Islands RTS: a Hybrid Haskell Runtime System for NUMA Machines

Marcin Orczyk

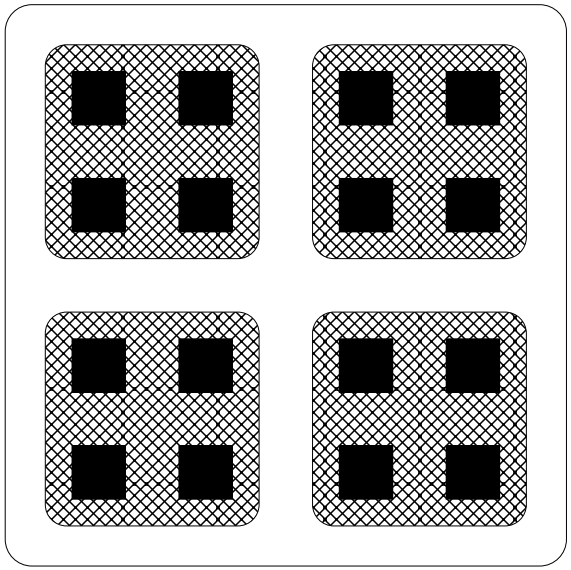
University of Glasgow

May 13, 2011

Paralellism

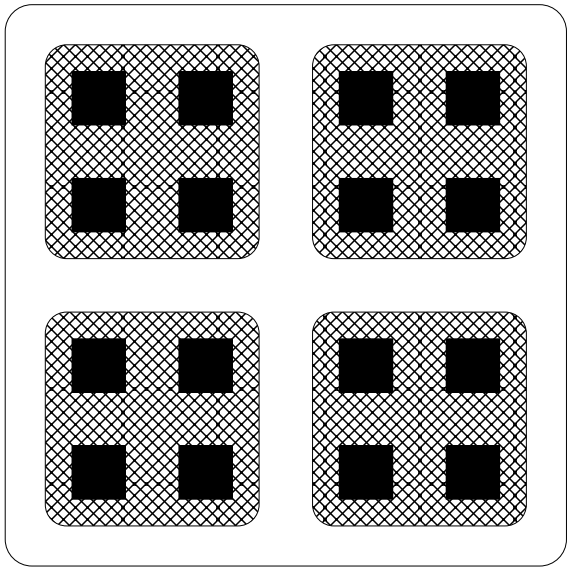
- CPU clock rates have plateaued
- the key to increased performance is *scalable* parallelism
- from multicore CPUs, through NUMA and massively parallel hardware (GPUs, FPGAs), to clusters and cloud computing

NUMA



16-core machine composed of 4 quad-core chips

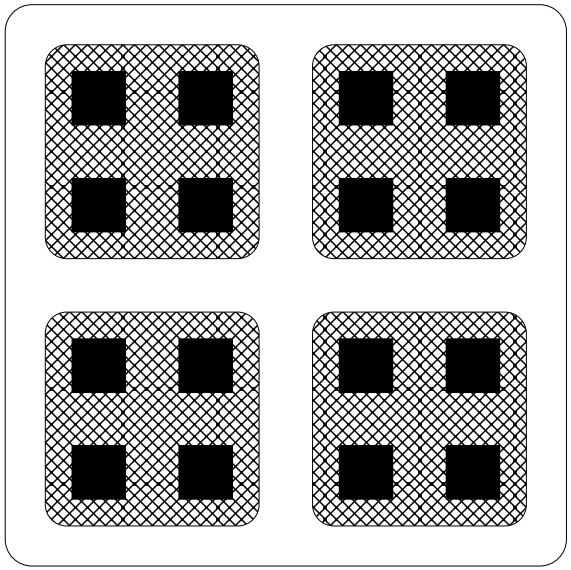
NUMA



16-core machine composed of 4 quad-core chips

- cache coherence is an overhead
- how high will it be at 20 cores? 100 cores? 500 cores?

NUMA



16-core machine composed of 4 quad-core chips

- cache coherence is an overhead
- how high will it be at 20 cores? 100 cores? 500 cores?

