest
  - default is -6
- Can be exploited e.g. by **zlib**
- Compress data with zlib on IBM z15 up to **42x faster** compared to software compression

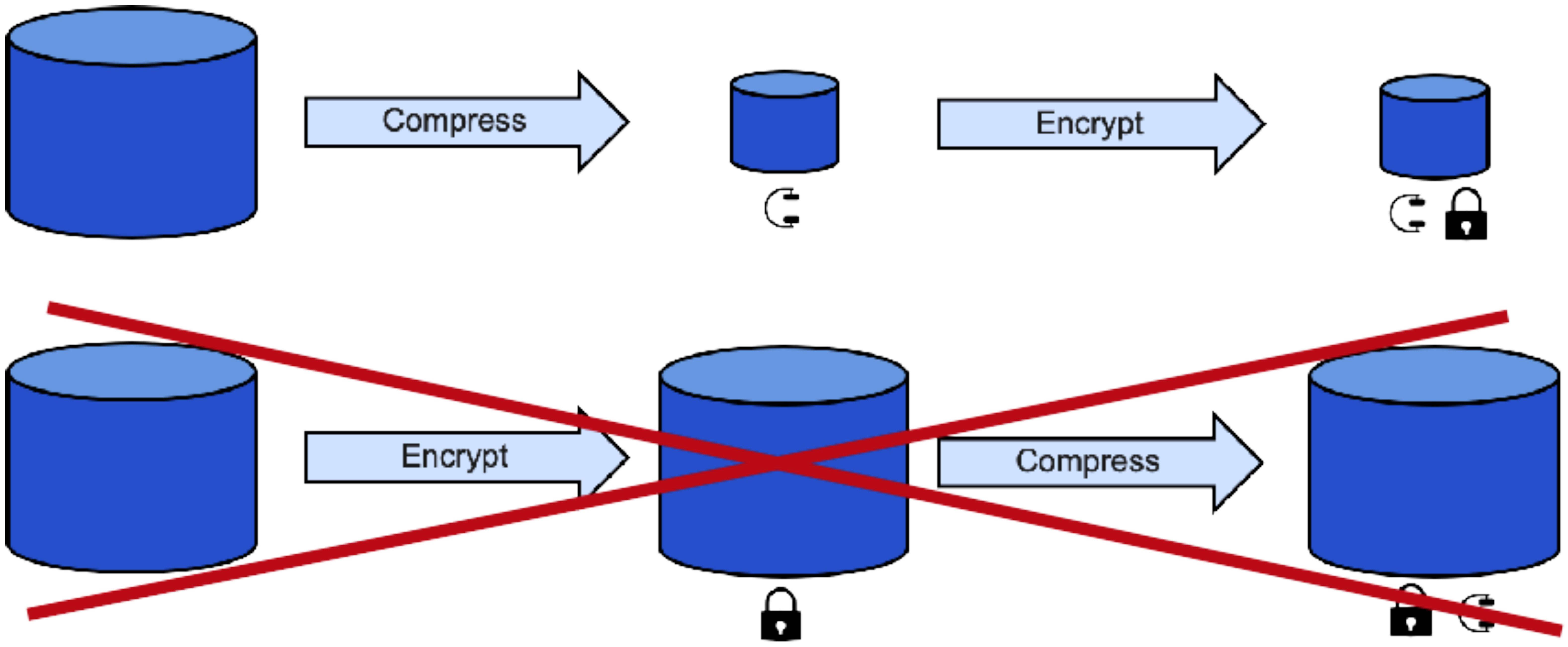


IBM Z

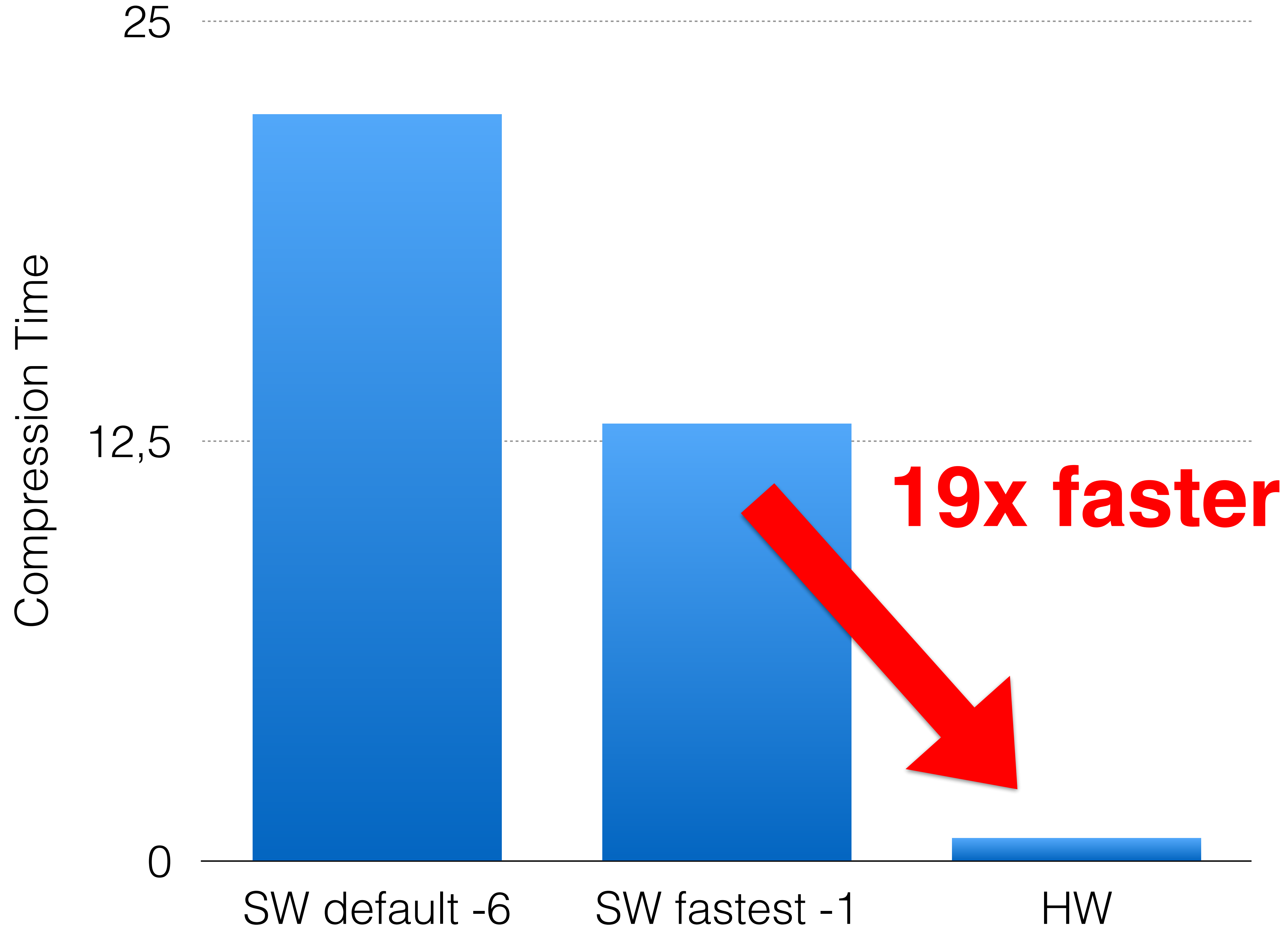
Dr. Stefan Reimbold

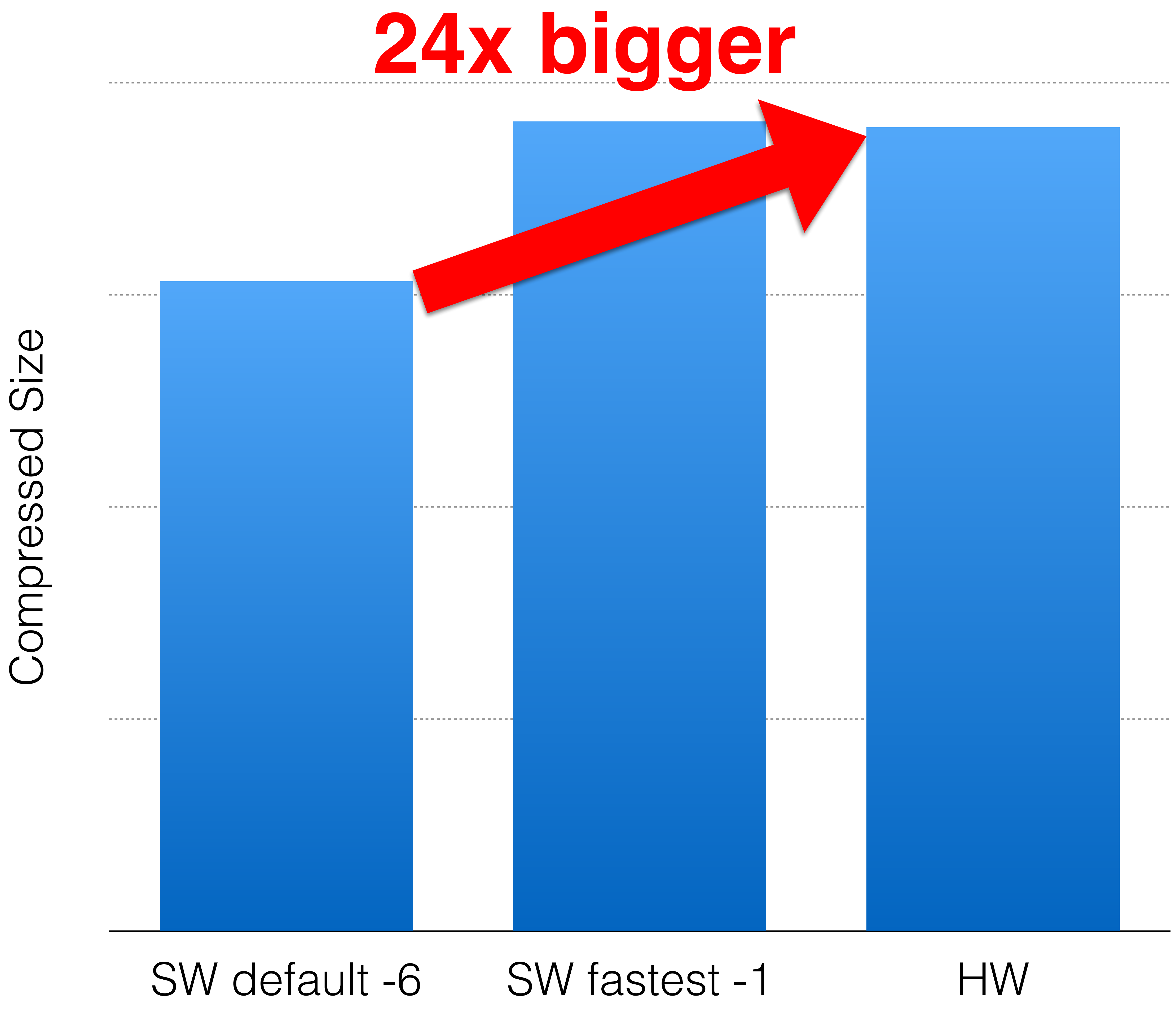
