Which Problems Does a **Multi-Language Virtual Machine** Need to Solve in the **Multicore/Manycore Era?**

Stefan Marr
Mattias De Wael, Michael Haupt, Theo D'Hondt
Software Languages Lab
Vrije Universiteit Brussel

VMIL Workshop 2011
2011-10-24

VMs today

Performance

Memory Management

IronPython Boo F# Groovy Scala X10

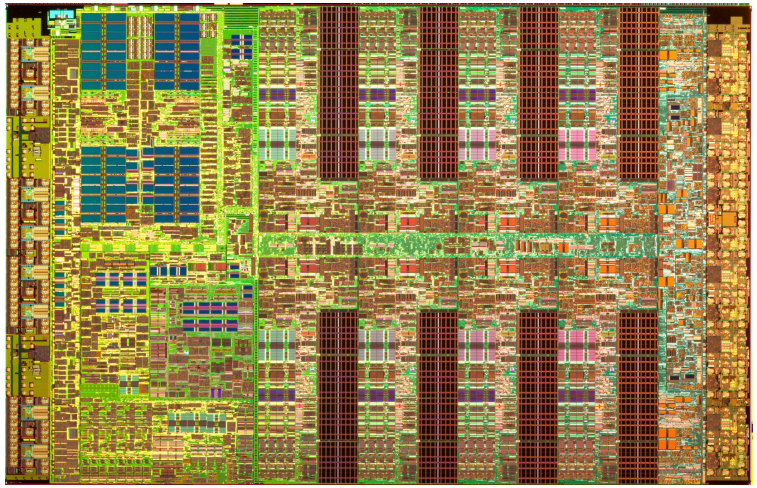
Multi-Language Eco System

Platform Independent

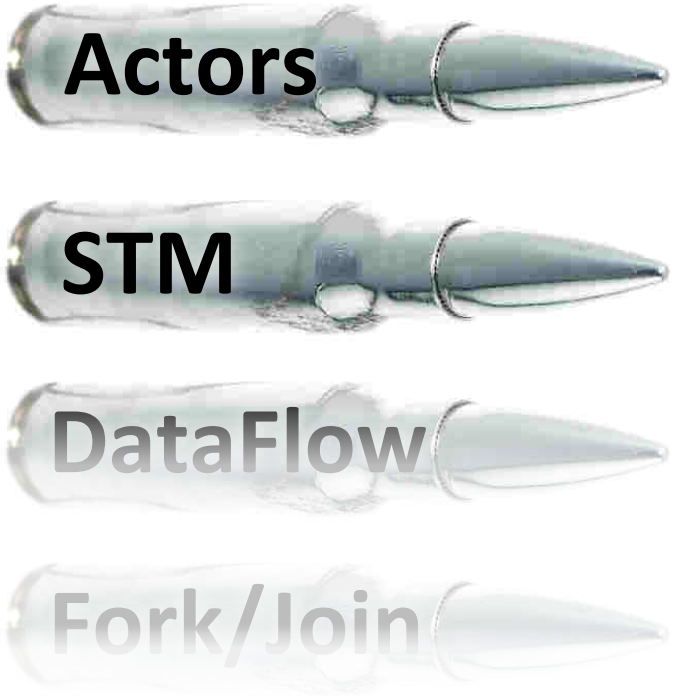
What do we need in the Multicore/Manycore Era?