- non-coherent or partially cache coherent architectures

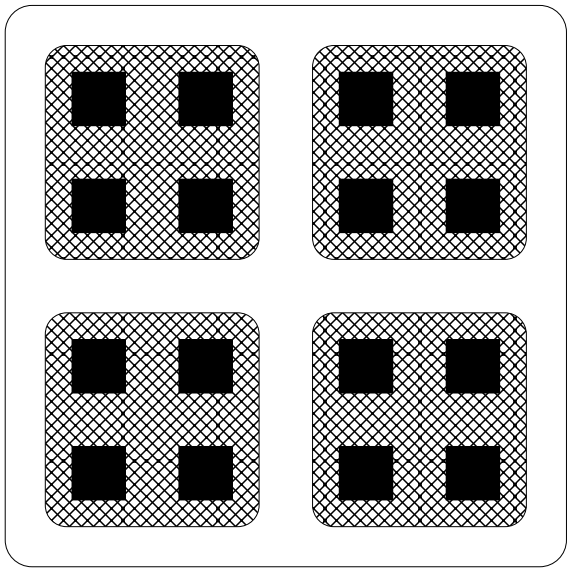
Islands RTS

- Haskell runtime system for partially cache coherent machines
- take advantage of shared memory/coherent cache when available

Islands RTS

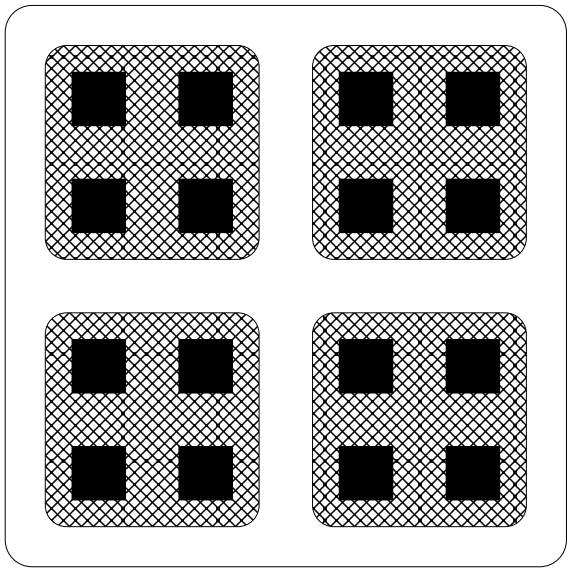
- Haskell runtime system for partially cache coherent machines
- take advantage of shared memory/coherent cache when available
- Islands RTS is a blend of two existing parallel Haskell implementations:
  - GHC: shared memory runtime system with support for parallel evaluation
  - GUM: distributed runtime system based on message passing

Island



- island: a set of processing cores sharing a cache coherent memory
- system is composed of a number of islands

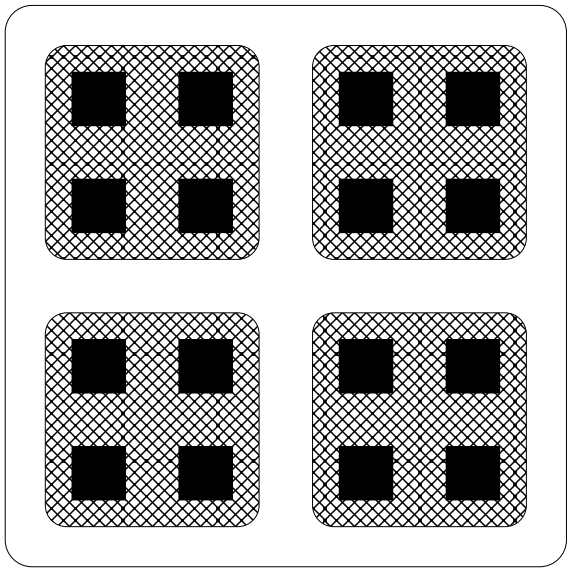
Island



system with 4 cache-coherent islands, each with 4 processing elements

- island: a set of processing cores sharing a cache coherent memory
- system is composed of a number of islands

Island



system with 4 cache-coherent islands, each with 4 processing elements

- island: a set of processing cores sharing a cache coherent memory
- system is composed of a number of islands

