OpenJ9

OpenJ9 on RISC-V

Cheng Jin (jincheng@ca.ibm.com)
OpenJ9 VM Software Developer, Runtime Technologies
IBM Ottawa Software Lab, Canada

OpenJ9

Important Disclaimers & Legal Notice

- THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.
- IBM AND THE IBM LOGO ARE TRADEMARKS OR REGISTERED TRADEMARKS OF IBM CORPORATION, IN THE UNITED STATES, OTHER COUNTRIES OR BOTH.
- JAVA AND ALL JAVA-BASED MARKS, AMONG OTHERS, ARE TRADEMARKS OR REGISTERED TRADEMARKS OF ORACLE IN THE UNITED STATES, OTHER COUNTRIES OR BOTH.
- OTHER COMPANY, PRODUCT AND SERVICE NAMES MAY BE TRADEMARKS OR SERVICE MARKS OF OTHERS.
- WHILST EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.
- ALL PERFORMANCE DATA INCLUDED IN THIS PRESENTATION HAVE BEEN GATHERED IN A CONTROLLED ENVIRONMENT. YOUR OWN TEST RESULTS MAY VARY BASED ON HARDWARE, SOFTWARE OR INFRASTRUCTURE DIFFERENCES.
- ALL DATA INCLUDED IN THIS PRESENTATION ARE MEANT TO BE USED ONLY AS A GUIDE.
- IN ADDITION, THE INFORMATION CONTAINED IN THIS PRESENTATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM, WITHOUT NOTICE.
- IBM AND ITS AFFILIATED COMPANIES SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.
- NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:
  - CREATING ANY WARRANT OR REPRESENTATION FROM IBM, ITS AFFILIATED COMPANIES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS

Agenda

- Introduction to OpenJ9
- Work on RISC-V

Part I

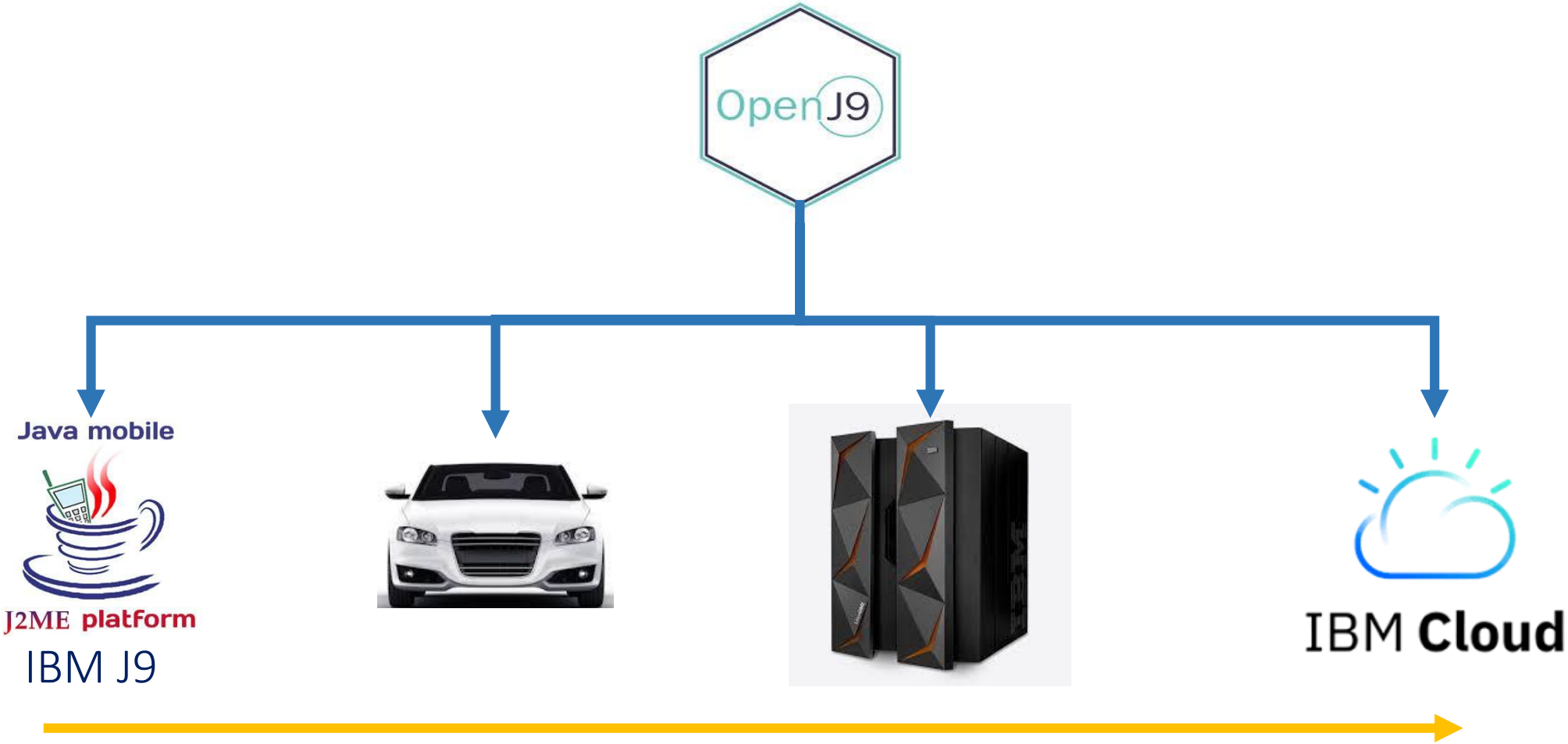
Introduction to OpenJ9

OpenJ9 or IBM J9?

1) Never heard of it
2) Hmm, might be Java9?

It’s been with us for many years. e.g. credit card transactions

Where is OpenJ9 ?



A Brief History of Open J9

Object Oriented Research Group
(1986, Carleton University)

Object Technology International (OTI)
founded in Ottawa, Ontario (Canada) in 1988

acquired in 1996
merged in 2003

IBM Ottawa Software Lab

+

Testarossa JIT (TR)

IBM Toronto Software Lab

VisualAge IDE
(SmallTalk/Java)

Eclipse IDE

https://en.wikipedia.org/wiki/Object_Technology_International

A Brief History of OpenJ9 (Cont.)

- J9 VM developed independently by IBM as Enterprise middleware over 20 years (high performance, scalability & reliability)
- Transition from ENVY/Smalltalk to C/C++ in recent years to lower barrier for developers
- Donated to the Eclipse Foundation in 2017 as Eclipse OpenJ9 (currently led by IBM)
- Dual License: Eclipse Public License v2.0 & Apache v2.0
- J9 is not Java 9 (the naming convention for the Smalltalk source code)

