关于「在 RISC-V 峰会召开前将 OpenJDK 移植到 RV32GC」结果却没有赶上 Deadline 这件事

PLCT-张定立

dingli@iscas.ac.cn

目录

1.Java的RISC-V支持现状

2.在PLCT我的工作

01 Java的RISC-V支持现状

OpenJDK for RV64G移植进度和现状

- Zero backend has been supported in the upstream

- Huawei contributed the initial porting in Bisheng JDK11(Based on RV64G)
https://gitee.com/openeuler/bishengjdk-11/tree/risc-v/

- Alibaba adopted Huawei’s patches in Alibaba Dragonwell 11(**internally**)

- This initial porting has been pushed to JDK Sandbox
https://github.com/openjdk/jdk-sandbox/tree/riscv-port-branch

- The test pipeline has been set up by using Eclipse Adoptium(formerly known as AdoptOpenJDK) script
http://ci.dragonwell-jdk.io/job/testGround/job/jobs/job/jdk/job/jdk-linux-riscv64-hotspot/22/

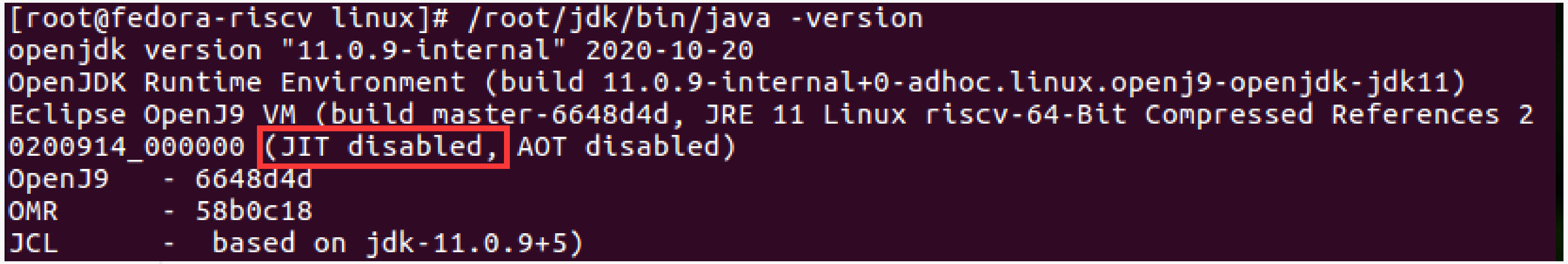
- The trial RISC-V OpenJDK build:
http://ci.dragonwell-jdk.io/userContent/jdk-16+14.tar.gz

参考资料：
https://static.sched.com/hosted_files/riscvforumdttc2021/32/riscv-java-v2-2.pdf

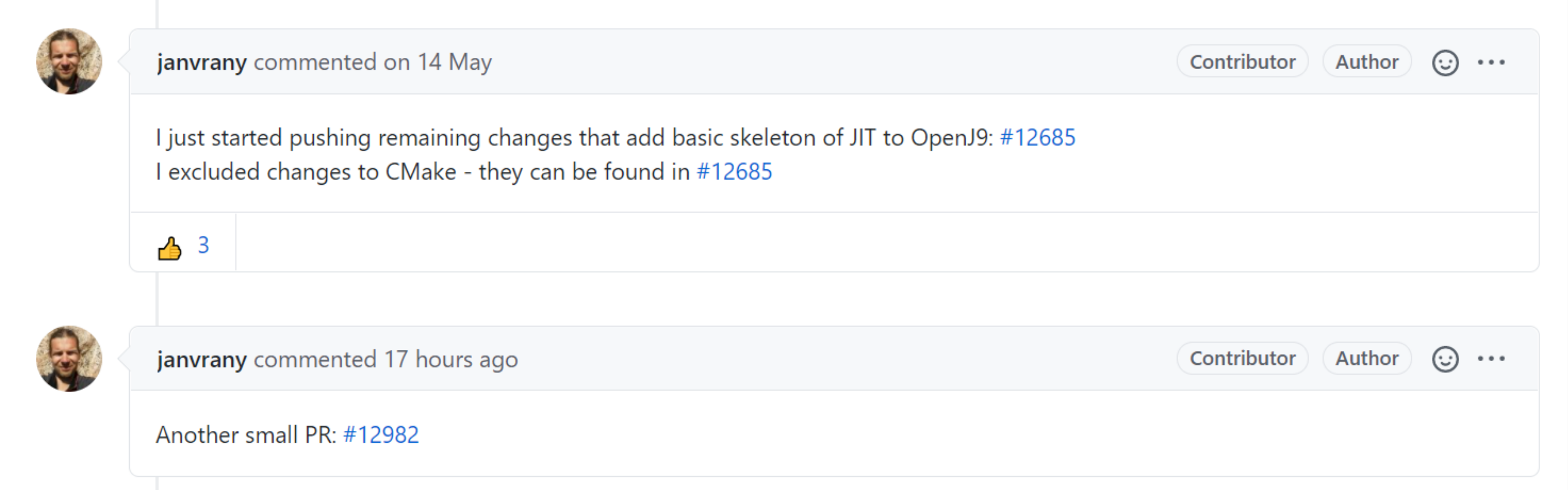
01 Java的RISC-V支持现状

OpenJ9 for RV64G移植进度和现状

- 移植基于OpenJDK11+OpenJ9+OMR，OpenJ9用以替代Hotspot
- 目前OpenJ9暂不支持针对RISCV64的JIT编译器，在这种情况下，无论是否在命令行上指定了-Xint选项，JDK默认都会以相同的输出结束。



```
[root@fedora-riscv linux]# /root/jdk/bin/java -version
openjdk version "11.0.9-internal" 2020-10-20
OpenJDK Runtime Environment (build 11.0.9-internal+0-adhoc.linux.openj9-openjdk-jdk11)
Eclipse OpenJ9 VM (build master-6648d4d, JRE 11 Linux riscv-64-Bit Compressed References 2
0200914_000000 (JIT disabled, AOT disabled)
OpenJ9   - 6648d4d
OMR      - 58b0c18
JCL      -  based on jdk-11.0.9+5)
```



janvrany commented on 14 May

I just started pushing remaining changes that add basic skeleton of JIT to OpenJ9: #12685
I excluded changes to CMake - they can be found in #12685

janvrany commented 17 hours ago

Another small PR: #12982

参考资料：
[1] https://github.com/eclipse/openj9/issues/5058
[2] https://github.com/eclipse/openj9/issues/11136

