WebSphere Liberty and Java performance

Vijay Sundaresan
STSM, WebSphere and Java Performance Architect, IBM Toronto Lab

Virtual WebSphere User Group Roadshow, Spring 2022

IBM

Agenda

Application server performance comparisons

Java optimizations for the cloud

Liberty performance across platforms

Please note

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at IBM’s sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

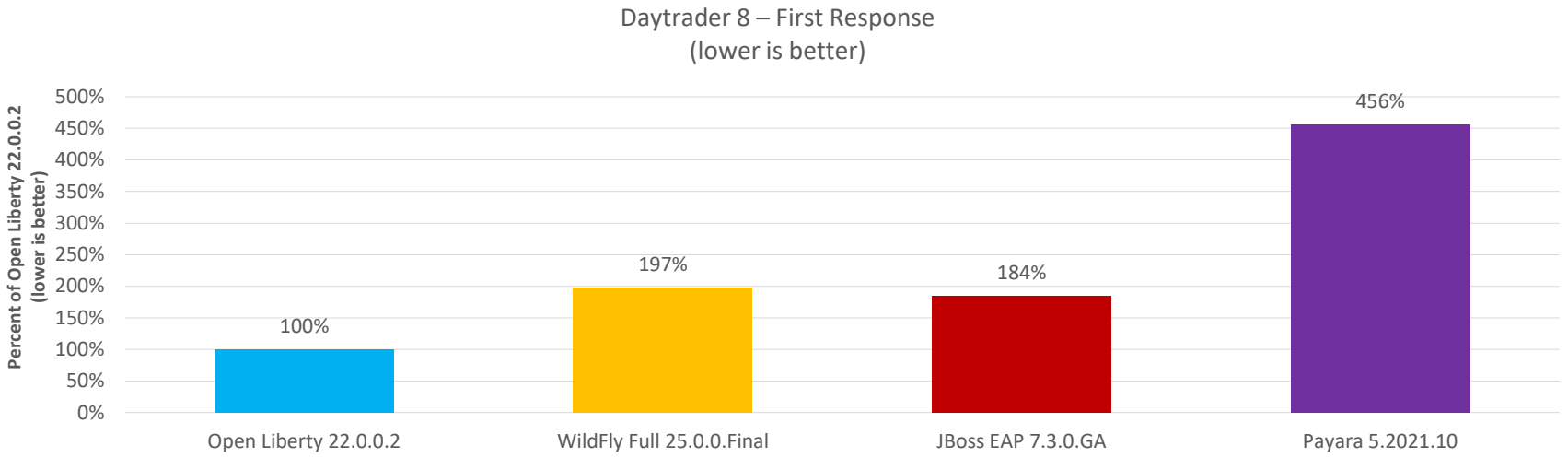
The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

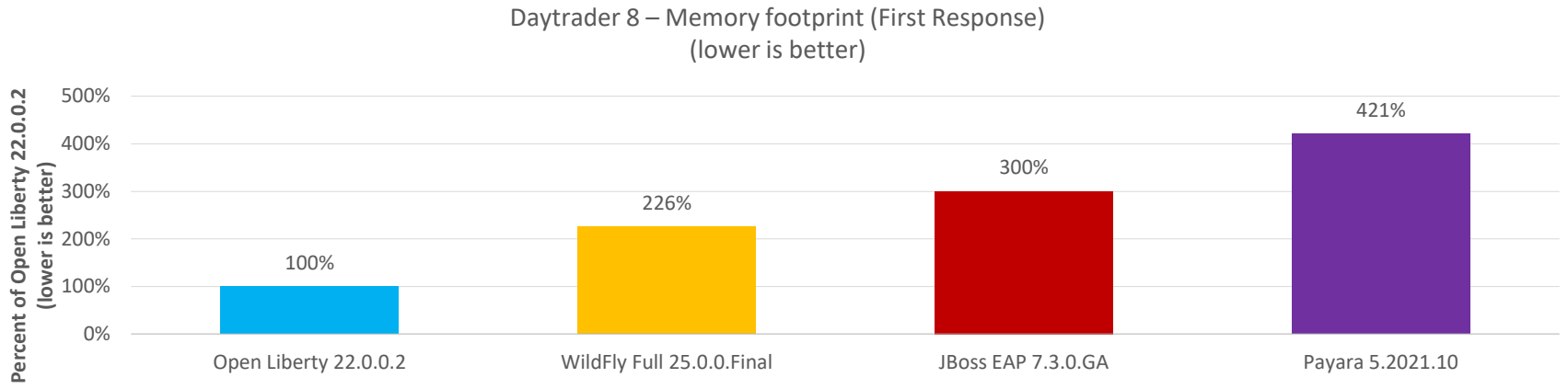
Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

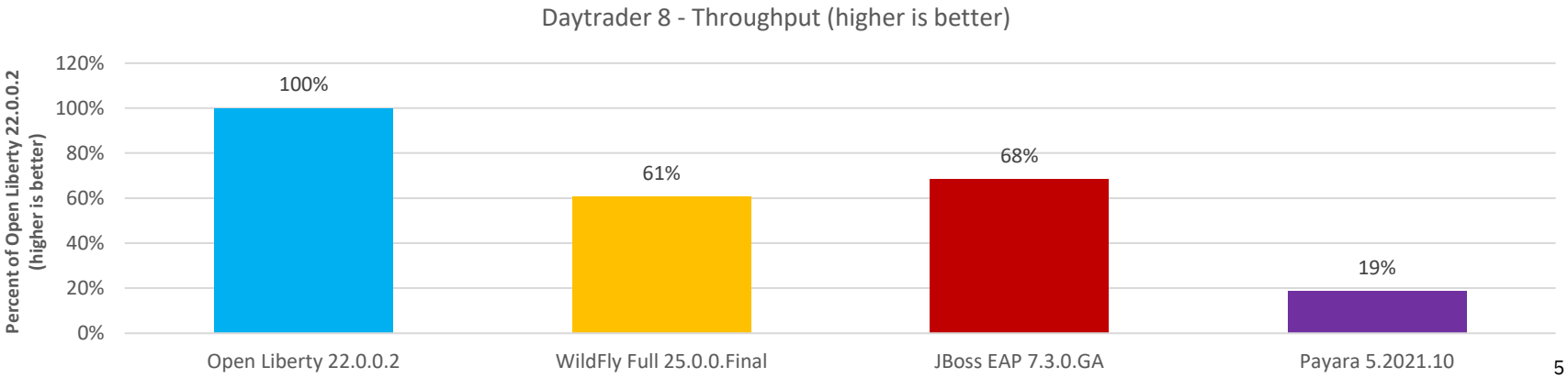
Application server performance comparisons

EE8 Performance (Daytrader8)

- Liberty outperforms others on all metrics for EE8 performance (startup time and memory footprint is almost half, throughput is 32% better)
- Comparisons used each application server's Docker image





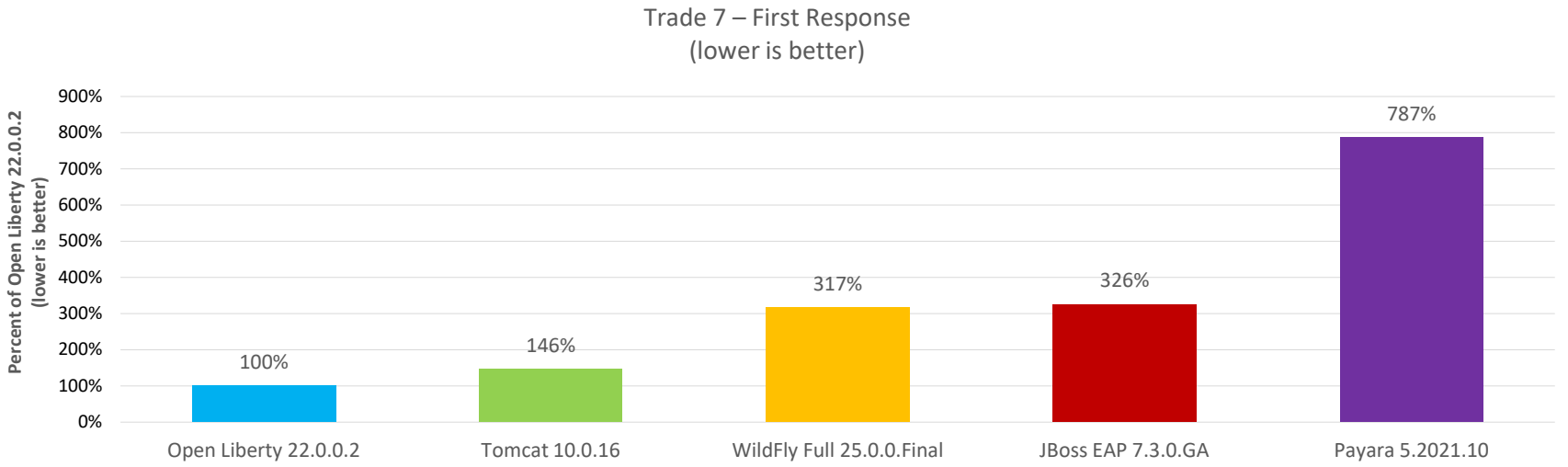


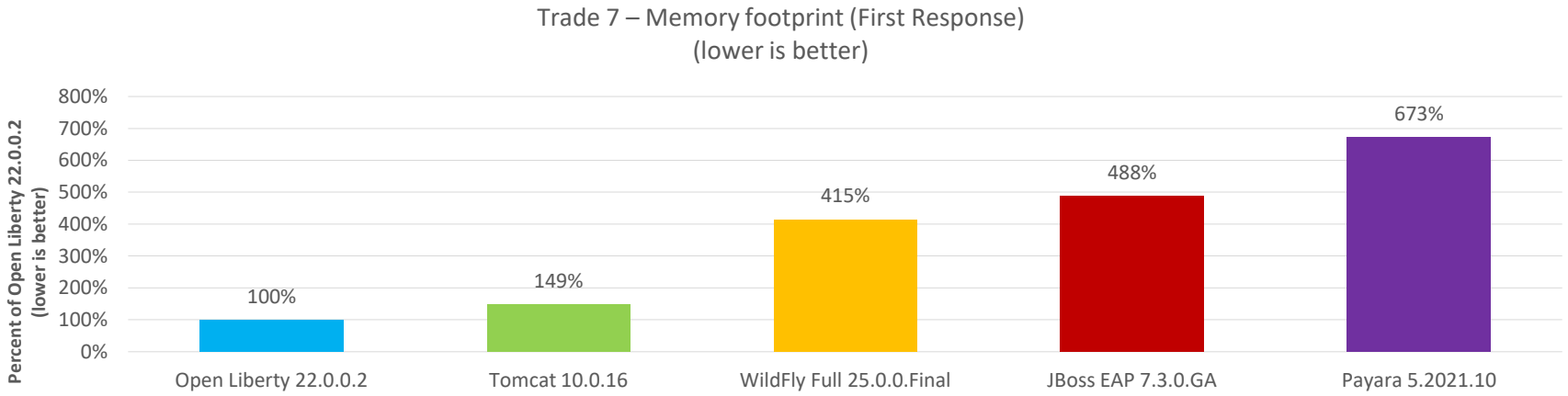
System Configuration:

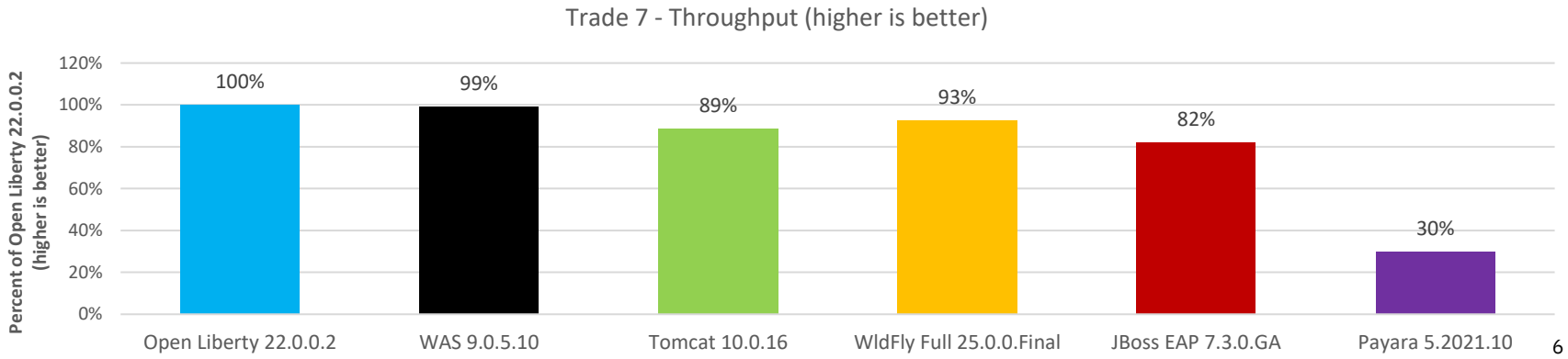
SUT: LinTel – SLES 12.4, Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, 4 cpus, 4GB RAM.
JDK version distributed with the docker images used for each server instance.

