

WebSphere Application Server for z/OS Version 8.5 Liberty Profile

David Follis
z/OS Runtime Architect



Disclaimer

- The information contained in this documentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this document, it is provided "as is" without warranty of any kind, express or implied.
- This information is based on IBM's current product plans and strategy, which are subject to change without notice. IBM will not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation.
- Nothing contained in this documentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of the IBM software.
- Performance is 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.
- 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.

What is the Liberty profile?

A lightweight, dynamic, composable runtime

Lightweight

- Server install is only about 55 MB
- Extremely fast server starts – typically well under 5 seconds

Dynamic

- Available features are user selected and can change at runtime
- Restarts are not required for server configuration changes

Composable

- Features are implemented as loosely coupled components with lazily resolved optional and mandatory dependencies
- The availability of features and components determines what Liberty *can* do and what's available to applications

What is the Liberty profile?

An easy to configure runtime environment

- Simple, extensible, and sparse configuration model
 - Configuration can live in a single XML document
 - Configuration is by exception
 - Defaults are provided by contributing feature
 - Only modifications to the defaults are required
- Flexible configuration structure
 - Include mechanism allows for shared configuration elements
 - Variable indirection mechanism allows for customization when distributed across multiple JVMs
 - Easily managed by version control systems if desired

What is the Liberty profile?

A transportable runtime for your applications

Use “server package” to generate an archive that contains a tested, self-contained, pre-configured server instance that includes your application

- Enables an application-centric deployment model that allows for easy scale-out
- Light-touch admin builds on the ND job manager infrastructure to manage Liberty server instances

A runtime environment with fidelity to full WAS

- Liberty is WebSphere
- Applications that are developed and tested on Liberty will run on the full profile

Lightweight configuration

```
<server description="tradeLiteServer">
  <featureManager>
    <feature>jsp-2.2</feature>
    <feature>jdbc-4.0</feature>
  </featureManager>
```

Features control what's available in the runtime

```
<logging consoleLogLevel="INFO" />
```

Singleton configurations specify properties for runtime services for which there is only one instance

```
<application type="war"
  id="tradelite"
  name="tradelite"
  location="${shared.app.dir}/webcontainer/tradelite.war" />
```

Instance configurations allow multiple instances of resources and applications to be declared

Includes can be used to implement an extensible configuration model

```
<include location="jdbc-drivers.xml" />
<include location="${user.home}/custom.xml" optional="true" />
```

```
<dataSource id="jdbc/DerbyTradeDataSource"
  jndiName="jdbc/TradeDataSource"
  jdbcDriverRef="DerbyEmbedded">
  <properties databaseName="${shared.resource.dir}/data/tradedb" />
</dataSource>
</server>
```

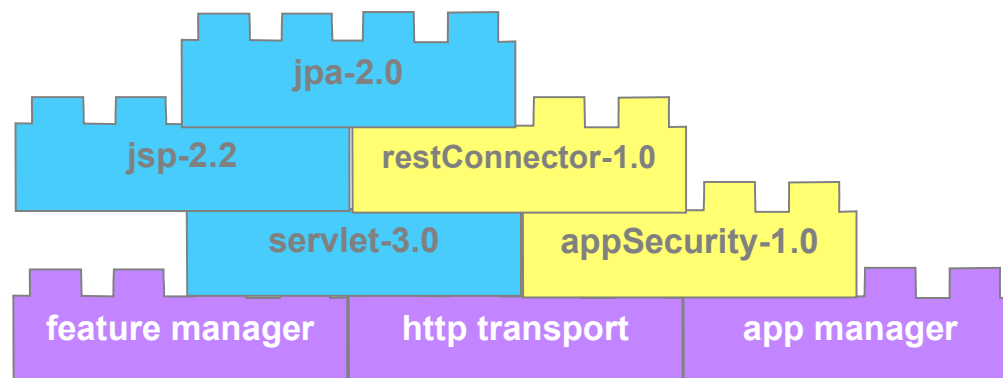
References can be used in multiple elements to point share a common definition

Composability – Based on *features*

```
<server description="composabilityIsKey">

  <featureManager>
    <feature>appSecurity-1.0</feature>
    <feature>jsp-2.2</feature>
    <feature>restConnector-1.0</feature>
    <feature>jpa-2.0</feature>
  </featureManager>

</server>
```