Integrated Accelerator for zEDC

- **Note:** Sequence of compression and encryption is essential



Archive - tar cvfz linux-5.0.20.tar.gz





tar Archives

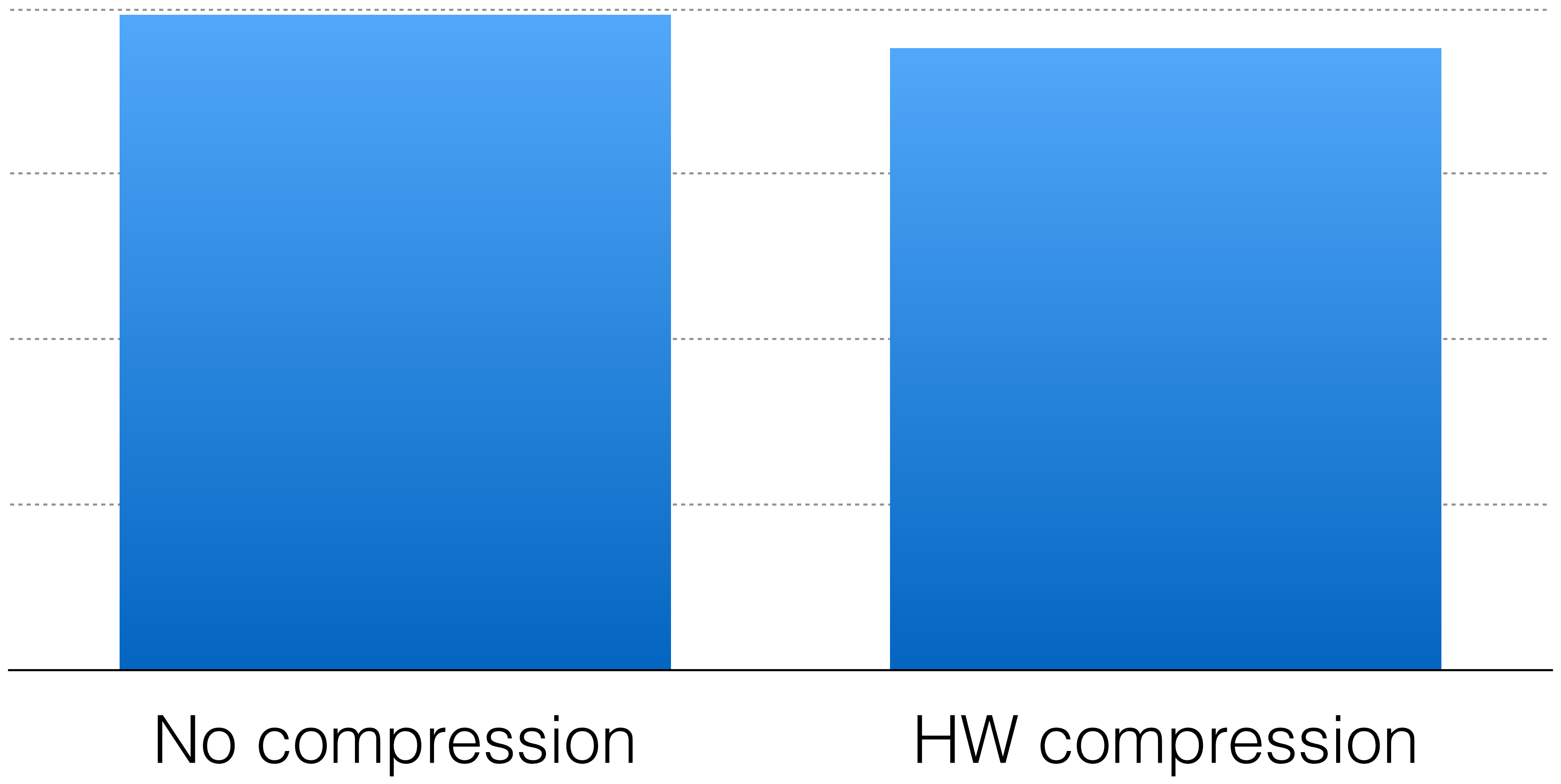
- No compression

```
# time tar cvf linux-5.0.20.tar linux-5.0.20 > /dev/null
real    0m0.704s
```

