GHC heap internals

Nikita Frolov <frolov@chalmers.se>
<table>
<thead>
<tr>
<th>RAM Footprint (approx)</th>
<th>Concurrent Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3 KB</td>
<td>Haskell ThreadId + MVar (GHC 7.6.3, 64-bit)</td>
</tr>
<tr>
<td>2.6 KB</td>
<td>Erlang process (64-bit)</td>
</tr>
<tr>
<td>8.0 KB</td>
<td>Go goroutine</td>
</tr>
<tr>
<td>9.0 KB</td>
<td>C pthread (minimum, 64-bit Mac OS X)</td>
</tr>
<tr>
<td>64.0 KB</td>
<td>Java thread stack (minimum)</td>
</tr>
<tr>
<td>513 KB</td>
<td>C pthread (default, 64-bit Mac OS X)</td>
</tr>
<tr>
<td>1024 KB</td>
<td>Java thread stack (default)</td>
</tr>
</tbody>
</table>

(courtesy of Bob Ippolito,
http://bob.ippoli.to/haskell-for-erlangers-2014/)
GHC RTS

https://ghc.haskell.org/trac/ghc/wiki/Commentary/Rts

- scheduler
- garbage collector
- I/O manager
Scheduler

• OS threads vs. Haskell Threads
  • Thread State Object + stack (+RTS -ki)

• Tasks (one per OS thread)
  • can hold a Capability
  • and hand over too

• Capabilities (one per CPU)
  • +RTS -N
Scheduler

- Run queue (one per Capability)
  - keeps Threads (`forkIO` or ...)
- Sparks
  - spark pool (one per Capability)
  - points to a thunk ⇒ can become a Thread
Sparks

• can be work-stolen by other Capabilities

• can *fizzle* if thunk already in WHNF

  • machine busy $\Rightarrow$ many fizzled sparks

• if thunks are never used, sparks will be collected (almost) immediately!
Heap

Everything is a closure!
Heap

- Roots
- **NOT** Threads (linked to by Run Queues)
- Run queue
- Spark pool
- Generations (+RTS -G)
- Nursery (+RTS -A)
Garbage collector

- Traverse from the root, copy, scrap the rest
- Oldest generations are collected least often
- Eager promotion: if pointed to by an old object
- Aging: don’t promote to quickly though
Garbage collector

- "Allocation wall"
- per-thread nurseries fitting into L2 cache
- but frequent collections will stop the world often
- running mutator and collector concurrently hurts cache
- Private heaps to every CPU!
Remember

• Play with heap and nursery sizes
  • Too big nursery: bad locality, less promotions
  • Too small nursery: unnecessary promotions
  • Too small starting heap: takes time to expand
• Distribute work evenly between sparks!
To read

• https://ghc.haskell.org/trac/ghc/wiki/Commentary/Rts

• http://www.haskell.org/ghc/docs/7.6.3/html/users_guide/runtime-control.html

• Runtime Support for Multicore Haskell

• Multicore Garbage Collection with Local Heaps

• Mio: A High-Performance Multicore IO Manager for GHC

• The Implementation of Functional Programming Languages
OS thread → Task → Capability

- run queue empty
- thread preempted

Capability → Spark pool

Spark pool → Run queue

- scheduler selects

Run queue → Haskell thread

- par
- converted

Haskell thread → Spark

Spark → Thunk

- par
- converted

Thunk → GC

- not referenced
- fizzled
- finished execution

GC → Haskell thread

- holds/releases
Questions?