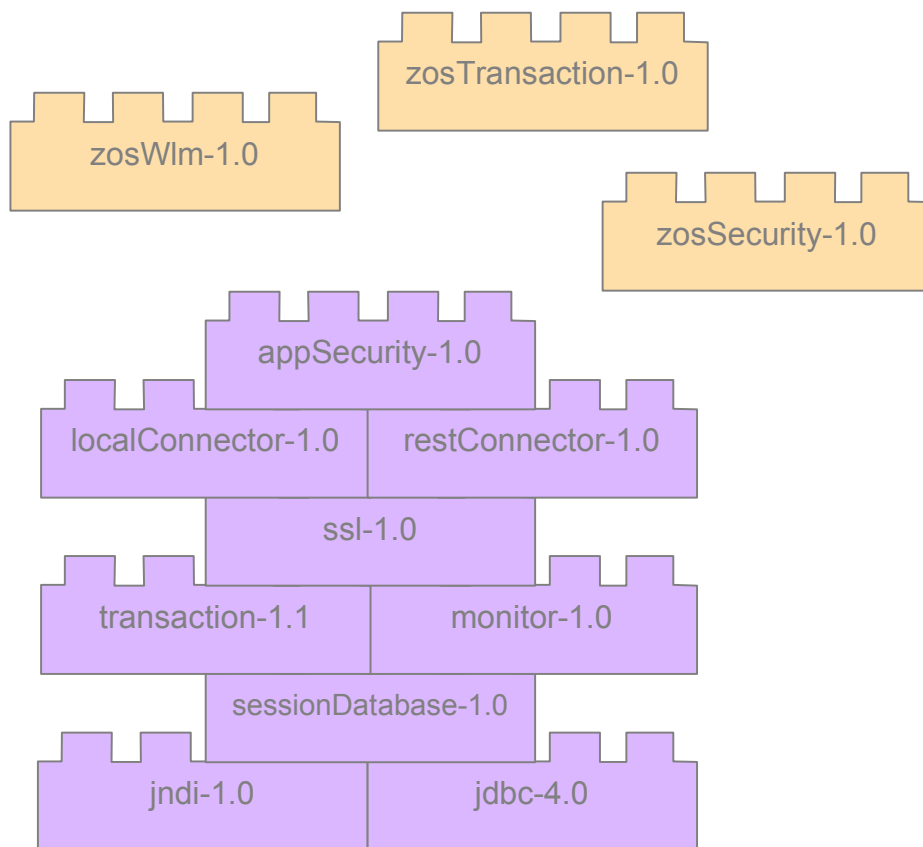


What is the WAS for z/OS Liberty profile?

The WAS for z/OS Liberty profile is Liberty with *optional*, independently enabled *extensions* that exploit z/OS facilities

- Only enable exploitation of z/OS features you need
- Only configure the z/OS functions you use

Focus of v8.5 is basic integration and exploitation



z/OS Feature Sets



Common Feature Sets

Liberty and traditional profile capabilities

There are functional differences between traditional WAS and the Liberty profile – Liberty provides a useful subset of traditional WAS

Liberty Profile

- Bean validation
- Blueprint
- Java API for RESTful Web Services
- Java Database Connectivity (JDBC)
- Java Naming and Directory Interface (JNDI)
- Java Persistence API (JPA)
- Java Server Faces (JSF)
- Java Server Pages (JSP)
- JMX
- Monitoring
- OSGi JPA
- Remote connector
- Secure Sockets Layer (SSL)
- Security
- Servlet
- Session Persistence
- Transaction
- Web application bundle (WAB)
- z/OS Security (SAF)**
- z/OS Transactions (RRS)**
- z/OS Workload Management**

Traditional WAS Profile

Everything Liberty has...