https://en.wikipedia.org/wiki/OpenJ9

OpenJDKs

Category 1: JDK designer & implementer

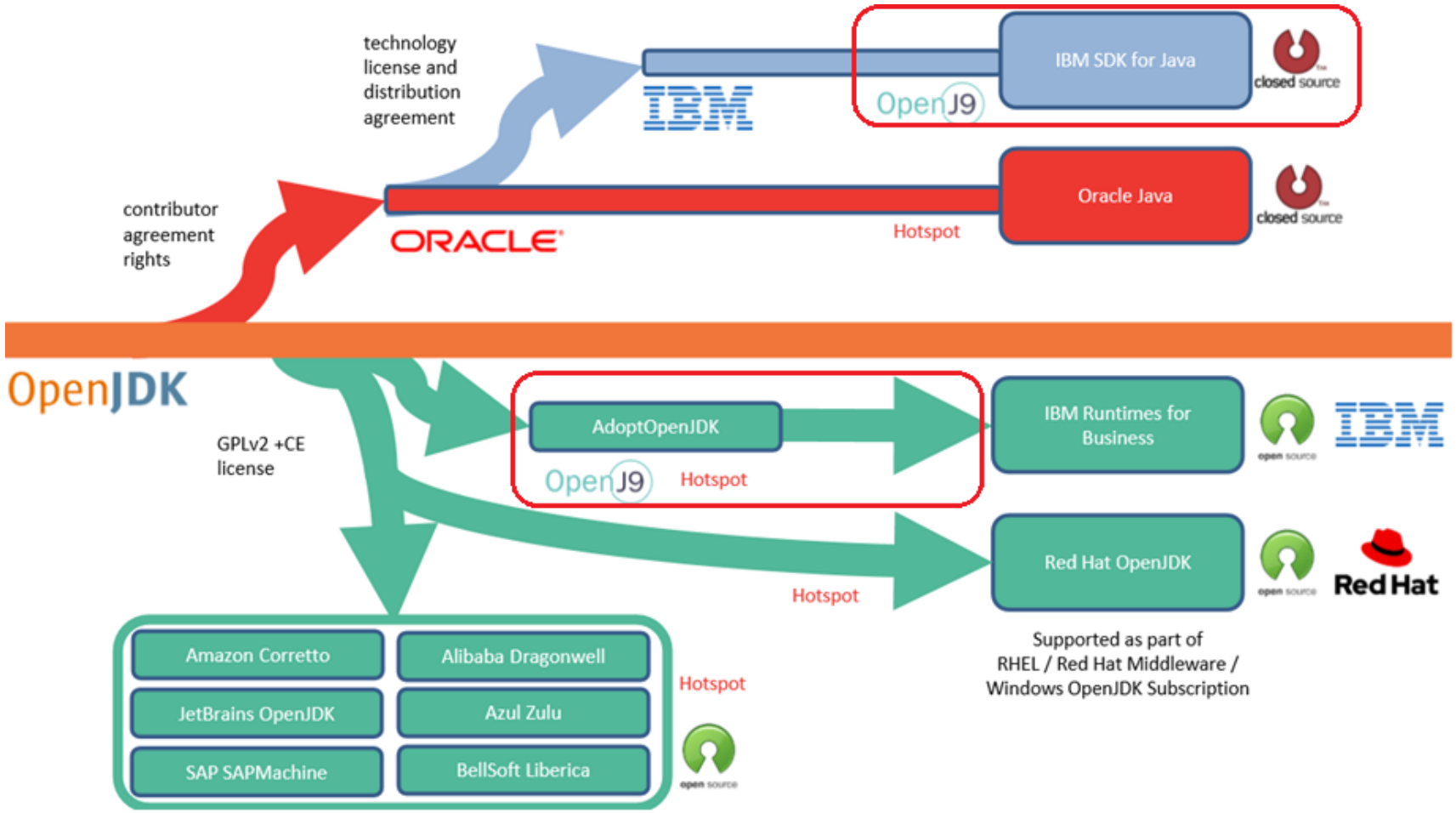
Oracle/Sun (Hotspot), IBM (OpenJ9)

Category 2: Customized JDKs (Hotspot-based)

Azul(Zulu), Amazon(Corretto), RedHat, Alibaba(Dragonwell), Huawei (Bisheng) …

https://en.wikipedia.org/wiki/OpenJDK

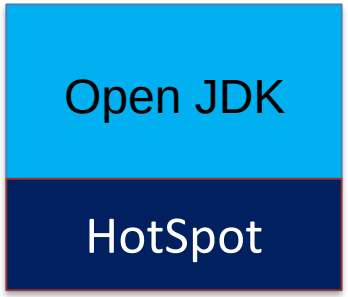
Packaging of JDKs



http://ibmhybridcloud.lookbookhq.com/c/ps07-java-strategy-a?x=5so0jp: Java Strategy And Roadmap

What is OpenJ9?

OpenJDK/Hotspot

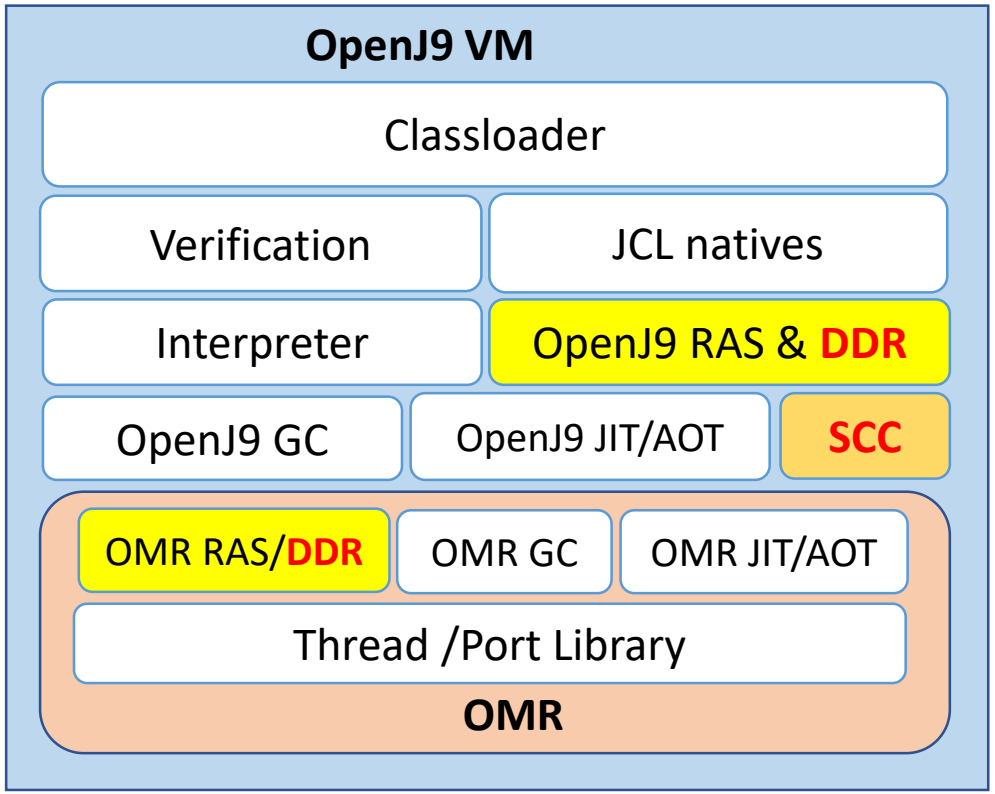


OpenJDK/OpenJ9

- OpenJDK (https://github.com/ibmruntimes/openj9-openjdk-jdk11): Building framework/Java Class libraries)
- OpenJ9 (https://github.com/eclipse/openj9): Java Virtual Machine Core (equivalent of Hotspot)
- OMR (https://github.com/eclipse/omr): Split from IBM J9 & refactored for polyglot runtimes

What is OpenJ9? (Cont.)

The Architecture of OpenJ9 VM



- **Direct Dump Reader (DDR):** diagnose issues of OpenJ9 in Java stacktraces at the bytecode level
  https://www.slideshare.net/Dev_Events/secrets-of-building-a-debuggable-runtime-learn-how-language-implementors-solve-your-runtime-issues
- **Shared Classes Cache (SCC):** store J9 ROM Classes & AOT code for better performance (startup time, footprint, etc)
  https://developer.ibm.com/technologies/java/tutorials/j-class-sharing-openj9/