Two Dimensions

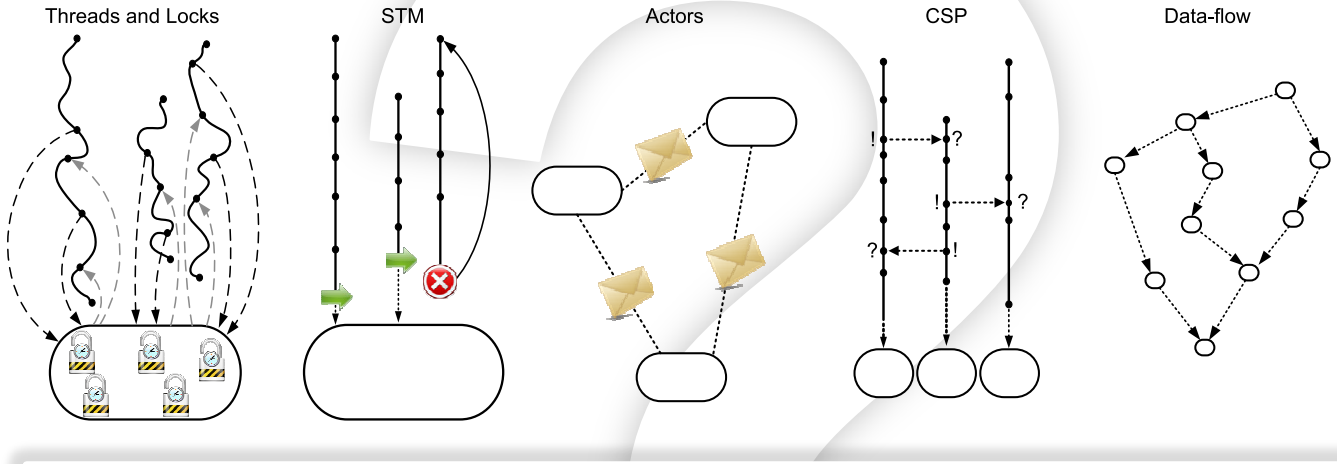
Processor Design



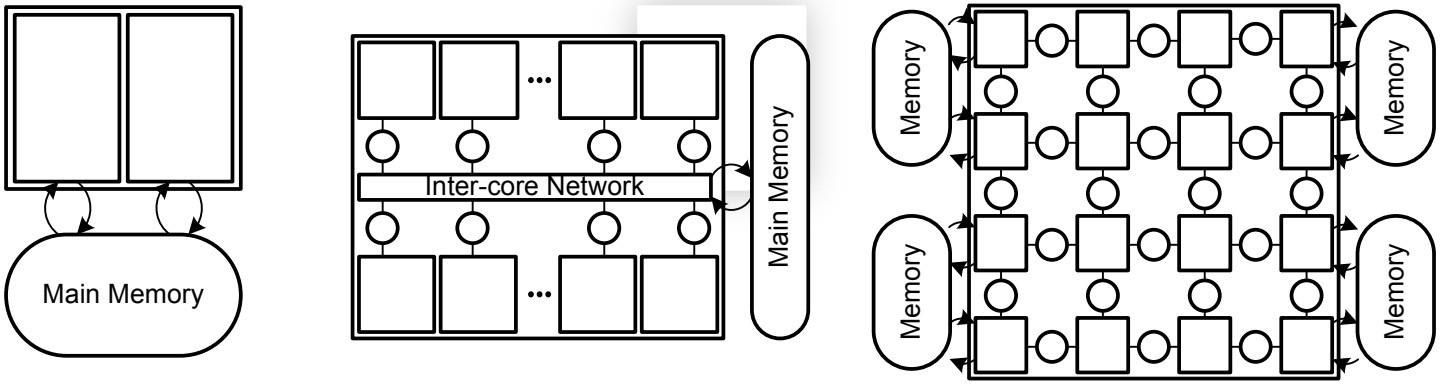
Concurrency Models

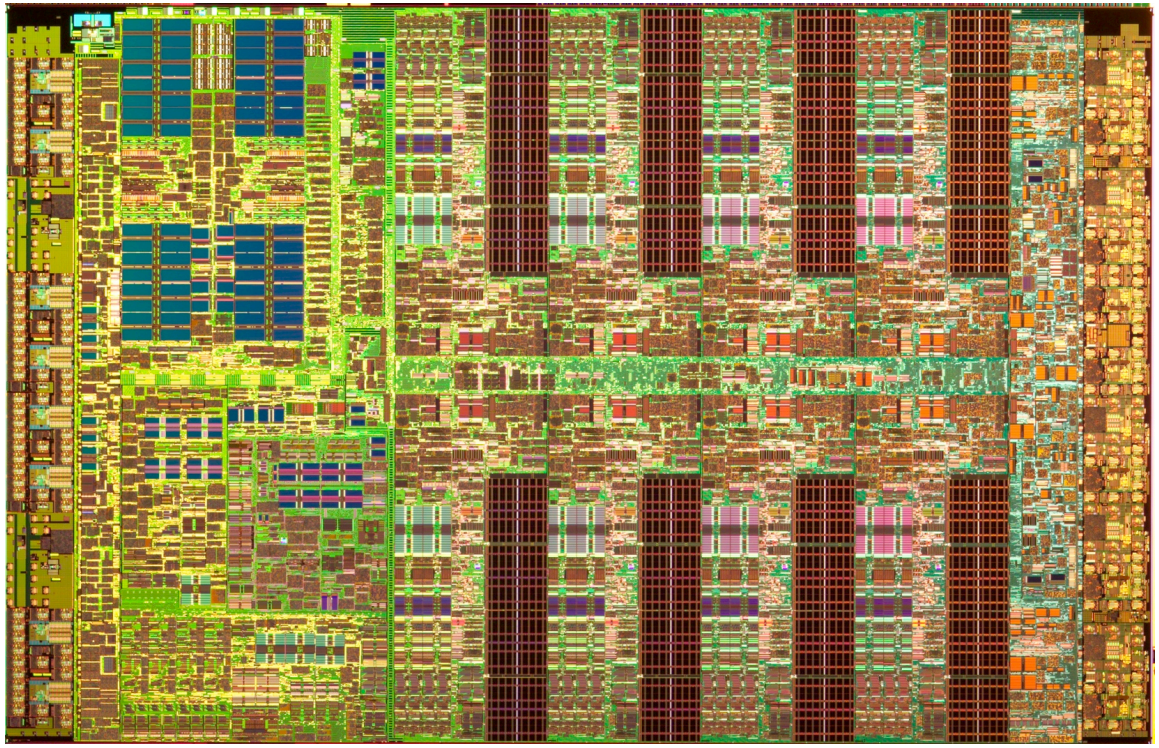


Processors and Concurrency Models



VM?





WHAT ARE THE RELEVANT CHANGES AT THE HARDWARE-LEVEL?

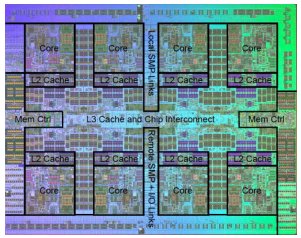
Brawny vs. Wimpy

- High sequential performance