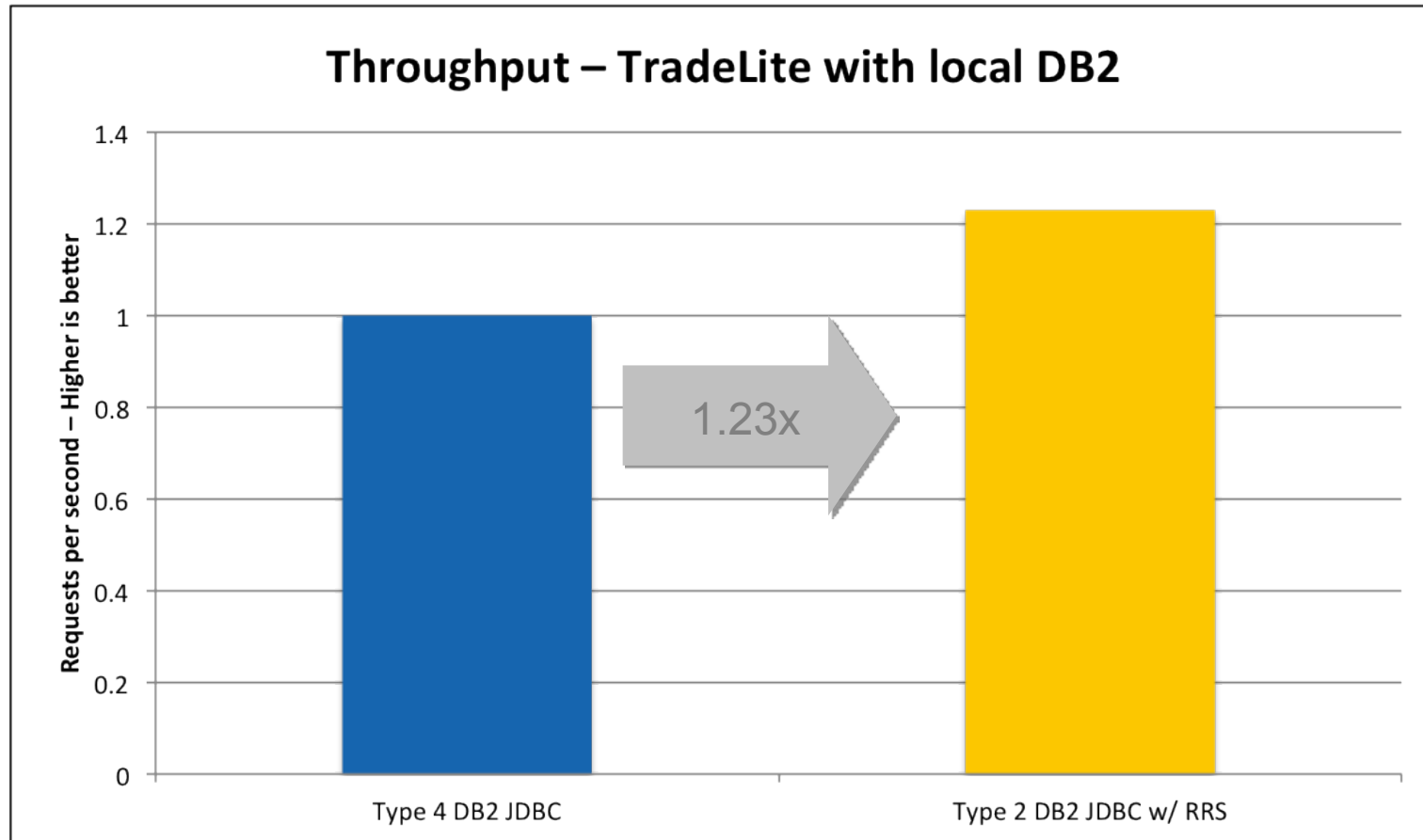


- Enterprise Java Beans (EJBs)
- Messaging (JMS)
- Web Services
- Service Component Arch (SCA)
- Java Connector Architecture (JCA)
- Clustering
- WebSphere Optimized Local Adapters
- Administrative Console
- WSADMIN scripting
- Multi-JVM Server Model**

And much more ...