02 我的工作

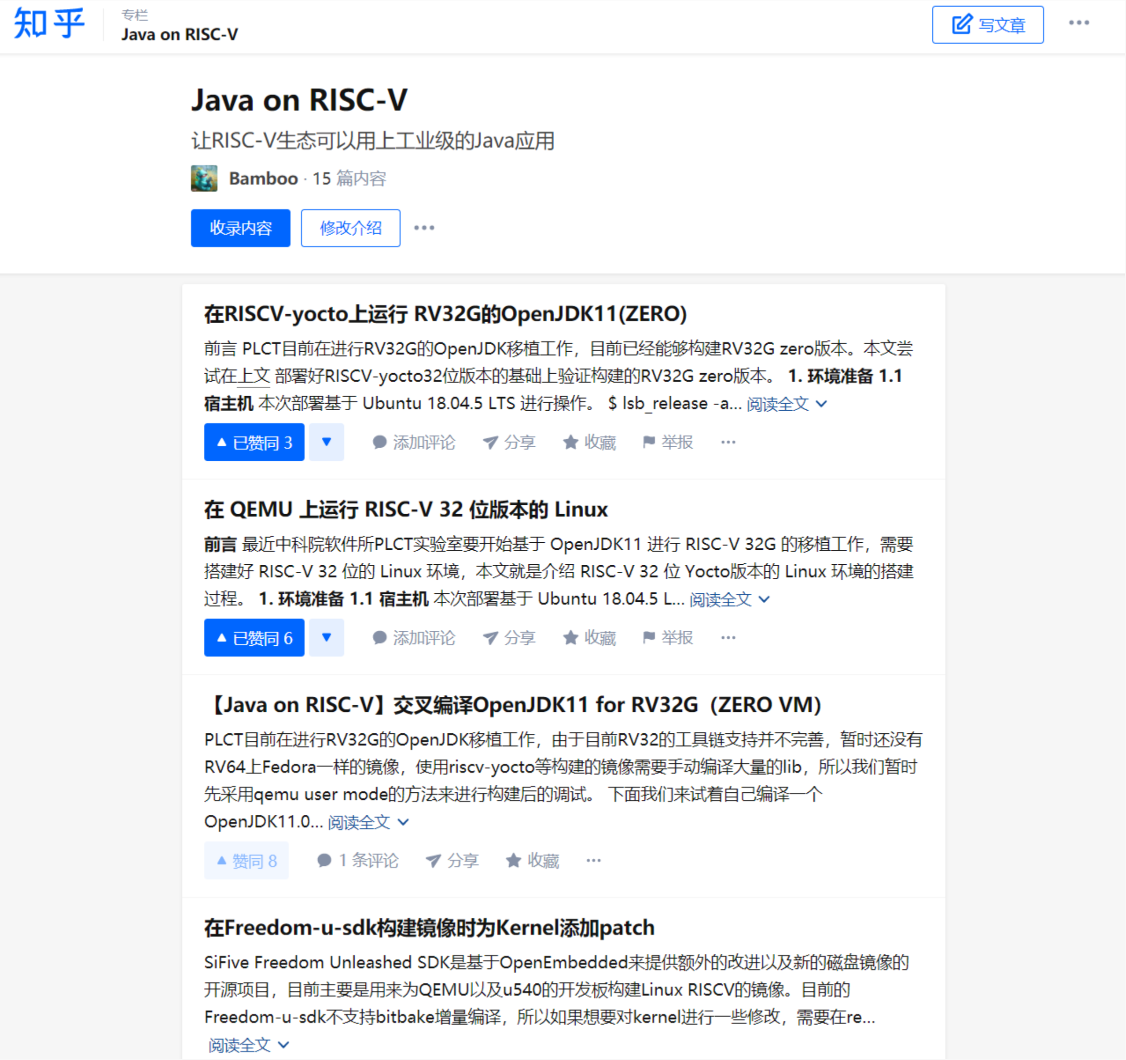
1.技术文章

2.毕昇JDK

3.OpenJDK for RV32G

02 我的工作

知乎专栏以及repo wiki



Java on RISC-V

让RISC-V生态可以用上工业级的Java应用

Bamboo · 15 篇内容

在RISCV-yocto上运行 RV32G的OpenJDK11(ZERO)

前言 PLCT目前在进行RV32G的OpenJDK移植工作，目前已经能够构建RV32G zero版本。本文尝试在上文 部署好RISCV-yocto32位版本的基础上验证构建的RV32G zero版本。 **1. 环境准备 1.1 宿主机** 本次部署基于 Ubuntu 18.04.5 LTS 进行操作。 $ lsb_release -a... 阅读全文

在 QEMU 上运行 RISC-V 32 位版本的 Linux

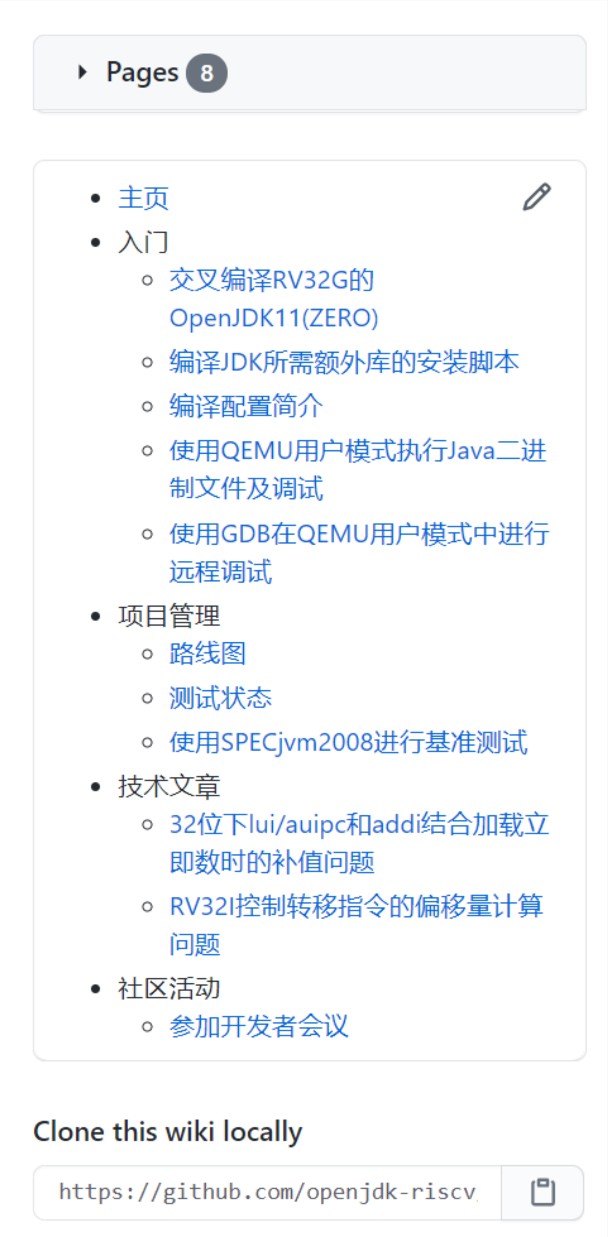
前言 最近中科院软件所PLCT实验室要开始基于 OpenJDK11 进行 RISC-V 32G 的移植工作，需要搭建好 RISC-V 32 位的 Linux 环境，本文就是介绍 RISC-V 32 位 Yocto版本的 Linux 环境的搭建过程。 **1. 环境准备 1.1 宿主机** 本次部署基于 Ubuntu 18.04.5 L... 阅读全文

【Java on RISC-V】交叉编译OpenJDK11 for RV32G（ZERO VM)

PLCT目前在进行RV32G的OpenJDK移植工作，由于目前RV32的工具链支持并不完善，暂时还没有RV64上Fedora一样的镜像，使用riscv-yocto等构建的镜像需要手动编译大量的lib，所以我们暂时先采用qemu user mode的方法来进行构建后的调试。 下面我们来试着自己编译一个OpenJDK11.0... 阅读全文

在Freedom-u-sdk构建镜像时为Kernel添加patch

SiFive Freedom Unleashed SDK是基于OpenEmbedded来提供额外的改进以及新的磁盘镜像的开源项目，目前主要是用来为QEMU以及u540的开发板构建Linux RISCV的镜像。目前的Freedom-u-sdk不支持bitbake增量编译，所以如果想要对kernel进行一些修改，需要在re... 阅读全文



- 主页
- 入门
  - 交叉编译RV32G的OpenJDK11(ZERO)
  - 编译JDK所需额外库的安装脚本
  - 编译配置简介
  - 使用QEMU用户模式执行Java二进制文件及调试
  - 使用GDB在QEMU用户模式中进行远程调试
- 项目管理
  - 路线图
  - 测试状态
  - 使用SPECjvm2008进行基准测试
- 技术文章
  - 32位下lui/auipc和addi结合加载立即数时的补值问题
  - RV32I控制转移指令的偏移量计算问题
- 社区活动
  - 参加开发者会议