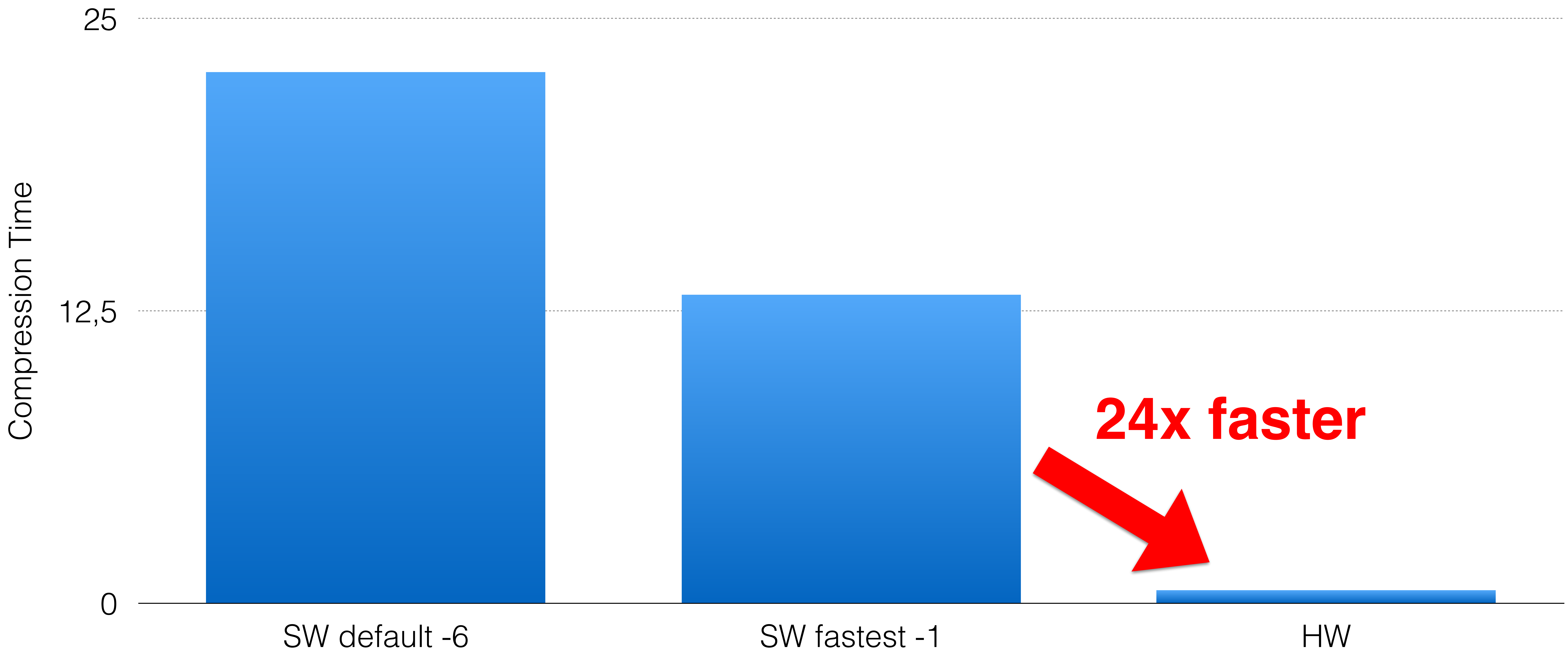
- Hardware Compression

```
# time GZIP=-1 tar cvfz linux-5.0.20.tar.gz linux-5.0.20 > /dev/null
real    0m0.668s
```

faster



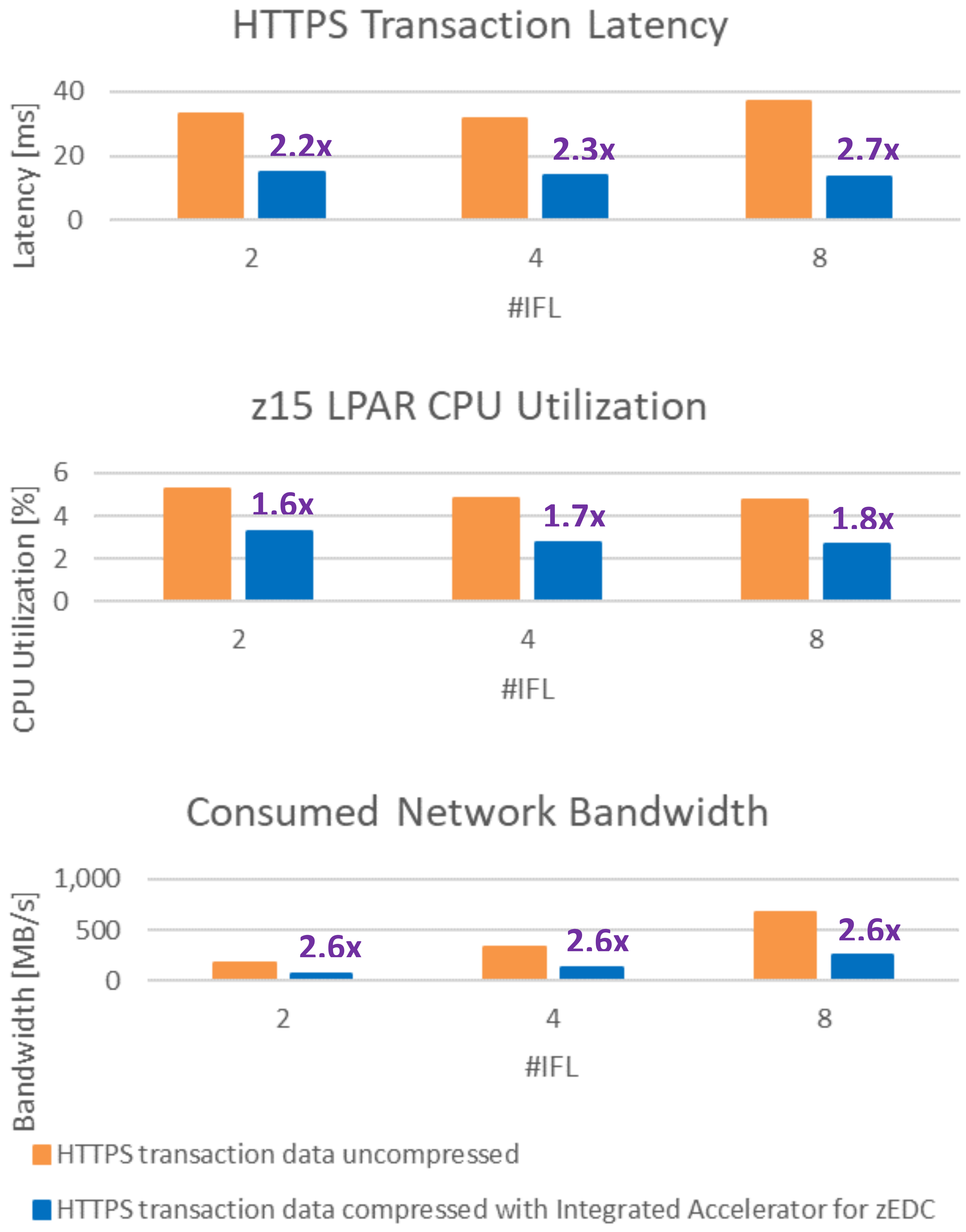
Compressor - gzip linux-5.0.20.tar



Competitive Performance

Compressing HTTPS Data before Encryption on z15

By compressing transaction data with the Integrated Accelerator for z Enterprise Data Compression prior to encryption, run secure web transactions with up to 2.7x lower latency, up to 1.8x less CPU utilization, and 2.6x less network bandwidth consumption on a z15 compared to running the transactions with encryption alone



DISCLAIMER: Performance results based on IBM internal tests running the wrk2 4.0.0.0 benchmarking tool (https://github.com/giltene/wrk2) remotely with a fix transaction rate against a NGINX 1.15.9 web server exploiting zlib (https://github.com/madler/zlib/pull/410) to compress transaction data before encryption versus not compressing transaction data before encryption. Data transmitted via NGINX webserver was the Silesia compression corpus (http://sun.aei.polsl.pl/~sdeor/index.php?page=Silesia). Results may vary. z15 configuration: LPAR with 8 dedicated IFL, 32 GB memory, 40 GB DASD storage, 200 GB FlashSystem 900 storage, SLES12 SP4 (SMT mode), running NGINX 1.15.9 with patch https://github.com/nginx/nginx/commit/cfa1316368dcc6dc1aa82e3d0b67ec0d1cf7eebb.