Feature – z/OS Transactions: Performance

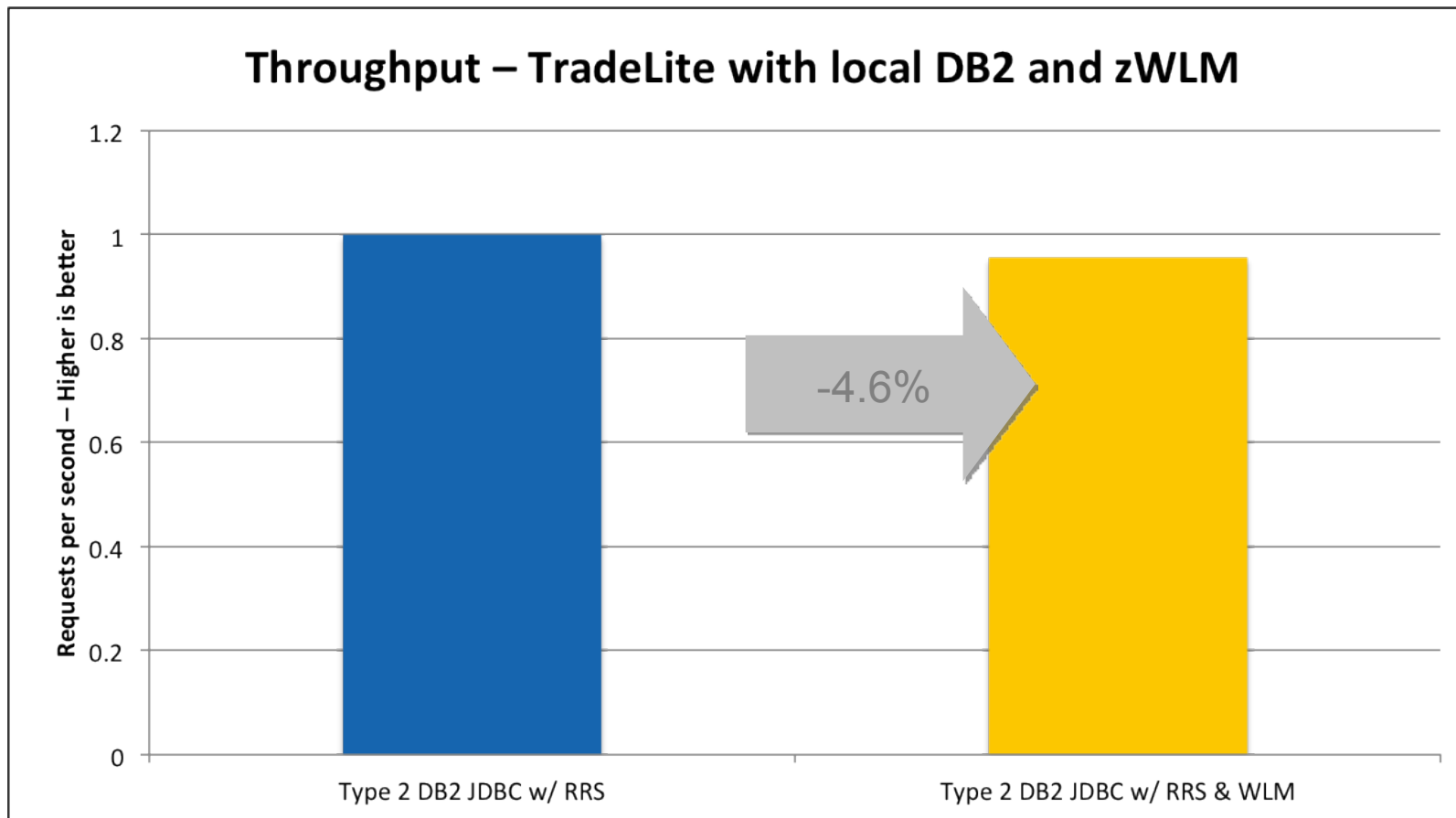
Optimized local connectivity for higher throughput



- z196, 4-way LPAR running z/OS 1.13
- 64bit IBM Java 6.0.1 with compressed references, 1M large pages, 2GB heap
- IBM DB2 for z/OS v10, JDBC with keepDynamic

