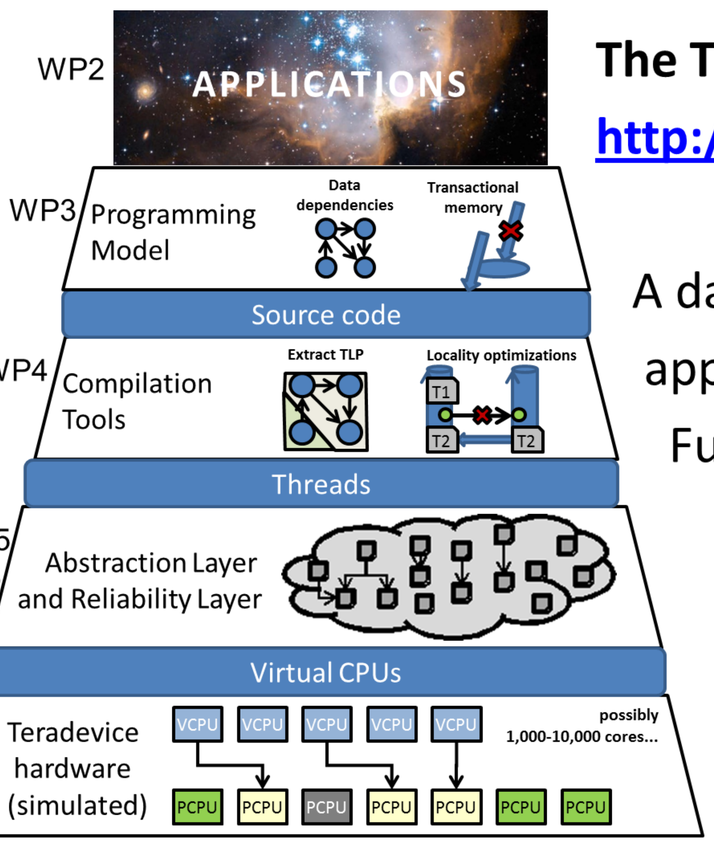
TERAFLUX: Exploiting Tera-device computing Challenges and possibilities