- one real heap per island
- virtual shared heap between islands

Virtual Shared Heap

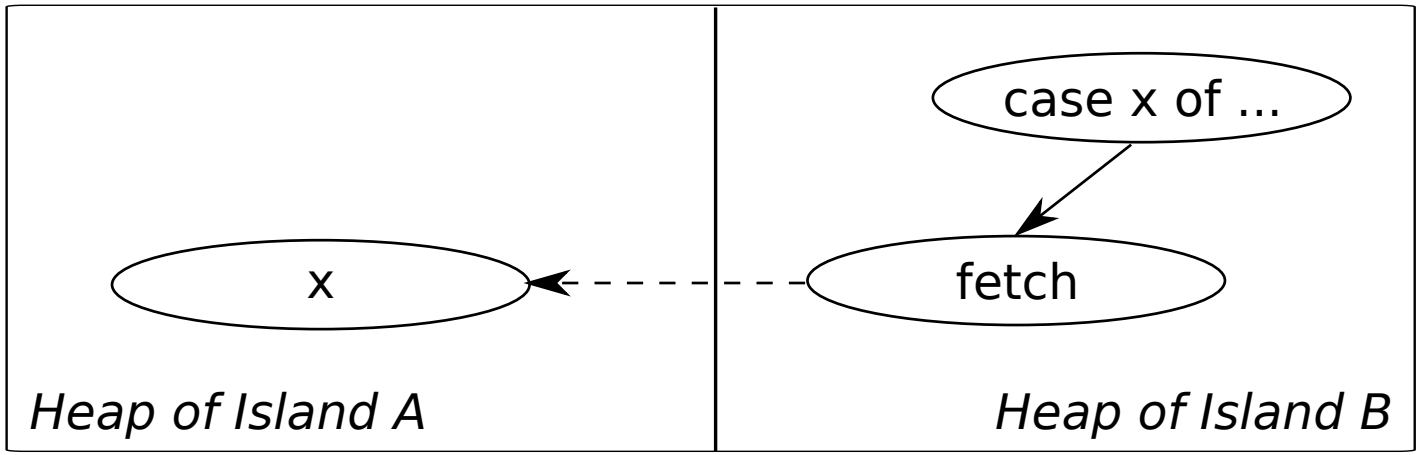
- following GUM, Islands RTS provides virtual shared heap
- closures can *transparently* reside on different islands

Virtual Shared Heap

- following GUM, Islands RTS provides virtual shared heap
- closures can *transparently* reside on different islands
- consider evaluation of the expression `(case x of ...)`, where `x` exists on a different island

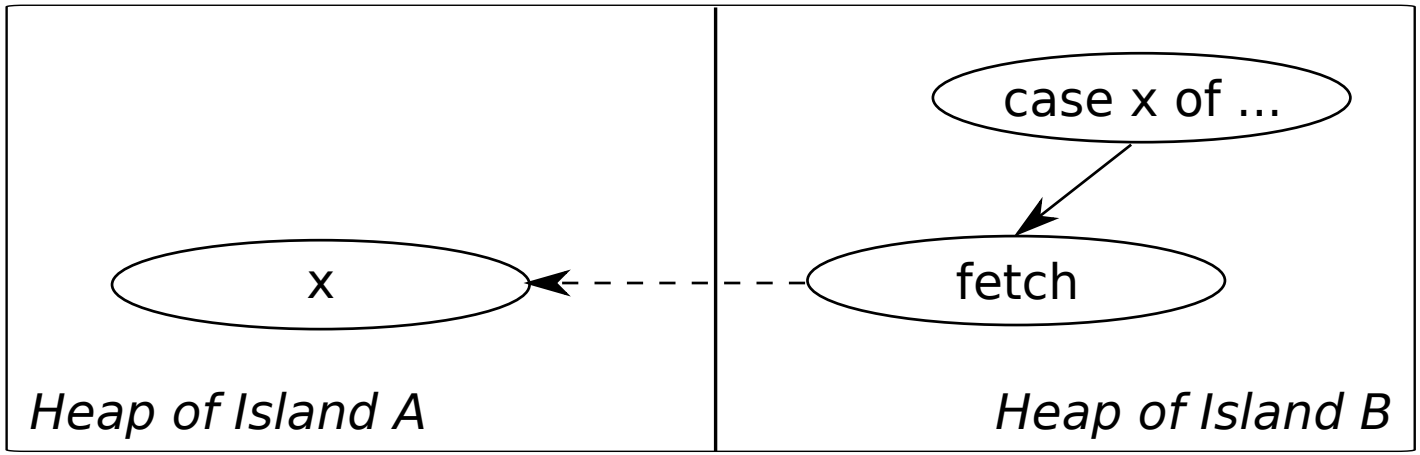
Virtual Shared Heap

- following GUM, Islands RTS provides virtual shared heap
- closures can *transparently* reside on different islands
- consider evaluation of the expression `(case x of ...)`, where `x` exists on a different island