- Few in numbers

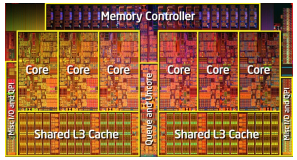
- Weak sequential performance
- High in numbers

And everything in-between!

Brawny

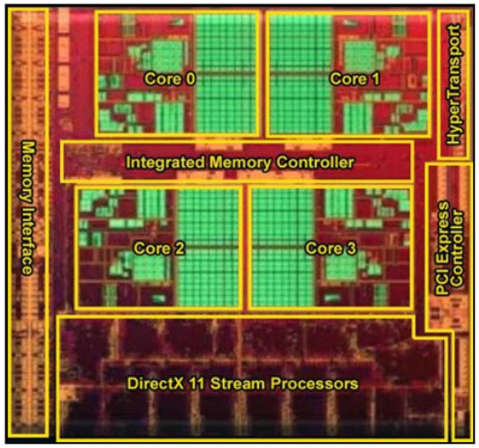


IBM Power7

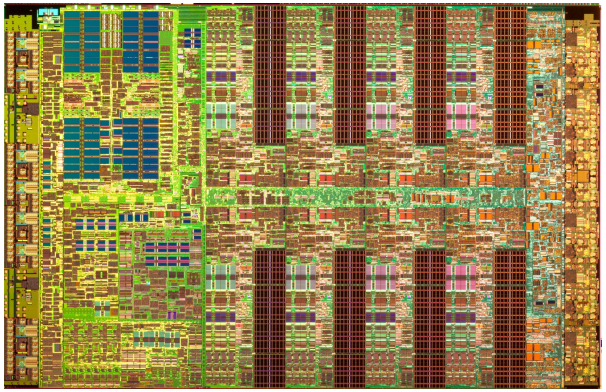


Intel i7

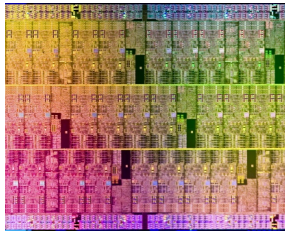
Hybrid



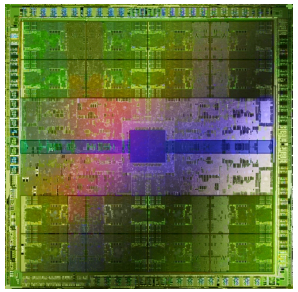
AMD Fusion



Cell B.E.