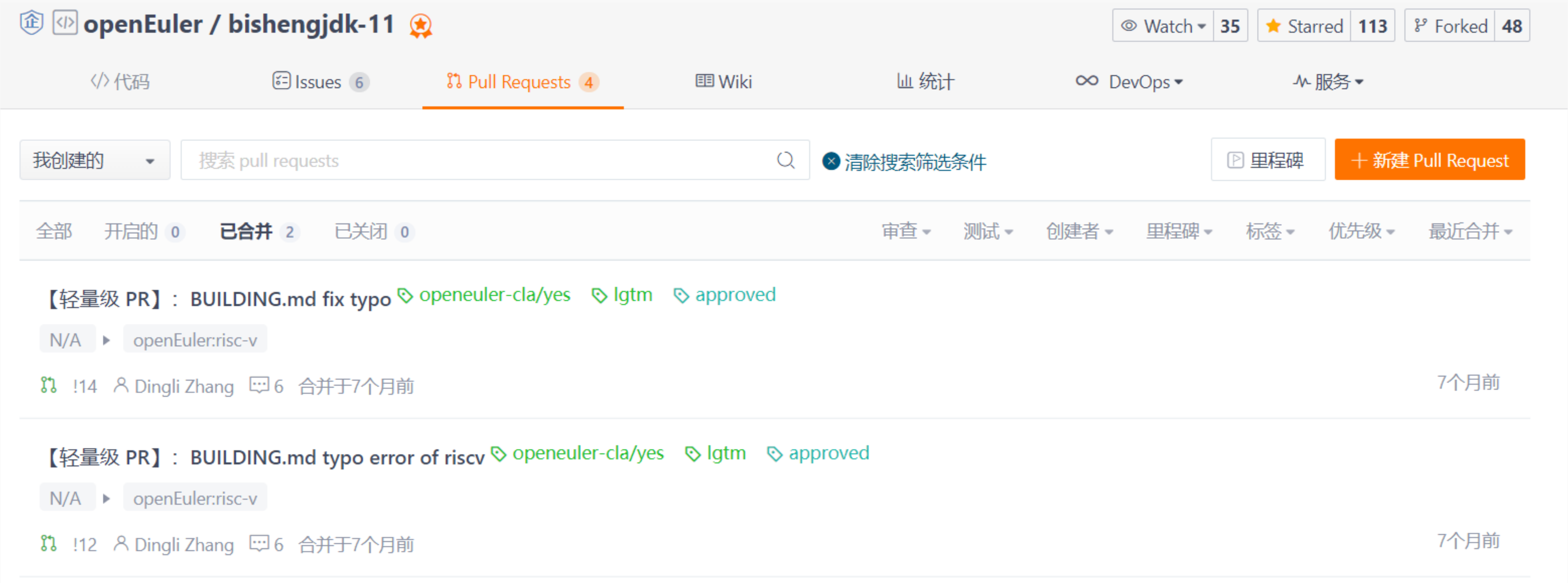
Clone this wiki locally

https://github.com/openjdk-riscv

参考资料：
[1] https://www.zhihu.com/column/c_1287750038518161408
[2] https://github.com/openjdk-riscv/jdk11u/wiki

02 我的工作：BishengJDK

Pull request for BishengJDK



【轻量级 PR】：BUILDING.md fix typo openeuler-cla/yes lgtm approved

N/A ▸ openEuler:risc-v

!14 Dingli Zhang 6 合并于7个月前 7个月前

【轻量级 PR】：BUILDING.md typo error of riscv openeuler-cla/yes lgtm approved

N/A ▸ openEuler:risc-v

!12 Dingli Zhang 6 合并于7个月前 7个月前

参考资料：

[1] https://gitee.com/openeuler/bishengjdk-11

02 我的工作：BishengJDK

Support for misaligned accessing

Solution of BishengJDK

```
static inline u2 get_native_u2(address p) {
  if ((intptr_t(p) & 1) == 0) {
    return *(u2*)p;
  } else {
    return ((u2)(p[1]) << 8) |
           ((u2)(p[0]));
  }
}

static inline void put_native_u2(address p, u2 x) {
  if ((intptr_t(p) & 1) == 0) {
    *(u2*)p = x;
  } else {
    p[1] = x >> 8;
    p[0] = x;
  }
}
```

Patch in OpenSBI

```
-  SET_RD(insn, regs, val.data_ulong << shift >> shift);
+  SET_RD(insn, regs, ((long)(val.data_ulong << shift)) >> shift);
```

各版本Linux在unleashed上对非对齐访问的支持

Linux for RV64版本	是否支持
Fedora	×
Ubuntu20.04	×
Freedom-u-sdk	✓

参考资料：

[1] https://gitee.com/openeuler/bishengjdk-11/commits/risc-v/src/hotspot/cpu/riscv64/bytes_riscv64.hpp

[2] https://github.com/riscv/opensbi/commit/7dcb1e1753e9c5daec0580779ea8c31778bff152

02 我的工作：BishengJDK

SPECjvm2008 benchmark

- Java虚拟机基准测试，其重点是执行单个应用程序的JRE的性能。
- 反映了硬件处理器和内存子系统的性能，但对文件I/O的依赖性较低。
- 利用现实生活中的应用程序和以区域为中心的基准测试。

下载安装包

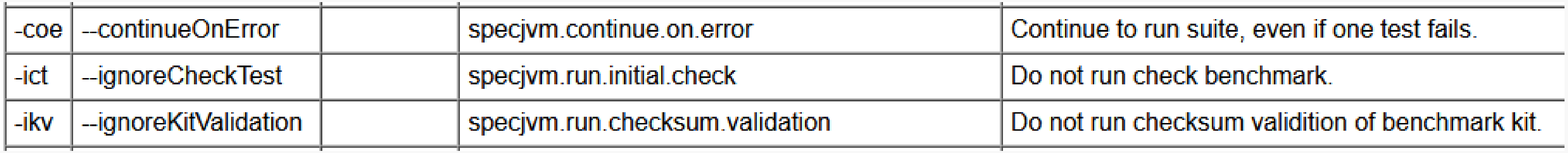
```
$ wget ftp://ftp.spec.org/dist/osg/java/SPECjvm2008_1_01_setup.jar
```

参考资料：
[1] http://www.spec.org/jvm2008/

02 我的工作：BishengJDK

SPECjvm2008 benchmark（Cont.）

测试bishengJDK for RV64G的一些额外参数



-coe	--continueOnError		specjvm.continue.on.error	Continue to run suite, even if one test fails.
-ict	--ignoreCheckTest		specjvm.run.initial.check	Do not run check benchmark.
-ikv	--ignoreKitValidation		specjvm.run.checksum.validation	Do not run checksum validition of benchmark kit.