DDR/jdmpview (1)

bin/jdmpview -core core.20201123.073314.15487.0001_140323-073326.dmp

> !threads

```
----->  !stack 0x020be500      !j9vmthread 0x020be500  !j9thread 0x7fb0740074e0       tid 0x25ac (9644) // (main)
        !stack 0x020c7f00      !j9vmthread 0x020c7f00  !j9thread 0x7fb0740b9690       tid 0x25ae (9646) // (JIT Compilation Thread-0)
        !stack 0x020cd000      !j9vmthread 0x020cd000  !j9thread 0x7fb0740b9c08       tid 0x25af (9647) // (JIT Compilation Thread-1 Suspended)
        !stack 0x020d2000      !j9vmthread 0x020d2000  !j9thread 0x7fb0740baba0       tid 0x25b0 (9648) // (JIT Compilation Thread-2 Suspended)
        !stack 0x020d7100      !j9vmthread 0x020d7100  !j9thread 0x7fb0740bb118       tid 0x25b1 (9649) // (JIT Compilation Thread-3 Suspended)
        !stack 0x020dc200      !j9vmthread 0x020dc200  !j9thread 0x7fb0740bc0c0       tid 0x25b2 (9650) // (JIT Diagnostic Compilation Thread-4 Suspended)
        !stack 0x020e1200      !j9vmthread 0x020e1200  !j9thread 0x7fb0740bc638       tid 0x25b3 (9651) // (JIT-SamplerThread)
        !stack 0x020e6300      !j9vmthread 0x020e6300  !j9thread 0x7fb0740f5220       tid 0x25b4 (9652) // (IProfiler)
        !stack 0x021aac00      !j9vmthread 0x021aac00  !j9thread 0x7fb0740f5798       tid 0x25b5 (9653) // (Signal Dispatcher)
        !stack 0x021afc00      !j9vmthread 0x021afc00  !j9thread 0x7fb074386a48       tid 0x25b7 (9655) // (GC Slave)
        !stack 0x021b9d00      !j9vmthread 0x021b9d00  !j9thread 0x7fb0743aa278       tid 0x25b9 (9657) // (Attach API wait loop)
...
```

DDR/jdmpview (2)

> !stackslots 0x021bcd00
<21bcd00> *** BEGIN STACK WALK, flags = 00400001 walkThread = 0x00000000021BCD00 ***
<21bcd00> ITERATE_O_SLOTS
<21bcd00> RECORD_BYTECODE_PC_OFFSET
<21bcd00> Initial values: walkSP = 0x00000000021D9A80, PC = 0x00007FB0742DD911, literals = 0x0000000002145368, A0 = 0x00000000021D9AD0, j2iFrame = 0x00000000021D9B58, ELS = 0x00007FB061AF6B28, decomp = 0x00007FB0743ECE60
<21bcd00> Bytecode frame: bp = 0x00000000021D9AB0, sp = 0x00000000021D9A80, pc = 0x00007FB0742DD911, cp = 0x00000000021462B0, arg0EA = 0x00000000021D9AD0, flags = 0x0000000000000000
<21bcd00> Method: com/ibm/jit/JITHelpers.getIntFromObject(Ljava/lang/Object;J)I **!j9method 0x0000000002145368** <-----
<21bcd00> **Bytecode index = 5** <-------
<21bcd00> Using debug local mapper
<21bcd00> Locals starting at 0x00000000021D9AD0 for 0x0000000000000004 slots
<21bcd00> O-Slot: a0[0x00000000021D9AD0] = **0x00000000E000B240** <--------------------
<21bcd00> O-Slot: a1[0x00000000021D9AC8] = 0x0000000000000000
<21bcd00> I-Slot: a2[0x00000000021D9AC0] = 0x0000000000000000
<21bcd00> I-Slot: a3[0x00000000021D9AB8] = 0x0000000000000038
…

DDR/jdmpview (3)

> j9object 0x00000000E000B240

```
!J9Object 0x00000000E000B240 {
    struct J9Class* clazz = !j9class 0x2145F00   // com/ibm/jit/JITHelpers <-----
    Object flags = 0x00000000;
    I lockword = 0x00000000 (offset=0) (java/lang/Object) <hidden>
```

>!j9class 0x0000000002145F00

```
J9Class at 0x2145f00 {
  Fields for J9Class:
      0x0: UDATA eyecatcher = 0x0000000099669966 (2573637990)
      0x8: struct J9ROMClass * romClass = !j9romclass 0x00007FB0742DCFB8
      0x10: struct J9Class ** superclasses = !j9x 0x0000000002146870
      0x18: UDATA classDepthAndFlags = 0x00000000000E0001 (917505)
      0x20: U32 classDepthWithFlags = 0x00000000 (0)
      0x24: U32 classFlags = 0x00000000 (0)
      0x28: struct J9ClassLoader * classLoader = !j9classloader 0x00007FB074055AD8
      0x30: j9object_t classObject = !j9object 0x00000000E000E858 // java/lang/Class
      0x38: volatile UDATA initializeStatus = 0x0000000000000001 (1)
      0x40: struct J9Method * ramMethods = !j9method 0x0000000002145268 // com/ibm/jit/JITHelpers.<init>()V
```

DDR/jdmpview (4)

> !j9method 0x0000000002145368

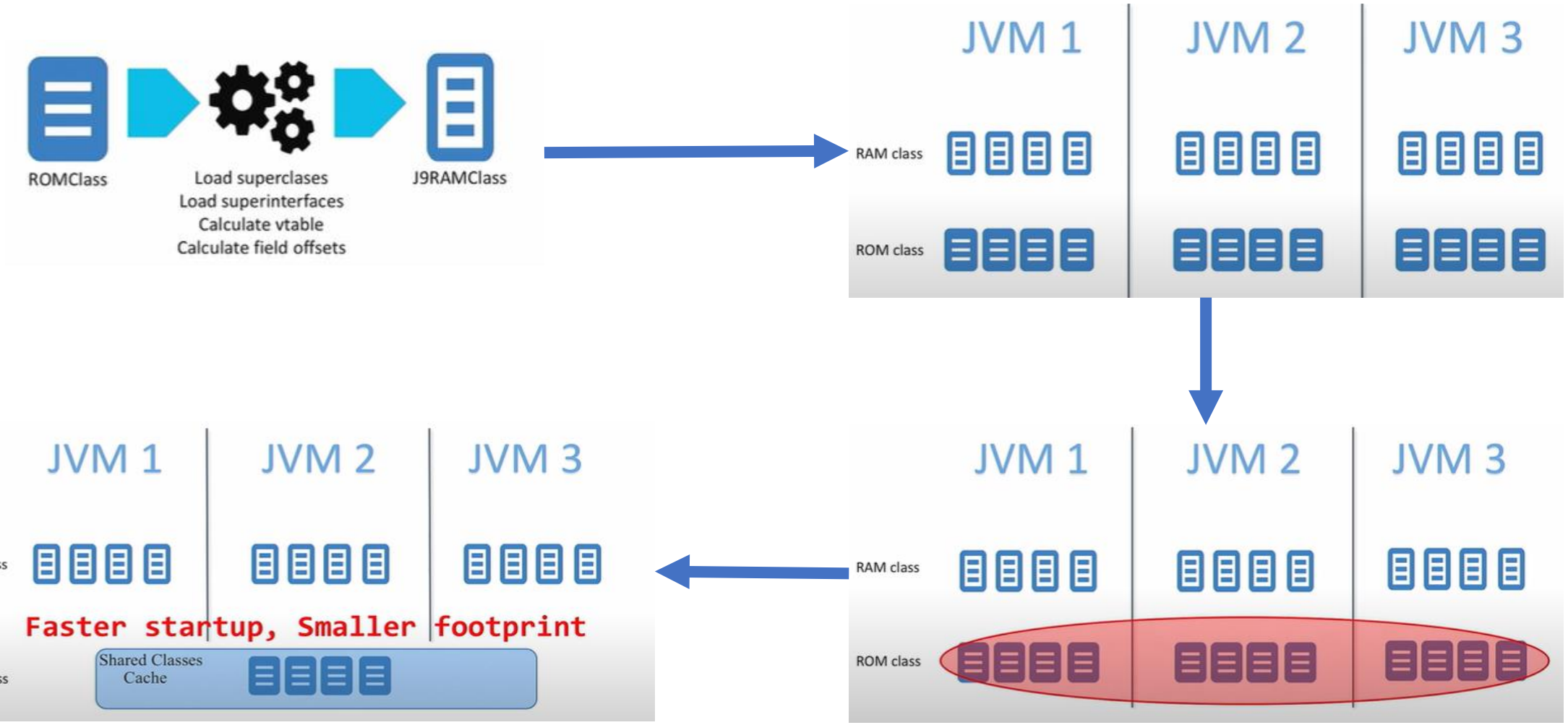
```
J9Method at 0x2145368 {
  Fields for J9Method:
      0x0: U8 * bytecodes = !j9x 0x00007FB0742DD90C
      0x8: struct J9ConstantPool * constantPool = !j9constantpool 0x00000000021462B0
      0x10: void * methodRunAddress = !j9x 0x0000000000000005
      0x18: void * extra = !j9x 0x0000000000000001
}
Signature: com/ibm/jit/JITHelpers.getIntFromObject(Ljava/lang/Object;J)I !bytecodes 0x0000000002145368 <------
```

> !bytecodes 0x0000000002145368

```
  Name: getIntFromObject
  Signature: (Ljava/lang/Object;J)I
  Access Flags (50001): public
  Max Stack: 4
  Argument Count: 4
  Temp Count: 0

   0 getstatic 9 com/ibm/jit/JITHelpers.unsafe Lsun/misc/Unsafe;
   3 aload1
   4 lload2
   5 invokevirtual 24 sun/misc/Unsafe.getInt(Ljava/lang/Object;J)I <------- Bytecode index = 5 indicated in stackslots
   8 return1
```

Shared Classes Cache (SCC)



https://www.youtube.com/watch?v=BUAESSl2sy8: The Eclipse OpenJ9 JVM a deep dive!