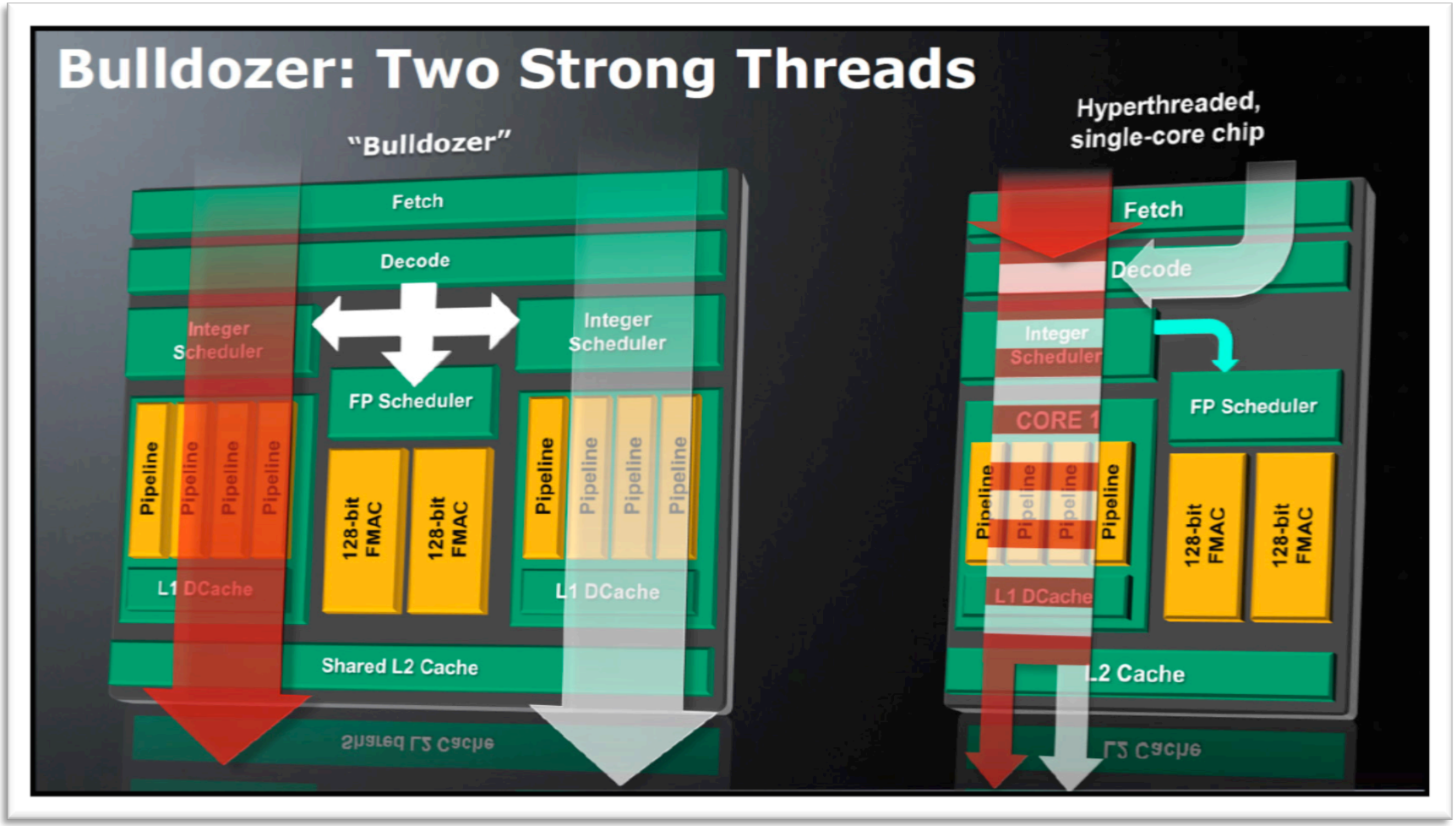


Intel MIC



Nvidia Fermi

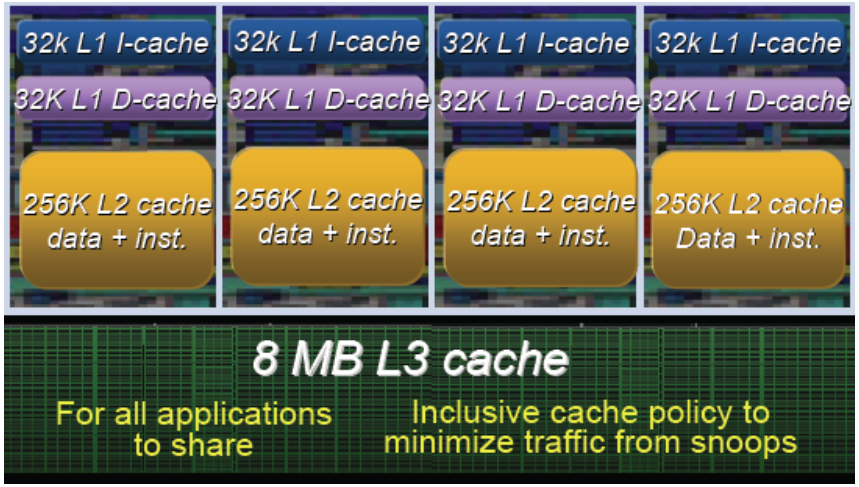
Cores vs. Threads vs. Shared PUs



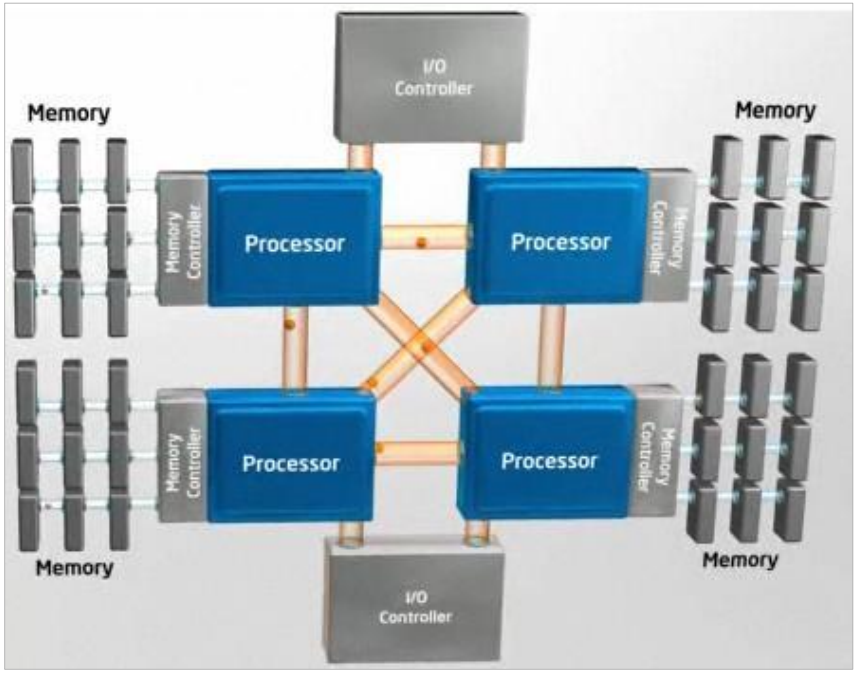
Source: AMD

Memory and Communication

Nehalem Cache Hierarchy



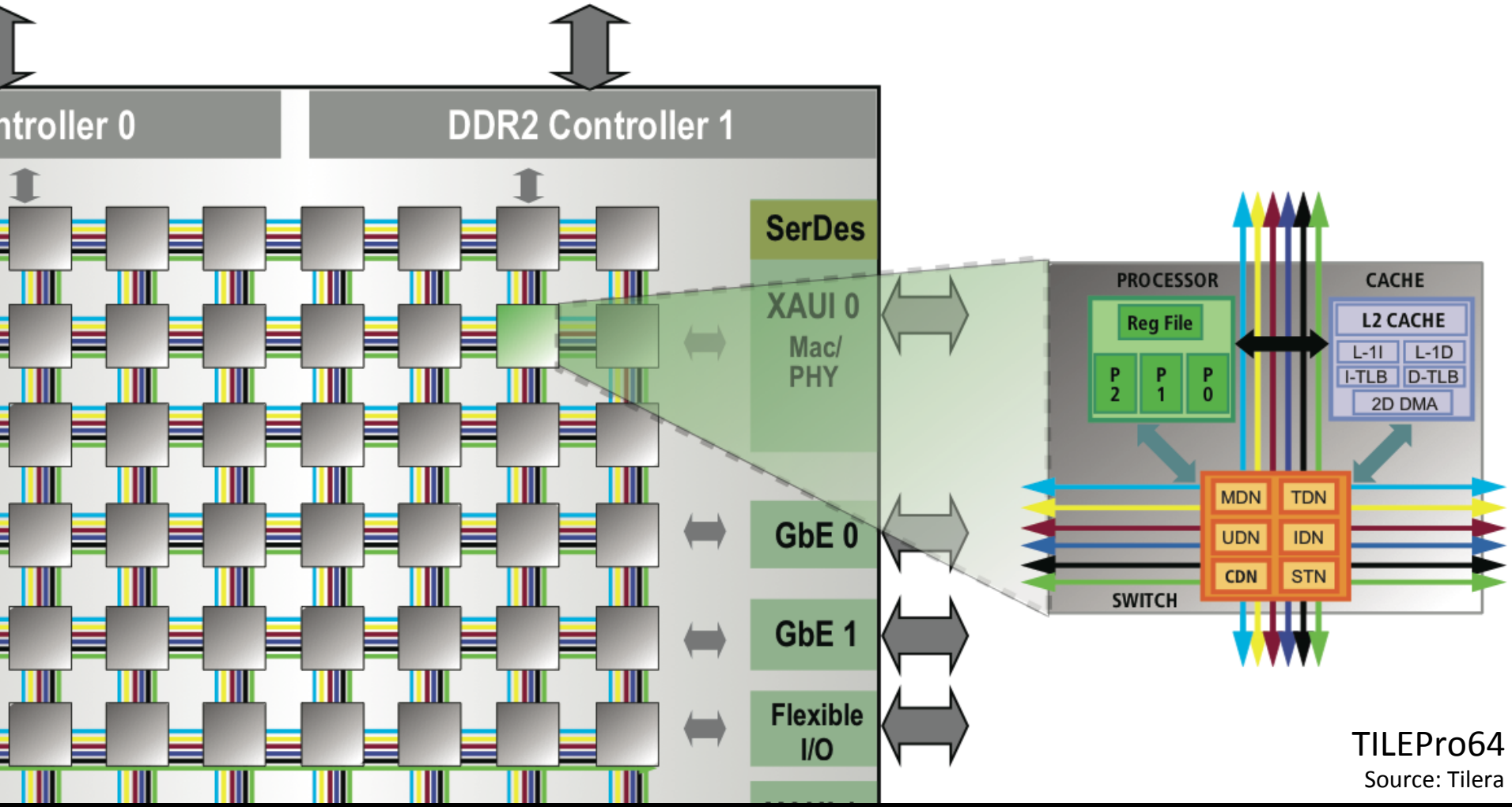
Four Processor Setup



Communication implicit via memory

Source: Intel