Feature – z/OS Workload Manager

The impact of enabling zWLM is under 5%



- z196, 4-way LPAR running z/OS 1.13
- 64bit IBM Java 6.0.1 with compressed references, 1M large pages, 2GB heap
- IBM DB2 for z/OS v10, T2 JDBC with keepDynamic

z/OS Operations – Choose your interface

- Run from a shell or as a started task with the provided launchers and PROCs
- Important messages routed as WTOs for automation
- Modify commands enable changes to trace specification or to request a diagnostic dump



```
sykesm@sykesm-MacBook:~/work/liberty-workspace/com.ibm.ws.security.authori...
sykesm@sykesm.../wlp/lib/features ... sykesm@sykesm...saf/build/classes

$ wlp/bin/server start tradeLiteServer
Server tradeLiteServer started with process ID 65682.
$ cat wlp/usr/servers/tradeLiteServer/logs/console.log
Launching tradeLiteServer (wlp-1.0.0.201203311104/websphere-kernel_1.0.0) on IBM J9 VM, vers
ion pmz6460_26sr1fp1-20120201_02 (SR1 FP1) (en_US)
[AUDIT ] CWWKE0001I: The server tradeLiteServer has been launched.
Listening on port localhost/127.0.0.1:5678 ...
[AUDIT ] J2CA8004I: The dataSource jdbc/TradeDataSource is available as jdbc/TradeDataSou
ce.
[AUDIT ] J2CA8000I: The jdbcDriver DerbyEmbedded is available.
[AUDIT ] CWWKZ0058I: Monitoring dropins for applications.
[AUDIT ] CWWKT0016I: Web application available (default_host): http://flash226.pok.ibm.com
:9080/snoop/*
[AUDIT ] CWWKZ0001I: Application snoop started in 0.12 seconds.
[AUDIT ] CWWKT0016I: Web application available (default_host): http://flash226.pok.ibm.com
:9080/tradelite/*
[AUDIT ] CWWKZ0001I: Application tradelite started in 0.59 seconds.
[AUDIT ] CWWKF0011I: The server tradeLiteServer is ready to run a smarter planet.
$
```