Elasticsearch

Dr. Stefan Reimbold

Elasticsearch

- Elasticsearch™ is a search engine based on Apache® Lucene® library
- Use cases:
  - Full text search: regular documents, HTML pages, source code, etc.
  - Logging and log analysis
  - Metrics: performance, availability, application usage statistics, etc.
  - Business analytics
- Most popular search engines

Elasticsearch

The following is ***important*** - please read it carefully

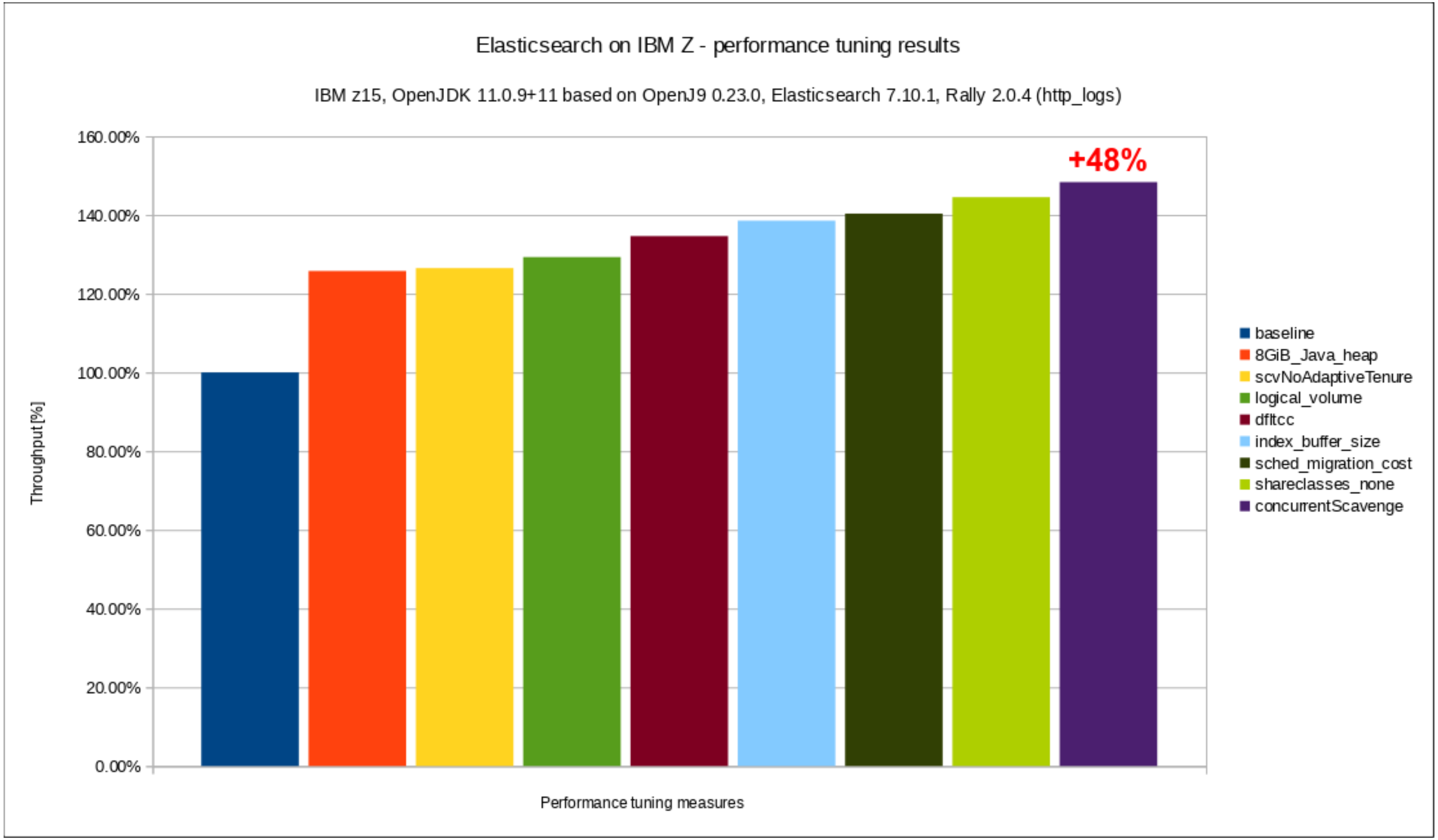
- The performance test results in the following charts were obtained in a **controlled lab environment** natively in LPAR. The measured differences in throughput might not be observed in real-life scenarios and environments other than native LPAR.

- All of the test runs were performed with Ubuntu® 20.04.2 LTS, Elasticsearch 7.10.1, and Rally 2.0.4. **Other** product versions might produce **different** performance results.

- All of the tests were specifically executed for **Elasticsearch**. The impact of the recommendations in this chart deck on **other** search engines might be **totally different**, including **adverse** performance effects.

- All of the tests were specifically executed for a heavy **indexing** workload. The impact of the recommendations in this chart deck on other types of workloads – query-only, for example – might be **totally different**, including **adverse** performance effects.

Elasticsearch



Elasticsearch

1. Increase **Java heap size** to at least 8 GiB
2. Use a **striped logical volume** for storing Elasticsearch data
3. Exploit the **Integrated Accelerator for zEDC**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
5. Consider lowering the Linux kernel **scheduler migration cost**
6. Disable Java's **shared class cache**
7. Enable **pause-less garbage collection**
8. Configure a **fixed tenure age** for the generations garbage collector

Elasticsearch

1. Increase **Java heap size** to at least 8 GiB
   - No general rule, but 1 GiB is **too small** for indexing-heavy workloads
   - To determine heap size do **analysis** of **verbose garbage collection**
   - Turned out to be the major tuning knob in this case - **25% increase in throughput**
2. Use a **striped logical volume** for storing Elasticsearch data
3. Exploit the **Integrated Accelerator for zEDC**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
5. Consider lowering the Linux kernel **scheduler migration cost**
6. Disable Java's **shared class cache**
7. Enable **pause-less garbage collection**
8. Configure a **fixed tenure age** for the generations garbage collector

Elasticsearch

1. Increase **Java heap size** to at least 8 GiB
2. Use a **striped logical volume** for storing Elasticsearch data
   - Device containing **index data** had average utilization over 95%
   - **Best Practice:** Use as many **stripes** as there are **physical** volumes
3. Exploit the **Integrated Accelerator for zEDC**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
5. Consider lowering the Linux kernel **scheduler migration cost**
6. Disable Java's **shared class cache**
7. Enable **pause-less garbage collection**
8. Configure a **fixed tenure age** for the generations garbage collector

Elasticsearch

1. Increase **Java heap size** to at least 8 GiB
2. Use a **striped logical volume** for storing Elasticsearch data
3. Exploit the **Integrated Accelerator for zEDC**
   - Analysis with perf was showing considerable amount of cycles spent in **compression**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
5. Consider lowering the Linux kernel **scheduler migration cost**
6. Disable Java's **shared class cache**
7. Enable **pause-less garbage collection**
8. Configure a **fixed tenure age** for the generations garbage collector

Elasticsearch

1. Increase **Java heap size** to at least 8 GiB
2. Use a **striped logical volume** for storing Elasticsearch data
3. Exploit the **Integrated Accelerator for zEDC**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
   - There's no clear recommendation to be found
   - 25% seems **reasonable** and higher values did **not** show further improvement
5. Consider lowering the Linux kernel **scheduler migration cost**
6. Disable Java's **shared class cache**
7. Enable **pause-less garbage collection**
8. Configure a **fixed tenure age** for the generations garbage collector

Elasticsearch

1. Increase **Java heap size** to at least 8 GiB
2. Use a **striped logical volume** for storing Elasticsearch data
3. Exploit the **Integrated Accelerator for zEDC**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
5. Consider lowering the Linux kernel **scheduler migration cost**
   - Linux `sysctl` setting that configures the **number of nanoseconds** the kernel will wait **before considering moving** the thread to another processor
   - Default setting is fine for most workloads
6. Disable Java's **shared class cache**
7. Enable **pause-less garbage collection**
8. Configure a **fixed tenure age** for the generations garbage collector