Virtual Shared Heap

- following GUM, Islands RTS provides virtual shared heap
- closures can *transparently* reside on different islands
- consider evaluation of the expression `(case x of ...)`, where `x` exists on a different island



- packing/unpacking closures
- global addresses
- message passing layer and protocol
- stub closures

Global Addresses

- triples

  (, ,)

Global Addresses

- triples

  (island, ,)

Global Addresses

- triples

  (island, slot,)

Global Addresses

- triples

  (island, slot, weight)

- weighted reference counting

Global Addresses

- triples

  (island, slot, weight)

- weighted reference counting
- prevent garbage collection
- slots are reused
- similar to stable pointers

Global Addresses

- triples

  (island, slot, weight)

- weighted reference counting
- prevent garbage collection
- slots are reused
- similar to stable pointers

- Islands RTS calls them “hard links”
- they are quite heavyweight and the guarantees they provide are often unnecessary
  - e.g. recognising duplicates, speculative evaluation

Global Addresses

Soft Links

- analogous to weak pointers
- triples

  (island, slot, slot2)

- do not prevent garbage collection
- slots never reused

Global Addresses

Soft Links

- analogous to weak pointers
- triples

  (island, slot, slot2)

- do not prevent garbage collection
- slots never reused

- GUM used only hard links
- reasons to distinguish between hard and soft links:
  - clarifies implementation
  - potentially improves performance

Message Passing

Meassages

- virtual shared heap
  - FETCH(ga-from, ga-to)
  - UPDATE(ga, data)
  - FREE(ga)
- work distribution
  - FISH
  - SPARK(data)
- startup, shutdown messages

Within Island

- based on GHC 7
- multiple islands in the same process
- changes:
  - eliminating global data structures
  - scheduler loop: hooks for message handling
  - garbage collector: hooks for global addresses
  - new closures