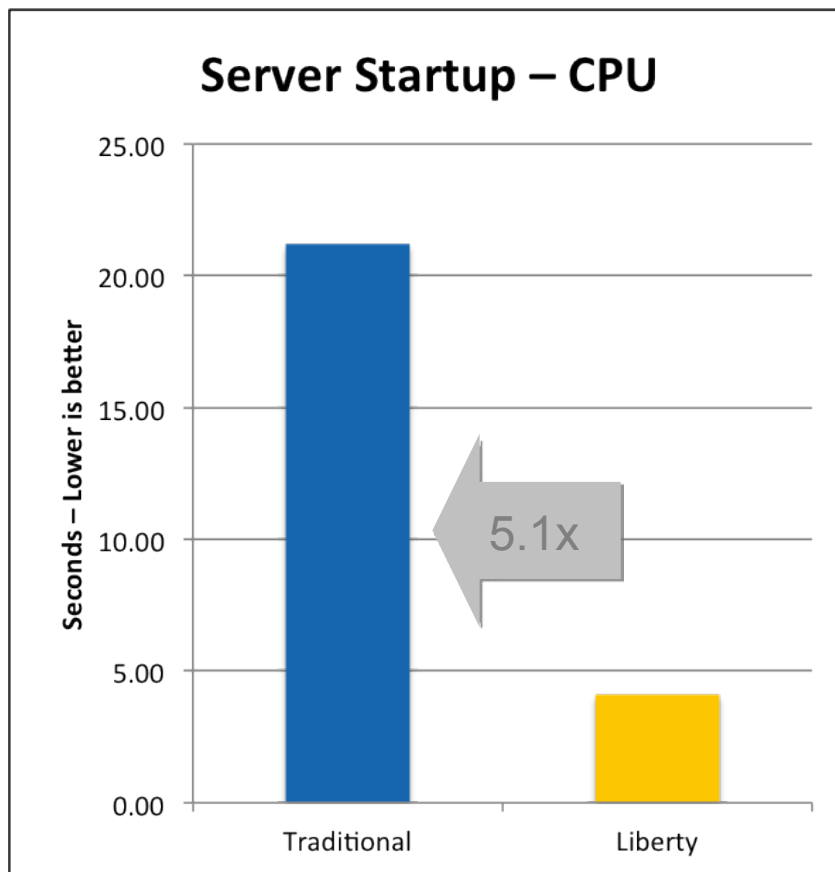
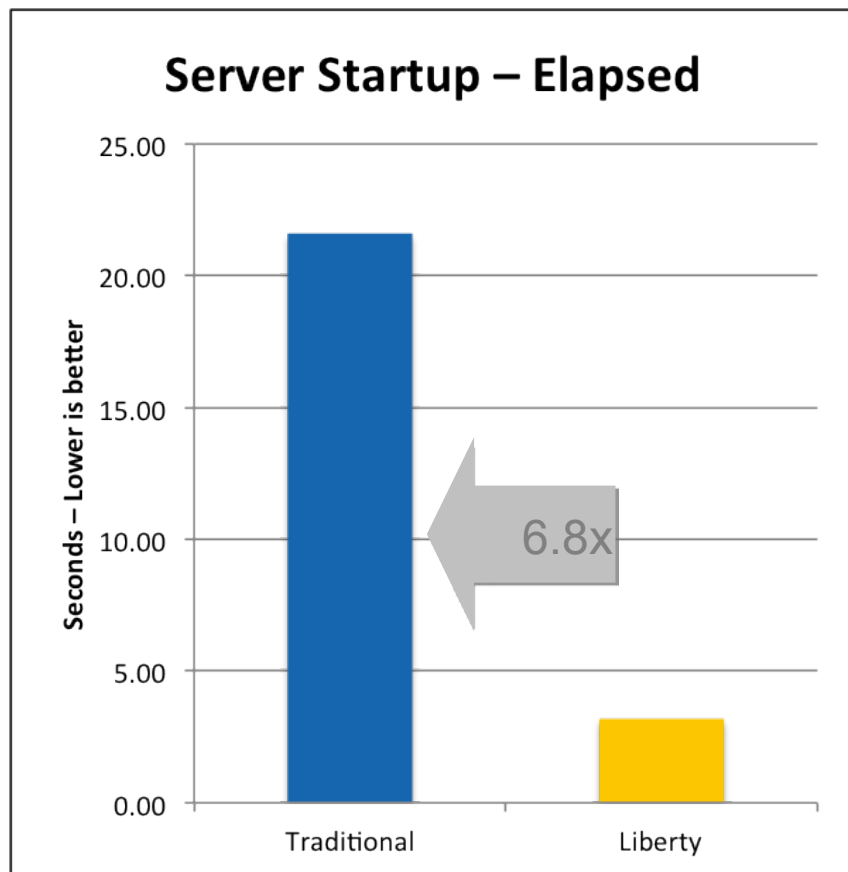
```
xa1
- SV1 start bbgzangl
- SV1 IAA812I PROFILE BBGZANGL.* (G) IN THE STARTED CLASS WAS USED
- TO START BBGZANGL WITH JOBNAME BBGZANGL.
SV1 $HASP100 BBGZANGL ON STCINADR
- SV1 $HASP373 BBGZANGL STARTED
SV1 CWWKB0056I INITIALIZATION COMPLETE FOR ANGEL
- SV1 start bbgzsrv,parms='tradeLiteServer'
- SV1 IAA812I PROFILE BBGZSRV.* (G) IN THE STARTED CLASS WAS USED
- TO START BBGZSRV WITH JOBNAME BBGZSRV.
SV1 $HASP100 BBGZSRV ON STCINADR
- SV1 $HASP373 BBGZSRV STARTED
SV1 +CWWKF0011I: The server tradeLiteServer is ready to run a smarter
planet.

IEE612I CN=C3E0SV1 DEUNUM=03E0 SYS=SV1
-
IEE163I MODE= RD

Wed 04 Apr 09:43
```

Why Liberty on z/OS?

