Loop Strip Mining Optimization in Hotspot C2 Just In Time Compiler

BangaloreJUG

Rahul Raghavan

rahul.v.raghavan@oracle.com

JavaYourNext (Cloud)

Nov 23, 2018

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

- **Why / What is loop strip mining?**
  - *related background, requirement*
- **Implementation, New flags**
- **Related OpenJDK process flow**
  - *testing / reviews / commit / bug trails / bug fixes*
- **Summary / takeaways**

Why loop strip mining?

- **Safepoint**
  - Stage of execution where the **state** of the executing thread is well described.
  - Java threads **poll** a 'safepoint flag' at reasonable intervals.

- Safepoint poll **locations**
  - Between any 2 bytecodes while running in the interpreter.
  - Method entry/exit in JIT compiled code.
  - On '**non-counted**' loop back edge in JIT compiled code.

- **Counted loops**
  - **By deafult** - No safepoints.
  - Problems for very long counted loops - irregular timeouts, non-responsive application etc.
  - Hurts **low latency GCs**.

- -XX:+**UseCountedLoopSafepoints**
  - Workaround by adding one safepoint per iteration.
  - Was disabled by default.
  - Performance penalty.

- ‘Loop Strip Mining’
  - Best of both worlds (UseCountedLoopSafepoints ON, OFF).
  - Low time to safepoint with a little to no impact on throughput.

C2 loop strip mining – JBS tasks

<JDK Bug System - bugs.openjdk.java.net>

- JDK-**5014723** - ‘implement strip mining loop optimization’
  - An old hotspot / compiler / P4 / bug created in Mar/2004; now closed after 8186027 fix

- JDK-**8186027** - ‘C2 loop strip mining’
  - [Created Aug 2017 / Resolved Nov 2017]
  - Proposed and contributed by Redhat / **Shenandoah-project team
  - JBS Description - “*We've implemented loop strip mining as a way to balance pause time and throughput in the Shenandoah repo. We've been using it for several months and it is stable as far as we can tell. We intend to propose it for jdk 10.*”

**‘Shenandoah’
  - http://openjdk.java.net/projects/shenandoah/ - “*Shenandoah is an ultra-low pause time garbage collector that reduces GC pause times by performing more garbage collection work concurrently with the running Java program. CMS and G1 both perform concurrent marking of live objects. Shenandoah adds concurrent compaction.*”

C2 loop strip mining – OpenJDK Contribution

JDK-8186027 - C2 loop strip mining
https://bugs.openjdk.java.net/browse/JDK-8186027

Extracts from JBS

- Type : Enhancement
- Priority : P4
- Component/s : hotspot
- Subcomponent : compiler
- Labels : c2, c2-loopopts
- Created : 2017-08-09
- Resolved : 2017-11-28
- Fix Version/s : 10
- Resolved In Build : b36

C2 loop strip mining - 8186027 fix

- ‘Loop Strip Mining’
  - Best of both worlds (UseCountedLoopSafepoints ON, OFF).
  - Low time to safepoint with a little to no impact on throughput.

- Converts loop:

```
for (int i = start; i < stop; i += inc) {
  // body
}
```

- To a loop nest:

```
i = start;
if (i < stop) {
  do {
    int next = MIN(stop, i+LoopStripMiningIter*inc);
    do {
      // body
      i += inc;
    } while (i < next);
    safepoint();
  } while (i < stop);
}
```

C2 loop strip mining - 8186027 fix

- Introduced new **LoopStripMiningIter** option.
  [*open/src/hotspot/share/opto/c2_globals.hpp*]

```
product(uintx, LoopStripMiningIter, 0,                  \
        "Number of iterations in strip mined loop")     \
        range(0, max_juint)                             \
```

- At present by default enabled for **G1 gc, ZGC and Epsilon gc** (In openjdk jdk/jdk repo)
  - **Default** option values: **-XX:+UseCountedLoopSafepoints -XX:LoopStripMiningIter=1000**
    [*G1Arguments::initialize() - open/src/hotspot/share/gc/g1/g1Arguments.cpp*]
    [*Similarly Zarguments::initialize(), EpsilonArguments::initialize()*]

- To **disable** loop strip mining: **-XX:LoopStripMiningIter=0 OR -XX:-UseCountedLoopSafepoints**

- Arguments validity, dependency, consistency checked at -
  [*CompilerConfig::check_args_consistency() - open/src/hotspot/share/compiler/compilerDefinitions.cpp*]

C2 loop strip mining - 8186027 fix

- Extracts from initial ‘RequestForReview’ email -

“… change was first pushed to the shenandoah repo several months ago
and we've been running with it enabled since.
The command line argument LoopStripMiningIter is the number of iterations between safepoints.
*In practice, with an arbitrary **LoopStripMiningIter=1000**,*
we observe time to safepoint on par with the current -XX:+UseCountedLoopSafepoints and
most performance regressions due to -XX:+UseCountedLoopSafepoints gone.”