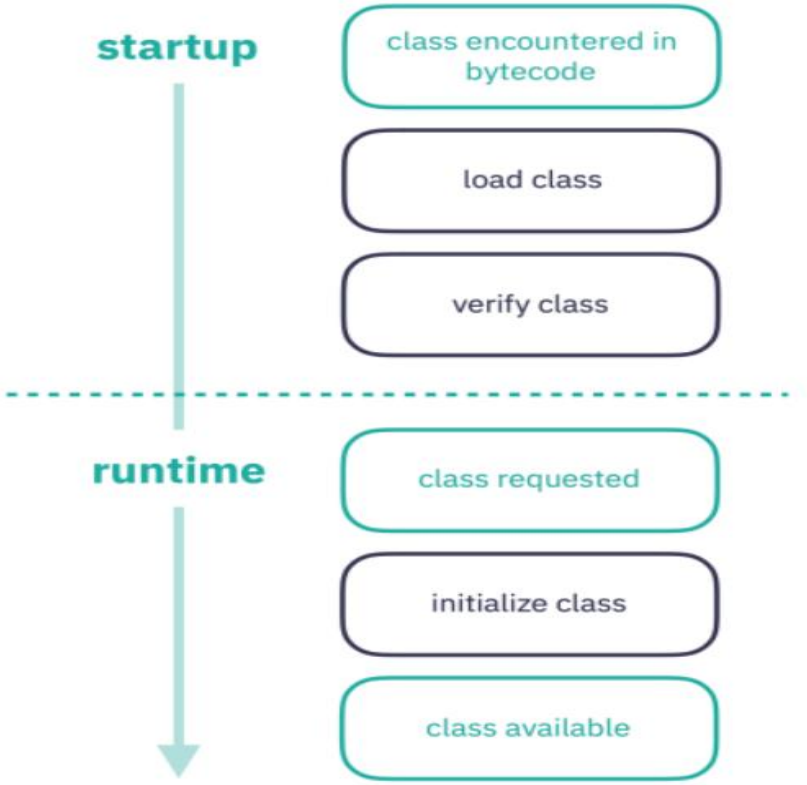
Shared Classes Cache (Cont.)

CDS ?

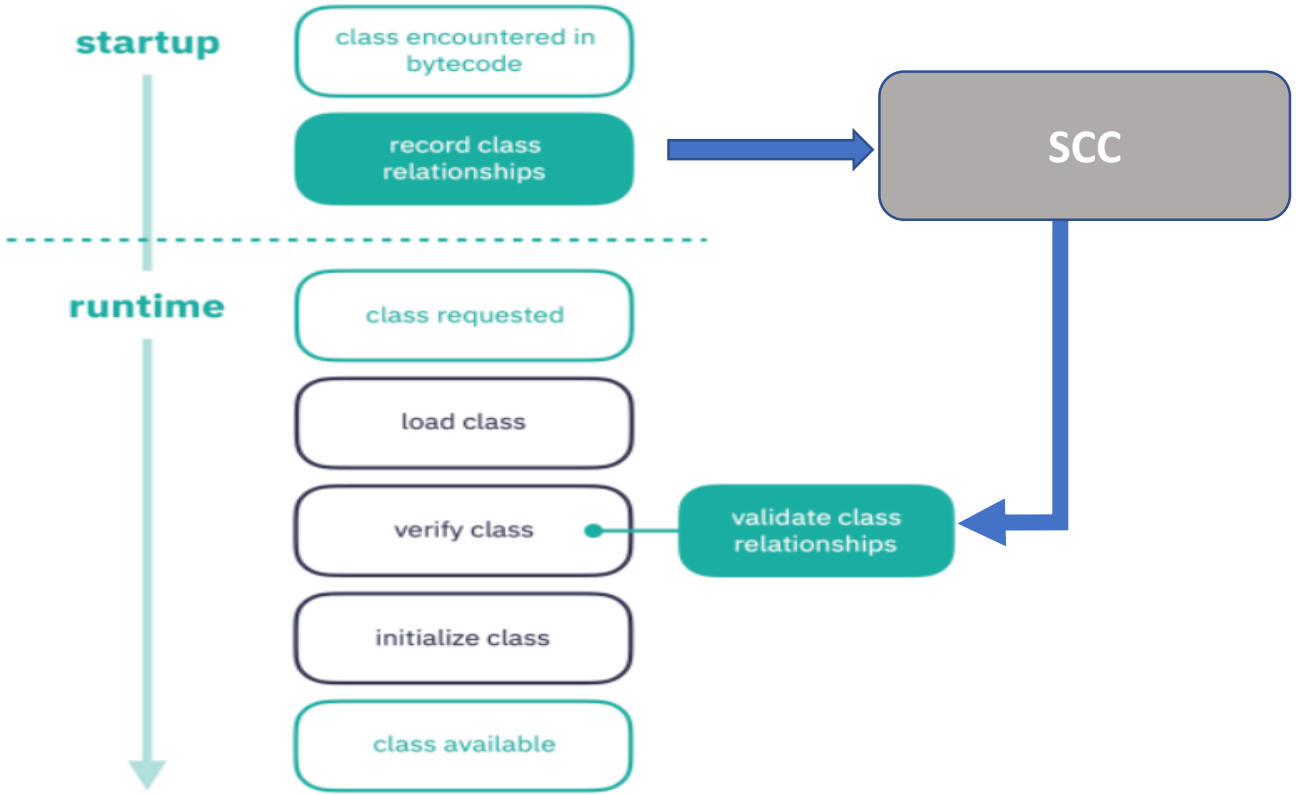
https://www.youtube.com/watch?v=BUAESSl2sy8: The Eclipse OpenJ9 JVM a deep dive!