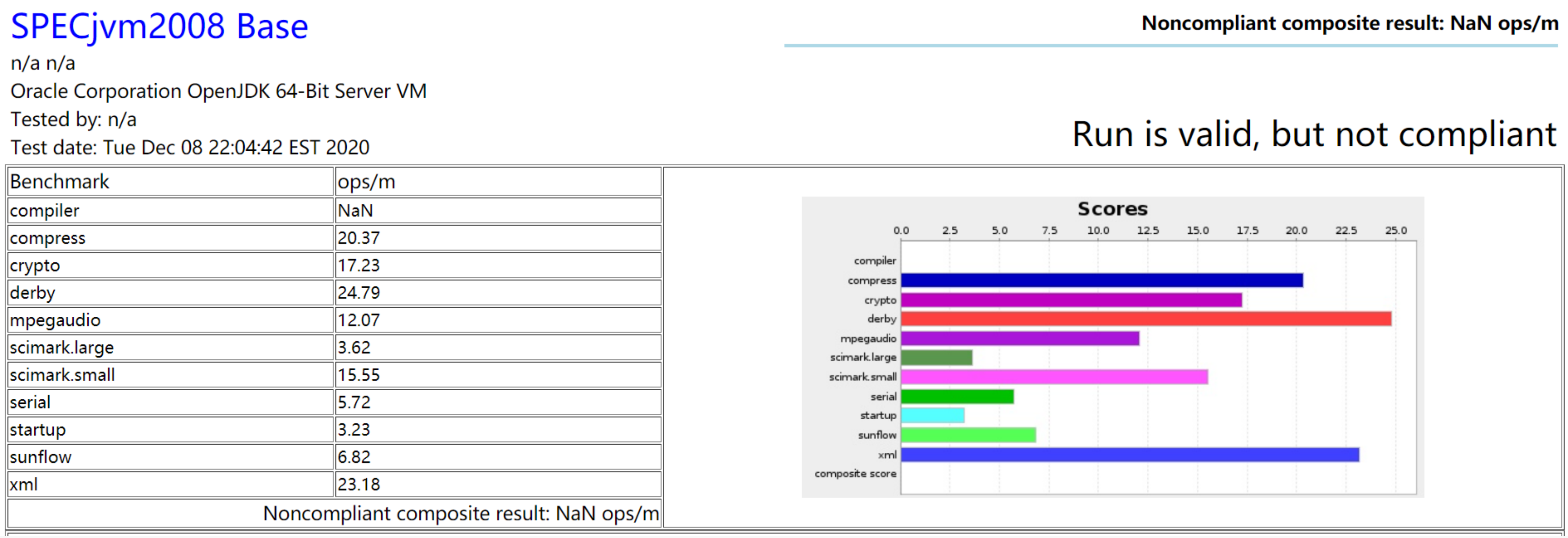
执行完整的测试并记录日志：

```
$ /path/to/jdk/java -jar SPECjvm2008.jar -ikv -ict -coe 2>&1 > SPECjvm2008.log
```

参考资料：
[1] http://www.spec.org/jvm2008/docs/UserGuide.html

02 我的工作：BishengJDK

SPECjvm2008 benchmark bishengJDK for RV64G的结果



SPECjvm2008 Base

Noncompliant composite result: NaN ops/m

n/a n/a
Oracle Corporation OpenJDK 64-Bit Server VM
Tested by: n/a
Test date: Tue Dec 08 22:04:42 EST 2020

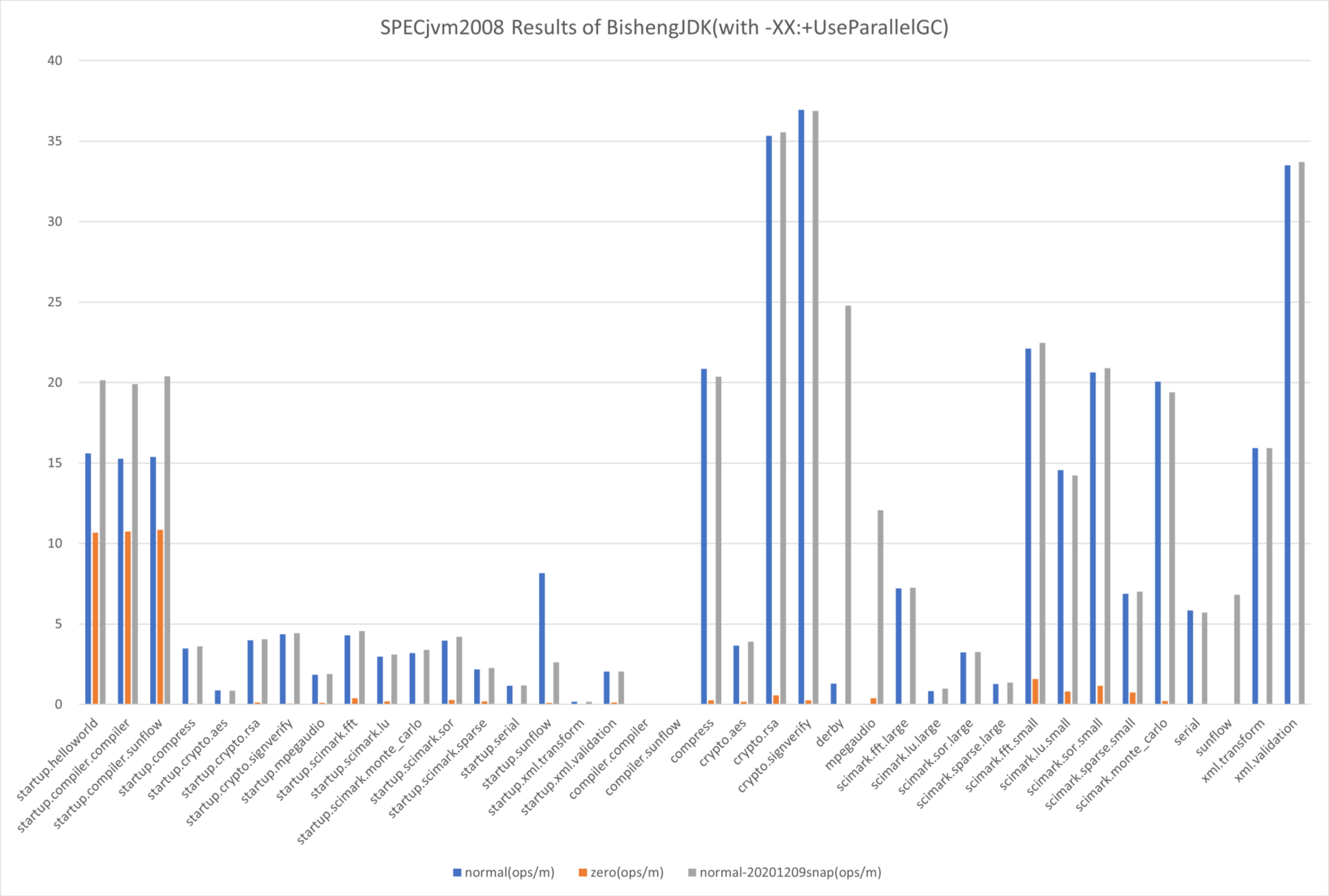
Run is valid, but not compliant

Benchmark	ops/m
compiler	NaN
compress	20.37
crypto	17.23
derby	24.79
mpegaudio	12.07
scimark.large	3.62
scimark.small	15.55
serial	5.72
startup	3.23
sunflow	6.82
xml	23.18
	Noncompliant composite result: NaN ops/m

```
$ /path/to/jdk/java -XX:+UseParallelGC -jar SPECjvm2008.jar -ikv -ict -coe
```

参考资料：
[1] http://www.spec.org/jvm2008/docs/UserGuide.html

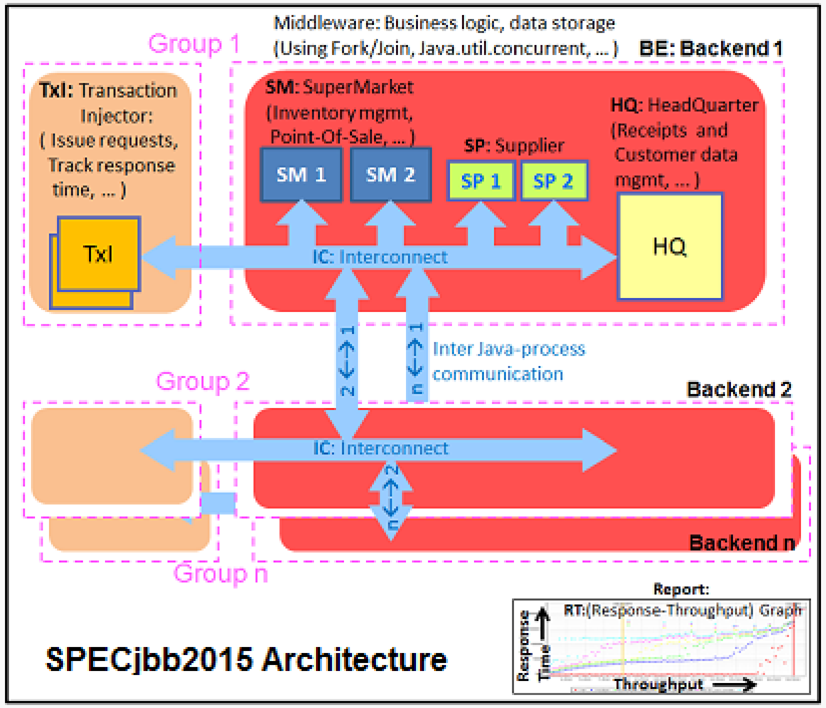
Cont.

