Islands RTS

GHC for a multikernel

Calum McCall
Marcin Orczyk
University of Glasgow

Islands RTS - cache coherence

- Haskell RTS for machines with *partial cache coherence*.
- Why only partial?
  - Cache coherence will not scale to 100s of cores.
  - Cache coherence does not apply in clusters.
  - But it is cheap and works well for smaller number of cores!
- Systems composed of a number of **islands**;
  - *island:* a group of cache-coherent cores.

Islands RTS - overview

- Based on GHC 7 and GUM:
  - really a modified *threaded* mode selected at build time.
- Memory organisation:
  - separate physical heap per island,
  - virtual shared heap built with message passing.
- Implements Glasgow parallel Haskell (GpH) model:
  - *par a b* -- returns *b* and marks *a* for parallel evaluation,
  - idle cores evaluate marked closures (called *sparks*) transparently to the programmer.

Islands RTS - between islands

- Inspired by and based on GUM.
- Key to virtual shared heap: *Fetch* closure
  - a remote reference to a closure on a different island,
  - evaluation triggers migration of the closure,
  - two migration policies available:
    - *move* (default for un-evaluated data)
    - *copy* (default for evaluated data)

Islands RTS - between islands

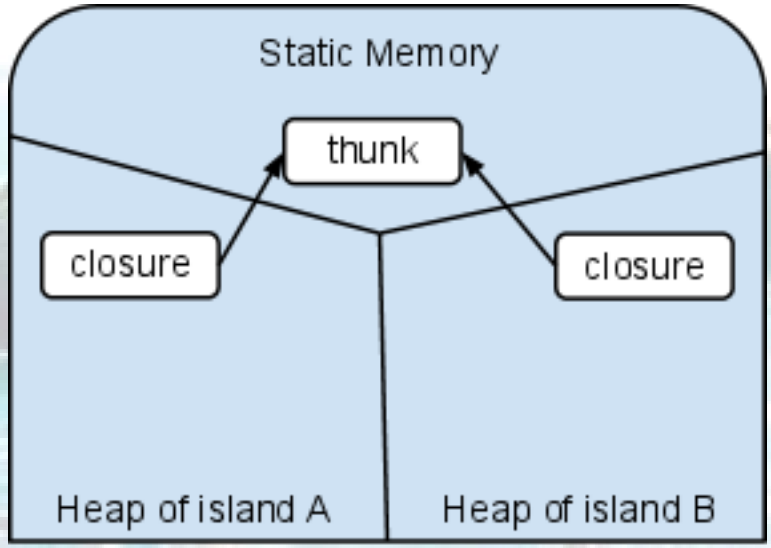
- Global addressing system:
  - inspired by GUM,
  - *hard links*: ref-counted, prevent GC, reusable,
  - *soft links*: volatile, cannot be fetched, non-reusable.
- Messaging protocol - five main message types:
  - FETCH, DATA, FREE, SPARK
  - work polling FISH message
- Packing/unpacking routines - GUM code ported to GHC 7.
- Message passing layer:
  - use different mechanisms at different levels,
  - e.g. shared memory when available.