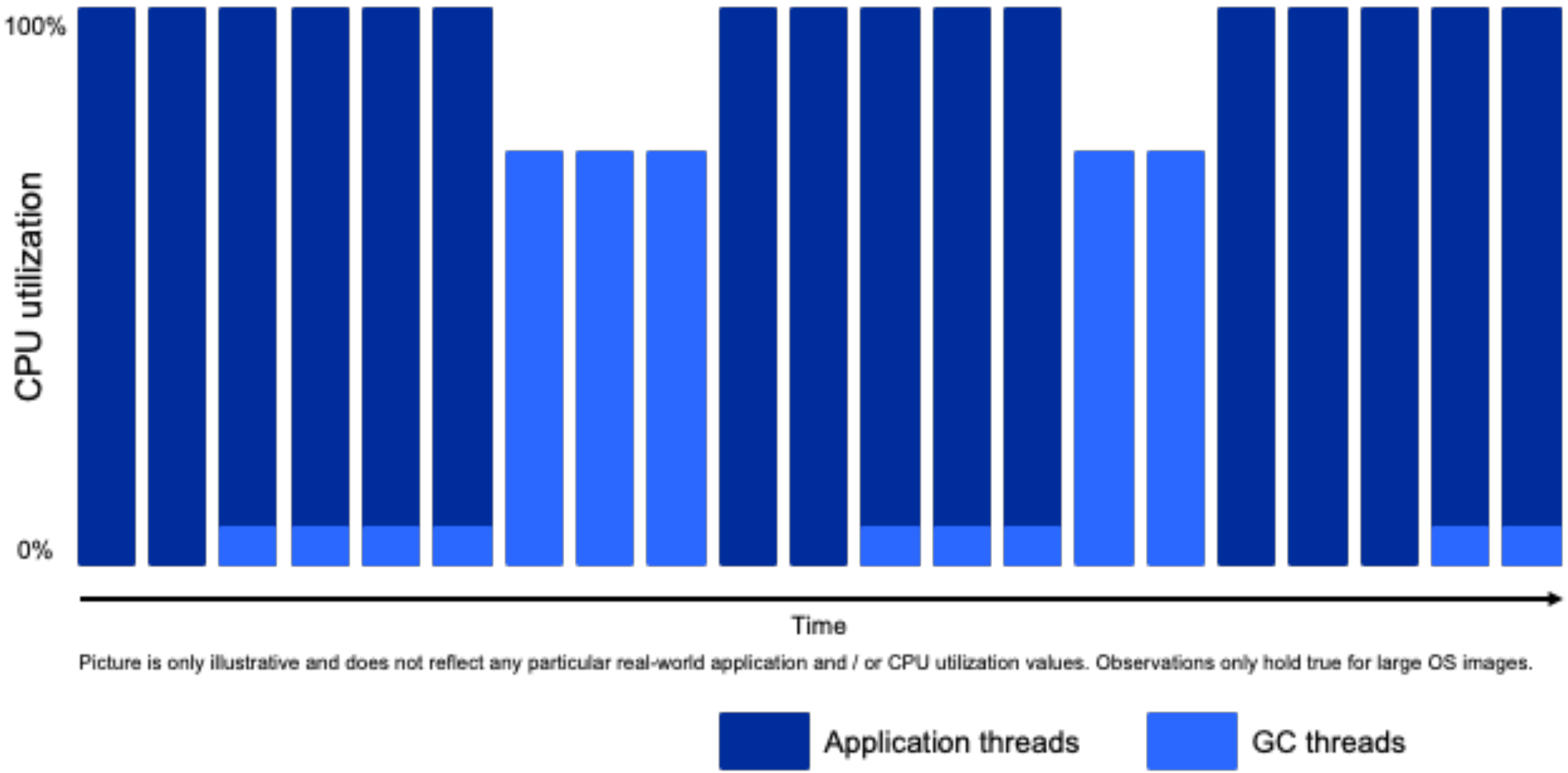
Elasticsearch

1. Increase **Java heap size** to at least 8 GiB
2. Use a **striped logical volume** for storing Elasticsearch data
3. Exploit the **Integrated Accelerator for zEDC**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
5. Consider lowering the Linux kernel **scheduler migration cost**
6. Disable Java's **shared class cache**
    - **Extremly useful** option for multi-JVM scenarios, in particular for microservices type of application
    - Only the **first** JVM needs to do bytecode verification, native compilation, etc.
    - Greatly improves JVM start-up time
    - One caveat is that code quality is **somewhat lower**
7. Enable **pause-less garbage collection**
8. Configure a **fixed tenure age** for the generations garbage collector

Elasticsearch

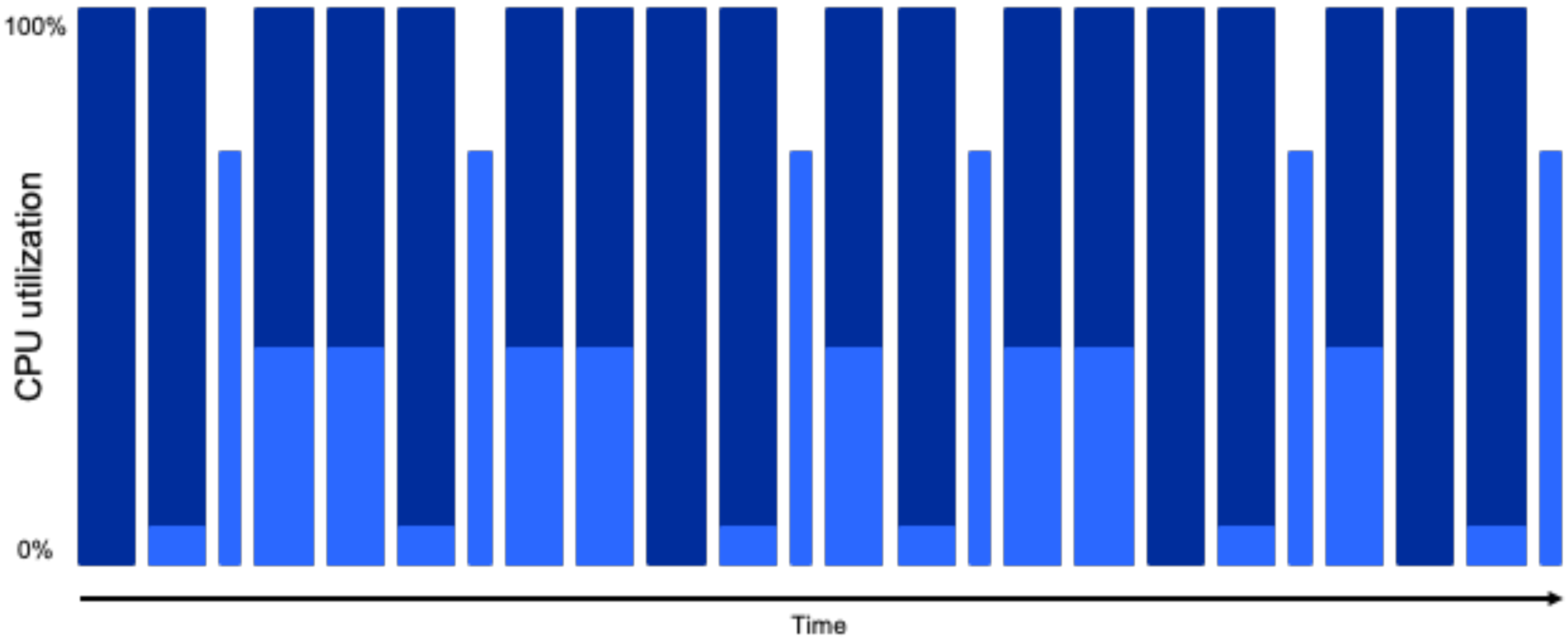
1. Increase **Java heap size** to at least 8 GiB
2. Use a **striped logical volume** for storing Elasticsearch data
3. Exploit the **Integrated Accelerator for zEDC**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
5. Consider lowering the Linux kernel **scheduler migration cost**
6. Disable Java's **shared class cache**
7. Enable **pause-less garbage collection**
   - **Important:** does not by magic improve CPU cost of garbage collection, CPU cost tends to be even higher
   - **Improves** the **pause times** significantly
8. Configure a **fixed tenure age** for the generations garbage collector

Regular Garbage Collection with gencon

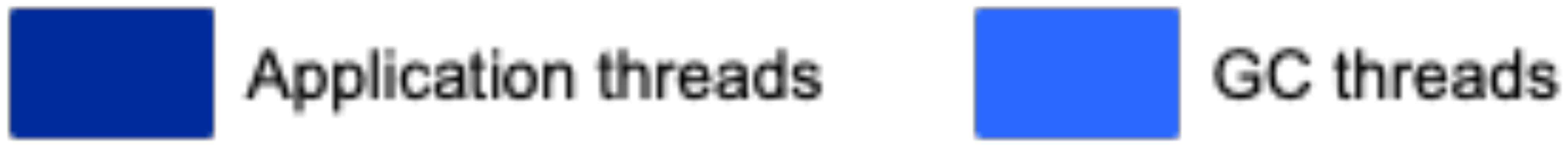


Picture is only illustrative and does not reflect any particular real-world application and / or CPU utilization values. Observations only hold true for large OS images.