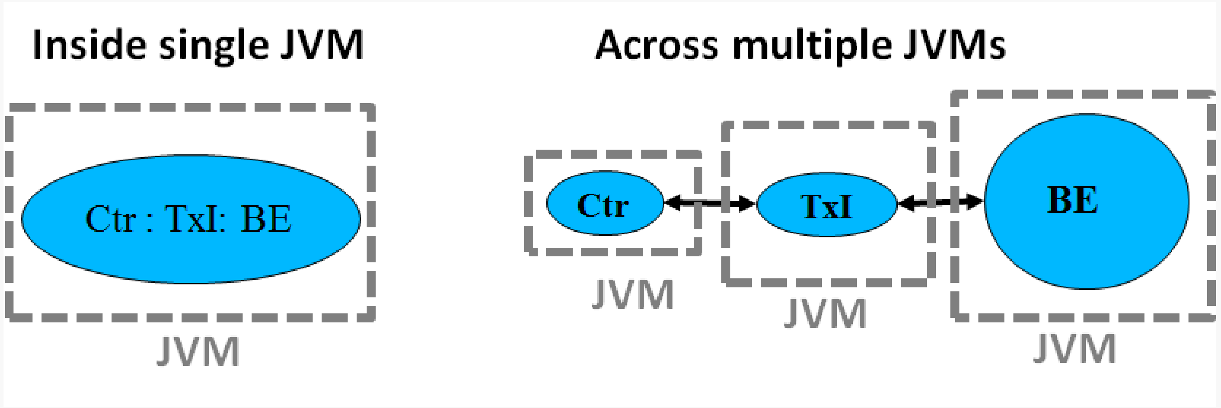

SPECjvm2008 Results of BishengJDK(with -XX:+UseParallelGC)
40
35
30
25
20
15
10
5
0
startup.helloworld
startup.compiler.compiler
startup.compiler.sunflow
startup.compress
startup.crypto.aes
startup.crypto.rsa
startup.crypto.signverify
startup.mpegaudio
startup.scimark.fft
startup.scimark.lu
startup.scimark.monte_carlo
startup.scimark.sor
startup.scimark.sparse
startup.serial
startup.sunflow
startup.xml.transform
startup.xml.validation
compiler.compiler
compiler.sunflow
compress
crypto.aes
crypto.rsa
crypto.signverify
derby
mpegaudio
scimark.fft.large
scimark.lu.large
scimark.sor.large
scimark.sparse.large
scimark.fft.small
scimark.lu.small
scimark.sor.small
scimark.sparse.small
scimark.monte_carlo
serial
sunflow
xml.transform
xml.validation
normal(ops/m)
zero(ops/m)
normal-20201209snap(ops/m)

02 我的工作：BishengJDK

SPECjbb2015 benchmark

- SPECjbb® 2015 基准测试基于一家拥有 IT 基础设施的公司，业务范围包括处理各种销售点请求、在线购买和数据挖掘等操作。
- SPECjbb 模拟了三层客户/服务器模型结构：第一层是用户（客户端输入）；第二层是商业应用逻辑；第三层是数据库。

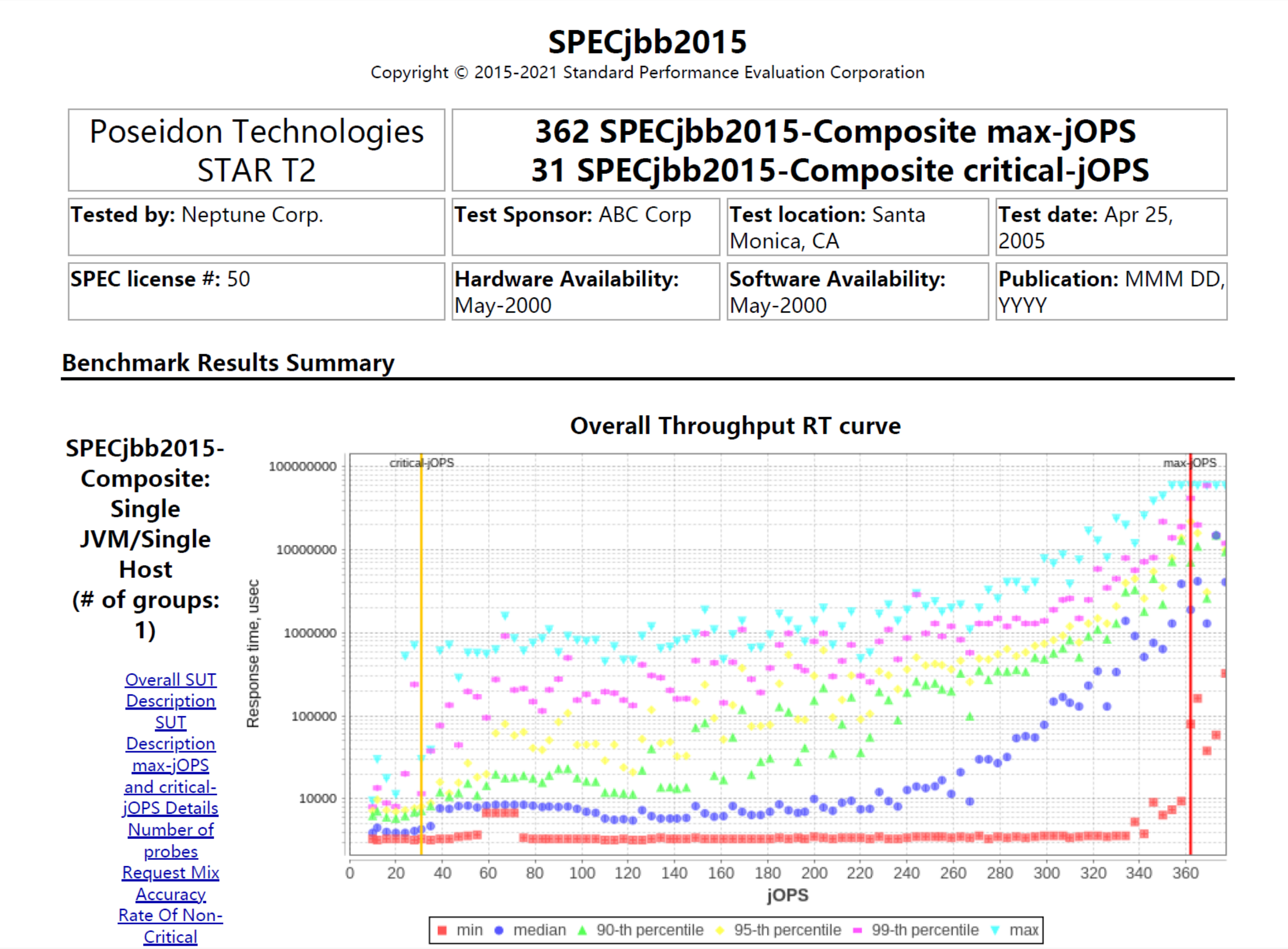

Middleware: Business logic, data storage
(Using Fork/Join, Java.util.concurrent, ...)
BE: Backend 1
Group 1
TxI: Transaction Injector: (Issue requests, Track response time, ...)
TxI
SM: SuperMarket (Inventory mgmt, Point-Of-Sale, ...)
SM 1
SM 2
SP: Supplier
SP 1
SP 2
HQ: HeadQuarter (Receipts and Customer data mgmt, ...)
HQ
IC: Interconnect
Inter Java-process communication
1 ↔ 2
1 ↔ n
Group 2
Backend 2
IC: Interconnect
2 ↔ n
Group n
Backend n
Report:
RT:(Response-Throughput) Graph
Response Time
Throughput
SPECjbb2015 Architecture