Antoni Portero Rania Mameesh Zhibin Yu Roberto Giorgi

Dipartimento di Ingegneria dell'Informazione
Via Roma, 56 53100 SIENA - Italy

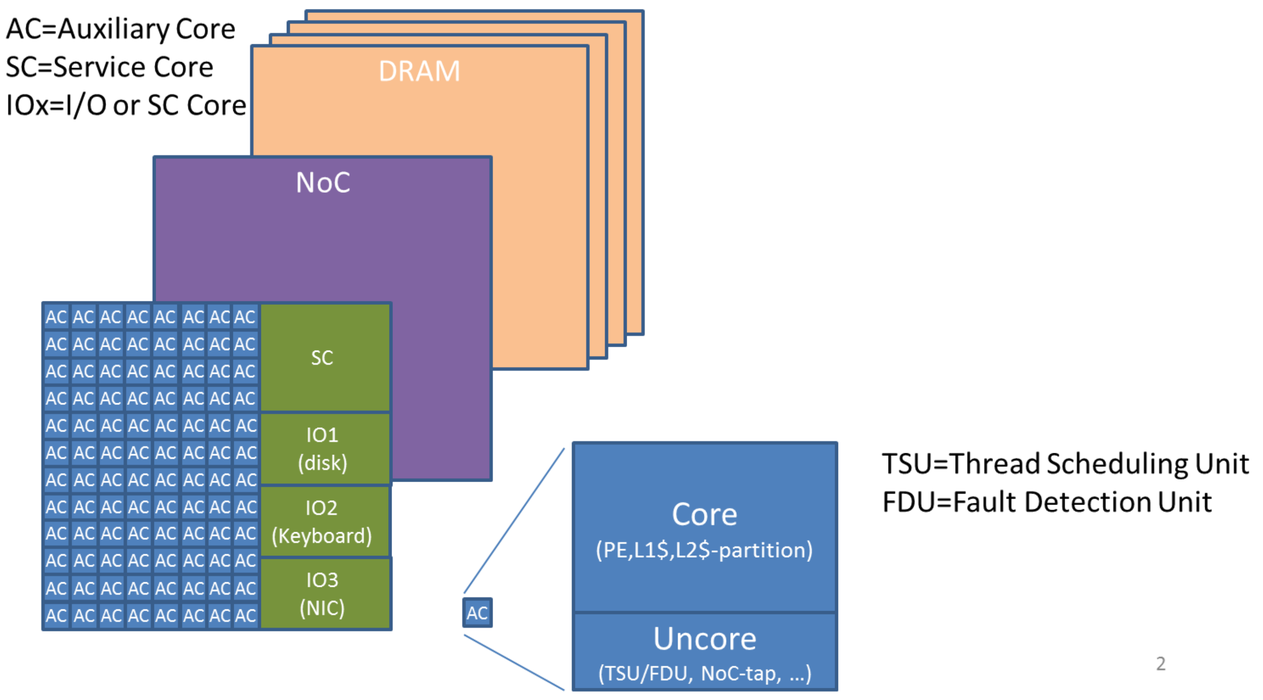


The TERAFLUX Project

http://teraflux.eu

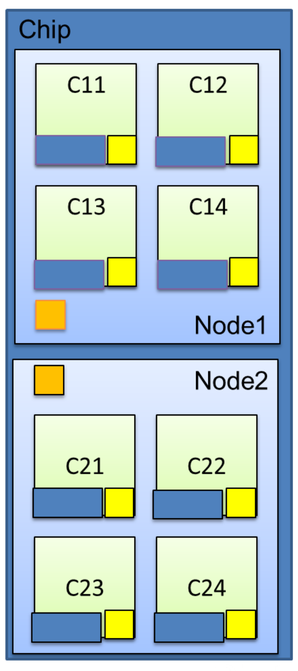
A dataflow approach to Future Many-Core

A possible TERAFLUX architectural instance

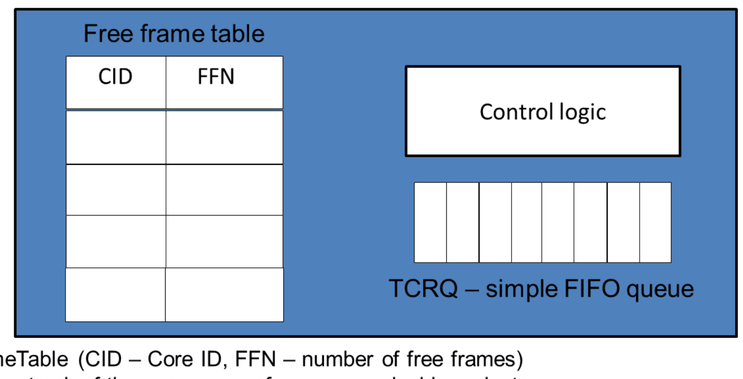


TSU=Thread Scheduling Unit
FDU=Fault Detection Unit

Distributed TSU: Elements

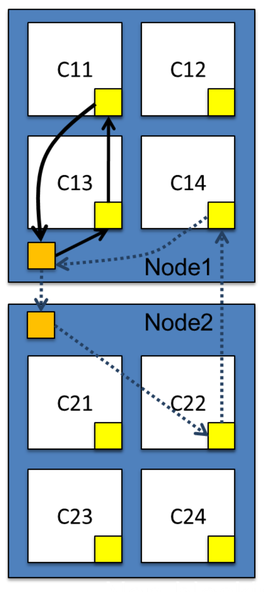


- Core (Cjk)
  - Off-the-shelf cores (may include L1, L2slice)
  - Minimal ISA extension
- Core Memory (Uncore)
  - Holds FM, OWM, TLS, CM per-thread pages
  - "Close" to the core (closeness set by NoC)
- Local Scheduling Element (LSE)
  - Manages threads on this Core
- Distributed Scheduling Element (DSE)
  - Distributes threads among Nodes
- Node
  - Groups Cores+Resources



- FreeFrameTable (CID – Core ID, FFN – number of free frames)
  - Keeps track of the occupancy of processors inside a cluster
  - Updated on each TDestroy and accepted TCreate
- TCRQ – ThreadCreateRequestQueue
  - Holds unserved ThreadCreateRequest messages
  - Message is pushed into queue when there are no free local resources
  - Message is popped from the queue when either TDestroyor BroadcastResponse arrives
- Control logic
  - Responsible for both inter and intra node communication and updating the messages inside a scheduler

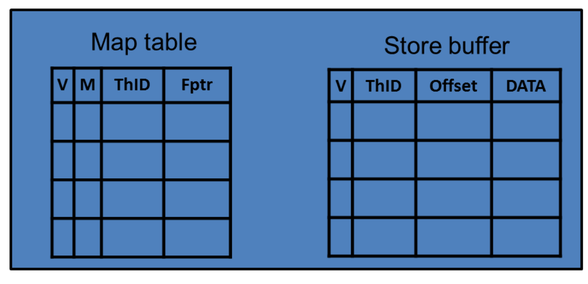
Scheduling Example



- LSE11 → DSE1:
  - I need a new frame
- DSE1 → LSE13:
  - You're available, give one frame to LSE11
- LSE13 → LSE11:
  - Here is a frame you can use

Fast communication → low overhead
What if every PE in the cluster is busy?