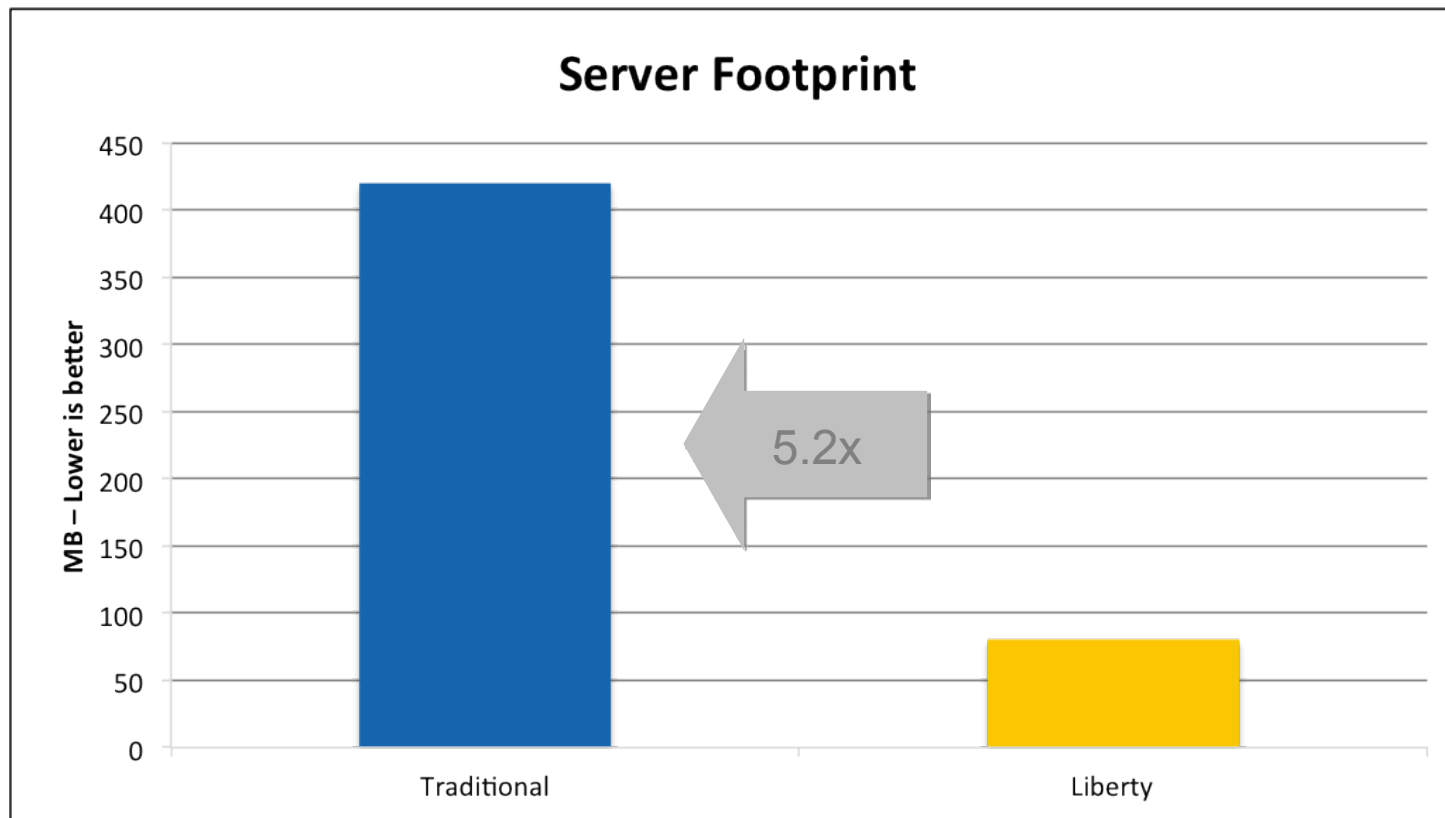
Performance: Startup time – 3.2 seconds!



- Liberty – 64bit IBM Java 6.0.1, 64/64MB min/max heap, 60MB shared class cache, TradeLite installed
- Traditional – 64bit IBM Java 6.0.1, 1SR, 128/256MB min/max CR heap, 256/512MB min/max SR heap, 75MB CR shared class cache, 75MB SR shared class cache, no applications installed

Why Liberty on z/OS?