参考资料：
[1] https://www.spec.org/jbb2015/

02 我的工作：BishengJDK

SPECjbb2015 benchmark bishengJDK for RV64G的结果



SPECjbb2015

Copyright © 2015-2021 Standard Performance Evaluation Corporation

Poseidon Technologies STAR T2	362 SPECjbb2015-Composite max-jOPS 31 SPECjbb2015-Composite critical-jOPS		
Tested by: Neptune Corp.	**Test Sponsor:** ABC Corp	**Test location:** Santa Monica, CA	**Test date:** Apr 25, 2005
SPEC license #: 50	**Hardware Availability:** May-2000	**Software Availability:** May-2000	**Publication:** MMM DD, YYYY

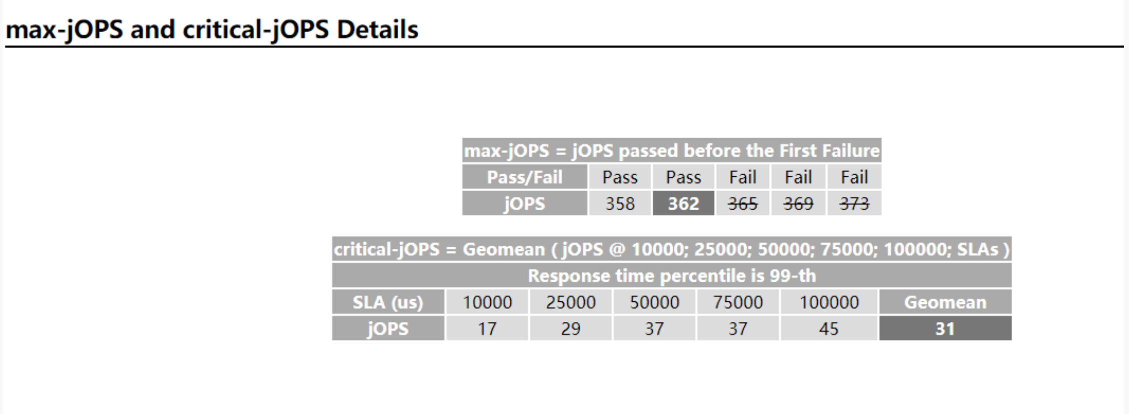
Benchmark Results Summary

SPECjbb2015-Composite: Single JVM/Single Host (# of groups: 1)

Overall SUT Description
SUT Description
max-jOPS and critical-jOPS Details
Number of probes
Request Mix Accuracy
Rate Of Non-Critical

Overall Throughput RT curve

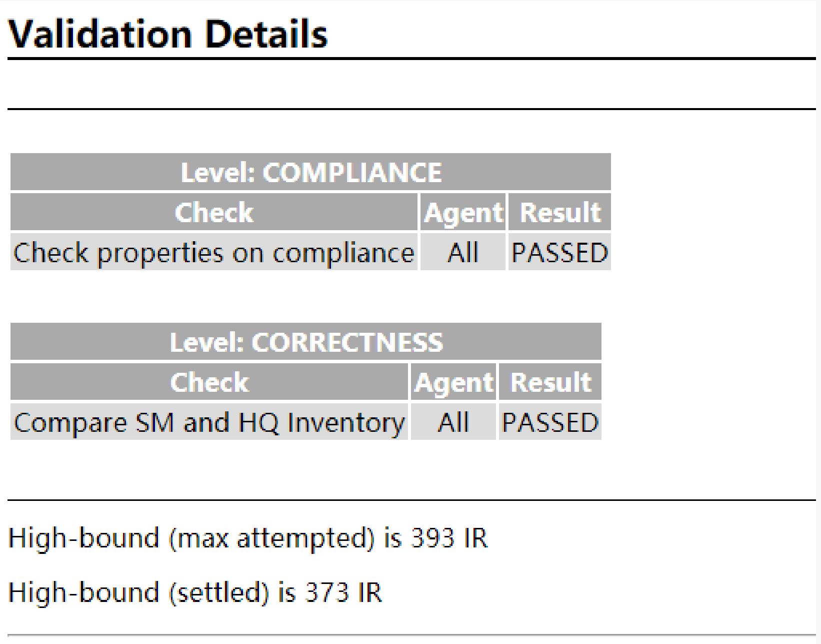
critical-jOPS
max-jOPS
Response time, usec
100000000
10000000
1000000
100000
10000
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
jOPS
■ min ● median ▲ 90-th percentile ◆ 95-th percentile ■ 99-th percentile ▼ max



max-jOPS and critical-jOPS Details

max-jOPS = jOPS passed before the First Failure					
Pass/Fail	Pass	Pass	Fail	Fail	Fail
jOPS	358	362	~~365~~	~~369~~	~~373~~

critical-jOPS = Geomean (jOPS @ 10000; 25000; 50000; 75000; 100000; SLAs)						
Response time percentile is 99-th						
SLA (us)	10000	25000	50000	75000	100000	Geomean
jOPS	17	29	37	37	45	31



Validation Details

Level: COMPLIANCE		
Check	Agent	Result
Check properties on compliance	All	PASSED

Level: CORRECTNESS		
Check	Agent	Result
Compare SM and HQ Inventory	All	PASSED

High-bound (max attempted) is 393 IR

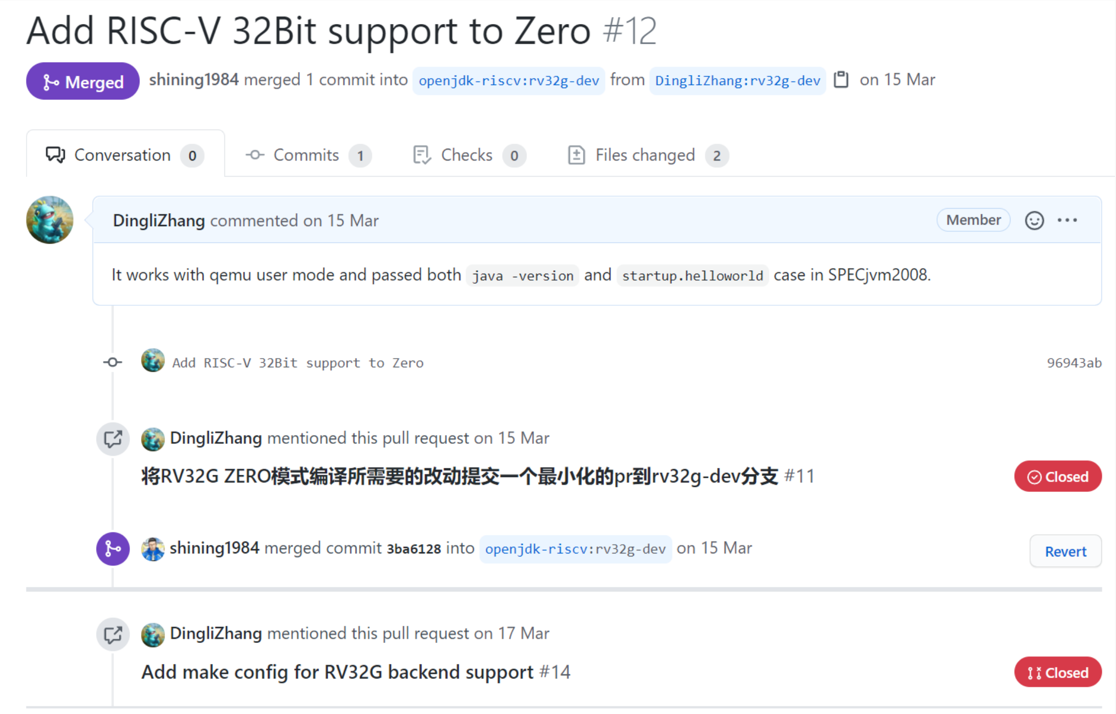
High-bound (settled) is 373 IR

执行命令：

```
$ /path/to/jdk/java -Xmx4g -jar specjbb2015.jar -m composite
```

03 我的工作：OpenJDK RV32G

Add zero support for OpenJDK RV32G



Add RISC-V 32Bit support to Zero #12

shining1984 merged 1 commit into `openjdk-riscv:rv32g-dev` from `DingliZhang:rv32g-dev` on 15 Mar

DingliZhang commented on 15 Mar

It works with qemu user mode and passed both `java -version` and `startup.helloworld` case in SPECjvm2008.

