JVM at Loongson

敖琪

Alibaba Java Meetup, 2018/08/11

LOONGSON 龙芯

2001, Institute of Computing Technology, Chinese Academy of Sciences began to develop Loongson CPUs

2010, Loongson Technology Corporation Limited

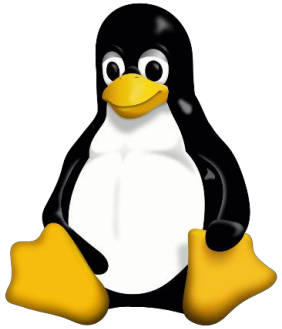
Loongson 1, 32-bit low-power, low-cost, for embedded and dedicated applications

Loongson 2, 64-bit low-power, single-core, mainly for industrial control and terminals

Loongson 3, 64-bit multi-core, mainly for desktops and servers

MIPS64 + LoongISA

We have to do:



OpenJDK

fedora

VxWORKS

Port Project for the MIPS Architecture

This project is sponsored by the OpenJDK Porters Group.

The purpose of this Project is to adapt and support OpenJDK on the MIPS family of architectures. Specific goals include:

- Keep the port up-to-date with the OpenJDK base. Currently, this means the version 7 line, but support for the new version 6 line is also an interesting possibility.
- Fill out missing pieces in the port. Currently, the template interpreter is complete but the client and server compilers are to be done.
- Generalize the port to other MIPS variants. The project scope is intended to be all MIPS variants, but principally modern MIPS64 and MIPS32, potentially with common recent ISA extensions. Currently, MIPS64 is the only implemented variant.

The Operating System for the port is Linux. It is intended to be distribution-agnostic.

Community

- Mailing list
  - mips-port

Loongson OpenJDK 8

MIPS64
Template Interpreter
C2
jdk8u181
JCK, jcstress, SPECjvm2008, SPECjbb2015...
Open Source Implementation of Java SE
Licensed under GPLv2 (with Classpath Exception)
JavaFX
Java Web Start
JNA

Loongson JVM Team's Work

1. Port OpenJDK to Loongson/MIPS
2. Solve Customers’ Java Related Problems
3. Verify low-level hardware and software, discover and adapt problems, and advice on the development of next-generation CPUs

OpenJDK 8 Source Code

Mercurial repository

```
ASSEMBLY_EXCEPTION  corba          jaxp   langtools  Makefile  README-builds.html
common              get_source.sh  jaxws  LICENSE    nashorn   test
configure           hotspot        jdk    make       README    THIRD_PARTY_README
```

hotspot

```
hotspot/src/cpu/   hotspot/src/os   hotspot/src/os_cpu/   hotspot/src/share/
├── mips           ├── aix          ├── aix_ppc           ├── tools
├── ppc            ├── bsd          ├── bsd_x86           └── vm
├── sparc          ├── linux        ├── bsd_zero
├── x86            ├── posix        ├── linux_mips
└── zero           ├── solaris      ├── linux_ppc
                   └── windows      ├── linux_sparc
                                    ├── linux_x86
                                    ├── linux_zero
                                    ├── solaris_sparc
                                    ├── solaris_x86
                                    └── windows_x86
```

Supported Architectures

- jdk6: sparc x86
- jdk7: ppc sparc x86
- jdk8: ppc sparc x86
- jdk9: aarch64 arm ppc s390 sparc x86 zero

C++ and Assembly Language

*.cpp *.hpp *.c *.h *.ad

How much code is needed to provide a full-featured OpenJDK on a new platform?

jdk8u/hotspot/src

- 1884 files
- 926141 lines

jdk8-mips64-public/hotspot/src

- 1990 files, +106
- 989994 lines, +63853

	X86	ppc	sparc	aarch64	mips
$arch + linux_$arch	125 files 105986 lines	83 files 48225 lines	101 files 59537 lines	108 files 62838 lines	106 files 61807 lines