EE7 Performance (Trade7)

- Liberty outperforms others on all metrics for EE7 performance (startup time and memory footprint is 50% better, throughput is better as well)
- Comparisons used each application server's Docker image





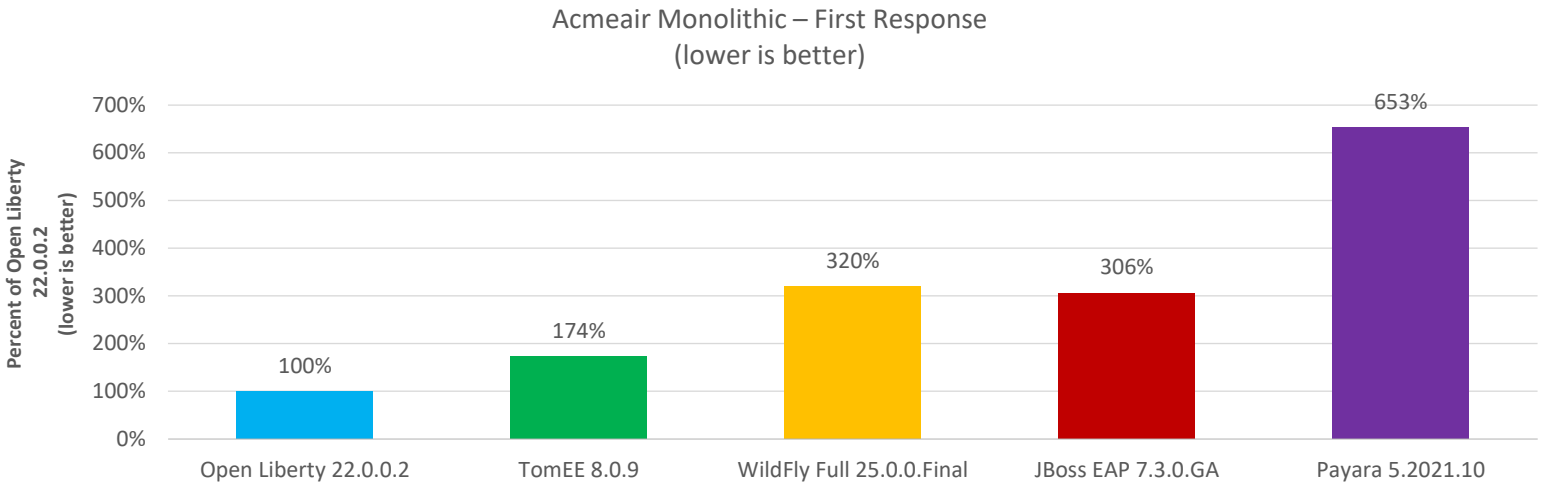


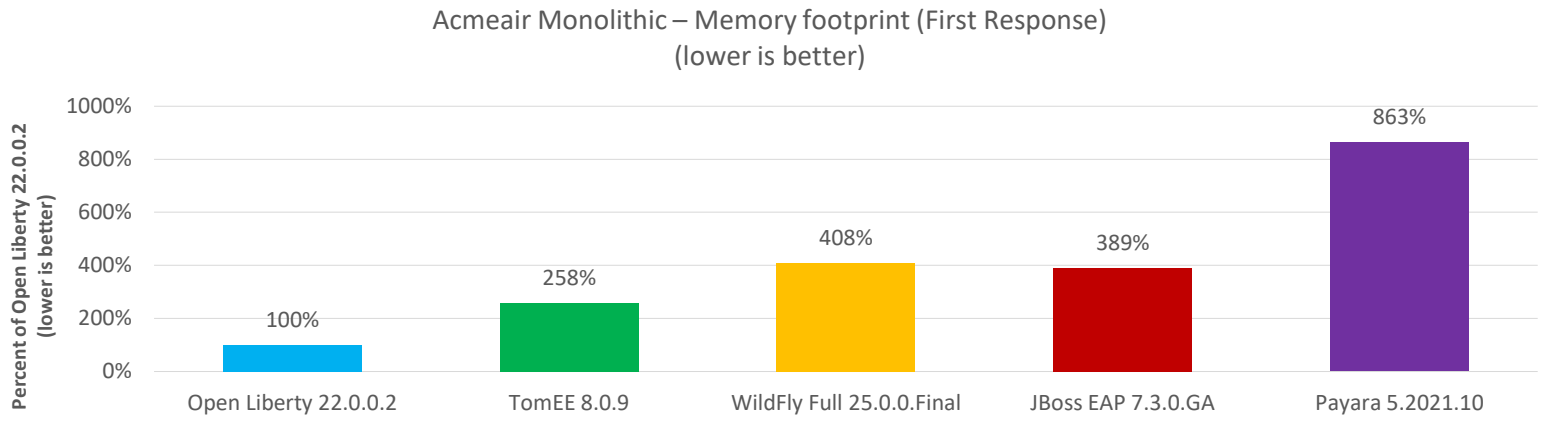
System Configuration:

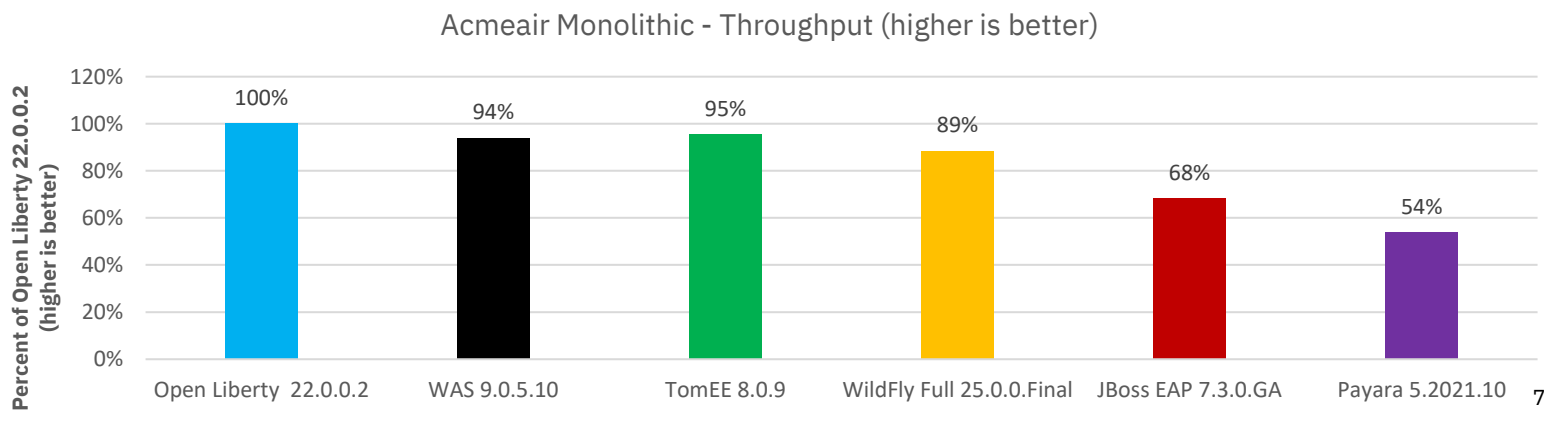
SUT: LinTel – SLES 12.4, Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, 4 cpus, 4GB RAM.
JDK version distributed with the docker images used for each server instance.

EE7 Performance (Acmeair Monolithic)

- Liberty outperforms others on all metrics for EE7 performance (startup time, throughput and memory footprint are much better)
- In this case Liberty throughput is better than WAS Traditional
- Comparisons used each application server's Docker image







System Configuration:

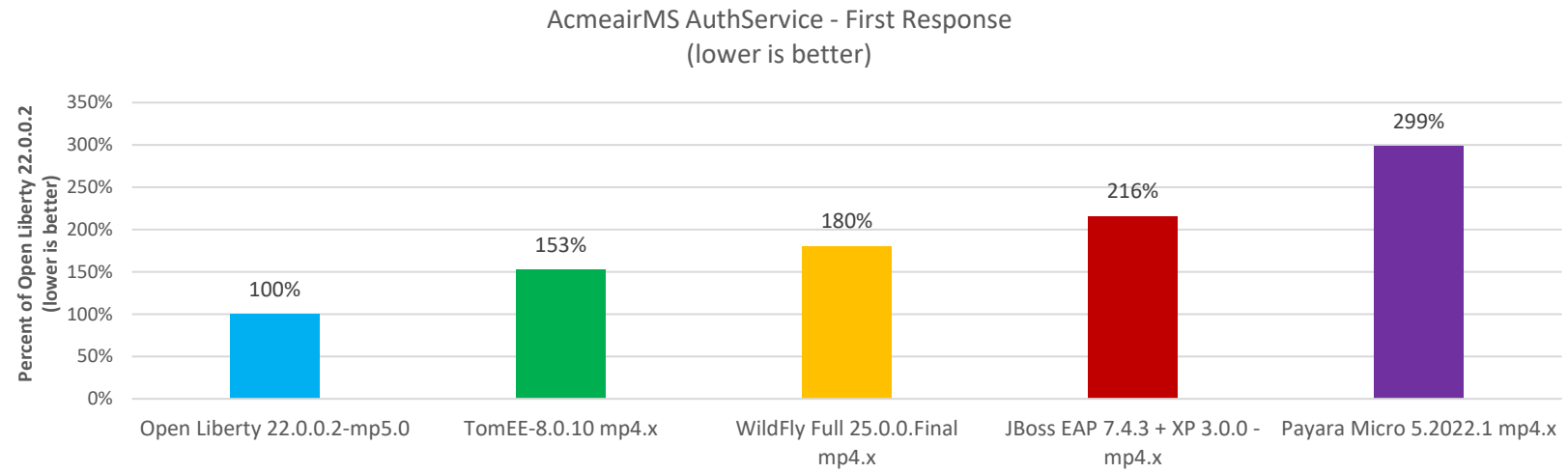
SUT: LinTel – SLES 12.4, Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, 4 cpus, 4GB RAM.
JDK version distributed with the docker images used for each server instance.

