Inner Workings of Safepoints

Johannes Bechberger

mostlynerdless.de
Kinds of Safepoints

Local Safepoint/Handshake

since 2017
Kinds of Safepoints

Global Safepoint/Handshake
And they get ever more important
```c
int mul(int a, int b) {
    int res = 0;
    while (b > 0) {
        res += a;
        b--;
    }
    return res;
}
```

Beware of inlining
int mul(int a, int b) {
    int res = 0;
    for (int i = 0; i < b; i++) {
        res += a;
    }
    return res;
}
int mul(int a, int b) {
    int res = 0;
    for (int j = 0; j < b; j += 1000) {
        for (int i = j; i < j + 1000; i++) {
            res += i;
        }
    }
    return res;
}

Loop Strip Mining
**Template interpreter produces no safepoint check for return bytecodes**

- **Type:** Bug
- **Priority:** P3
- **Affects Version/s:** 22
- **Component/s:** hotspot
- **Labels:** interpreter
- **Subcomponent:** runtime
- **Resolved In Build:** b12
- **CPU:** arm, aarch64, riscv

**Resolved**

**Details**

The template interpreter produces a safepoint check for return bytecodes (`TemplateTable::_return(TosState state)`) on x86 [1] and other platforms, but not on aarch64, arm, and riscv.

I describe the bug in more detail at https://mostlynerdless.de/blog/2023/07/31/the-inner-workings-of-safepoints/.

[1] https://github.com/openjdk/jdk/blob/5d193193a3a4c519e7b3d77b27e6b2bf1b11c7f9/src/hotspot/cpu/x86/templateTable_
[2] https://github.com/openjdk/jdk/blob/5d193193a3a4c519e7b3d77b27e6b2bf1b11c7f9/src/hotspot/cpu/aarch64/templateTable_aarch64.cpp#L2174C27-L2174C27
Safepoint biased profilers are:

- Tricksy
- Sneaksy
- Filthy
- All of the above

― Nitsan Wakart
Implementation
We could just...

```cpp
if (thread->at_safepoint()) {
    SafepointMechanism::process();
} else {
    // do nothing
}
```

Slow, only in interpreted mode
Emit in MacroAssembler

testb(Address(thread_reg, JavaThread::polling_word_offset()), SafepointMechanism::poll_bit());

// handshake bit set implies poll
jcc(Assembler::notZero, slow_path);

https://github.com/openjdk/jdk/blob/a18b03b86fdd0eef773badbcd46607a8e5a068a/src/hotspot/cpu/x86/macroAssembler_x86.cpp#L3073
We could just...

```cpp
if (thread->at_safepoint()) {
    SafepointMechanism::process(); // do nothing
} else {
    rare
    // do nothing
    often
}
```
We could just...

```cpp
if (thread->at_safepoint()) {
    SafepointMechanism::process(); // slow path
} else {
    // do nothing // fast path
}
```
Read from pointer

- Data
- Segmentation fault
void method() {
    ...
    // check
}

Current Thread

<table>
<thead>
<tr>
<th>name</th>
<th>good page</th>
</tr>
</thead>
<tbody>
<tr>
<td>polling_page</td>
<td>bad page</td>
</tr>
</tbody>
</table>

Safepoint

_disabled

enabled

⚠️ Segfault

Safepoint
Initialize per thread

```c++
char* bad_page = polling_page;
char* good_page = polling_page + page_size;

os::protect_memory(bad_page, page_size, os::MEM_PROT_NONE);
os::protect_memory(good_page, page_size, os::MEM_PROT_READ);

_poll_page_armed_value =
    reinterpret_cast<uintptr_t>(bad_page);
_poll_page_disarmed_value =
    reinterpret_cast<uintptr_t>(good_page);
```

https://github.com/openjdk/jdk/blob/70e7cddcb9ecedde455cf701b5c7ad05286ac0067d/src/hotspot/share/runtime/safepointMechanism.cpp#L65
Emit e.g. in C1

```c
int LIR_Assembler::safepoint_poll(
    LIR_Opr tmp, CodeEmitInfo* info) {

    int offset = __ offset();
    const Register poll_addr = rscratch1;

    __ movp.ptr(poll_addr,
                Address(r15_thread,
                        JavaThread::polling_page_offset()));
    // ...
}
```

https://github.com/openjdk/jdk/blob/70e7cdcb9eced455cf701b5c7ad05286ac0067d/src/hotspot/cpu/x86/c1_LIRAssembler_x86.cpp#L543
Arming local safepoints

```c++
void SafepointMechanism::arm_local_poll(
    JavaThread* thread) {
  thread->poll_data()
      ->set_polling_word(_poll_word_armed_value);
  thread->poll_data()
      ->set_polling_page(_poll_page_armed_value);
}
```
Arming global safepoints

```c
_state = _synchronizing;
for (JavaThreadIteratorWithHandle jtiwh;
     JavaThread *cur = jtiwh.next();)
     {
     SafepointMechanism::arm_local_poll(cur);
    }
```
Tracking Safepoints
SafepointBegin

Category: Java Virtual Machine / Runtime / Safepoint

Safepointing begin

<table>
<thead>
<tr>
<th>Configuration</th>
<th>enabled</th>
<th>threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>true</td>
<td>10 ms</td>
</tr>
<tr>
<td>profiling</td>
<td>true</td>
<td>0 ms</td>
</tr>
</tbody>
</table>

Field | Type  | Description
--- | --- | ---
 safepointId | ulong | Safepoint Identifier
 totalThreadCount | int | Total Threads The total number of threads
 jniCriticalThreadCount | int | JNI Critical Threads The number of thread sections

Examples 3