Lazy Verification via SCC

Current Verification Process

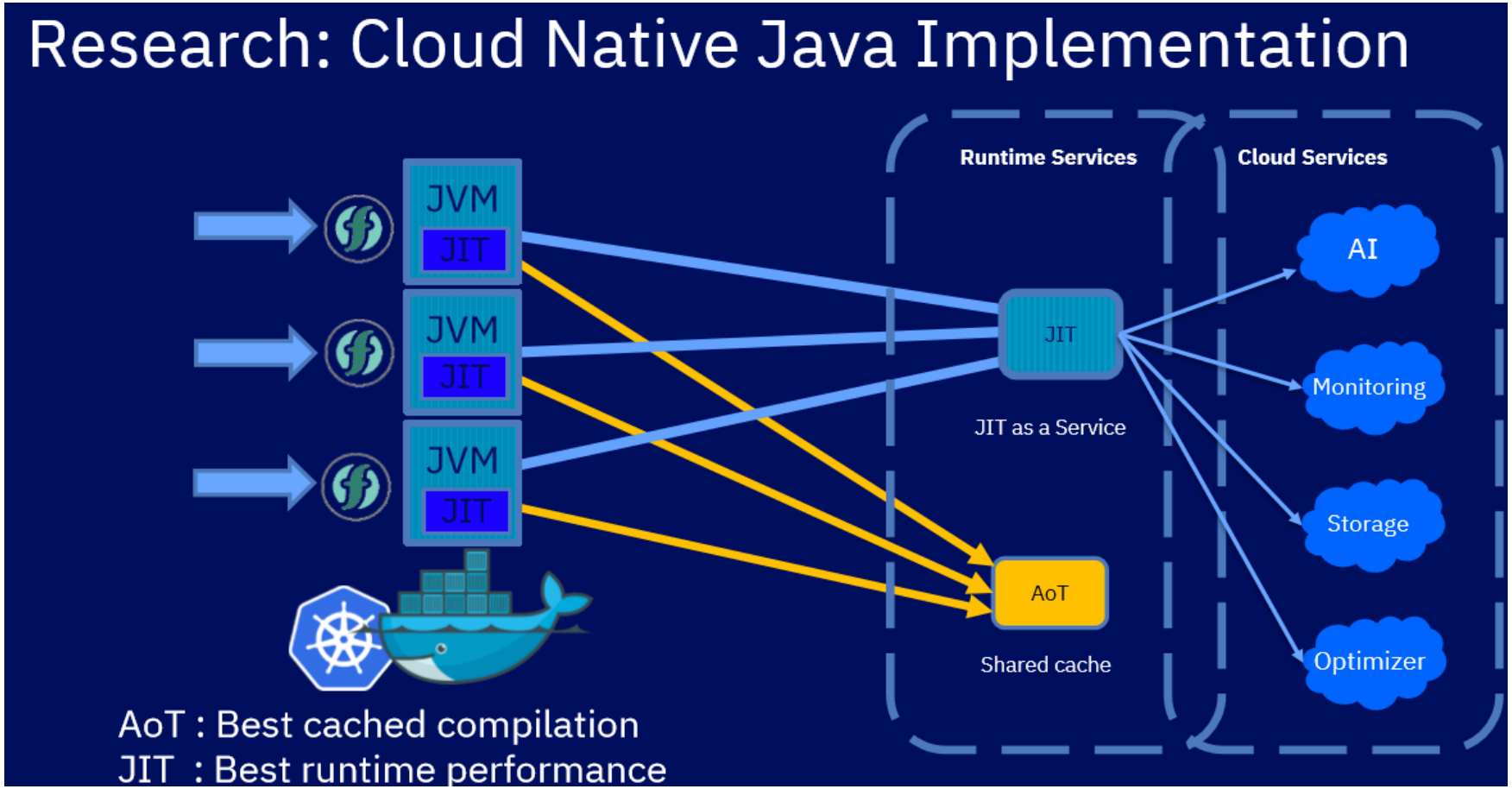


Optimized Verification Process using -XX:+ClassRelationshipVerifier



https://blog.openj9.org/2019/10/29/relationship-verification-lets-get-lazy/

JITaaS with SCC



http://ibmhybridcloud.lookbookhq.com/c/ps07-java-strategy-a?x=5so0jp: Java Strategy And Roadmap

The challenges of Java/JVM

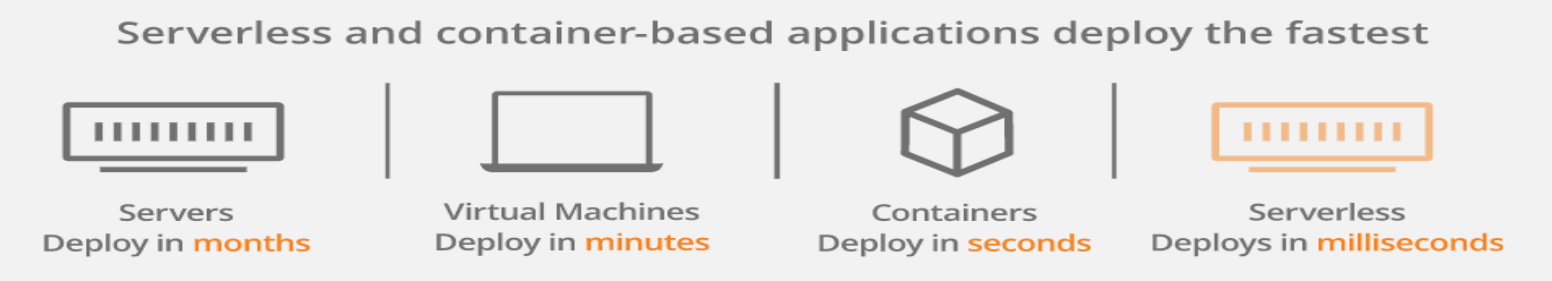
OpenJ9

Monolith

Microprofile/Microservice

Serverless/FaaS
(Amazon Lambda)

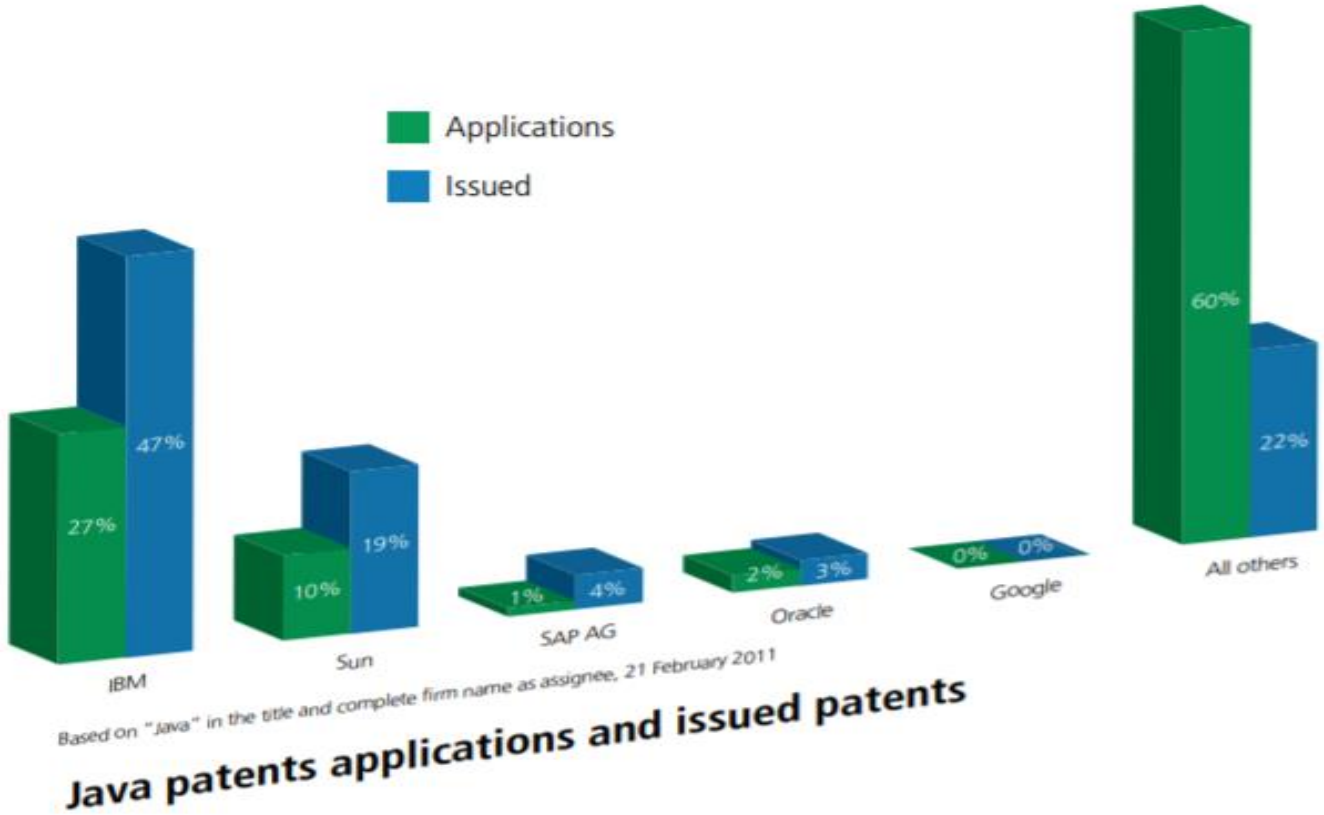
Cloud Native/(RedHat Quarkus)