`Add RISC-V 32Bit support to Zero` 96943ab

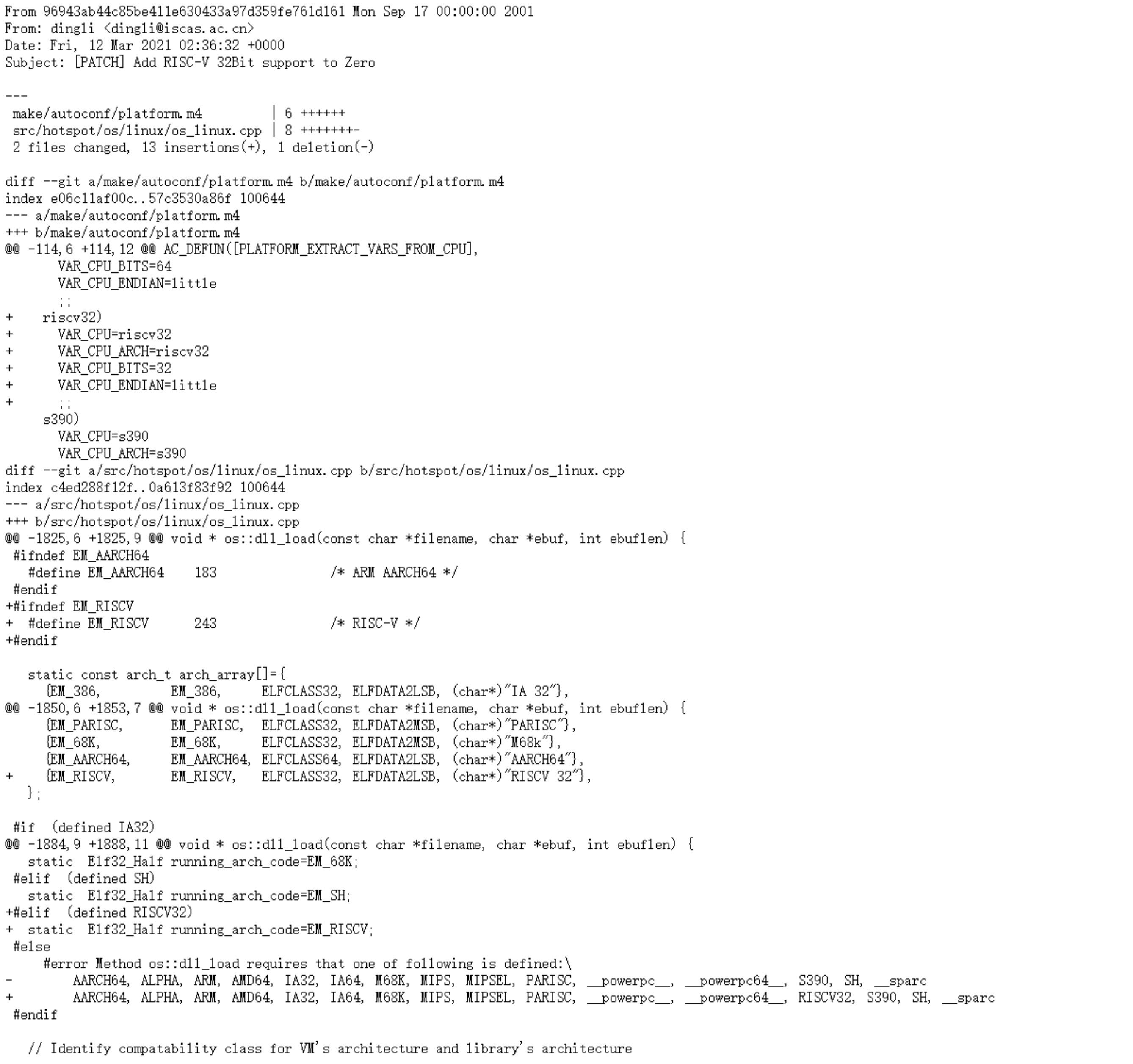
DingliZhang mentioned this pull request on 15 Mar