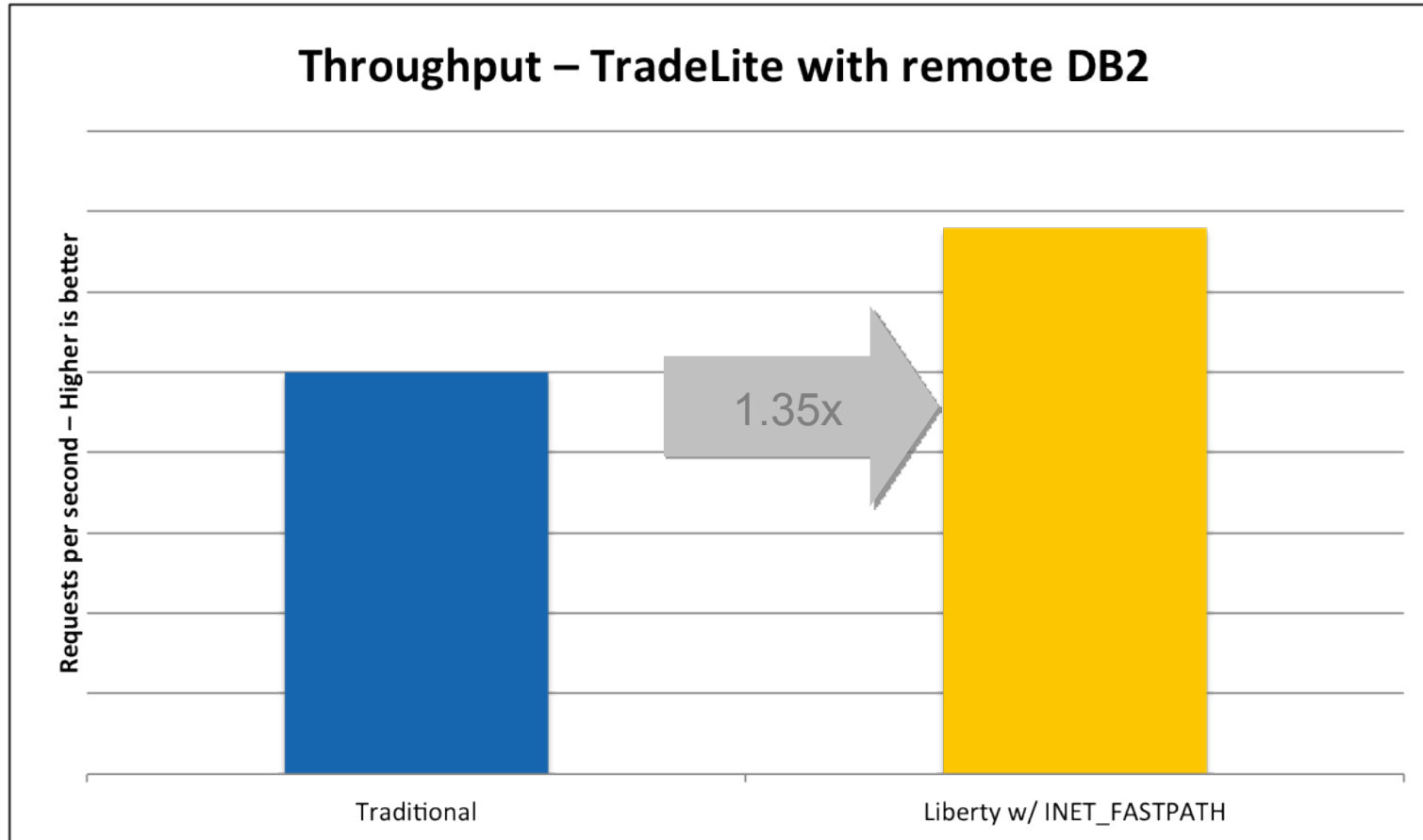
Performance: Memory footprint – 80% reduction



- Liberty – 64bit IBM Java 6.0.1, 64/64MB min/max heap, 60MB shared class cache, TradeLite installed
- Traditional – 64bit IBM Java 6.0.1, 1SR, 128/256MB min/max CR heap, 256/512MB min/max SR heap, 75MB CR shared class cache, 75MB SR shared class cache, no applications installed

Why Liberty on z/OS?

Performance: Throughput – Up to 35% improvement



- z196, 2-way LPAR running z/OS 1.13
- 64bit IBM Java 6.0.1 with compressed references, 1M large pages, 2GB heap
- IBM DB2 for z/OS v10, JDBC T4 with keepDynamic
- `_BXK_INET_FASTPATH=*` set to enable CommServer “fast path” for Liberty