Memory and Communication

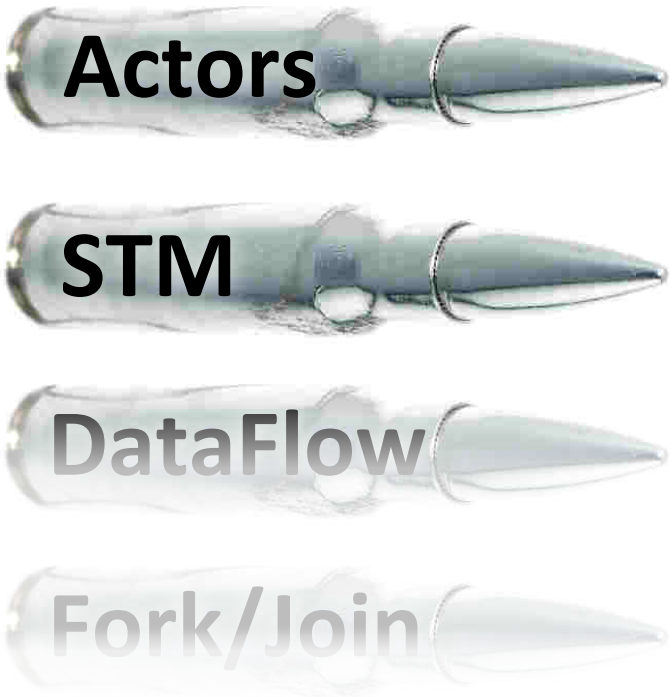


TILEPro64

Source: Tilera

What do VMs need to account for?

- Appropriate utilization
  - Varying performance of cores/units?
  - Caches and *simultaneous multithreading*?
- Non-uniform memory and communication
  - Cache hierarchies?
  - Explicit inter-core communication?



WHAT ARE TYPICAL IMPLEMENTATION PROBLEMS OF CONCURRENCY MODELS?

Encapsulation

```
actor BreakEncapsulation:
  def main_loop(self):
    receive:
      (Message) =>
        Message.msg = "broken"
# main
msg = Message("Hello World!")
broken ! (msg)
# wait
print msg.msg
```

IMPORTANT: Messages can be any kind of object but have to be immutable. Scala can't enforce immutability (yet) so this has to be by convention.