将RV32G ZERO模式编译所需要的改动提交一个最小化的pr到rv32g-dev分支 #11

shining1984 merged commit `3ba6128` into `openjdk-riscv:rv32g-dev` on 15 Mar

DingliZhang mentioned this pull request on 17 Mar

Add make config for RV32G backend support #14

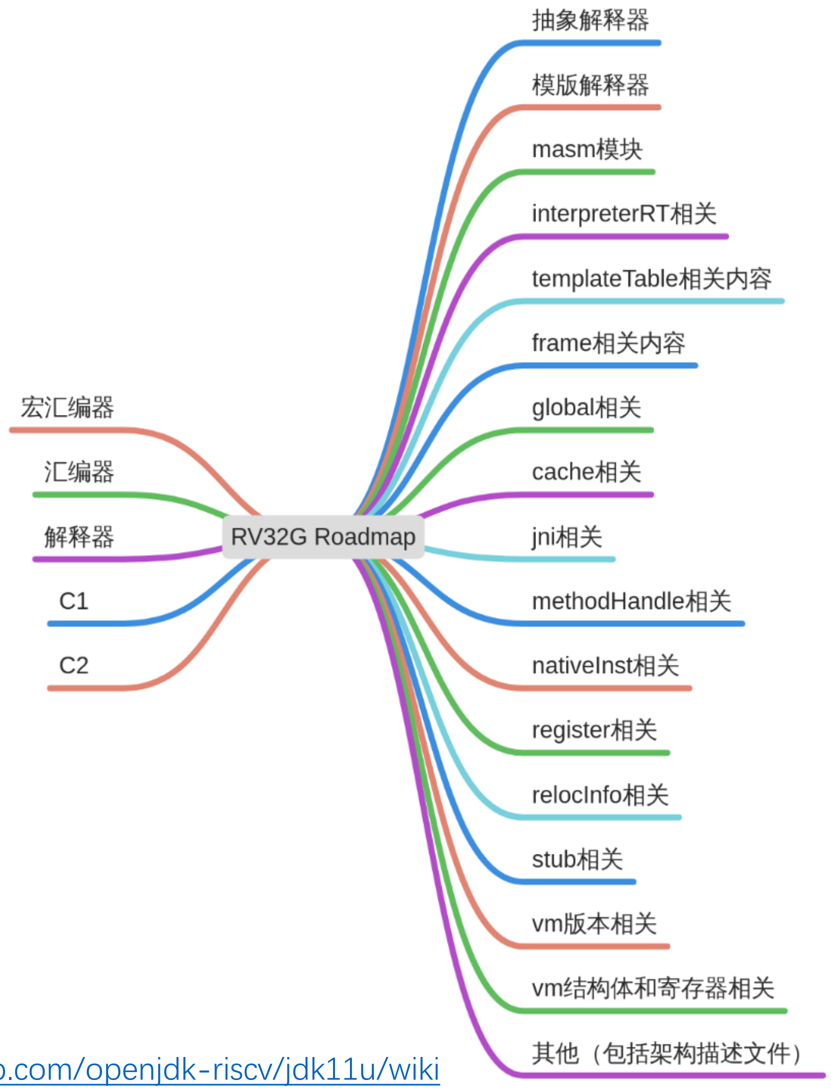


```
From 96943ab44c85be411e630433a97d359fe761d161 Mon Sep 17 00:00:00 2001
From: dingli <dingli@iscas.ac.cn>
Date: Fri, 12 Mar 2021 02:36:32 +0000
Subject: [PATCH] Add RISC-V 32Bit support to Zero

---
 make/autoconf/platform.m4          | 6 ++++++
 src/hotspot/os/linux/os_linux.cpp | 8 +++++++-
 2 files changed, 13 insertions(+), 1 deletion(-)

diff --git a/make/autoconf/platform.m4 b/make/autoconf/platform.m4
index e06c11af00c..57c3530a86f 100644
--- a/make/autoconf/platform.m4
+++ b/make/autoconf/platform.m4
@@ -114,6 +114,12 @@ AC_DEFUN([PLATFORM_EXTRACT_VARS_FROM_CPU],
       VAR_CPU_BITS=64
       VAR_CPU_ENDIAN=little
       ;;
+    riscv32)
+      VAR_CPU=riscv32
+      VAR_CPU_ARCH=riscv32
+      VAR_CPU_BITS=32
+      VAR_CPU_ENDIAN=little
+      ;;
     s390)
       VAR_CPU=s390
       VAR_CPU_ARCH=s390
diff --git a/src/hotspot/os/linux/os_linux.cpp b/src/hotspot/os/linux/os_linux.cpp
index c4ed288f12f..0a613f83f92 100644
--- a/src/hotspot/os/linux/os_linux.cpp
+++ b/src/hotspot/os/linux/os_linux.cpp
@@ -1825,6 +1825,9 @@ void * os::dll_load(const char *filename, char *ebuf, int ebuflen) {
 #ifndef EM_AARCH64
   #define EM_AARCH64    183               /* ARM AARCH64 */
 #endif
+#ifndef EM_RISCV
+  #define EM_RISCV      243               /* RISC-V */
+#endif

   static const arch_t arch_array[]={
     {EM_386,         EM_386,     ELFCLASS32, ELFDATA2LSB, (char*)"IA 32"},
@@ -1850,6 +1853,7 @@ void * os::dll_load(const char *filename, char *ebuf, int ebuflen) {
     {EM_PARISC,      EM_PARISC,  ELFCLASS32, ELFDATA2MSB, (char*)"PARISC"},
     {EM_68K,         EM_68K,     ELFCLASS32, ELFDATA2MSB, (char*)"M68k"},
     {EM_AARCH64,     EM_AARCH64, ELFCLASS64, ELFDATA2LSB, (char*)"AARCH64"},
+    {EM_RISCV,       EM_RISCV,   ELFCLASS32, ELFDATA2LSB, (char*)"RISCV 32"},
   };

 #if  (defined IA32)
@@ -1884,9 +1888,11 @@ void * os::dll_load(const char *filename, char *ebuf, int ebuflen) {
   static  Elf32_Half running_arch_code=EM_68K;
 #elif  (defined SH)
   static  Elf32_Half running_arch_code=EM_SH;
+#elif  (defined RISCV32)
+  static  Elf32_Half running_arch_code=EM_RISCV;
 #else
     #error Method os::dll_load requires that one of following is defined:\
-        AARCH64, ALPHA, ARM, AMD64, IA32, IA64, M68K, MIPS, MIPSEL, PARISC, __powerpc__, __powerpc64__, S390, SH, __sparc
+        AARCH64, ALPHA, ARM, AMD64, IA32, IA64, M68K, MIPS, MIPSEL, PARISC, __powerpc__, __powerpc64__, RISCV32, S390, SH, __sparc
 #endif

  // Identify compatability class for VM's architecture and library's architecture
```

参考资料：
[1] https://github.com/openjdk-riscv/jdk11u/pull/12

03 我的工作：OpenJDK RV32G

Roadmap for OpenJDK RV32G



入门文章

- 交叉编译RV32G的OpenJDK11(ZERO)
- 编译JDK所需额外库的安装脚本
- 编译配置简介
- 使用QEMU用户模式执行Java二进制文件及调试
- 使用GDB在QEMU用户模式中进行远程调试

参考资料：
[1] https://github.com/openjdk-riscv/jdk11u/wiki

03 我的工作：OpenJDK RV32G

跳转优化

```
void Assembler::movptr_with_offset(Register Rd, address addr, int32_t &offset) {
  ...
assert(is_unsigned_imm_in_range(imm32, 31, 0) || (imm32 == (uintptr_t)-1), "32-bit overflow in address constant");
  // Load 32 bits
  int32_t imm = imm32;
  int32_t upper = imm, lower = imm;
  lower = (lower << 20) >> 20;
  upper -= lower;
  lui(Rd, upper);
  addi(Rd, Rd, lower);
  // This offset will be used by following jalr/ld.
  offset = 0x0;
}

void Assembler::movptr(Register Rd, address addr) {
  int offset = 0;
  movptr_with_offset(Rd, addr, offset);
  addi(Rd, Rd, offset);
}
```

```
void Assembler::movptr(Register Rd, address addr) {
  int offset = 0;
  auipc(Rd, (int32_t)addr);
  addi(Rd, Rd, ((int32_t)addr << 20) >> 20);
}
```

参考资料：
[1] https://github.com/openjdk-riscv/jdk11u/pull/132

03 我的工作：OpenJDK RV32G

使用GDB在QEMU用户模式中进行远程调试

当我们在x86主机上使用QEMU用户模式开发运行RV32G JDK的时候，需要用到GDB进行远程调试。
首先进入到构建好的JDK镜像的bin目录下，然后在后台使用QEMU执行java -version:

```
$ cd /path/to/jvm/openjdk-11.0.9-internal/bin
$ /path/to/qemu/bin/qemu-riscv32 -L /path/to/riscv32/sysroot -g 33334 ./java -version &
```

接着运行工具链中的GDB:

```
$ /path/to/riscv32/bin/riscv32-unknown-linux-gnu-gdb --args ./java –version
```

接下来在GDB中连接调试端口（最开始步骤中-g后的参数即为端口号）：

```
(gdb) target remote localhost:33334
```

我们通常将main作为第一个断点，这样就会在src/java.base/share/native/launcher/main.c中的JLI_InitArgProcessing(jargc > 0, const_disable_argfile);这里打上断点：

```
(gdb) b main
```

参考资料：
[1] https://github.com/openjdk-riscv/jdk11u/wiki/Debug-with-GDB-in-QEMU-user-mode

03 我的工作：OpenJDK RV32G

使用GDB在QEMU用户模式中进行远程调试（cont.）

之后我们需要添加shared library的path，多个路径用:隔开：

```
(gdb) set solib-search-path /path/to/jvm/openjdk-11.0.9-internal/lib:/path/to/jvm/openjdk-11.0.9-internal/lib/jli:/path/to/jvm/openjdk-11.0.9-internal/lib/server
```

接着我们在需要调试的地方打上断点（以macroAssembler_riscv32.cpp:2632为例）：

```
(gdb) b macroAssembler_riscv32.cpp:2632
No source file named macroAssembler_riscv32.cpp.
Make breakpoint pending on future shared library load? (y or [n])
(gdb) y
```

继续输入c往下调试：

```
(gdb) c
```

这里会出现报错：Thread 1 received signal SIGTRAP, Trace/breakpoint trap.，我们需要切换到第二个进程：

```
(gdb) t 2
```

参考资料：
[1] https://github.com/openjdk-riscv/jdk11u/wiki/Debug-with-GDB-in-QEMU-user-mode

谢 谢

欢迎加入我们

知乎专栏：
Java on RISC-V : 让RISC-V生态可以用上工业级的Java应用
https://www.zhihu.com/column/c_1287750038518161408

Github：
https://github.com/openjdk-riscv/jdk11u