Assembler, MacroAssembler, Disassembler

Interpreter

- cppInterpreter
- templateInterpreter

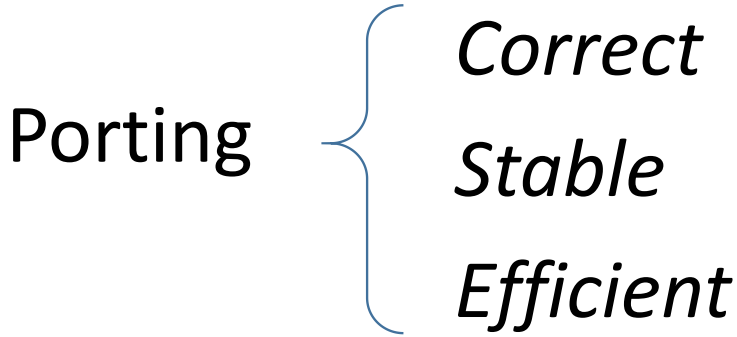
C1

- c1_*
- HIR => LIR => Assembly

C2

- opto
- .ad file(Architecture Description File)
- Register Definitions, Operand, Instruction
- Ad => ADL Compiler => hpp/cpp
- x86.ad ~5k lines, x86_32.ad ~13k lines, x86_64.ad ~12k lines
- sparc.ad ~11k lines
- ppc.ad ~12k lines
- mips_64.ad ~15k lines

Runtime


Porting
Correct
Stable
Efficient

regression, jvm98, jtreg, JCK

Dacapo, SPECjvm2008, SPECjbb2015...

Adapt to different CPUs, one binary compatible all

- java, javac, … => mips64el
- JIT => mips64el, gs464, gs464e

Verify low-level hardware and software

Advice on the development of next-generation CPUs

OCTLA Signatories List

The following organizations and individuals have signed the OpenJDK Community TCK License Agreement (OCTLA) and been granted access to the JCK.

- Signatories for Java SE 9, or later
  - Azul Systems, Inc.
  - BellSoft
  - Canonical
  - Fujitsu Technology Solutions GmbH
  - London Jamocha Community
  - Loongson Technology Co., Ltd.
  - MicroDoc Software GmbH
  - Red Hat
  - SAP
  - SUSE Linux GmbH
  - Twitter
- Signatories for Java SE 8
  - Alibaba Group Holding Limited
  - Amazon Fulfillment Services, Inc.
  - Azul Systems, Inc.
  - BellSoft
  - Canonical
  - Cavium
  - Emmanuel Bourg
  - The FreeBSD Foundation
  - Fujitsu Technology Solutions GmbH
  - Huawei Technologies Co. Ltd.
  - Intel
  - Linaro
  - London Jamocha Community
  - Loongson Technology Co., Ltd.
  - MicroDoc Software GmbH
  - Myriad Group AG
  - Red Hat
  - SAP
  - Supercomputing Systems AG
  - SUSE Linux GmbH
  - Twitter

Weak Memory Model Support

- MIPS is a "weak memory model" architecture
- additional memory barriers
- Parallel GC
- jsctress

Performance Optimization

- Unaligned Access Elimination
- LoongISA
- Vector
- Inline Method
- Mathematical Operation
- Memory Access Optimization
- Redundant Type Conversion Elimination
- Array Copy
- Get Thread
- Load Immediate
- Jump Optimization To Improve Branch Prediction Hit Rate
- Interpreter Instruction Dispatch
- Global Register
- Register Allocation Optimization