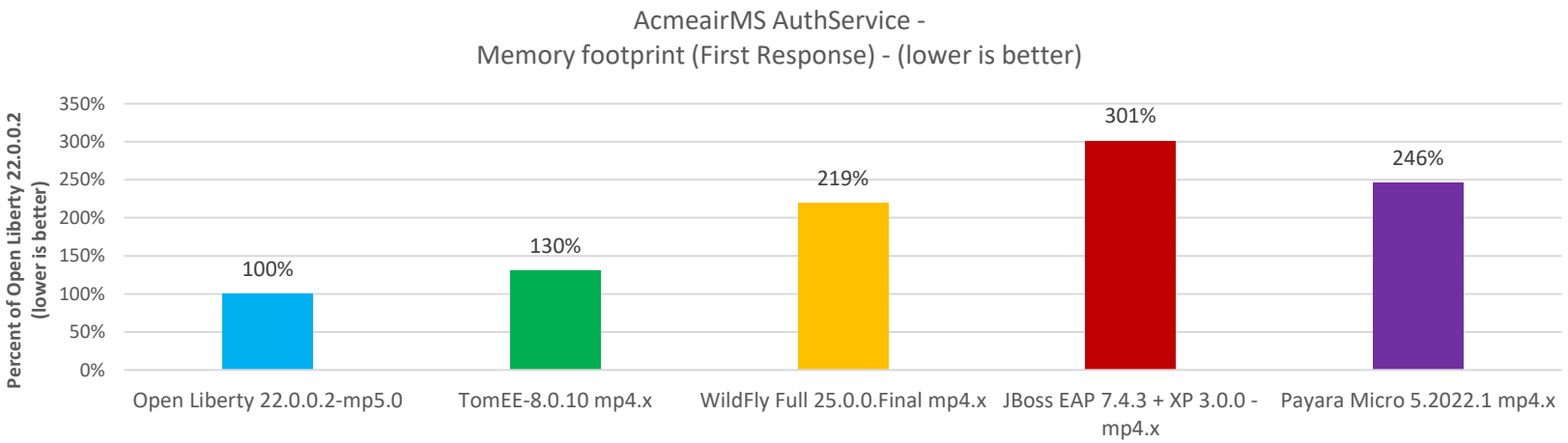
MicroProfile Performance (Acmeair Microservices)

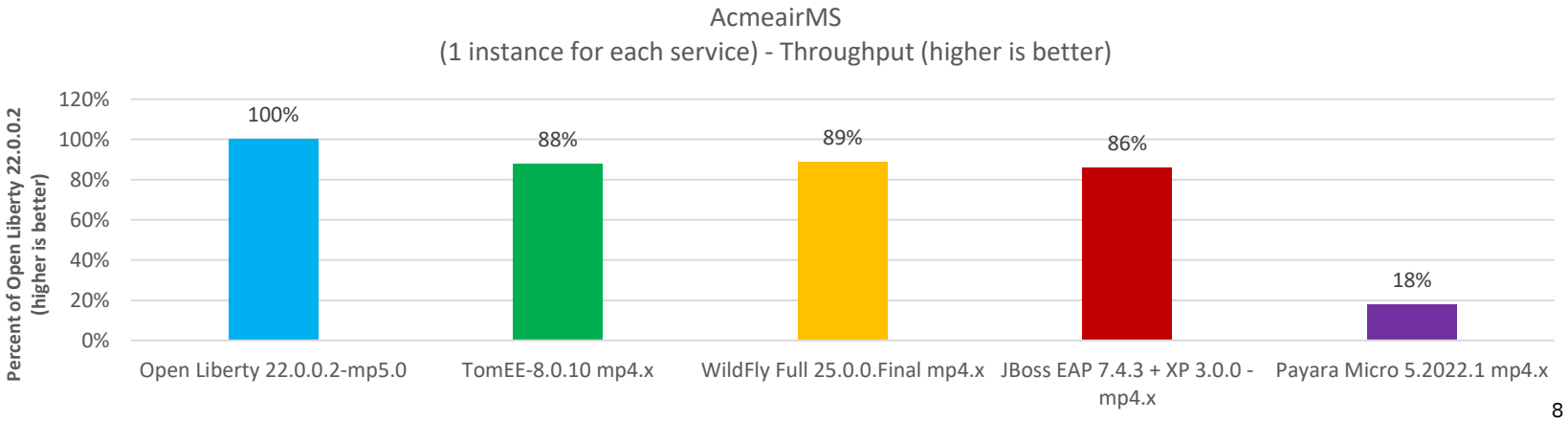
- Liberty provides the most balanced performance across all MicroProfile implementations (startup time, throughput and memory footprint are all best-in-class)
- Comparisons used each application server's MicroProfile (latest spec version supported) Docker image

System Configuration:

SUT: LinTel – Ubuntu 16.04.6 LTS Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, 2 cpus and 1gb memory set in docker container.
JDK version distributed with the docker images used for each server instance.

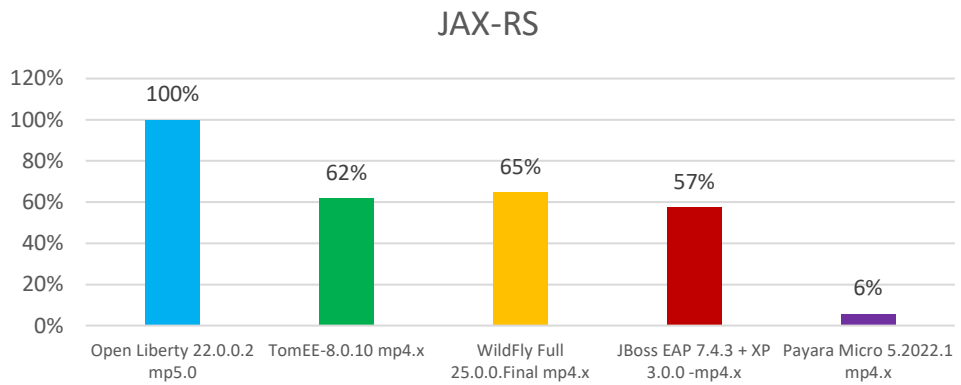


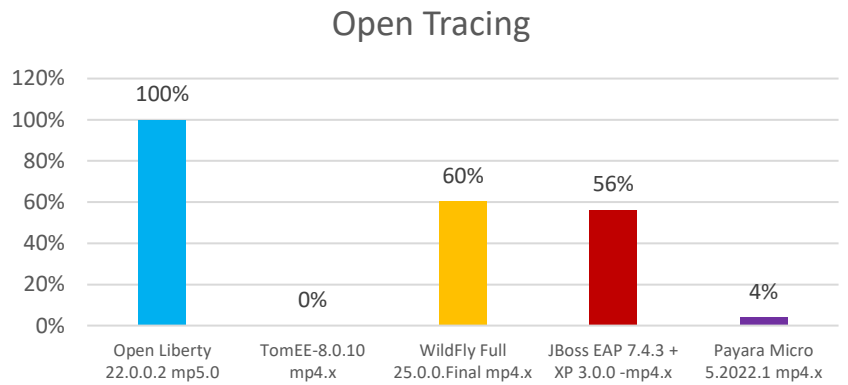


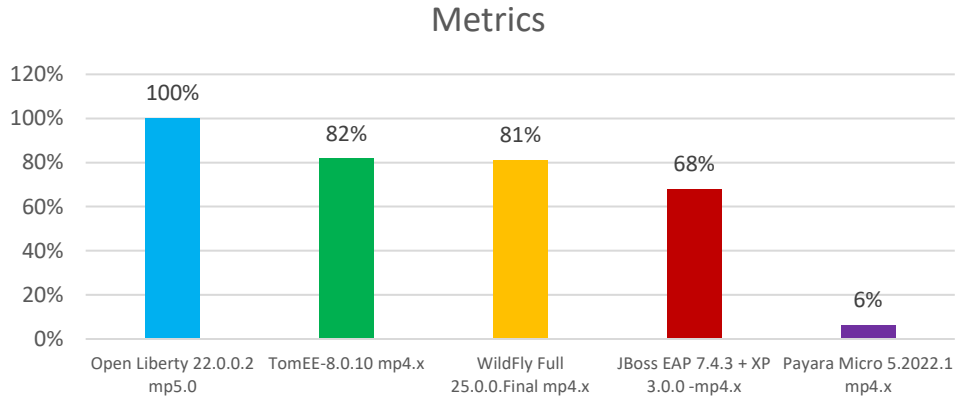


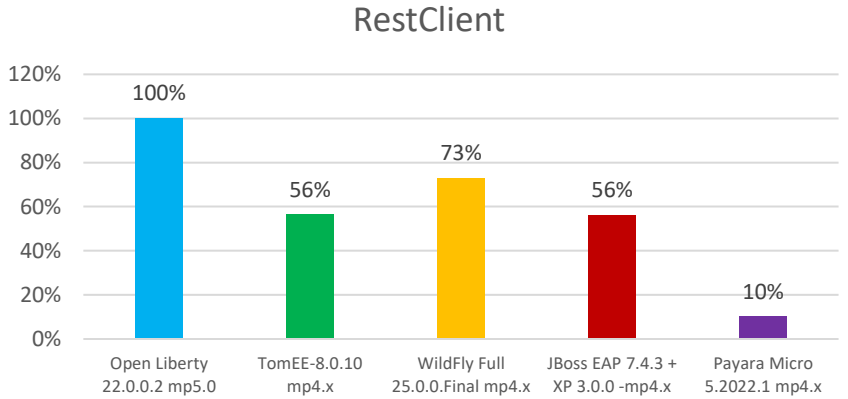
MicroProfile Performance (Acmeair Microservices Primitives)

- Comparisons used each application server's (latest spec version) MicroProfile Docker image
- Liberty provides the best implementation of all the MicroProfile features measured using primitives focused on different parts of the spec