Traditional/Legacy Applications → Cloud

https://www.cloudflare.com/learning/serverless/serverless-vs-containers/

Innovation on Java



https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/IMM/I110401F.pdf (Intellectual Property Magazine April, 2011)

Part II

Work on RISC-V

Background & Motivation of OpenJ9/RISC-V

Cloud Computing vs Edge Computing

Cloud Computing (Centralized)

- Long distance from the data source
- Network bottleneck for huge volume of data
- Security risk for sensitive data

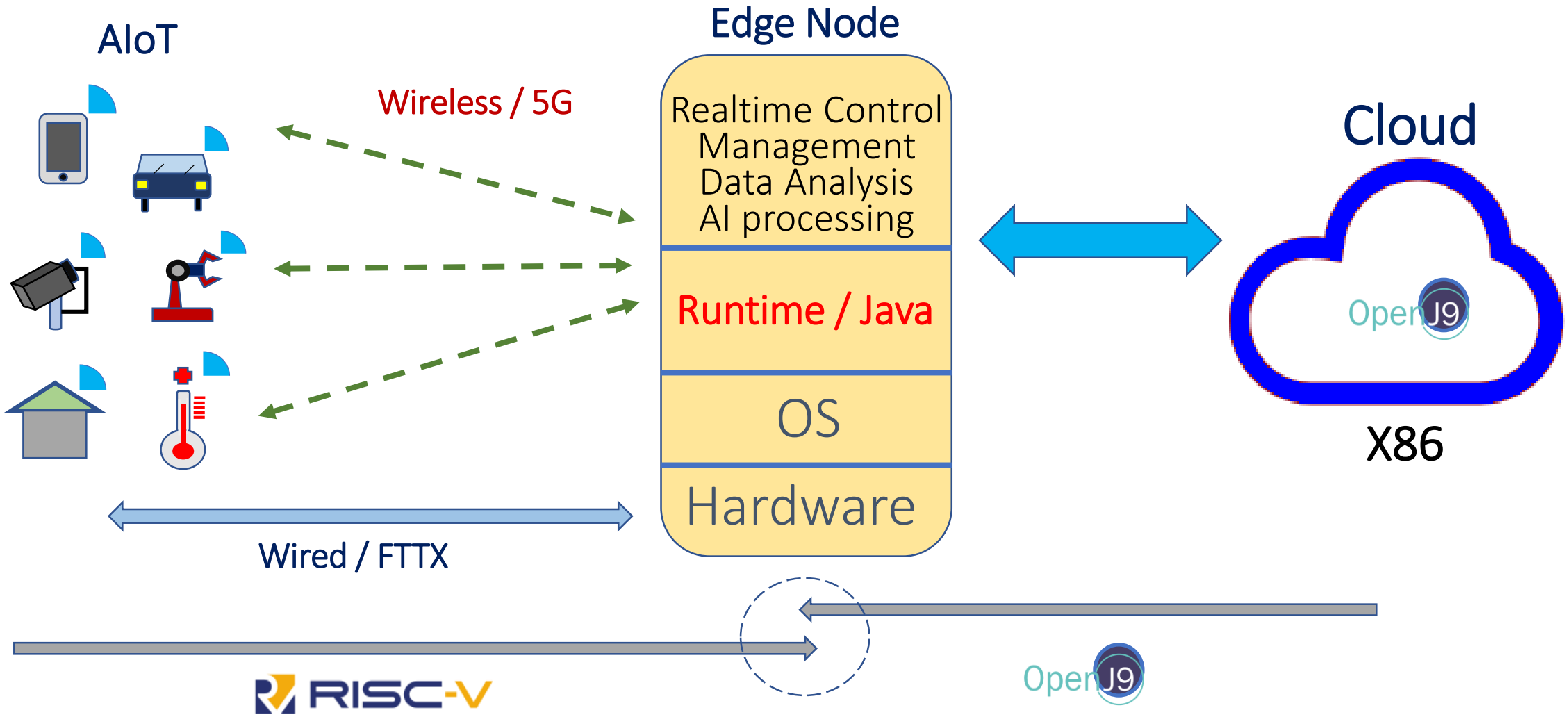
Edge Computing (Decentralized)

- Close to the data source
- Fast response to real-time data
- Data localization

"Around 10% of enterprise-generated data is created and processed outside a traditional centralized data center or cloud. By 2025, Gartner predicts this figure will reach 75%"

https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders

Background & Motivation of OpenJ9/RISC-V (Cont.)



A solution for AIoT & Edge with lower cost & Lower power consumption is preferable to others

Porting OpenJ9 JDK to RISC-V (1)

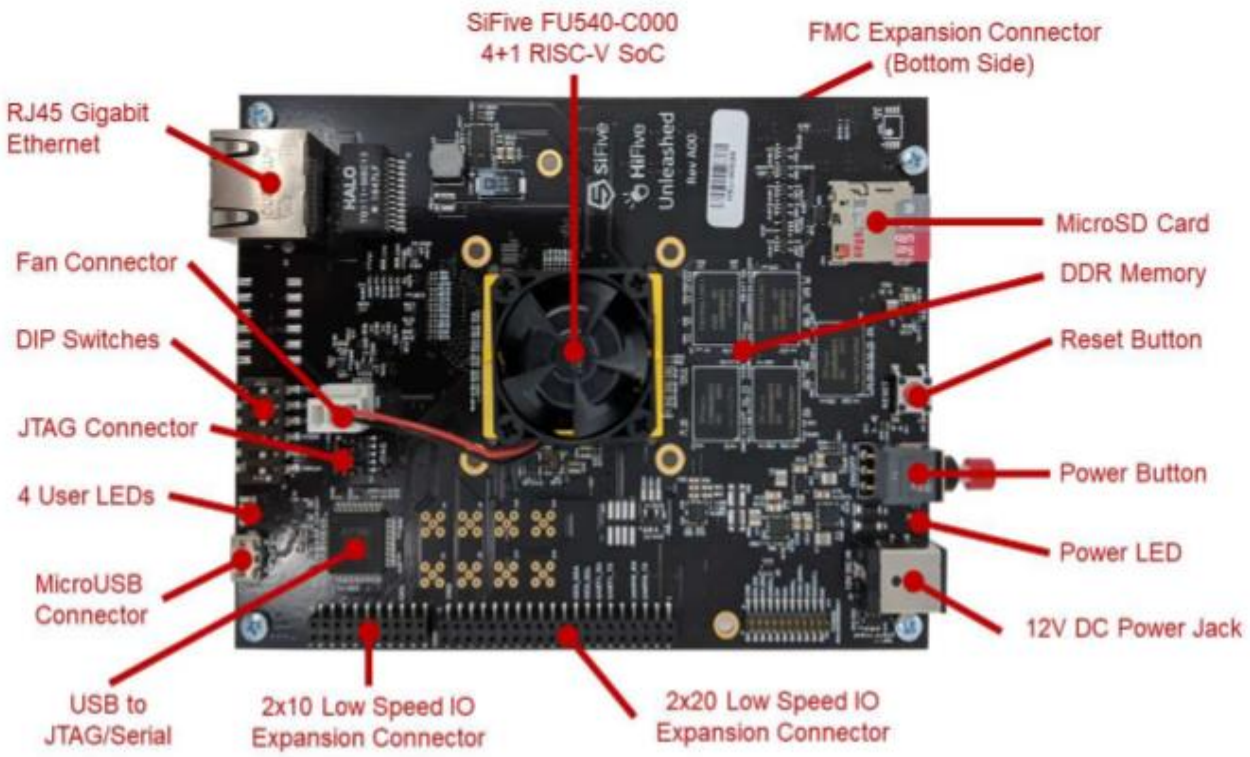
Preparation of Software

- Toolchain for cross-compilation
  - RISC-V GNU Compiler Toolchain at https://github.com/riscv/riscv-gnu-toolchain (compiled from source)
- Emulator on the Host system
  - RISC-V QEMU (riscv64) > 5.0 at https://www.qemu.org/ (compiled from source)
  - The building instruction at https://wiki.qemu.org/Documentation/Platforms/RISCV/
- Linux-based Host System
  - Ubuntu v19 (Fedora v32, Debian v10, etc)
- Linux-based Target System (https://github.com/riscv/riscv-software-list)
  - 64 bit Fedora/RISC-V at https://dl.fedoraproject.org/pub/alt/risc-v/repo/virt-builder-images/images/
  - 64 bit Debian/RISC-V at https://github.com/janvrany/riscv-debian (customized to HiFive U540)