Review Email Thread for C2 loop strip mining 8186027

Subject	From	Correspondents	Date	Location
RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Tuesday 03 October 2017 06:49 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Nils Eliasson	Nils Eliasson	Wednesday 11 October 2017 06:45 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Wednesday 11 October 2017 07:23 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Nils Eliasson	Nils Eliasson	Monday 23 October 2017 07:46 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Tuesday 24 October 2017 02:07 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Vladimir Kozlov	Vladimir Kozlov	Wednesday 25 October 2017 04:32 AM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Wednesday 25 October 2017 07:59 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Vladimir Kozlov	Vladimir Kozlov	Friday 27 October 2017 09:20 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Friday 27 October 2017 09:39 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Vladimir Kozlov	Vladimir Kozlov	Saturday 28 October 2017 02:49 AM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Monday 30 October 2017 10:32 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Vladimir Kozlov	Vladimir Kozlov	Tuesday 07 November 2017 02:25 AM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Tuesday 07 November 2017 10:19 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Vladimir Kozlov	Vladimir Kozlov	Tuesday 07 November 2017 10:44 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Monday 20 November 2017 03:57 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Vladimir Kozlov	Vladimir Kozlov	Wednesday 22 November 2017 03:06 AM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Wednesday 22 November 2017 08:27 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Vladimir Kozlov	Vladimir Kozlov	Thursday 23 November 2017 01:28 AM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Thursday 23 November 2017 07:48 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Nils Eliasson	Nils Eliasson	Friday 24 November 2017 03:29 AM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Andrew Haley	Andrew Haley	Friday 24 November 2017 02:08 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Thomas Schatzl	Thomas Schatzl	Friday 24 November 2017 02:52 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Vladimir Kozlov	Vladimir Kozlov	Monday 27 November 2017 11:49 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Thomas Schatzl	Thomas Schatzl	Tuesday 28 November 2017 01:14 AM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Tuesday 28 November 2017 04:30 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Vladimir Kozlov	Vladimir Kozlov	Tuesday 28 November 2017 10:46 PM	hotspot-compiler-dev@openjdk.java.net
Re: RFR(L): 8186027: C2: loop strip mining	Roland Westrelin	Roland Westrelin	Wednesday 29 November 2017 09:03 PM	hotspot-compiler-dev@openjdk.java.net

Testing, final changeset for C2 loop strip mining 8186027

"…. Beyond verifying performance results with the usual benchmarks, when I implemented that change, I wrote test cases for (hopefully) every loop optimization and verified by inspection of the generated code that the loop opt triggers correct with loop strip mining.

Roland."

HG Updates added a comment - 2017-11-28 23:27

URL: http://hg.openjdk.java.net/jdk/hs/rev/f913f6dba2d3
User: kvn
Date: 2017-11-28 18:02:13 +0000

HG Updates added a comment - 2017-12-07 14:29

URL: http://hg.openjdk.java.net/jdk/jdk/rev/f913f6dba2d3
User: jwilhelm
Date: 2017-12-07 09:00:22 +0000

OpenJDK / jdk / jdk
changeset 48145:f913f6dba2d3

```
8186027: C2: loop strip mining
Reviewed-by: kvn, neliasso
```

author roland
date Tue, 28 Nov 2017 11:59:16 +0100 (10 months ago)
parents 364207a23251
children 646ed97b7e0d
files src/hotspot/share/gc/g1/g1Arguments.cpp src/hotspot/share/opto/c2_globals.hpp src/hotspot/share/opto/cfgnode.cpp src/hotspot/share/opto/classes.hpp src/hotspot/share/opto/compile.cpp src/hotspot/share/opto/ifnode.cpp src/hotspot/share/opto/loopPredicate.cpp src/hotspot/share/opto/loopTransform.cpp src/hotspot/share/opto/loopUnswitch.cpp src/hotspot/share/opto/loopnode.cpp src/hotspot/share/opto/loopnode.hpp src/hotspot/share/opto/loopopts.cpp src/hotspot/share/opto/macro.cpp src/hotspot/share/opto/node.hpp src/hotspot/share/opto/superword.cpp src/hotspot/share/runtime/arguments.cpp test/hotspot/jtreg/compiler/loopopts/UseCountedLoopSafepointsTest.java
diffstat 17 files changed, 1152 insertions(+), 211 deletions(-) [+]

C2 loop strip mining Related tests

* *(jdk/jdk repo / as of Nov 2018)*

- open/test/hotspot/jtreg/compiler/loopstripmining/
  - BackedgeNodeWithOutOfLoopControl.java
  - CheckLoopStripMiningIterShortLoop.java
  - LimitSharedwithOutOfLoopTest.java
  - StripMinedLoopReorgOffsets.java
  - UnexpectedNodeInOuterLoopWhenCloning.java
  - UnexpectedPinnedNodeInOuterLoop.java

- open/test/hotspot/jtreg/compiler/loopopts/
  - UseCountedLoopSafepointsTest.java
  - TestCountedLoopSafepointBackedge.java
  - TestOneIterationStripMined.java
  - TestStripMinedBackToBackIfs.java

Summary, Takeaways

- Be Aware (and Beware) of
  - **LoopStripMiningIter** option value - Number of iterations between safepoints in strip mined loop
  - (and also **LoopStripMiningIterShortLoop** value)
  - Dependecy of Loop Strip Mining options with **UseCountedLoopSafepoints** option.

- Complexity, testing requirements of hotspot / compiler changes.
- OpenJDK community contributions.
- Role played by – Early Access (EA) builds.

References, Links

- https://bugs.openjdk.java.net/browse/JDK-8186027
- https://bugs.openjdk.java.net/browse/JDK-5014723
- http://mail.openjdk.java.net/pipermail/hotspot-compiler-dev/2017-October/027193.html
- https://bugs.openjdk.java.net/browse/JDK-8196296
- http://mail.openjdk.java.net/pipermail/hotspot-compiler-dev/2018-January/028155.html
- http://mail.openjdk.java.net/pipermail/hotspot-compiler-dev/2018-January/028161.html

- http://openjdk.java.net/projects/shenandoah/
- http://openjdk.java.net/census#shenandoah
- http://hg.openjdk.java.net/shenandoah
- https://wiki.openjdk.java.net/display/shenandoah/Main
- http://openjdk.java.net/jeps/189
- https://bugs.openjdk.java.net/browse/JDK-8046179

Thank you for your attention!