Pause-less Garbage Collection



Picture is only illustrative and does not reflect any particular real-world application and / or CPU utilization values. Observations only hold true for large OS images.



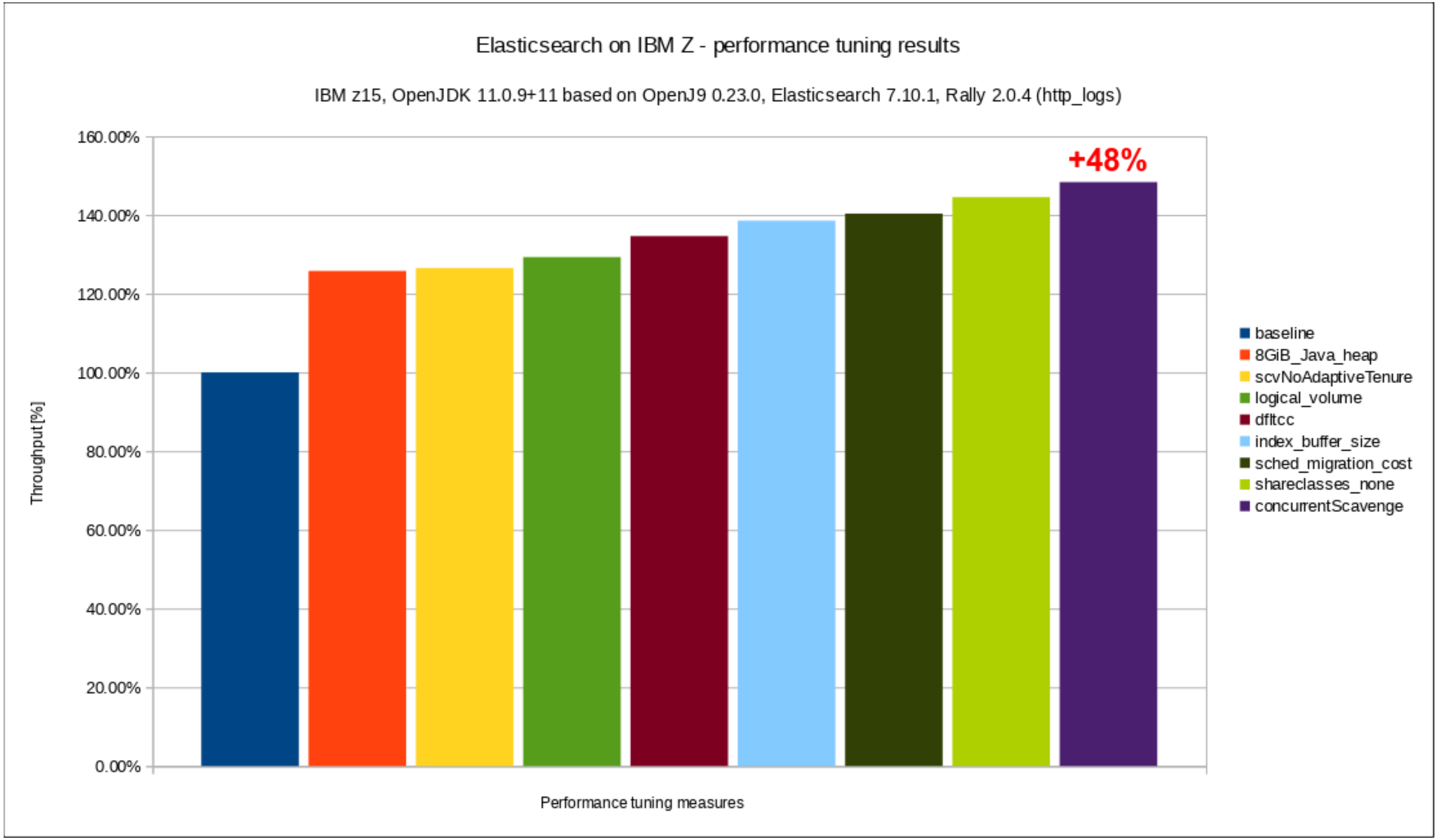
Elasticsearch

1. Increase **Java heap size** to at least 8 GiB
2. Use a **striped logical volume** for storing Elasticsearch data
3. Exploit the **Integrated Accelerator for zEDC**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
5. Consider lowering the Linux kernel **scheduler migration cost**
6. Disable Java's **shared class cache**
7. Enable **pause-less garbage collection**
8. Configure a **fixed tenure age** for the generations garbage collector
   - Typically response time is more important in a **query-heavy** scenario
   - Nonetheless this might also be of interest for indexing

Elasticsearch

1. Increase **Java heap size** to at least 8 GiB
2. Use a **striped logical volume** for storing Elasticsearch data
3. Exploit the **Integrated Accelerator for zEDC**
4. Increase the size of the **Elasticsearch indexing buffer** to 25% of the available Java heap
5. Consider lowering the Linux kernel **scheduler migration cost**
6. Disable Java's **shared class cache**
7. Enable **pause-less garbage collection**
8. Configure a **fixed tenure age** for the generations garbage collector