HotSpot VM Options

UseLoongsonISA

Use3A2000

UseCodeCacheAllocOpt

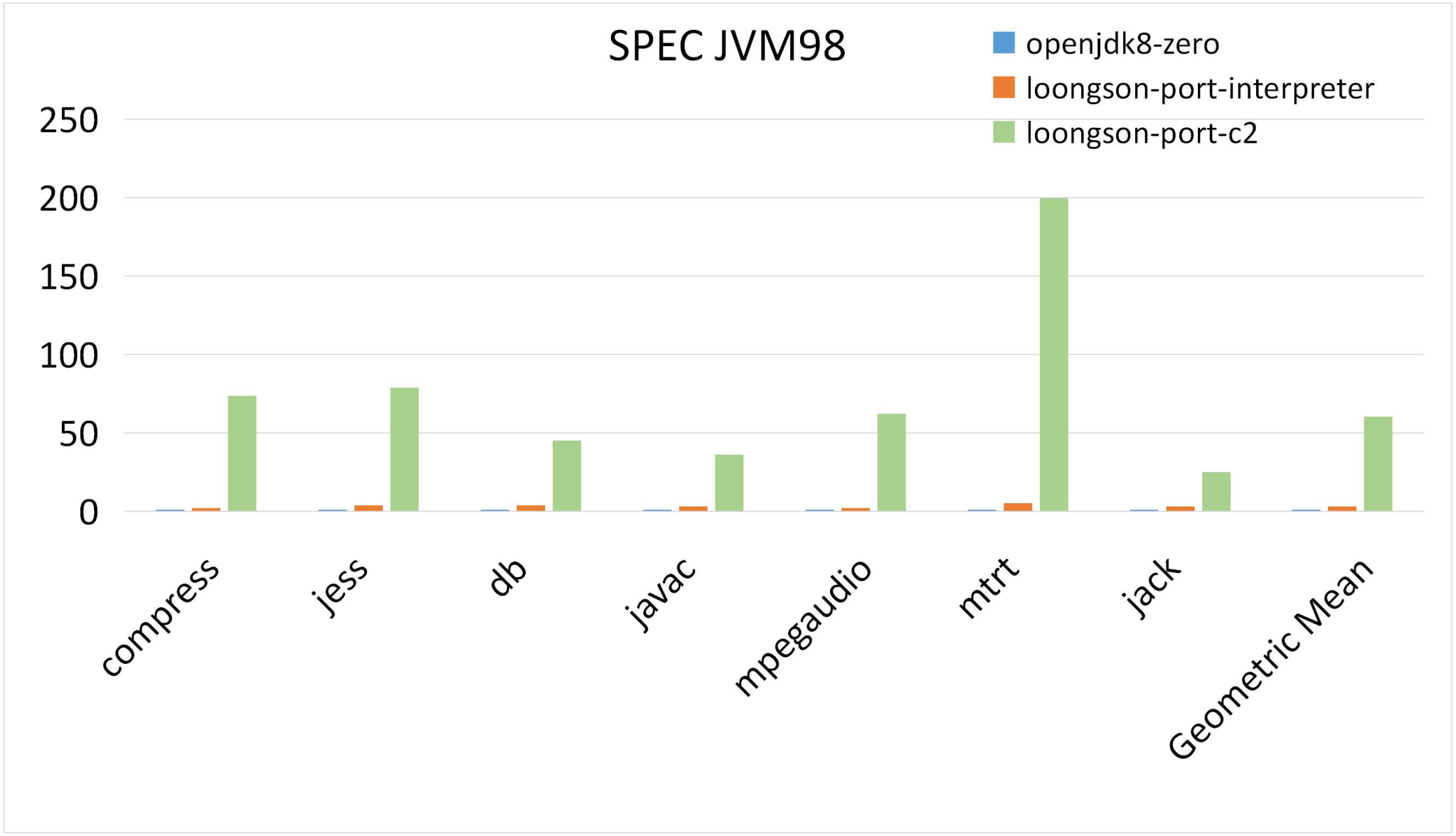
UseSyncLevel

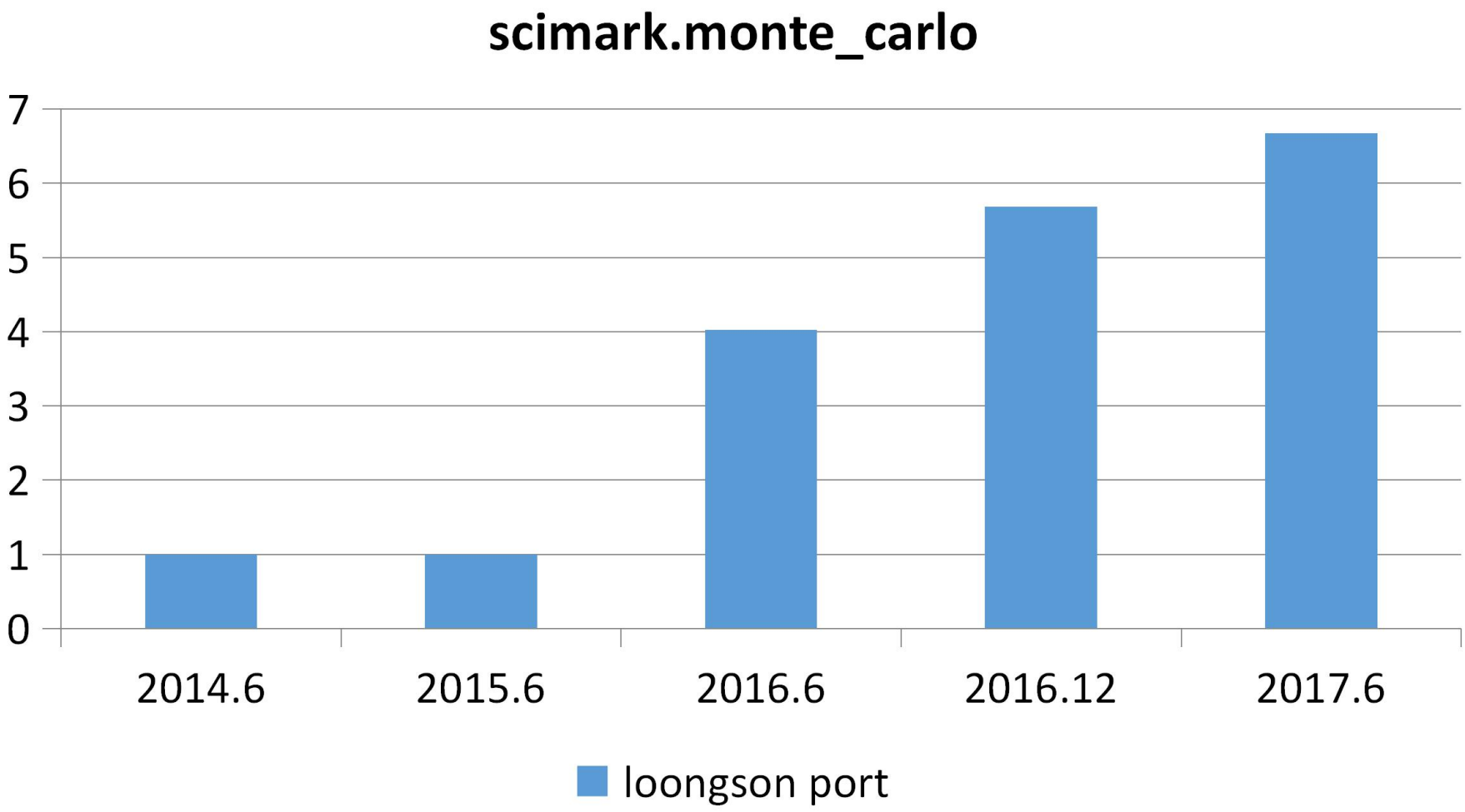
UseBoundCheckInstruction

SetFSFOFN

UseCountLeadingZerosInstructionMIPS64

UseCountTrailingZerosInstructionMIPS64


SPEC JVM98
openjdk8-zero
loongson-port-interpreter
loongson-port-c2
250
200
150
100
50
0
compress
jess
db
javac
mpegaudio
mtrt
jack
Geometric Mean


scimark.monte_carlo
7
6
5
4
3
2
1
0
2014.6
2015.6
2016.6
2016.12
2017.6
loongson port

Lessons Learned

Developing and debugging are hard

- Not much documentation, not many experienced people to ask, long time to explore, low efficiency
- Training a Developer

reseach => product

Testing is hard

- test suite
- jdk8u: *a new patch* + Officially Supported Port + Kernel + Library + OS + Boardcard + BIOS + CPU
- jdk8u-mips-public: *a new patch* + MIPS Port + Kernel + Library + OS + Boardcard + BIOS + CPU

Contributing to OpenJDK is hard

- OCA, OCTLA v2.0, OCTLAv3.0, JSPA,...
- New Port: 60k+ lines code
- Fast Moving

Not much used

- Need more usage to expose problems

repo

http://hg.loongnix.org/jdk8-mips64-public

Homepage

http://www.loongnix.org/index.php/Java

Mailing list

http://lists.loongnix.org/mailman/listinfo/java

龙芯问答

http://ask.loongnix.org/?/topic/JAVA

Next Steps

C1

Contributing to OpenJDK MIPS Project

Maintain and optimize jdk8u-mips64

Upgrade to the latest version of Openjdk, maybe 13, 14, 15, ...

Loongson JVM Team

Employee + Graduate Student
ICT, CAS. has graduated 3 doctors, more than 5 masters
~10 papers
More than 20 patents
JCP Full Member

about me

- Received a PhD in Computer Architecture from ICT in 2015
- aoqi@loongson.cn

Welcome to join us!

Thanks.
Question?