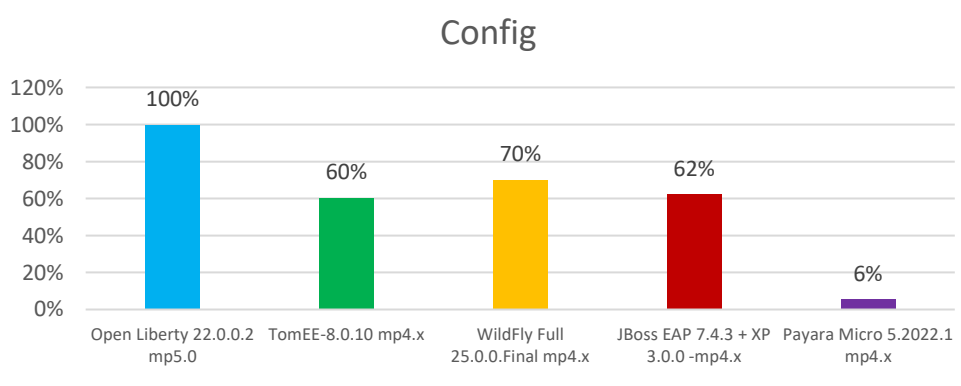


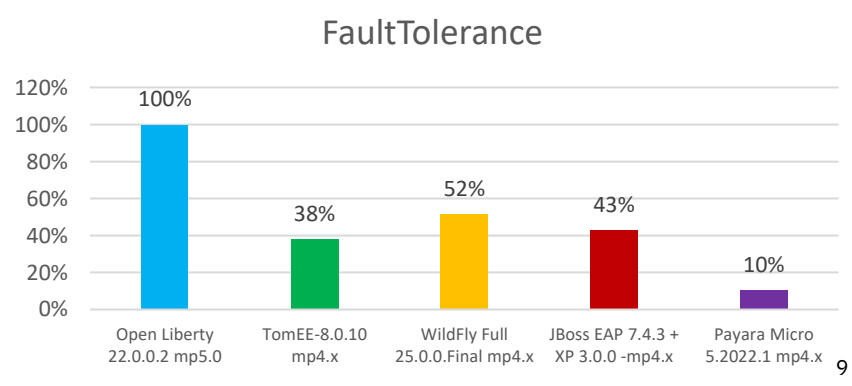




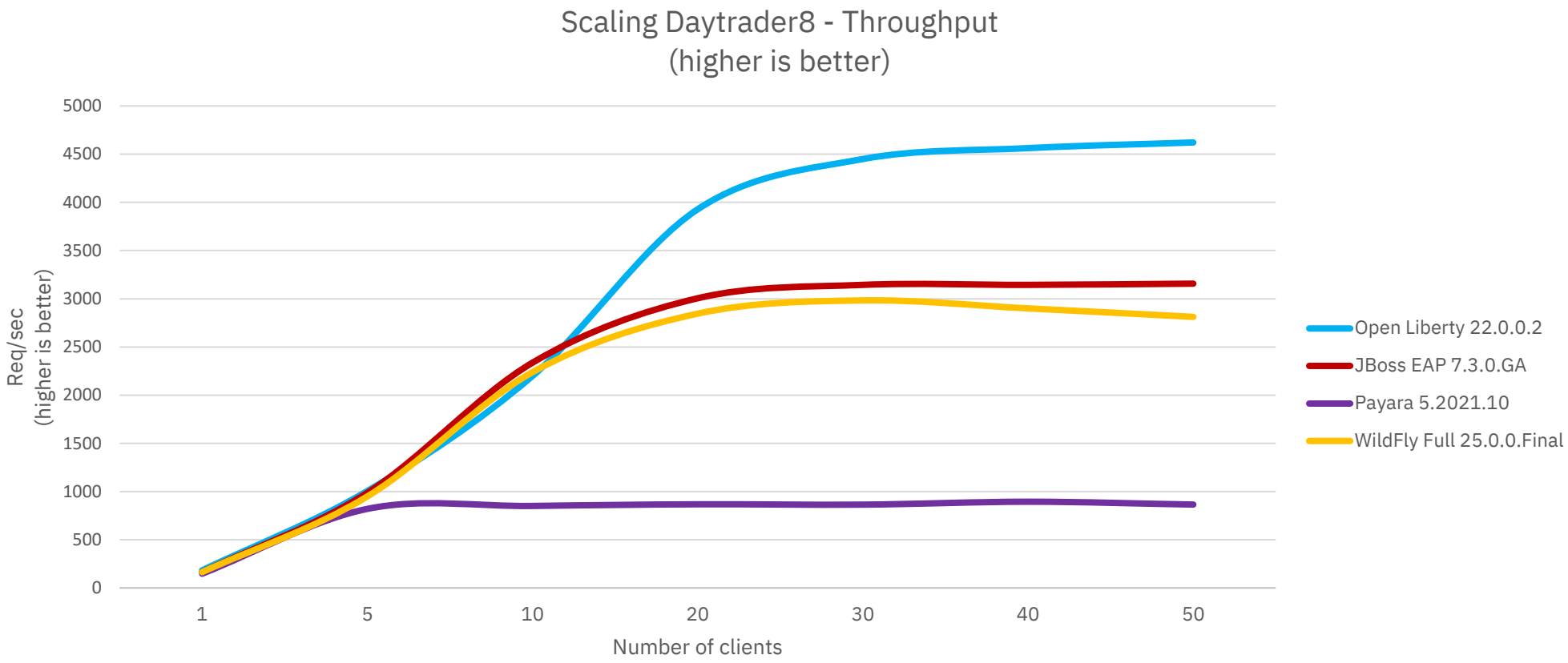
System Configuration:

SUT: LinTel – Ubuntu 16.04.6 LTS Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, 2 cpus and 1gb memory set in docker container. JDK version distributed with the docker images used for each server instance.





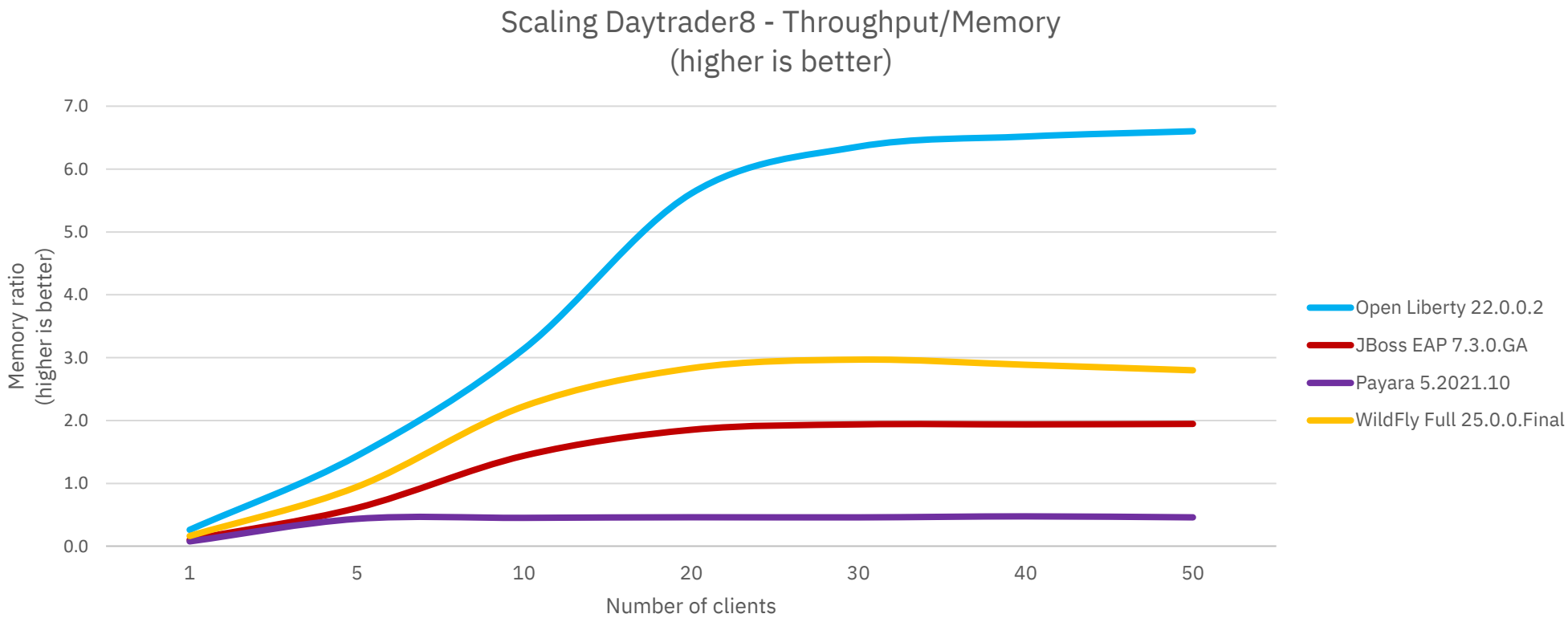
Liberty scales better than other frameworks (raw throughput)



System Configuration:

SUT: LinTel – SLES 12.4, Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, 2 cpus, 2GB RAM.
JDK version distributed with the docker images used for each server instance.

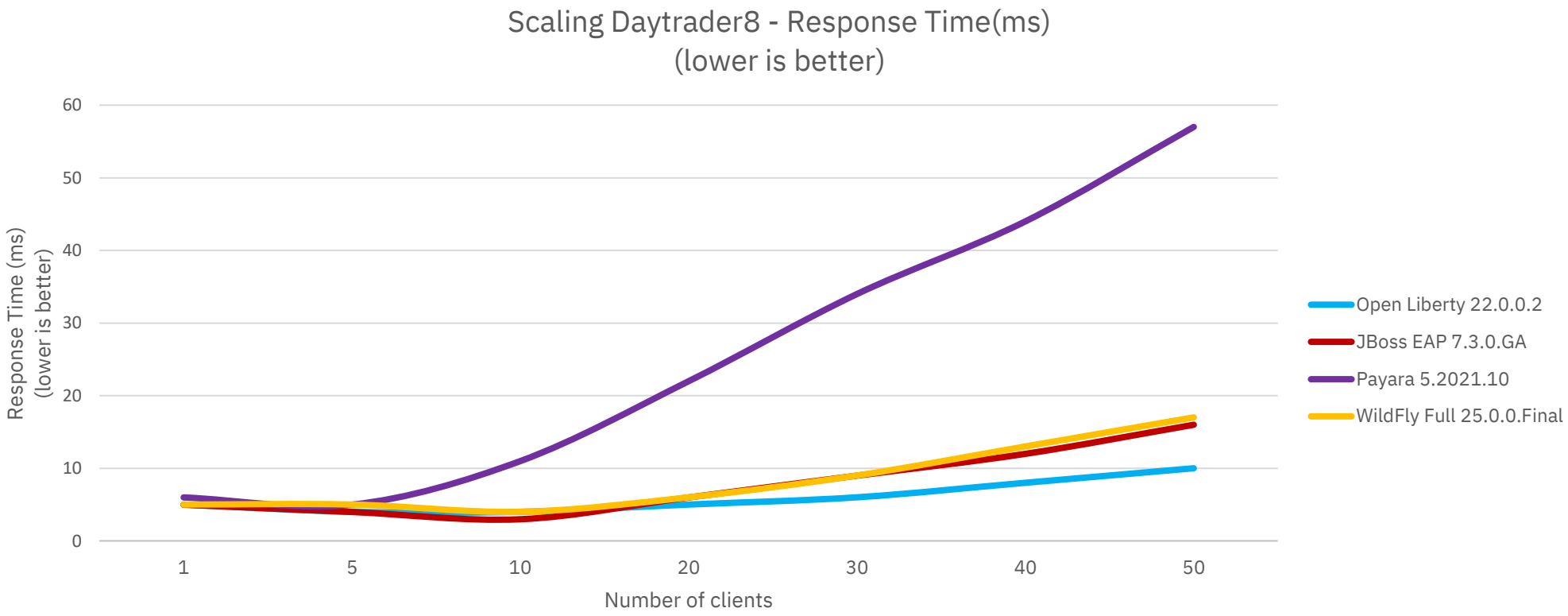
Liberty scales better than other frameworks (throughput:memory ratio)



System Configuration:

SUT: LinTel – SLES 12.4, Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, 2 cpus, 2GB RAM.
JDK version distributed with the docker images used for each server instance.

Liberty scales better than other frameworks (response time)

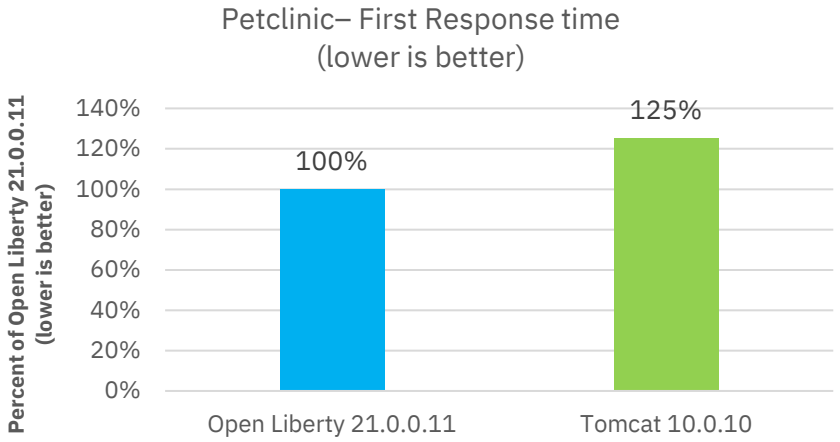


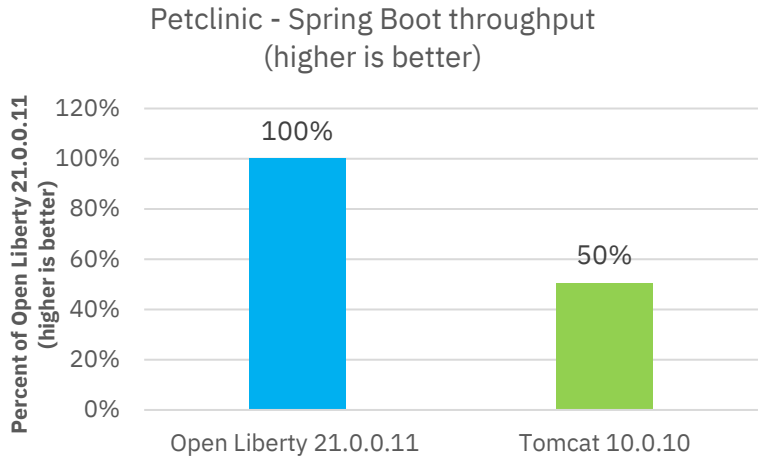
System Configuration:

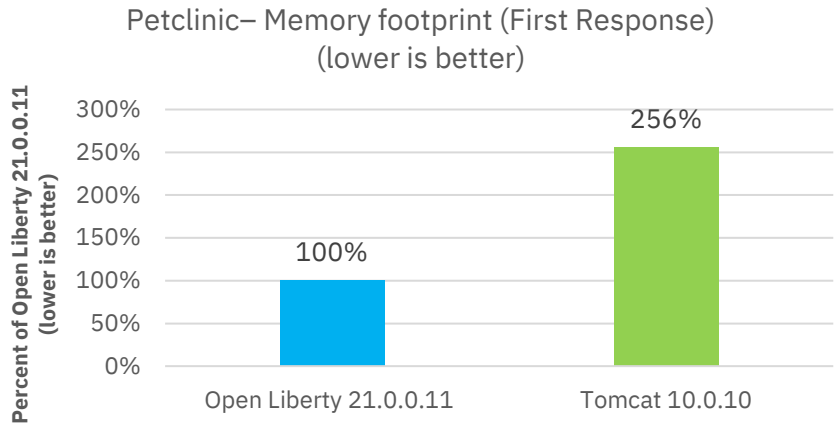
SUT: LinTel – SLES 12.4, Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, 2 cpus, 2GB RAM.
JDK version distributed with the docker images used for each server instance.