Elasticsearch

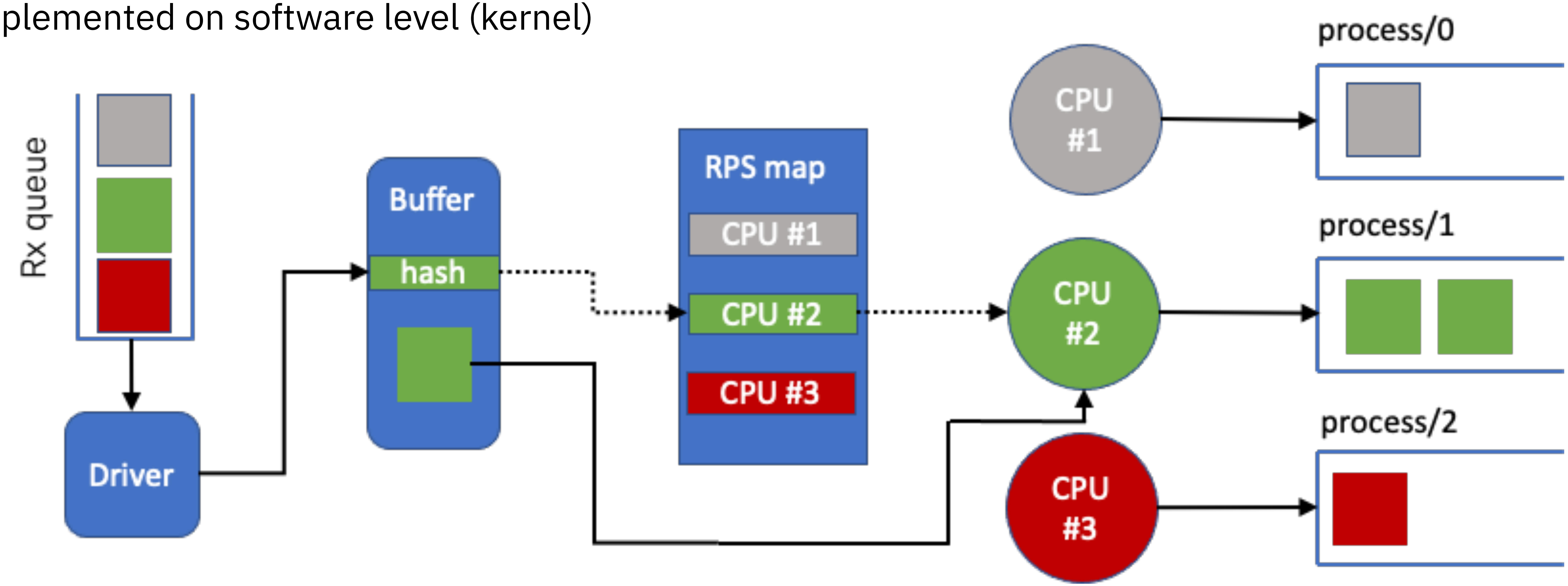


Red Hat OpenShift on IBM Z

Dr. Stefan Reimbold
10. Juni 2021

Received Packet Steering - RPS

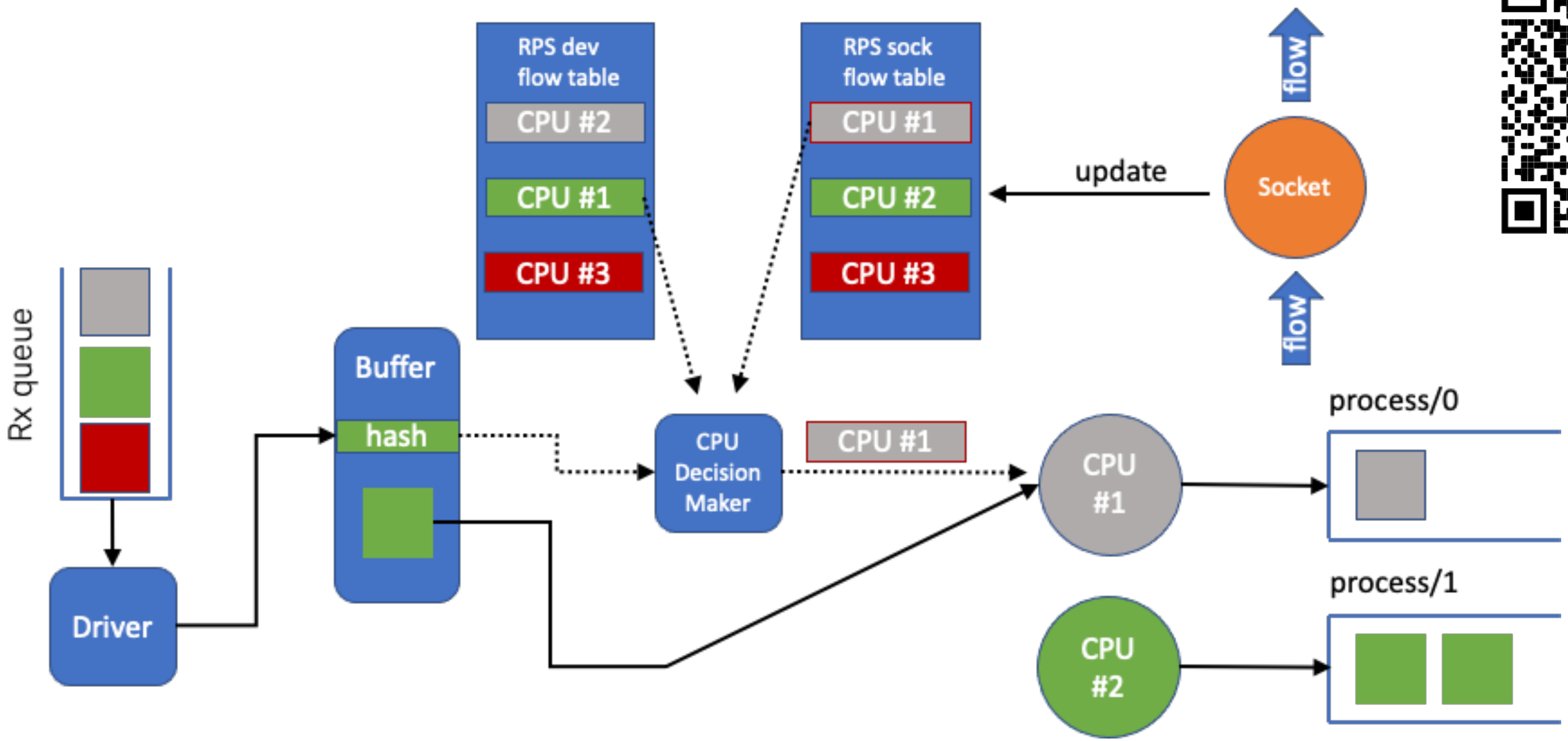
- Prevents hardware queue of network card being bottleneck
- Direct packets to specific CPUs
- Implemented on software level (kernel)



Latency can be improved significantly with RPS setting on (depending on workload)

Received Flow Steering - RFS

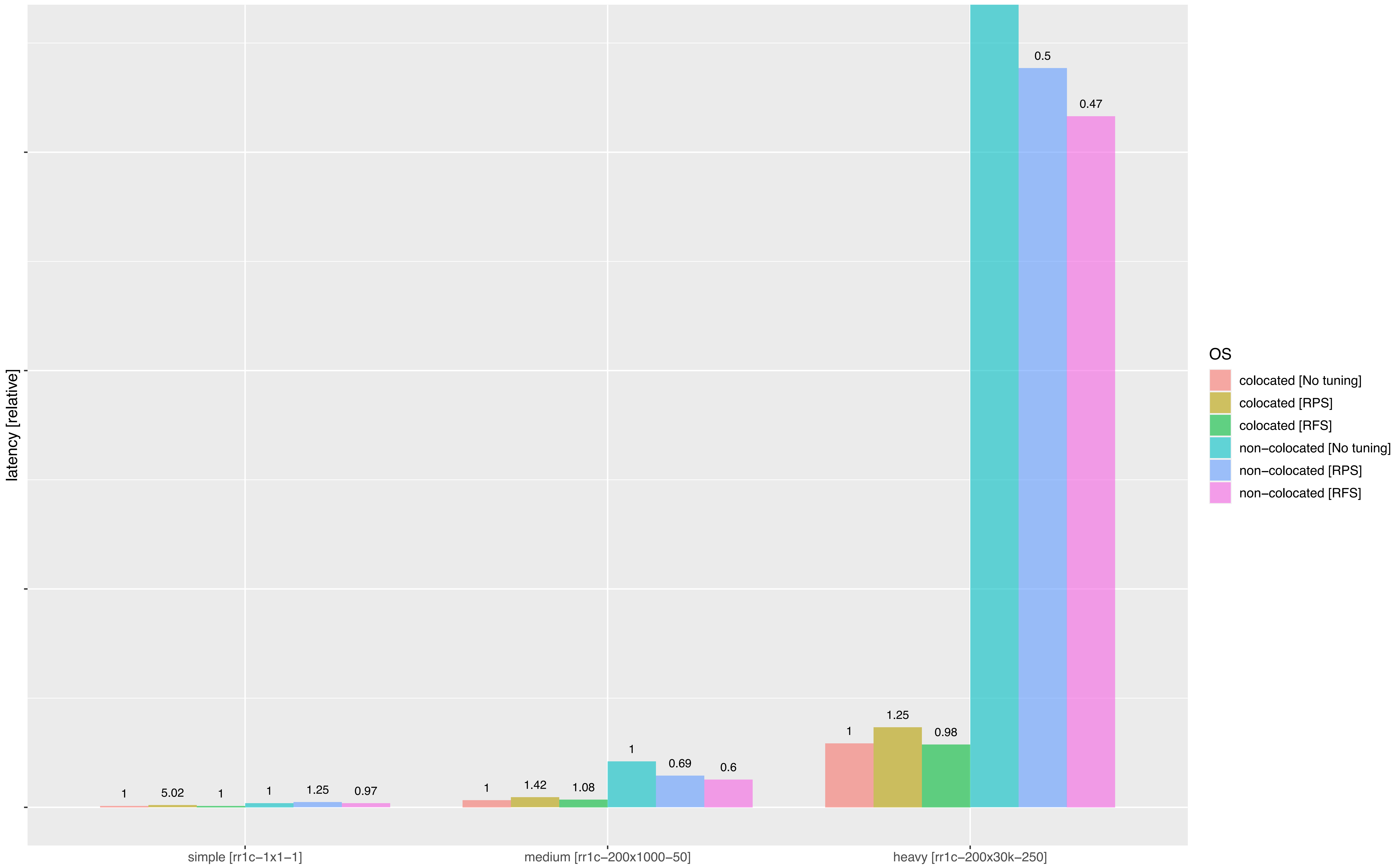
How to Tune with RFS:



- RFS checks if packet processing is running on CPU of destination thread
- If not, the table is updated and packet processing is performed on the target core
- This uses positive cache effects more efficient