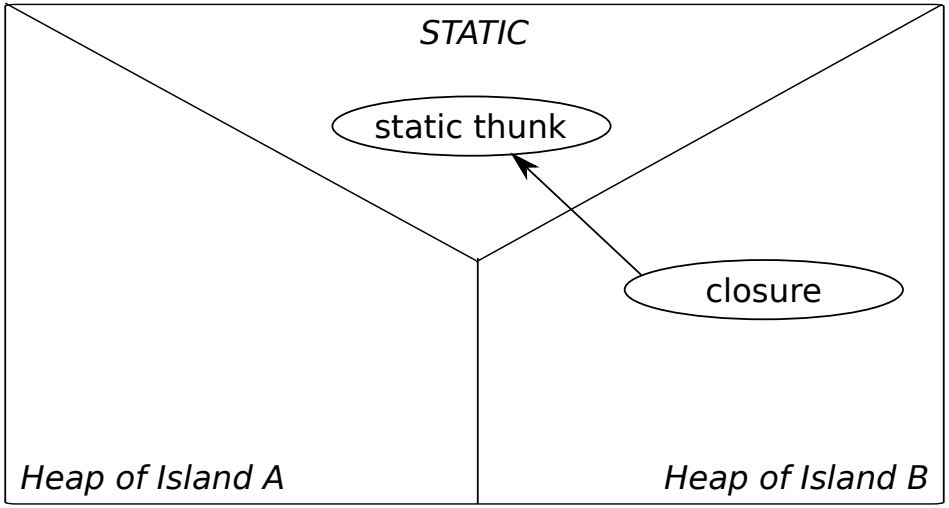
Static Thunks

- compiler allocates certain closures statically
- some of them are thunks

Static Thunks

- compiler allocates certain closures statically
- some of them are thunks

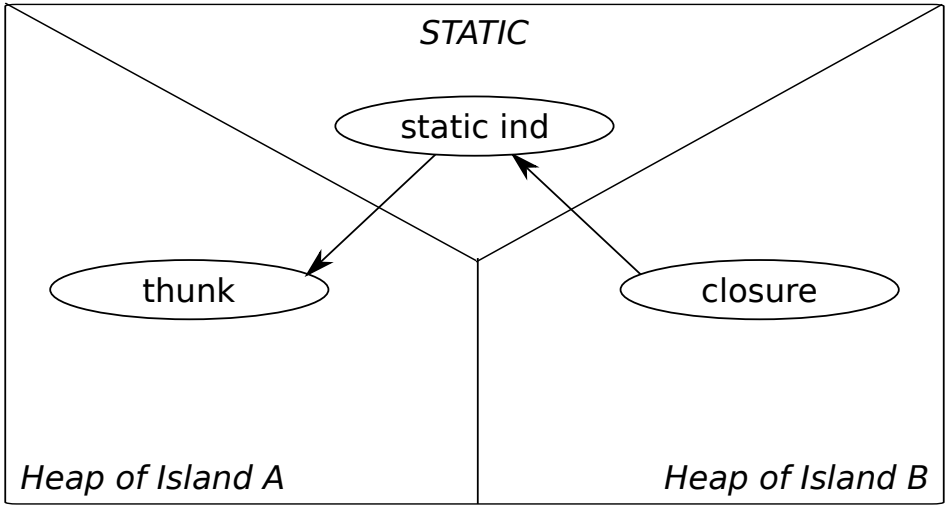
- closure on island B refers to a static thunk



Static Thunks

- compiler allocates certain closures statically
- some of them are thunks

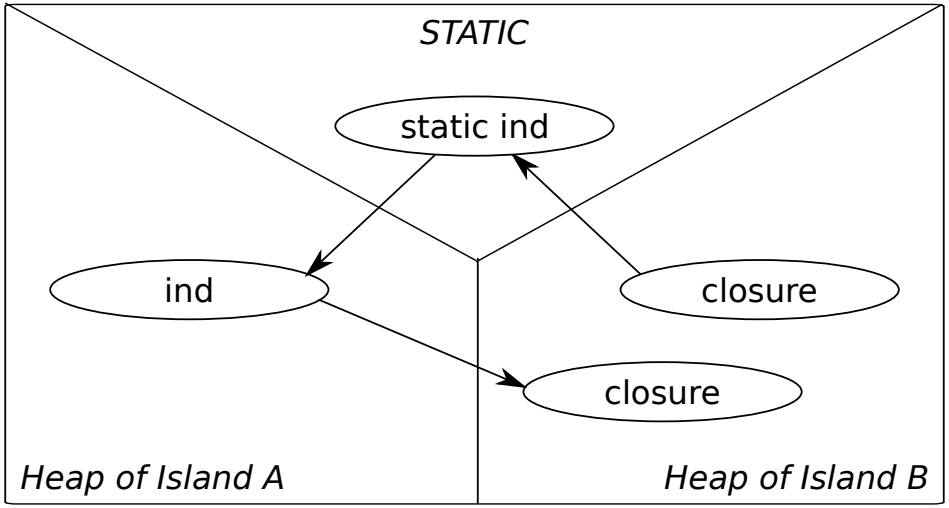
- closure on island B refers to a static thunk
- island A evaluates the static thunk



Static Thunks

- compiler allocates certain closures statically
- some of them are thunks

- closure on island B refers to a static thunk
- island A evaluates the static thunk
- island B evaluates thunk in A's heap