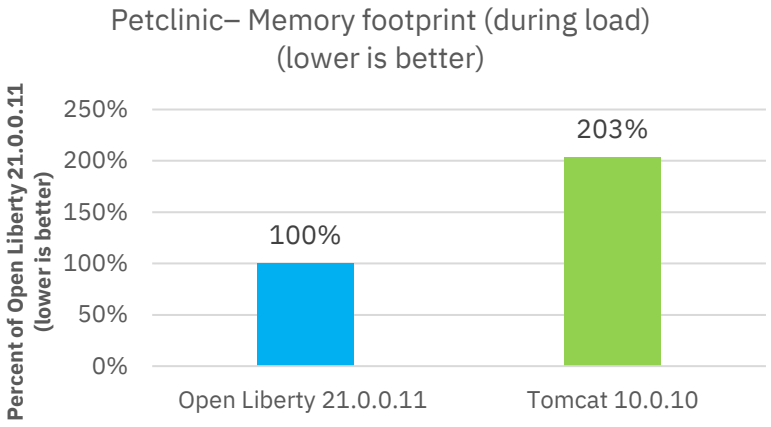
Better performance with Spring Boot on Liberty

- Open Liberty 21.0.0.11 outperforms Tomcat on the Pet Clinic Spring Boot application across all metrics
- **Open Liberty provides 2x throughput using less than half the memory footprint, and starts up 25% faster**







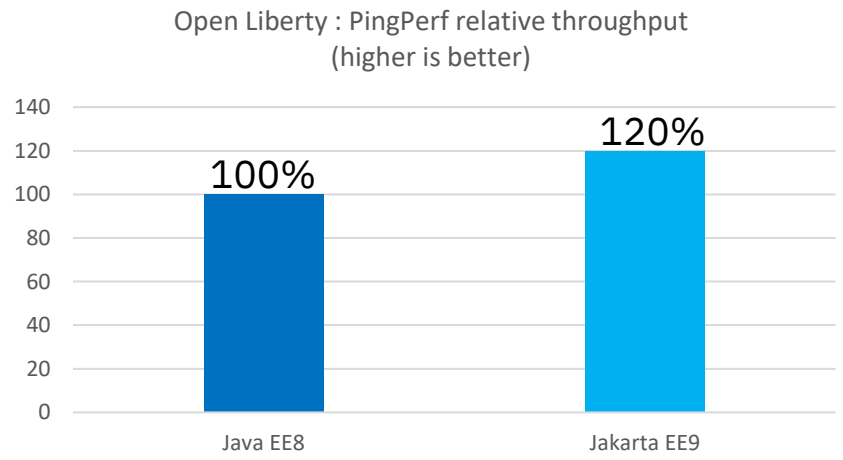


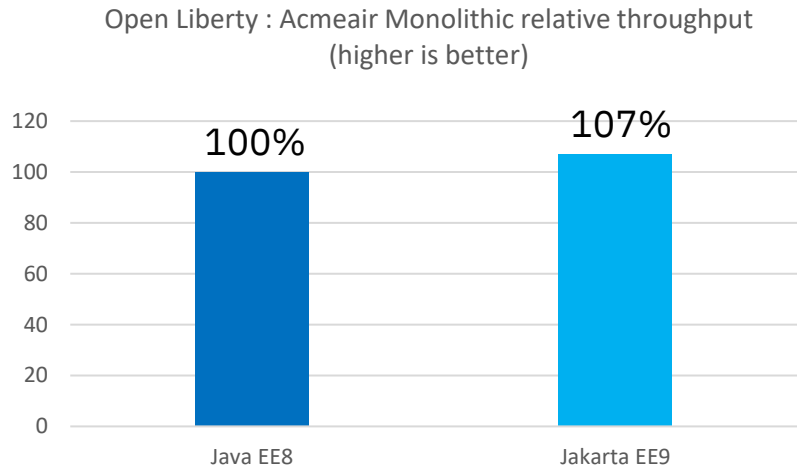
System Configuration:

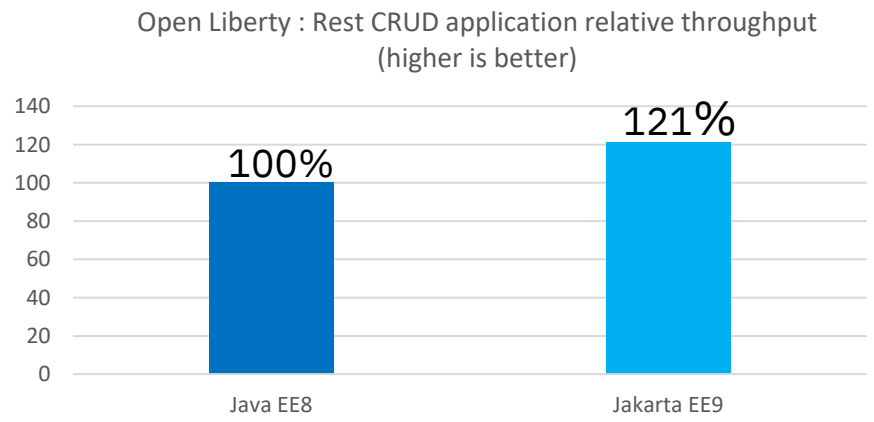
***SUT**:* LinTel – Ubuntu 20.04.4 LTS Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz, 2 physical cores, 4GB RAM.
JDK version distributed with the docker images used for each server instance.

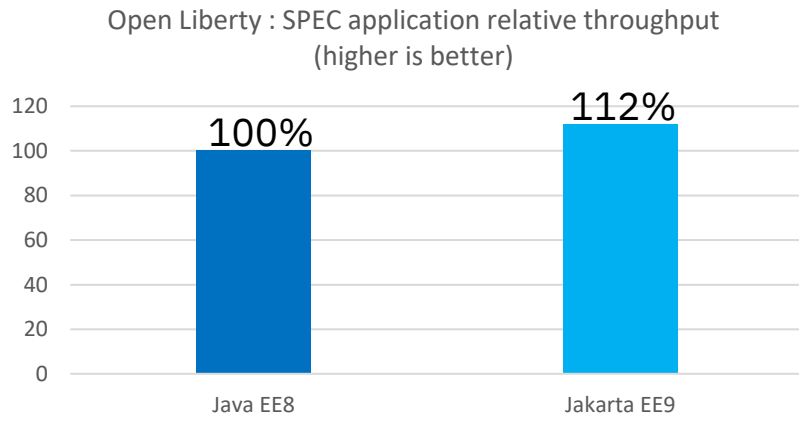
Better throughput with Jakarta EE9 implementation

- **Open Liberty Jakarta EE9 implementation has more performant REST implementation based on RestEasy**
  - Significant throughput benefits seen across different applications









System Configuration:

SUT: LinTel – SLES 12 sp3 - Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70 GHz, 256GB RAM. 2 physical cores allocated except for SPEC which used all 48 cores.

Summary: Open Liberty provides industry leading performance

- Open Liberty is the most performant and versatile Java framework
- Wins across metrics : throughput, startup time, memory footprint, scaling
- Wins across programming models : EE8, EE7, MicroProfile, Spring Boot
- Wins across deployment scenarios : containers as well as bare metal

Java optimizations for the cloud

OpenJ9 JVM : optimized to run in containers

Computing resources ==

Small deployment size	Recognize resource limits imposed by environment	Use computing resources judiciously	Fast start-up
• Use JRE instead of JDK • Use Jlink with JDK11 and later releases to minimize dependencies • Choose smaller OS base image, e.g. alpine	• Container awareness : CPUs and memory limits • Allow user to specify heap size as a percent of container memory size • Use larger –Xmx default value in containers	• Reduced memory use to work well in shared environment by default • -Xtune:virtualized enables further reduction in CPU and memory use	• More content in shared classes cache (SCC) and better quality AOT compiled code • Embed SCC in containers • Multi-layered SCC for different container layers • Portable AOT : easier container build with embedded SCC

Goal : Keep each container small, scale out new container instances quickly

Java landscape : from native image to full JVM

Positives

1. Extremely fast startup time (less than 50 ms)
2. Small memory footprint (less than 30mb) on startup
3. Small on-disk footprint (no bytecodes, no interpreter, no JIT etc.)
4. Very small container image size
5. Native OS executable image

Negatives

1. Only runs a subset of existing Java workloads and tooling
   - Developer experience mismatch
2. Not designed for intensive / long running applications
3. Long time (tens of minutes sometimes) to build native image
4. Peak throughput is much lower than that of JVM mode
5. GC technology is not as good as in JVM mode
6. Requires work to keep up with new Java versions
7. Different technology = different bugs and behavior