Porting OpenJ9 JDK to RISC-V (2)

Preparation of Hardware

- HiFive Unleashed 540 (64bit U540 SoC)
  - Linux-capable RISC-V processor with multi-core
  - 8GB DDR4 RAM with ECC
  - MicroSD Card (Removable Storage)
  - MicroUSB connector
  - RJ45 Gigabit Ethernet port



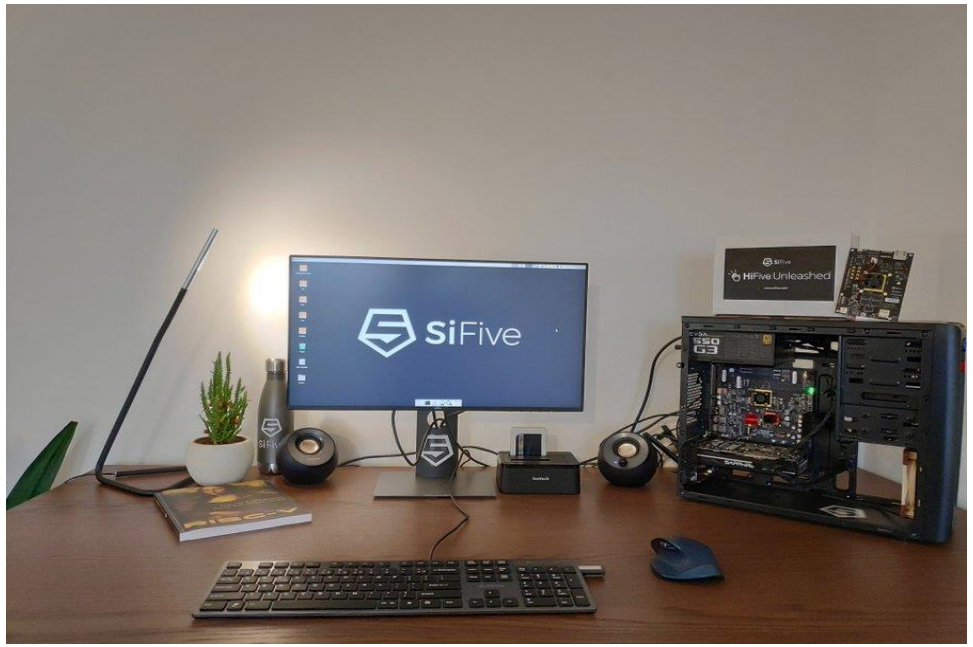
https://sifive.cdn.prismic.io/sifive/8328bc5d-b7ea-4885-a7f9-30fc108b7222_HiFive_Unleashed_Getting_Started_Guide-v1p2.pdf

Porting OpenJ9 JDK to RISC-V (2)

Preparation of Hardware (Cont.)

- HiFive Unmatched 740 PC board (SiFive FU740 SoC) (pre-order/ Q4 2020)

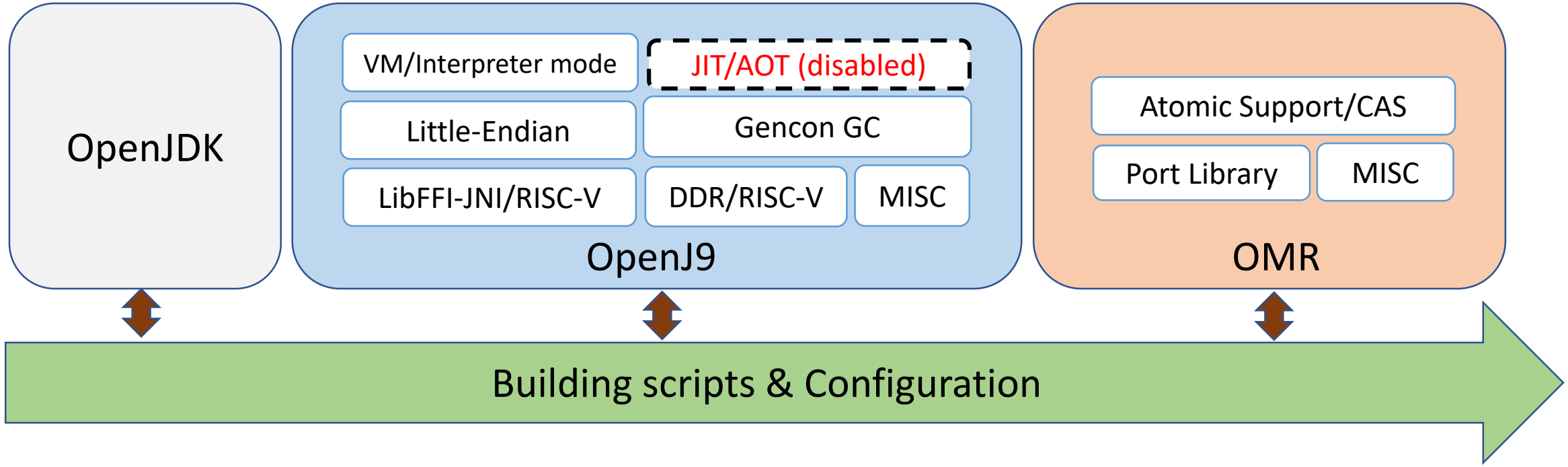
https://techcrunch.com/2020/10/29/sifives-new-pc-is-bringing-open-source-computing-closer-to-reality/



https://twitter.com/bitmask_reg/status/1321865548565827590

Porting OpenJ9 JDK to RISC-V (3)

Development of OpenJ9 JDK on RISC-V

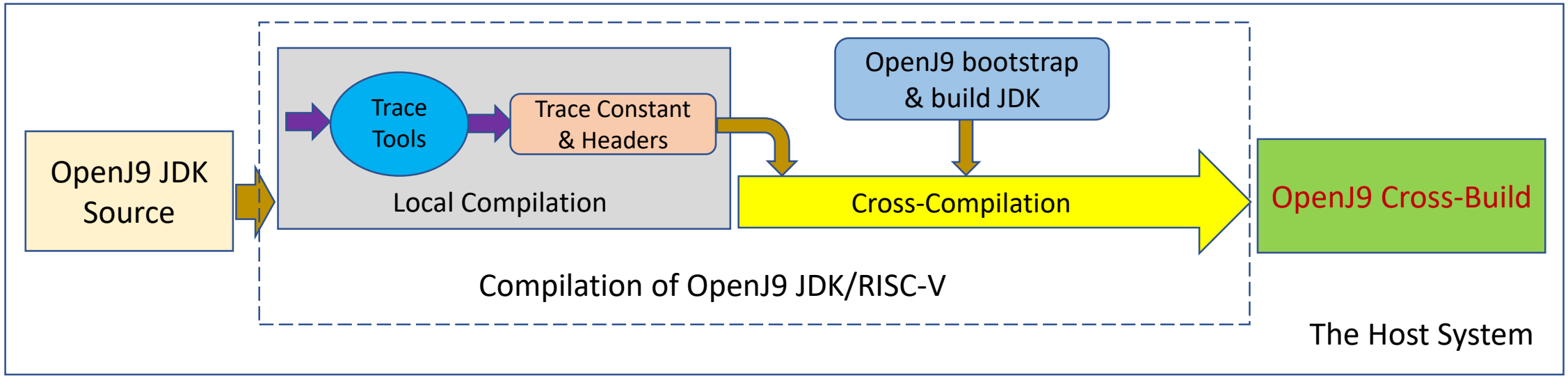


Java Native Interface (JNI) support

- the Foreign Function Interface (FFI) library on RISC-V from https://github.com/libffi/libffi/tree/master/src/riscv