JCSP: obj-by-ref, no warning

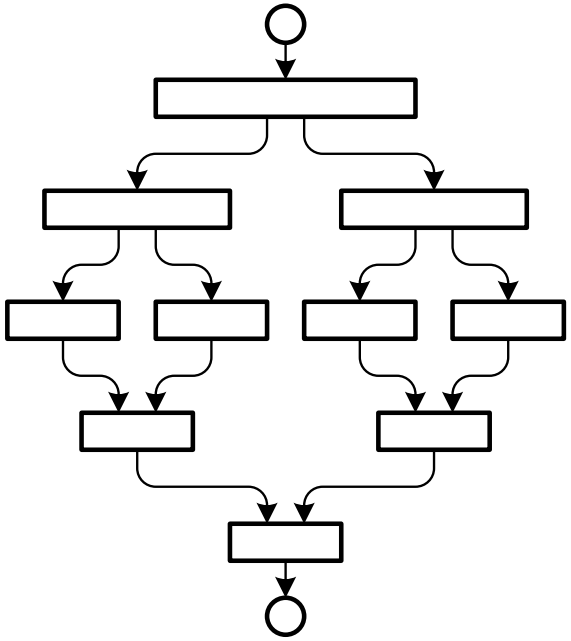
Kilim: zero-copy type system not "reflection-proof"

Clojure: STM + pure Java Agents + pure Java

Swing UI: "The single-thread rule"

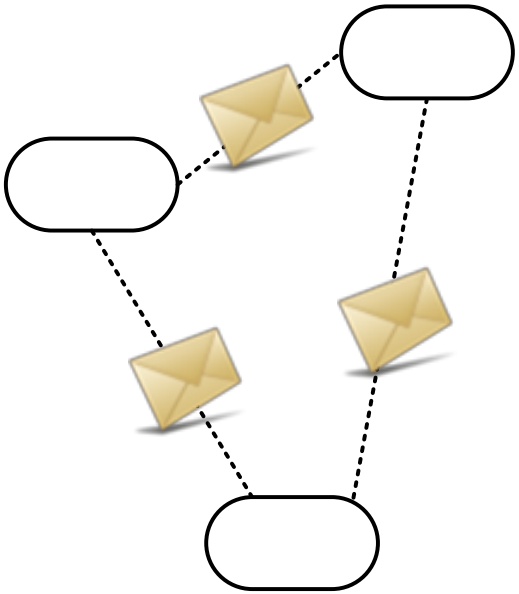
Scheduling Guarantees

Fork/Join



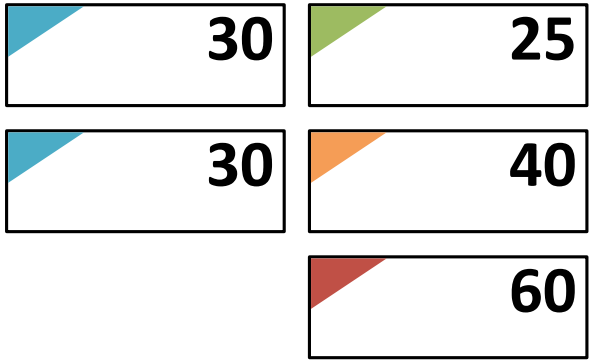
work-stealing

Actors



fair scheduling

Prioritized Tasks

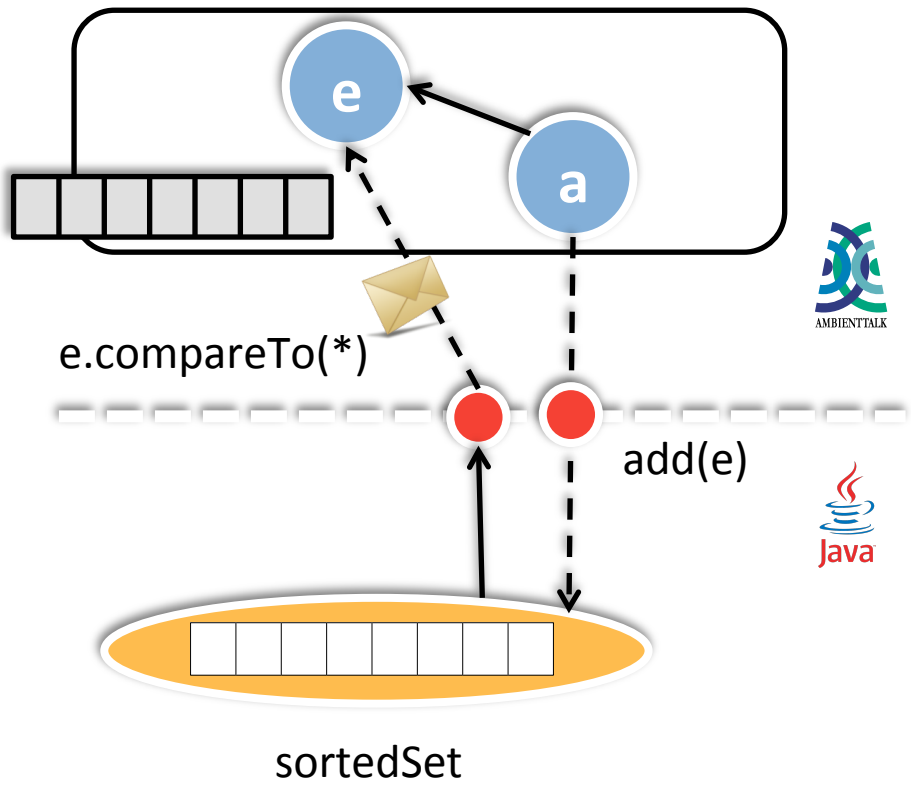


priority-based scheduling

Immutability

Source: Star Wars Episode V: The Empire Strikes Back.

