Runtime partitions for scaling GHC programs

Ben Gamari

28 Aug 2020
The Problem
The Problem

GC synchronization leads to poorly scaling across many cores.

Server applications have competing pressures:

▶ Low-latency wants small nurseries $\rightarrow$ lots of synchronization

▶ Multi-core scaling demands large nurseries (minimize synchronization)
A Solution: Local heaps

Marlow 2011\(^1\):

- Give each core its own independent nursery
- Prohibit references between local nurseries, allowing nursery GCs to happen without synchronizations
- “Globalise” data which is needed by other cores
- Implicit globalisation: non-obvious heuristics, hard to reason about performance

\(^1\)S. Marlow & S. Peyton-Jones. “Multicore garbage collection with local heaps.” Proceedings of the 10th International Symposium on Memory Management (2011)
A Solution: Multiple processes

forkMap
:: (NFData a, NFData b) → Int
   (a → b)
   StaticPtr

Communicate via pipes
StaticPtrs allow sharing of code references between processes
Compact normal forms can be used for efficient(?) serialisation
Feature request: CNFs could be mmap’d between processes
Explicit heap partitions within a process.

Partition 1
- nursery block
- gen 0 block
- gen 1 block
- gen 1 block
- CNF block
- TSO

Partition 2
- nursery block
- gen 0 block
- gen 0 block
- gen 1 block
- TSO

PChan
A Solution: Partitioning (message-passing)

-- | Similar to a TChan, but allowing sharing of values across

```haskell
data PChan a

writeTChan :: PChan a -> a -> IO ()
readTChan :: PChan a -> IO a
```
A Solution: Partitioning

```
-- | A reference-counted "globalized" value
data PRef a

newPRef :: a -> IO (PRef a)

-- | Use a value held in a 'PRef'.
withPRef :: (a #-> b) -> PRef a -> b

-- Sadly, this is broken due to laziness.
```
“Arenas” via partitioning

*ephemeral* partitions:

▶ Run a thread in a partition
▶ After finished, tear down the world

Avoids GC entirely for sufficiently short-lived partitions.
Implementation

- Block descriptor: identify owner partition
- Capabilities can run threads from any partition
- `mut_list` must be flushed to global (non-capability-local) remembered set when a capability switches partitions
The challenge of CAFs

CAFs are a shared resource.

A few options:

▶ Don’t update CAFs (yuck!)
▶ Introduce a new “per-partition” indirection type
▶ Globalize all CAF evaluations?
Volunteers?

Who wants to implement this?