Dr. Stefan Reimbold

Received Flow Steering - RFS

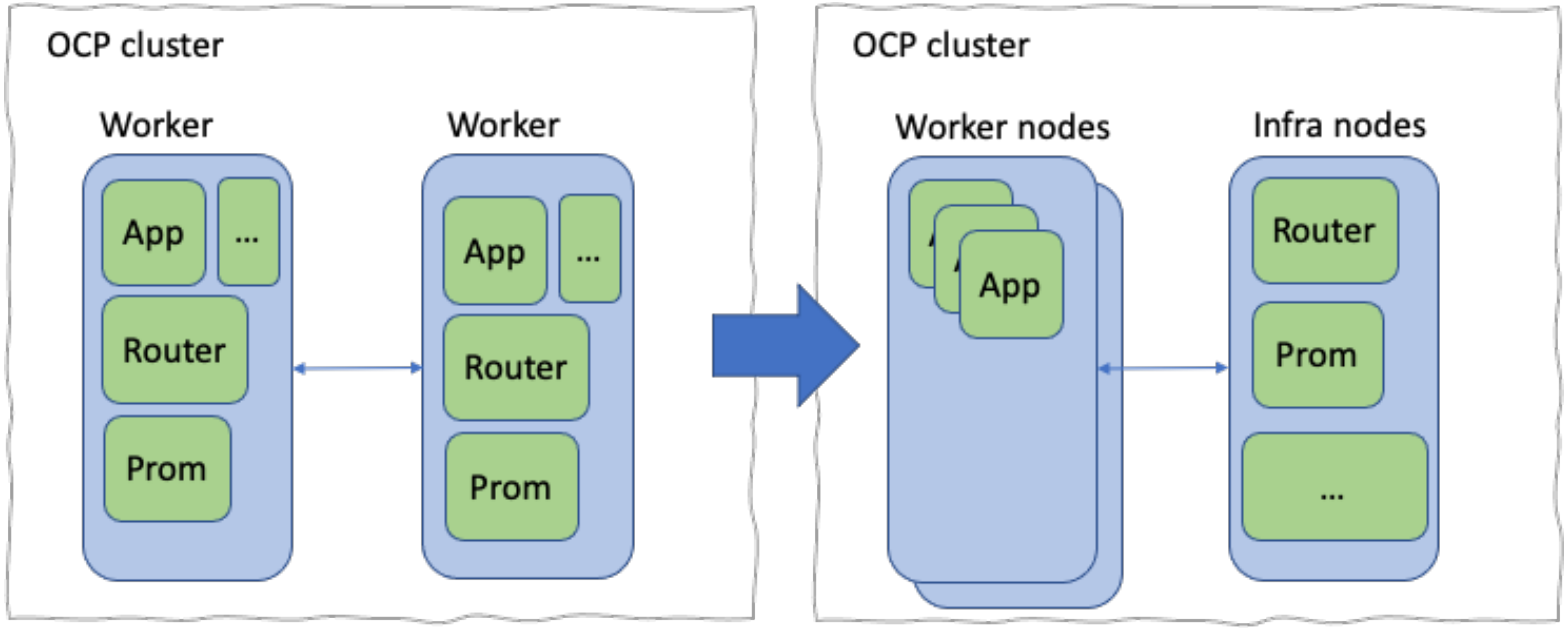


IBM Z

Dr. Stefan Reimbold

CoreOS/OCP Optimization - Using Infrastructure Nodes

How to setup infra nodes:



- Move all infrastructure services to infrastructure nodes to keep workers for application workloads exclusively
- Can improve performance significantly
- OCP services, such as monitoring solutions (e.g. Prometheus) and router run in worker nodes per default
- Consume resources of worker and slow down applications

https://www.linkedin.com/pulse/boosting-performance-using-infrastructure-nodes-your-cluster-miranda

Summary

Dr. Stefan Reimbold

Thank You !

- Dr. Axel Busch
- Marc Beyerle
  Java Performance Analyst
- Dominic Röhm

Summary

- Integrated Accelerator for zEDC
- Elasticsearch
- Red Hat OpenShift on IBM Z

Links

Documentation

- Linux on Z and LinuxONE Knowledgecenter

  https://www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_main.html

Webcasts

- In-depth sessions on Linux on Z topics
- Provided by Linux on Z development team

  For future sessions and session recordings

  https://developer.ibm.com/tv/linux-ibm-z/

Blogs

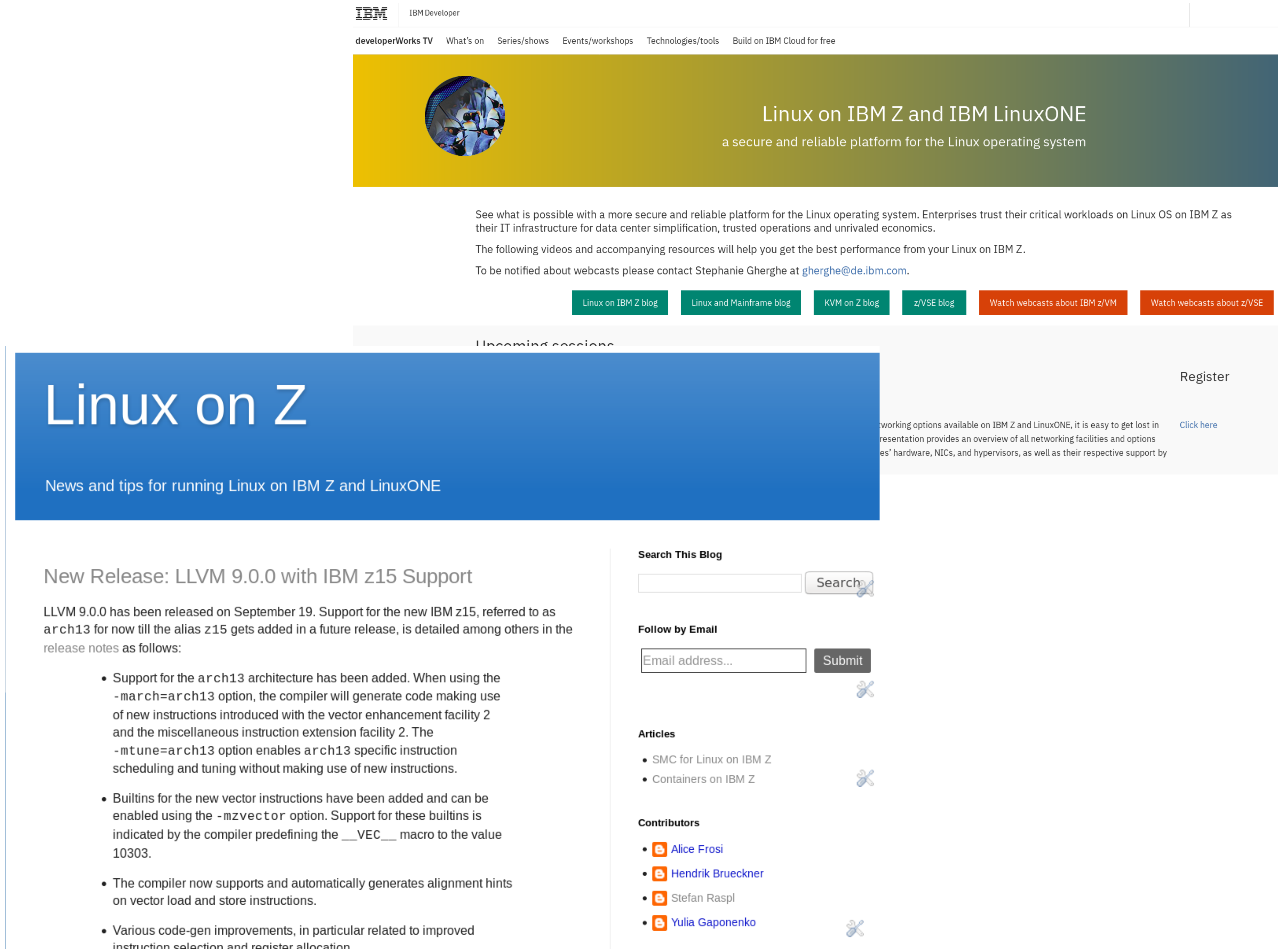
- Latest news from the development team
- Focus on upstream submissions
- Feature articles on specific in-depth topics

  Linux on Z, including containers

  http://linux-on-z.blogspot.com/

  KVM on Z

  http://kvmonz.blogspot.com/



Questions ?

Dr. Stefan Reimbold
Performance Analyst

Linux on IBM Z
Performance Evaluation

Schoenaicher Strasse 220
D-71032 Boeblingen

Mail: Postfach 1380
D-71003 Boeblingen
Phone +49-7031-16-2368
stefan.Reimbold@de.ibm.com

vcard