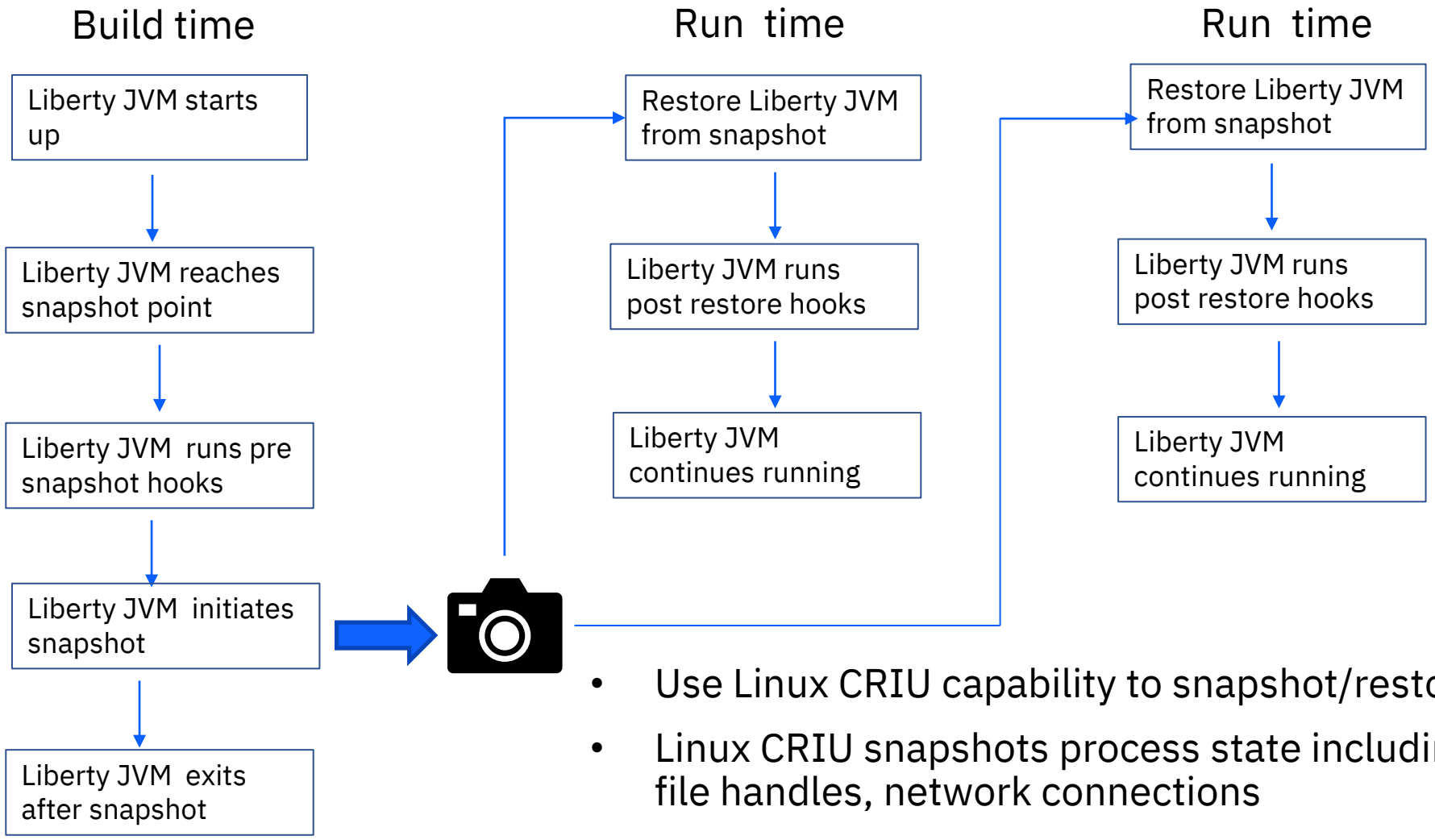
Positives

1. Full Java capabilities available
   - Dynamic class loading, Reflection, Serialization
   - Monitoring agents, JNI
   - ....
2. Runs full suite of applications and tooling
3. Dynamically adapts to program behavior
   - AOT, Interpreter, JIT compiler

Negatives

1. Runtime memory use typically higher
2. Slower startup
3. Larger on-disk footprint

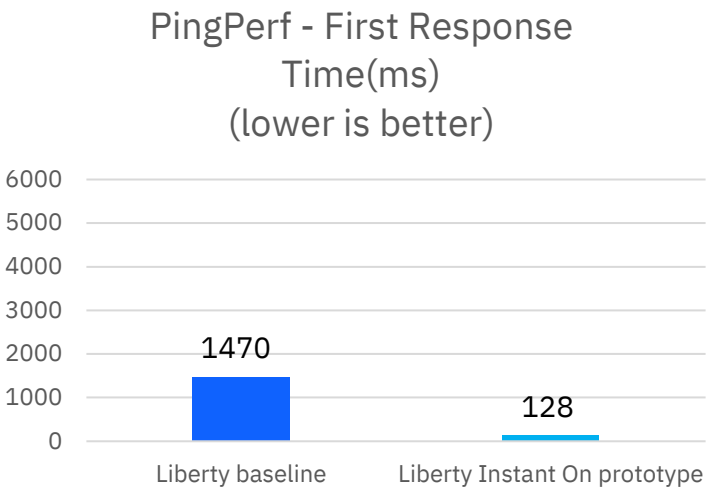
What is Liberty Instant On ?

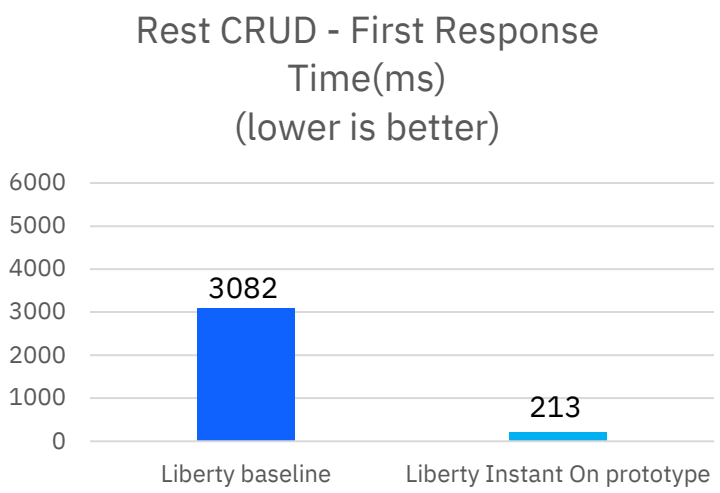


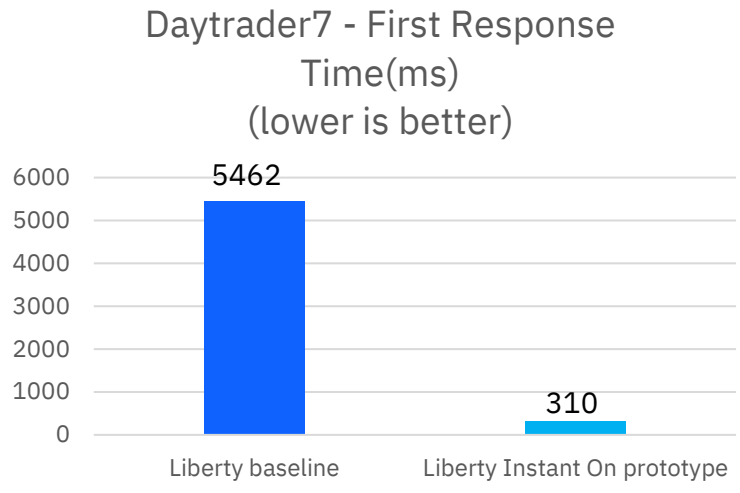
- Use Linux CRIU capability to snapshot/restore Liberty process
- Linux CRIU snapshots process state including memory pages, file handles, network connections
- Liberty and OpenJ9 run pre-snapshot and post-restore hooks to enable seamless user experience

Goal : “instant on” without limiting programming model/capabilities substantially

Linux CRIU capable of reducing startup time dramatically







- ~10X reduction in first response time by using Liberty Instant On **prototype** using CRIU
- Liberty Instant On is an effort to explore productization of CRIU use with Liberty
  - Main design goals
    - Usability : make it easy for an end user to consume the feature
    - Functionality : make it possible for end user to run their application without changes

System Configuration:

SUT: LinTel – SLES 15 sp3 - Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz, 2 physical cores(4 cpu), 32GB RAM.

Liberty Instant On : making it easy to use

Liberty will make it easy to specify the startup phase where to take a snapshot

- After we have started up all the features, or we have started up the application
- SPI allows Liberty features and user features to participate in the prepare/restore steps

Pre-requisites packaged into Liberty container image to avoid user having to do so

- Brings the right versions of Liberty, OpenJ9, CRIU, OS together via the Liberty docker file
- Specifies command line options needed to acquire the relevant privilege levels

Make it easy for user to build Liberty application container image with a snapshot

- Run application until Liberty initiates CRIU snapshot if the user requests it
- Bundles CRIU snapshot in new layer as part of Liberty application container build step

Sensible Liberty defaults to make end user experience as seamless as possible

- Tries to restore from CRIU snapshot automatically if snapshot files are detected
- Maximizes chances of successful restore, do not penalize JVM mode startup code paths
- If CRIU restore failed for some reason, start Liberty in JVM mode

Liberty Instant On : making it easy to try existing apps

Co-operative “hook” architecture across Liberty/OpenJ9 will perform fix-ups

- Checkpoint hooks run before snapshot is taken, Restore hooks run after restore
- Can specify ordering amongst the different hook methods to run
- Can specify if a hook is to be run either in single/multi threaded periods

Liberty/OpenJ9 hooks will handle differences in snapshot and restore environments

- CPU versions, number of CPUs, amount of RAM, different locale, timezone etc.
- Immutable variables or variables from restore (deployment) environment, e.g. for config

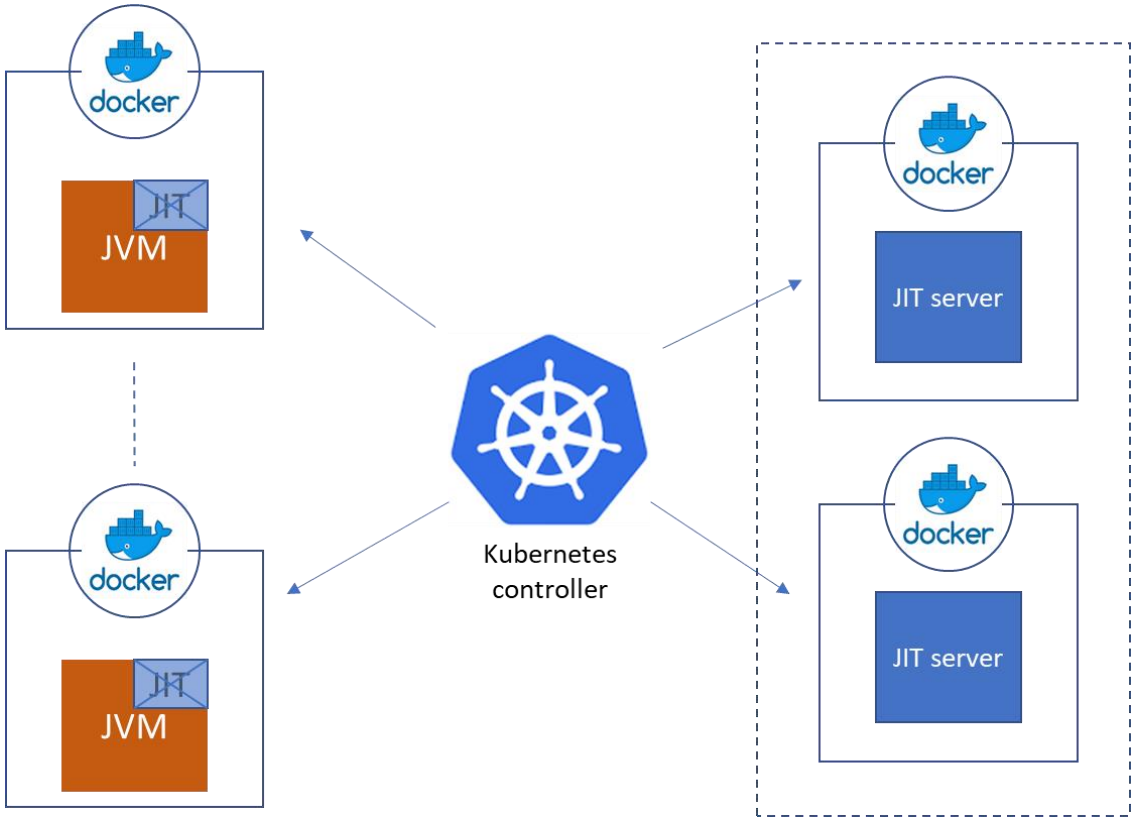
Liberty/OpenJ9 will remove any Java security “state” from the snapshot

- Only instantiate a barebones security provider and clear it before snapshot is taken
- Instantiate a full function security provider upon restore

Liberty/OpenJ9 will preserve “expected” application behaviors

- Different processes restored from one snapshot use different Random seeds
- Timer instances restored from snapshot must behave normally
- Snapshot taken before ports are opened, connections established after restore

What is JIT server ?

