Architecture of the OpenJDK PPC Port

This page describes architectural decisions in the initial HotSpot VM PPC port from 2013. It explains extensions we made to shared code needed for PPC, but which might be also useful for other platforms. Further it gives details of implementations in the PPC part.

Supported operating systems and processor implementations

AIX: The port supports AIX 5.3 and later. It is tested to be built with xlC 10. Starting with 8u40, we only support xlC 12. Patch level 12.01.0000.0008 or higher is required.

Linux: We build and run the VM on SLES 10.3 with gcc 4.1.2, but support it on other Linux distributions, too. Google contributed a port for the new little-endian Linux variants.

We do not yet support iSeries PASE but support is upcoming.

Processor: The port supports only 64-bit PPC machines. It recognizes Power chips 5 through 7 and generates code optimized for these processors. So far, it lacks an instruction scheduler targeted to Power 6, which is an in-order issueing processor. Jdk 9 will contain support for Power 8.

Interpreter and Compiler

The port is derived from a VM targeted towards server applications. Therefore it focuses on maximizing peak performance. Thus, it utilizes the high end C2 optimizing compiler of HotSpot. First, we ported the CPP interpreter, but by now also added a port of the template interpreter. Since 8u20 big- and little-endian vairants should be built using the template interpreter, which is the default.

The port does not support the C1 compiler. SAP is working on this.

Supported HotSpot features

We support a wide range of basic functional flavours of the HotSpot VM. As of garbage collection, we support

- The default GC.
- Concurrent Mark and Sweep (-XX:+UseConcMarkSweepGC).

Support for G1 is also implemented, but we still test and fix issues.

We support **biased locking** to improve the locking performance. This is essential on PPC, as the lock instructions are expensive. Further we support **comp ressed Oo**ps to reduce memory usage.

The C2 compiler uses **implicit null checks**, which required some adaptions to shared code as on AIX the zero page is not protected.

The port does not support tiered compilation, as the C1 compiler is not ported.

Here you find more detailed informations:

- C++ interpreter features
- C2 compiler extensions and new optimizations