Crossing Borders and Language-Levels




```
jThread.sortedSet =
        FarRefWrapper(sortSet)

jThread.sortedSet
        .add(FarRefWrapper(e))



// add(fr_e) (wrapper semantics)
fr_e.compareTo(obj) {
  return send_wait(e,
                   compareTo,
                   obj)
}
```

Language/Application-Level

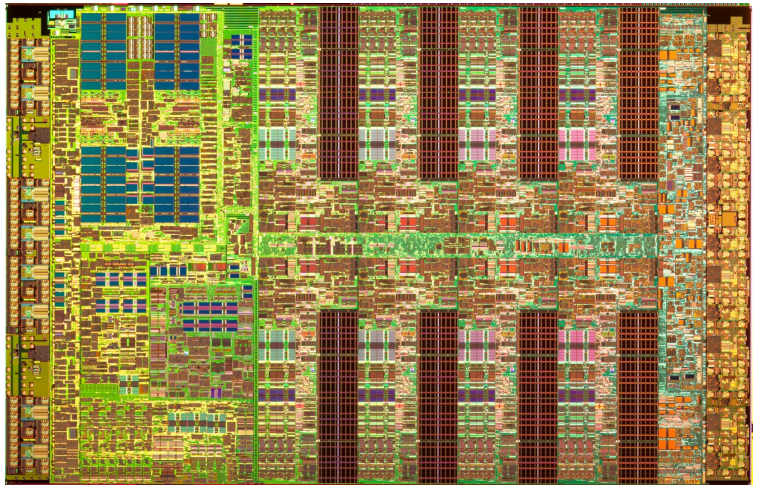
Implementation/Meta-Level

What do VMs need to account for?

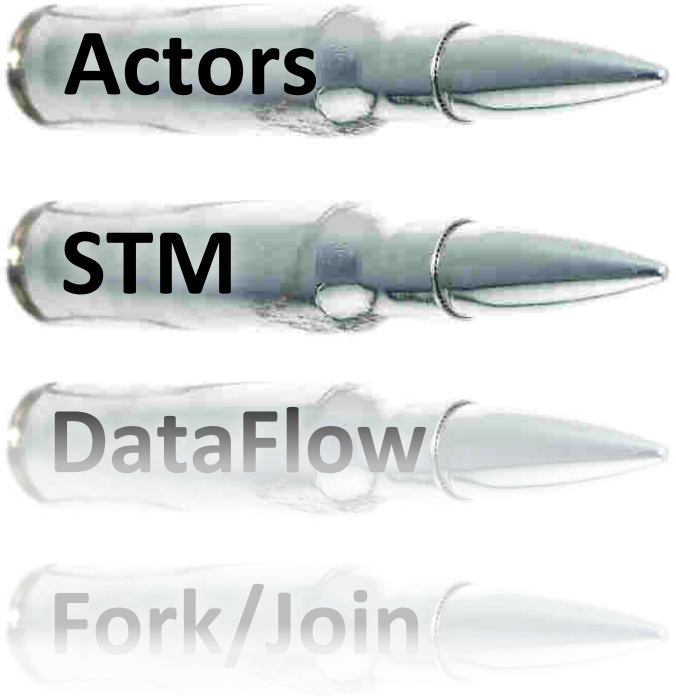
- Enforcing language semantics?
- Flexible concurrent encapsulation?
  - What is the relation to immutability?
- Enforceable scheduling guarantees?
- Normal vs. reflective operations?

What do VMs need to account for?

Processor Design



Concurrency Models



What do VMs need to account for?

Processor Design

- Appropriate utilization
  - Varying performance?
  - Hyperthreads?
- Non-uniform memory and communication
  - Cache hierarchies?
  - Explicit inter-core communication?

Concurrency Models

- Flexible concurrent encapsulation?
  - Flexible Immutability?
- Enforceable scheduling guarantees?
- Normal vs. reflective operations?