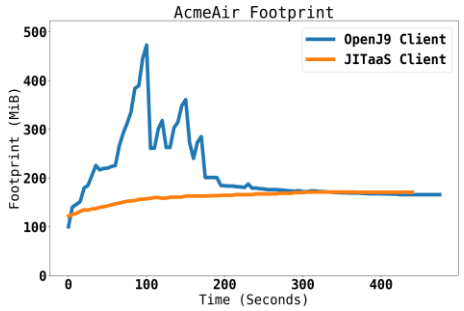


JVM client identifies methods to compile, but asks server to do JIT compiles securely

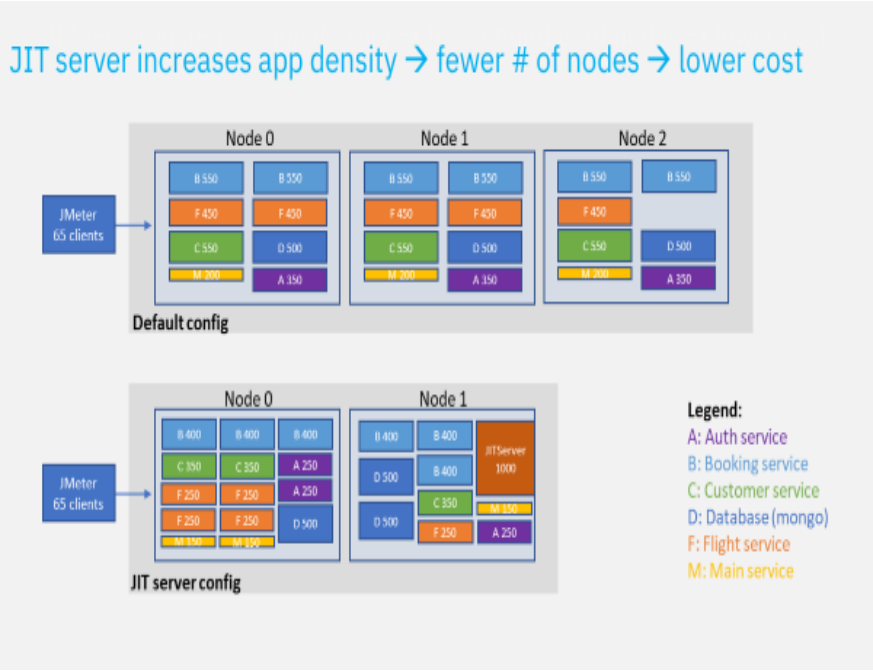
- JIT server asks questions to the client JVM (about classes, environment, etc.)
- JIT server sends generated code back to be installed in JVM client's code cache
- Decoupling of JIT from the JVM allows the client and the JIT to be scaled out independently
- JIT server is a supported feature in OpenJ9 on Linux X86 and Power platforms

Benefits of remote compilation at JIT server

- Remove JIT induced CPU and memory spikes from JVM clients
  - With JIT server, resource consumption driven by the Java application
  - Customers can right size their containers without considering JIT
  - Smaller application containers since JIT activity is remote
  - Reduction in cost from smaller containers
  - Better performance from increased container density



- Faster time to ramp up to peak throughput
  - JIT server can have more CPU and memory than client
  - More compilations can be done in parallel than at client
  - AOT code caching at the server further improves ramp up
  - Particularly important with a horizontal pod auto-scaler (HPA)



- Future : JIT server can do complex compiler analyses
  - May not be feasible in a traditional (in-process JIT) mode

Liberty performance across platforms

IBMJCEPlus is now the default JCE provider on most platforms

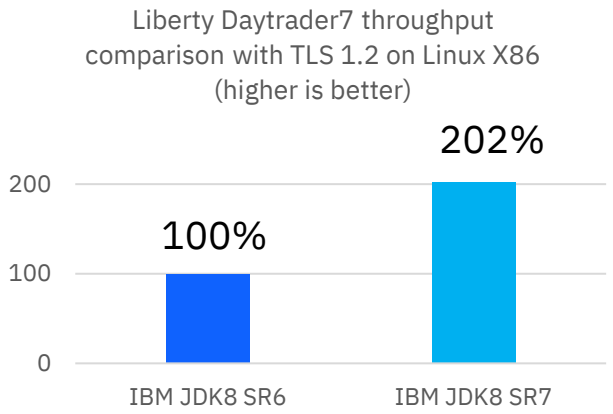
Default JCE provider in IBM SDK for Java 8 SR7 was changed on all platforms except z/OS

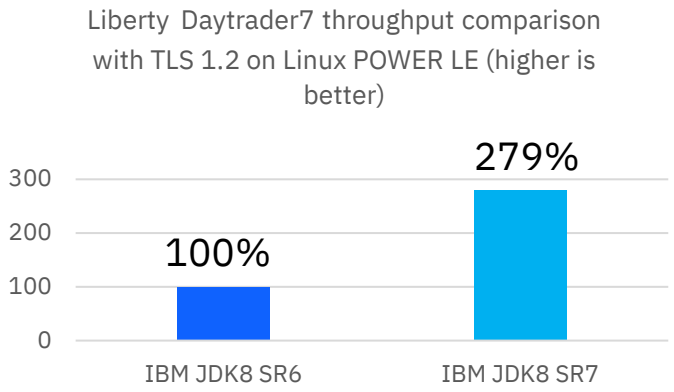
- Old defaults : IBMJCE and IBMJCEFIPS
- **New defaults : IBMJCEPlus and IBMJCEPlusFIPS**

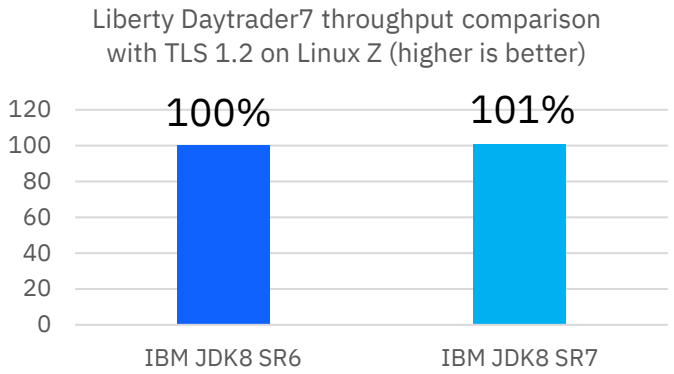
What is IBMJCEPlus ?

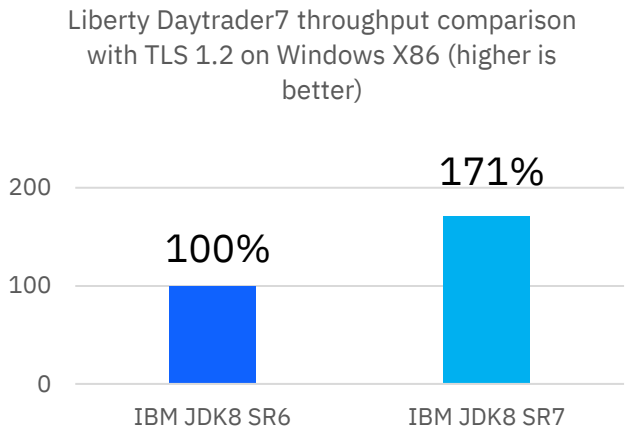
- Improves performance of several JCE algorithms used for authentication, encryption
- Based on existing IBM component called GSKit that is based on OpenSSL
  - OpenSSL has a large set of crypto algorithms that are extremely well tuned across platforms
  - IBMJCEPlus performs very well by offloading cryptography work to GSKit bundled with the JDK
  - TLS 1.3 support : newer algorithms required by TLS 1.3 are supported and accelerated
  - Some small number of capabilities implemented by IBMJCE are not implemented in IBMJCEPlus
  - New providers are available on z/OS as well, but not default (though it may change in future)

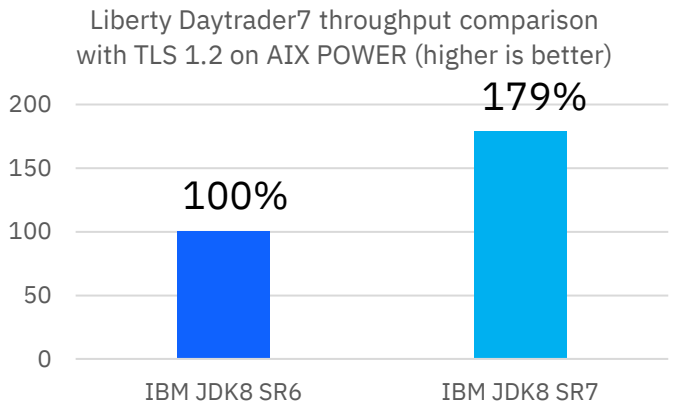
Liberty TLS 1.2 throughput improvements in IBM JDK8 SR7









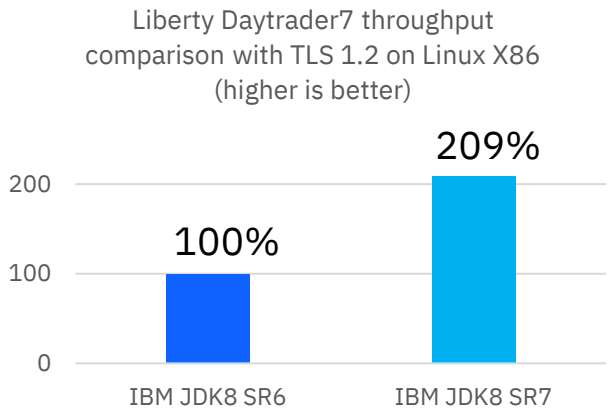


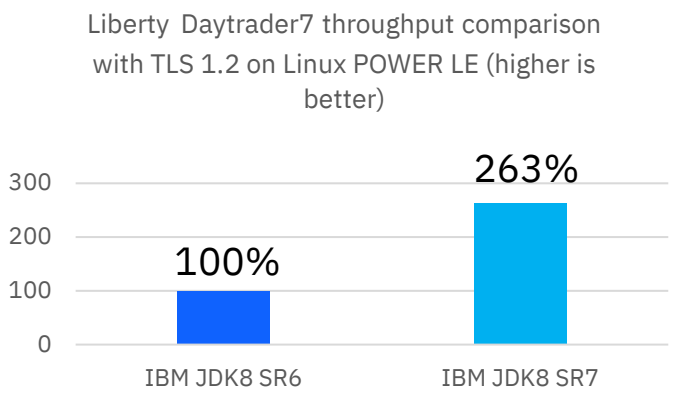
Note : IBMJCE was very performant on IBM Z and so the switch to IBMJCEPlus in IBM JDK8 SR7 did not improve performance on Linux on Z

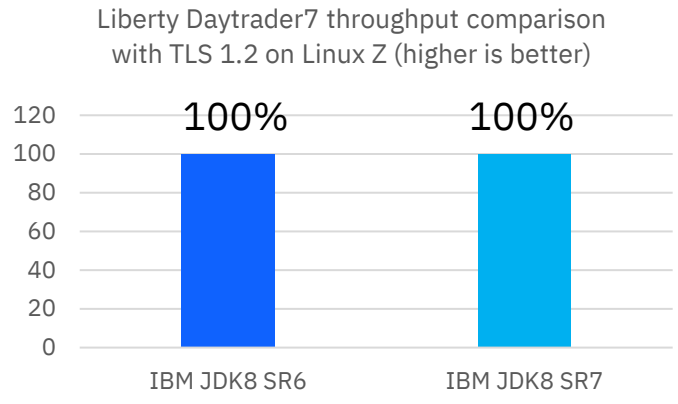
Large throughput improvements on X86 and POWER platforms were seen out-of-the-box for most JCE algorithms

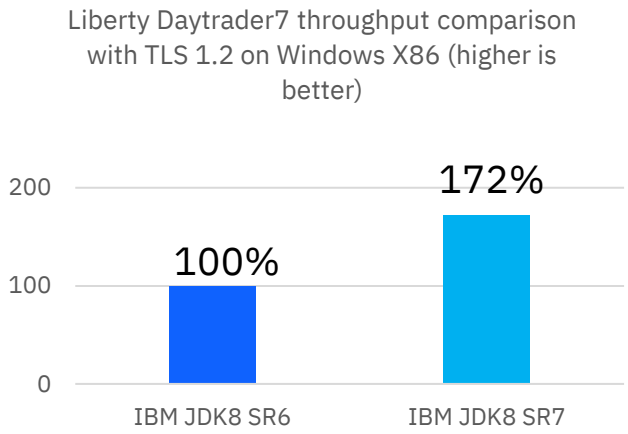
- **TLSv1.2 with https.cipherSuites=TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (sessions reused)**
- **Significant improvement in TLS overhead with IBMJCEPlus provider made default in IBM JDK8 SR7**