- LSE14 → DSE1:
  - "I need a new frame"
- DSE1 → DSE2:
  - LSE14 needs a frame, I don't have it
- DSE2 → LSE22:
  - Give a frame to LSE14
- LSE22 → LSE14:
  - Here is a frame you can use

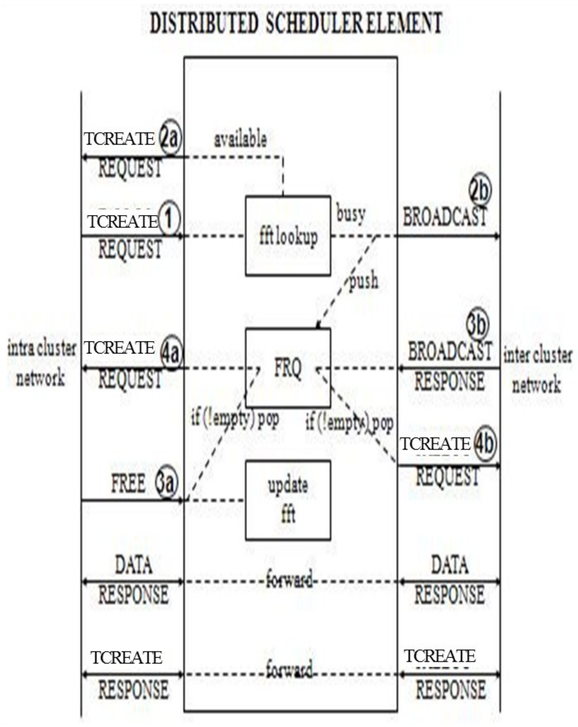


- Map table (V – Valid, M – Mapped, ThID – thread ID, Fptr – Frame pointer)
  - Keeps track of the issued resource requests for the execution of new threads
  - a ThID is assigned to a thread when new TCreate instruction occurs; it is used just inside that core
  - a Fptr is assigned when a TCreateResponse message arrives; it is unique globally in the system
  - Can be cleared on the thread completion

- Store buffer (V – valid, ThID – thread ID, offset – for storing in a frame, DATA – data to store)
  - Keeps track of the TStores issued for the threads that didn't receive a TCreateResponse yet (those kept in Map table and still not mapped)
  - On each TStore for the new thread that still doesn't have resources assigned, a new entry is created
  - When TCreateResponse arrives, all entries are checked and TStore messages are sent (entry invalidated) if there is any matching
  - If TStore occurs for a thread that already has its resources assigned, there is no need to use the buffer

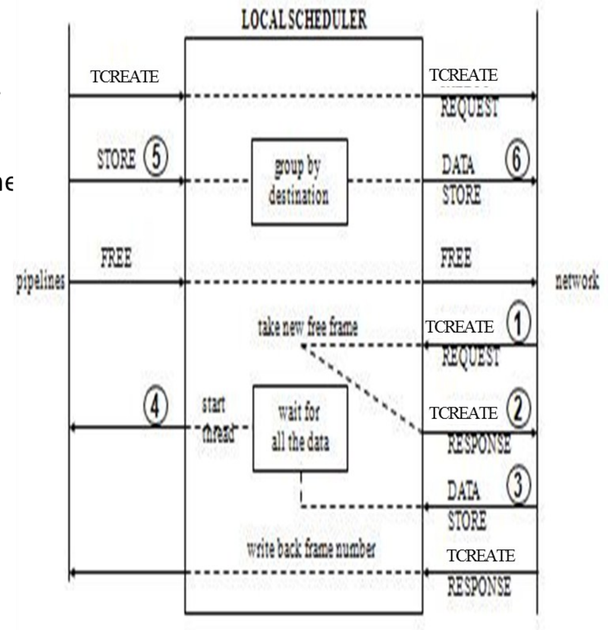
Distributed Scheduler Element (D-TSU)

- On new TCreateRequestMessage checks the availability in local node
  - If yes – forwards it
  - If no – put the message in FRQ and send broadcast
- Message is removed from FRQ when FfreeMessage or BroadcastResponse arrive
- Other messages are just forwarded



Local Scheduler Element (L-TSU)

- On TCreateRequestMessage
  - Choose a free frame for execution of the new thread
  - Send TCreateResponseMessage to the issuing processor
- On TCreateResponseMessage simply update the continuation with the frame identifier
- On store send DataStore message (group them if the destination is the same)



Giorgi, R., Popovic, Z., Puzovic, N. , **DTA-C: A Decoupled multi-Threaded Architecture for CMP System. Computer Architecture and High Performance Computing**, 2007. SBAC-PAD 2007. 19th International Symposium on, vol., no., pp.263-270, 24-27 Oct. 2007.