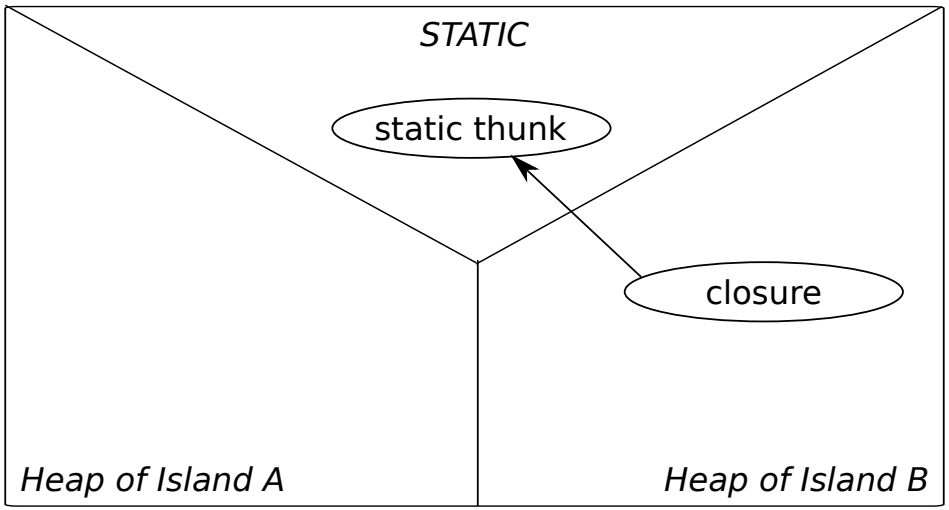


Static Thunks - Solution

- a layer of indirection

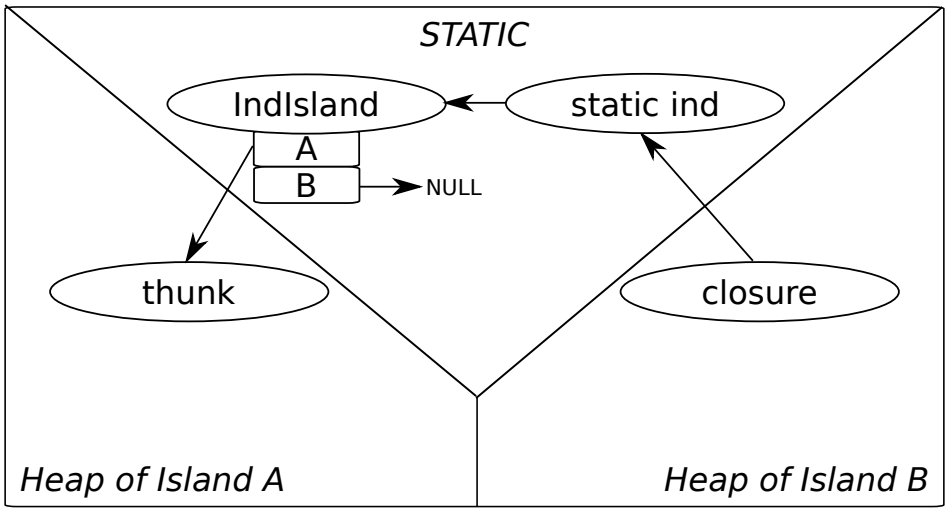
Static Thunks - Solution

- a layer of indirection
- closure on island B refers to a static thunk



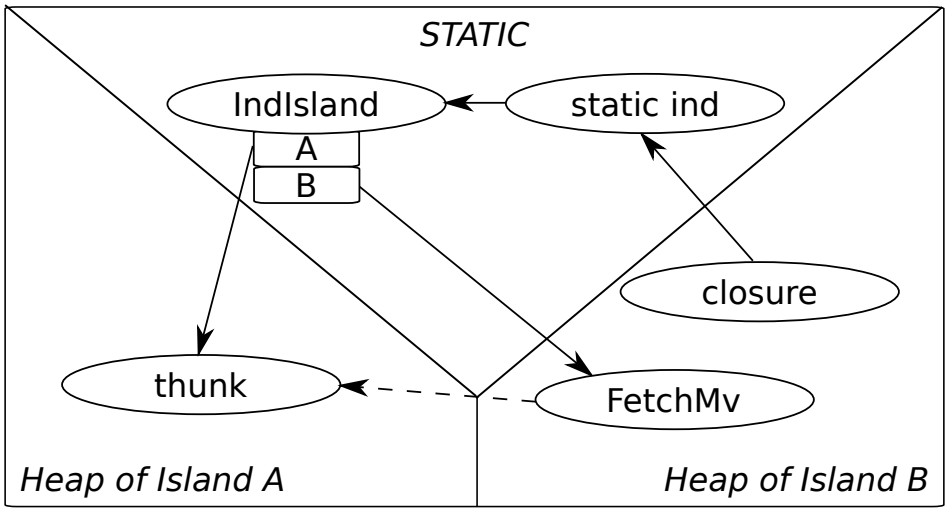
Static Thunks - Solution

- a layer of indirection
- closure on island B refers to a static thunk
- island A evaluates the static thunk



Static Thunks - Solution

- a layer of indirection
- closure on island B refers to a static thunk
- island A evaluates the static thunk
- island B accesses the static thunk



Static Thunks - Solution

- overhead on evaluation and access
- memory overhead proportional to the number of local islands
- additional, nontrivial complexity

- suggestions for solving this problem are welcomed

Summary

- parallel hardware becomes hierarchical
- Islands RTS matches it with the hierarchical architecture of the runtime itself
  - within a cache coherent island - shared memory graph reduction
  - between the islands - virtual heap based on message passing
- enables exploiting most appropriate mechanisms at each level

- future directions
  - non-coherent shared memory
  - port to Barrelfish
  - remote islands
  - heterogenous islands

Questions?

Questions?

Uniqueness of Soft Links

we can have tons!