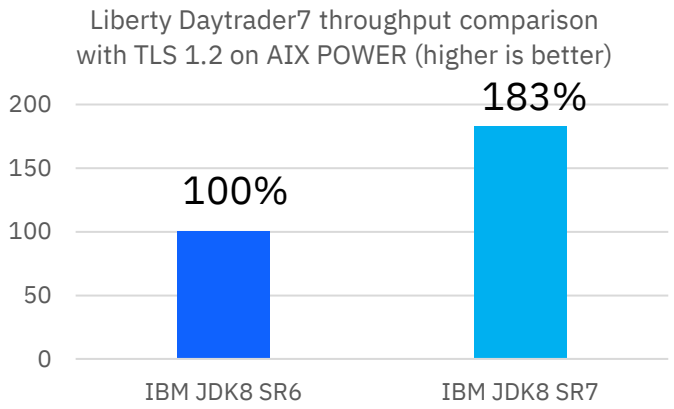
Liberty TLS 1.2 throughput improvements in IBM JDK8 SR7











Note : IBMJCE was very performant on IBM Z and so the switch to IBMJCEPlus in IBM JDK8 SR7 did not improve performance on Linux on Z

Large throughput improvements on X86 and POWER platforms were seen out-of-the-box for most JCE algorithms

- **TLSv1.2 with https.cipherSuites=TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (sessions reused)**
- **Significant improvement in TLS overhead with IBMJCEPlus provider made default in IBM JDK8 SR7**

POWER10 improved Java and Liberty throughput significantly

Application	Per-core POWER10/POWER9 throughput ratio	Per-socket POWER10/POWER9 throughput ratio
Java SE transactional benchmark	1.35X	1.6X
Liberty Daytrader7 (Java EE7) benchmark	1.6X	2X

Above results were collected on Linux POWER LE but AIX results are similar
POWER10 socket has 15 cores, POWER9 socket has 12 cores leading to additional per-socket differences

- Liberty on OpenShift (OCP) comparison running on POWER10 vs Intel CascadeLake
  - **POWER10 achieved ~4x pod density while maintaining similar transaction latency**

- POWER10 technology that helps Liberty and Java applications
  - **Larger L2 cache (2X), higher memory bandwidth, more sophisticated instruction pipelines**

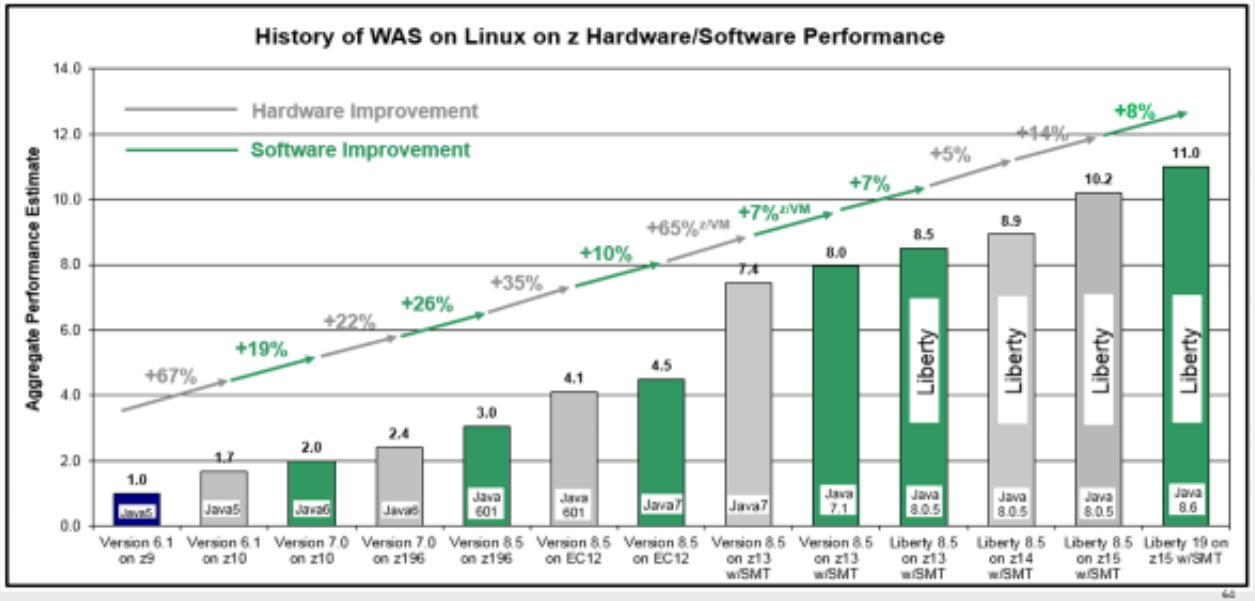
Java and Liberty performance on IBM Z

The new IBM Telum chip

will power the next generation of IBM Z systems

Stay tuned for more on this front as Java will be exploiting IBM Telum features for performance!

History of WebSphere Throughput Performance on Linux on Z



- Historical trend is for a good uplift in WAS and Liberty throughput performance from every new hardware-software generation
- New cache structure, SIMD and crypto enhancements will be exploited

Summary : Innovations galore in Java and Liberty performance

- Open Liberty + OpenJ9 : an awesome combination to run Java
- Open Liberty : the most flexible framework for all deployments
- Exciting innovations coming for Liberty in containers and on IBM Systems
- Stay current : performance features being released through continuous delivery

IBM TechCon 2022

Automate, integrate and turn vision into action

The technical conference for thought leaders, practitioners, and creators

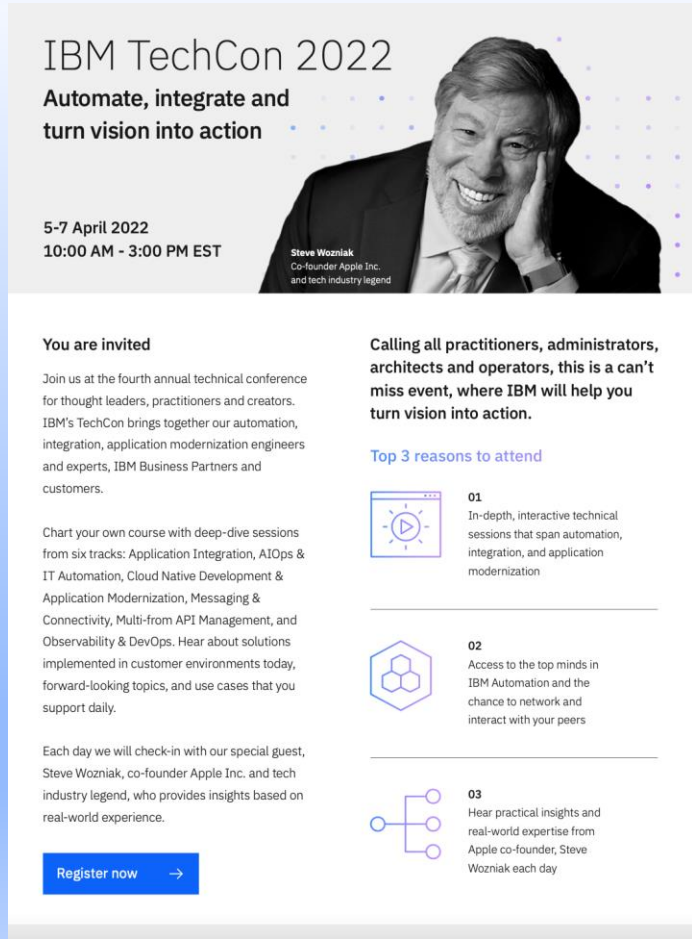
April 5-7, 10-3p ET

Registration is open!

W3 site with invitation, social, latest info here

72 live virtual sessions in 6 tracks

- Application Integration
- AIOps & IT Automation
- Cloud Native Dev & App Mod
- Messaging & Connectivity
- API Management
- Observability & DevOps



Top 3 Reasons To Attend:

- In depth, interactive technical sessions
- Access to the top IBM Architects and Engineers
- Insights from a Tech Industry legend, Steve Wozniak, each day

Join the WAS CAB

WebSphere Customer Advisory Board

Email:
claudiab@us.ibm.com

Webex:
https://ibm.webex.com/meet/claudiab

Community Resource:
http://ibm.biz/WASCABCommunityResources

Sign up:
http://ibm.biz/WebSphereAdvisoryBoard

Weekly meetings

Thursday and Friday
9:15 am ET

Join →

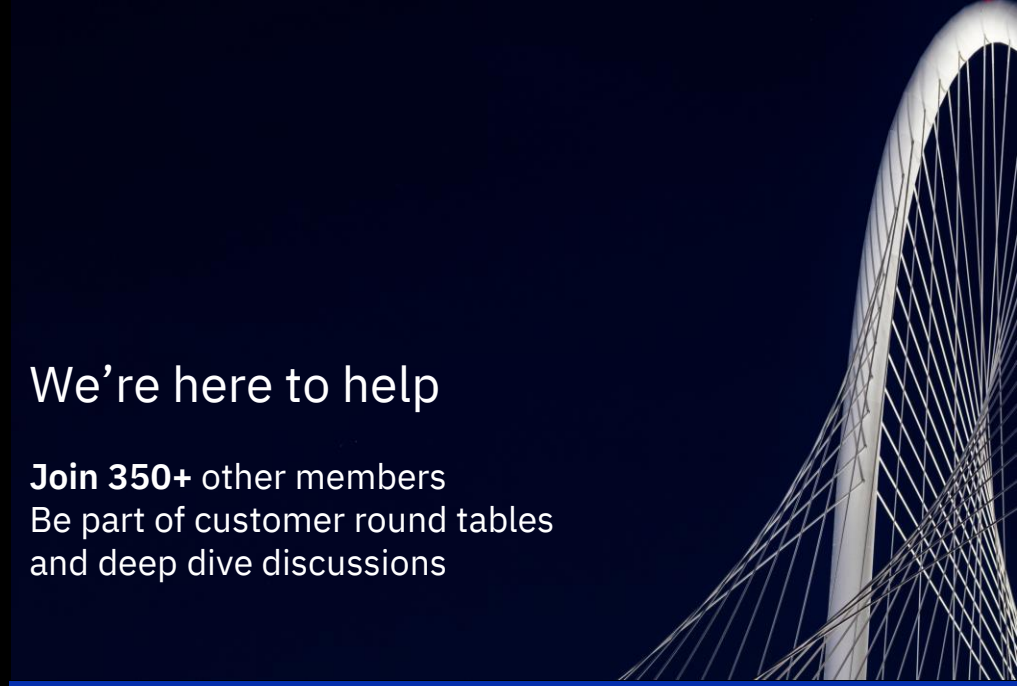
Monthly meetings

Time zone friendly sessions

Join →

Other Programs

- Feedback Programs
- Previews, Demos
- Labs, workshops
- 1-on-1



We're here to help

Join 350+ other members
Be part of customer round tables and deep dive discussions

Engage at your own pace:

- Stay in the loop at meetings
- Share solutions and pain points
- Connect with other customers
- Access to resources and experts
- Customized meetings
- Special offers
- All Partners are welcome

Be heard and get a chance at $100

The annual WebSphere & Liberty Community survey is underway, and two survey participants will win a $100 gift card.

Help shape future offerings, services, and events!

Go to the survey →

Open Liberty

Useful Links

Why choose Liberty for Microservices
https://ibm.biz/6ReasonsWhyLiberty

Choosing the right Java runtime
https://ibm.biz/ChooseJavaRuntime

How to approach application modernization
https://ibm.biz/ModernizeJavaApps

Open Liberty Site
https://www.openliberty.io

Open Liberty Guides
https://www.openliberty.io/guides

https://openliberty.io →

Explore Liberty & Runtimes

Read

Forrester's: Total Economic Impact of Liberty within WebSphere Hybrid Edition

Watch

Liberty: reducing costs and increasing agility in hybrid-cloud

Join

WAS & Liberty Community

Read

CCS Insight Report: Renewal and Transition for Java Technology

Watch

Java Revolution through Renewal and Transition

Thank you.