Islands RTS - within island

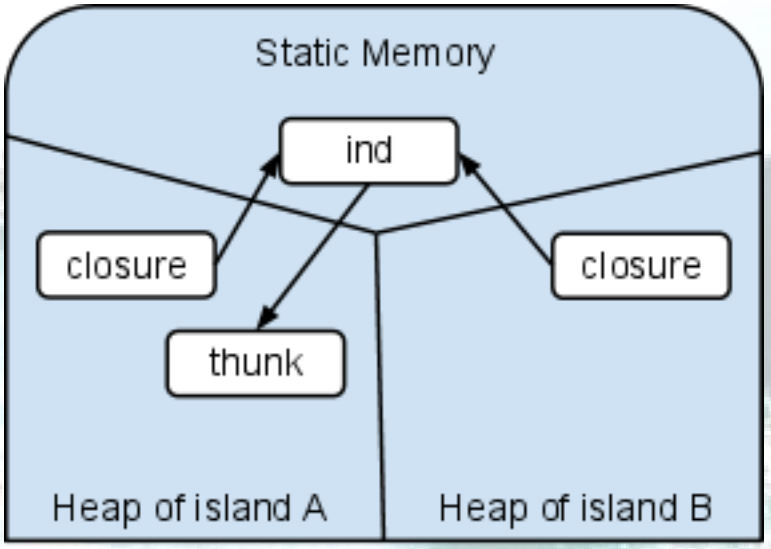
- Based on GHC 7:
  - essentially a "threaded" (SMP) GHC RTS per island,
  - message handling hooked into scheduler,
  - global addresses management hooked into GC.

- [Bad Idea] Multiple islands in the same process:
  - required wide changes to GHC RTS code
    - switch from global to per-island data structures
    - *giant* time sink
    - pulling patches from GHC HEAD - difficult
  - evil static thunks
    - thunks are un-evaluated closures
    - compiler allocates some closures in static memory

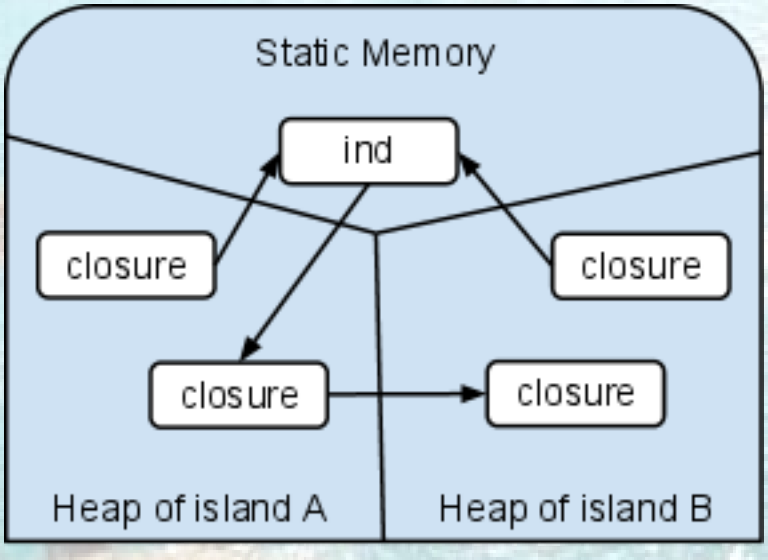
Islands RTS - evil static thunks



Initial state



A evaluates static thunk



B evaluates thunk in A's heap

Islands RTS - results

- Working proof-of-concept implementation on x86 NUMA machine:
  - provides parallel speed-ups (not great performance),
  - incomplete implementation of packing/unpacking,
  - supports only multiple-islands-per-process.
- Identified problems with multiple islands per process:
  - large cost in terms of modified LOC,
  - static thunks are a significant issue (evil).

Islands RTS - future

- A reimplementation on top of current GHC HEAD:
  - one island per process,
  - support islands on different machines,
  - pull patches regularly.
- Support other concurrency/parallelism models of GHC:
  - look for common primitives,
  - leverage virtual heap and messaging.
- Implement on Barrelfish.
- Heterogenous islands:
  - different CPU architectures,
  - GPUs, FPGAs...

Islands on Barrelfish

- Need to represent islands on Barrelfish, Dispatcher fits this well.
  - multi-threaded islands need cache-coherent memory
  - otherwise one island per core
- IDC for message passing between islands

GHC RTS on Barrelfish

- GHC already partially ported by Ross McIlroy, this includes the RTS.
- This needs to be updated to current GHC version
- Currently, RTS is multithreaded but needs more work.
- Need to adapt GHC build system to use Barrelfish build system


Questions?