Porting OpenJ9 JDK to RISC-V (4)

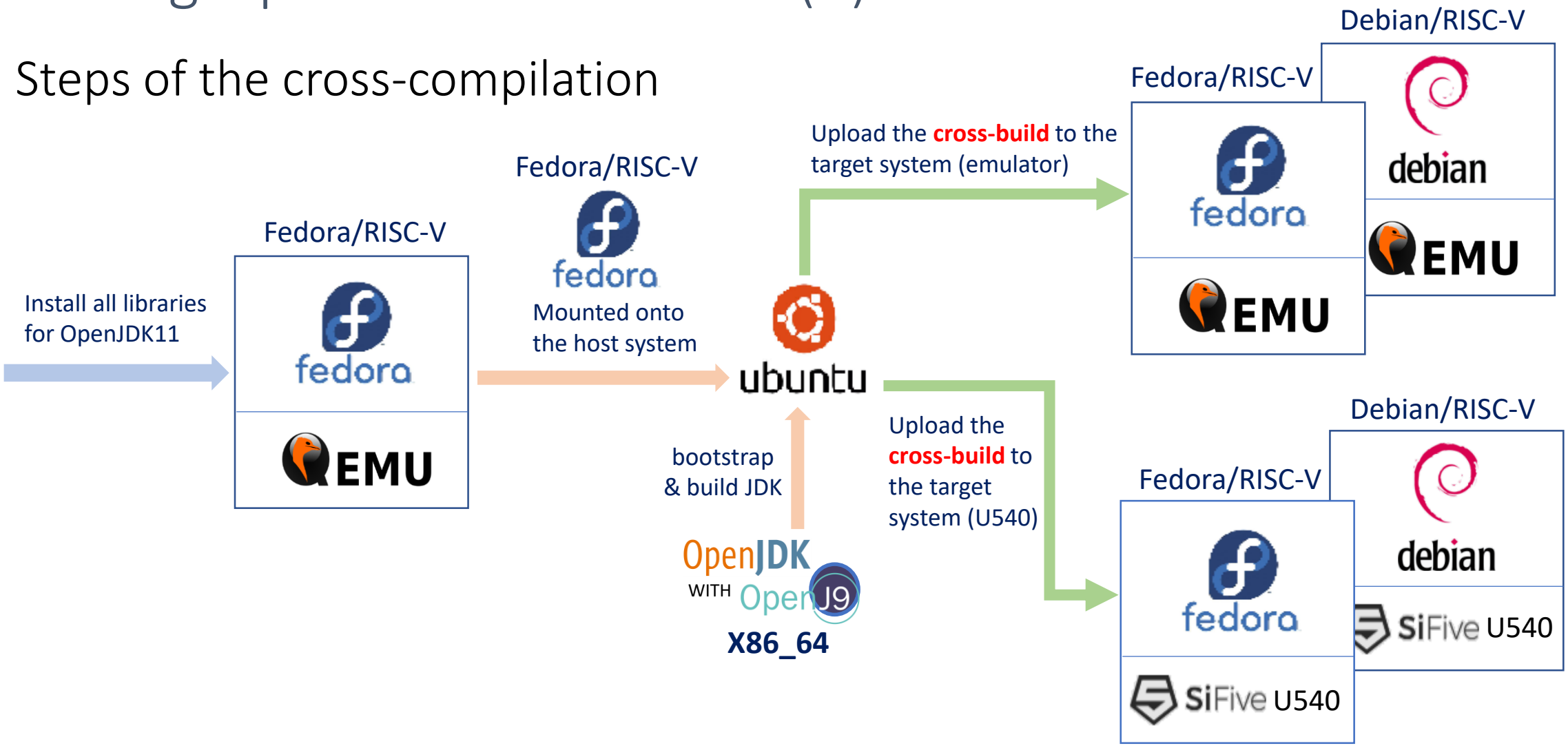
Mix of local complication & cross-compilation in OpenJ9 JDK on RISC-V



- Local toolchain (on the host system): /usr/bin/gcc & g++, etc
- Cross-toochain (on the host system): riscv64-linux-gnu-gcc & g++, etc

Porting OpenJ9 JDK to RISC-V (5)

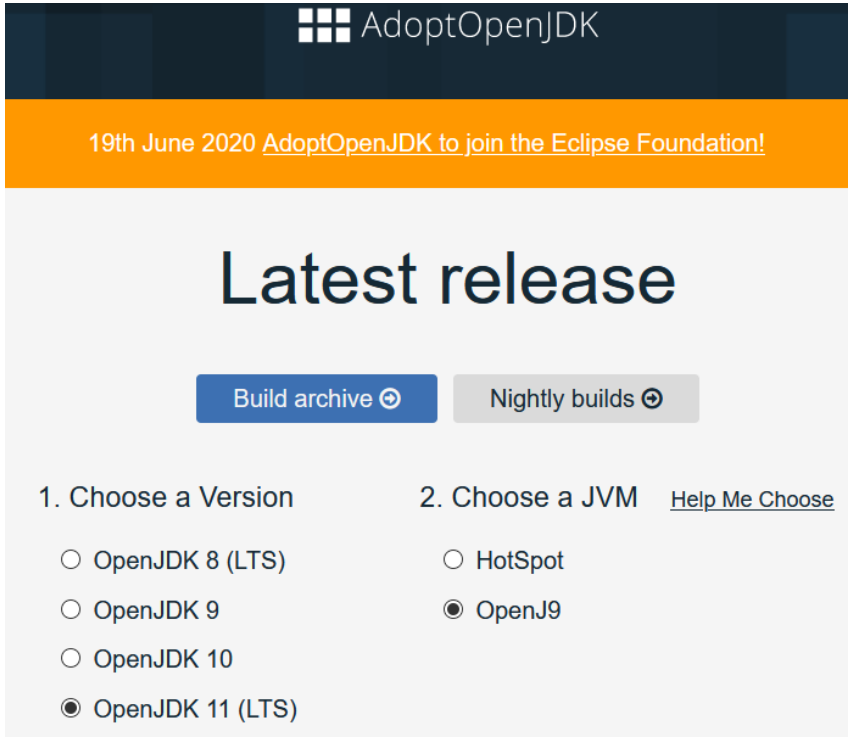
Steps of the cross-compilation



- OpenJ9 on RISC-V: https://github.com/eclipse/openj9/blob/master/doc/build-instructions/Build_Instructions_V11.md#riscv64
- Fedora/RISCV on U540: https://fedoraproject.org/wiki/Architectures/RISC-V/Installing

Porting OpenJ9 JDK to RISC-V (6)

Downloading links of bootstrap JDK at AdoptOpenJDK



https://adoptopenjdk.net/releases.html?variant=openjdk11&jvmVariant=openj9

Status of OpenJ9 JDK on RISC-V

Execution of OpenJ9 JDK (Interpreter mode)

QEMU (Emulator)

1. Debian/RISC-V
   - ✓ Debian (Linux kernel 5.0) with Berkeley Boot Loader (BBL)
   - ✓ RISC-V (Linux kernel > 5.3) with Open Source Supervisor Binary Interface (OpenSBI) (supported since linux kernel > 5.3)
2. Fedora/RISC-V
   - ✓ Fedora stage4 + Linux kernel 4.19 (first bootstrap image) with BBL
   - ✓ Linux kernel > 5.3 with OpenSBI

HiFive U540 (Dev board)

1. Debian (Linux kernel 5.0) with BBL
2. Java exception on Fedora & Debian (Linux kernel > 5.3) with OpenSBI (under investigation)

Future Work

- JIT support (TR JIT vs Lightweight JIT)
- Various GC strategies
- DDR generation via cross-compilation
- Java 8 & other JDK LTS support

OpenJ9

Q & A