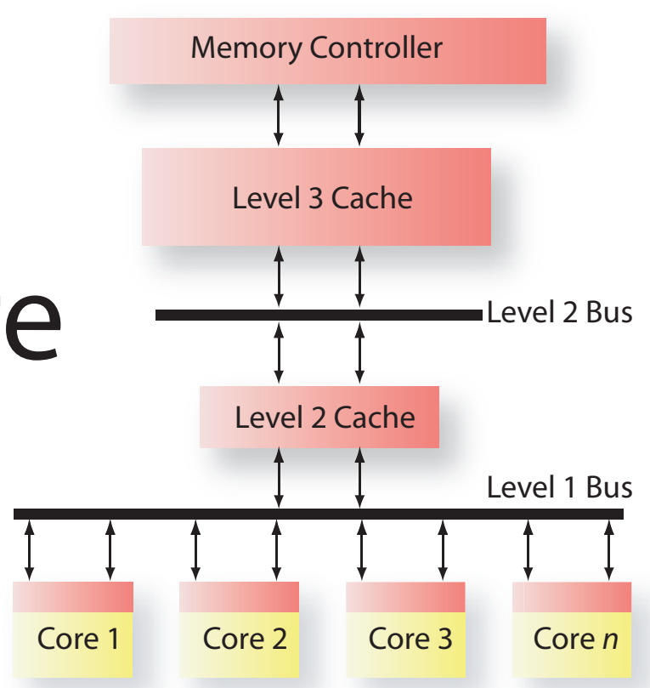
Virtualization and legacy support, parallelization and speculation

Dr. Ian Rogers, Prof. Ian Watson

Hardware

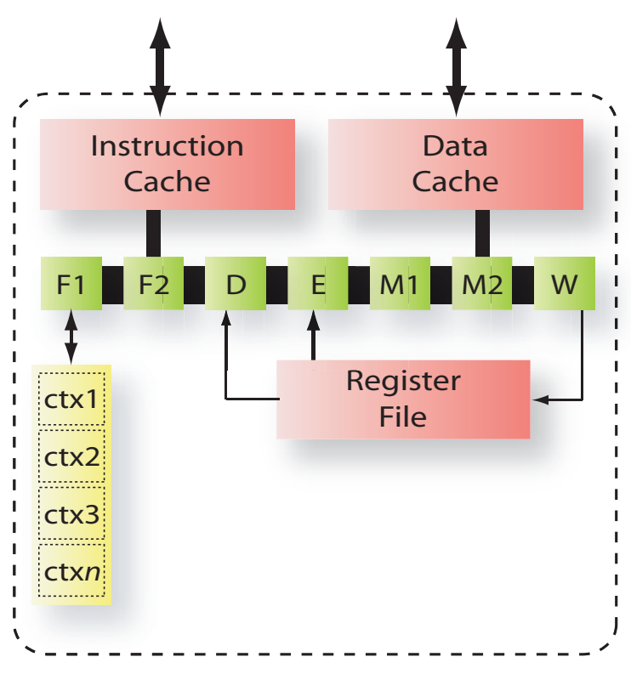
Chip Multiprocessor

Parallel processors and systems-on-a-chip are becoming the norm.



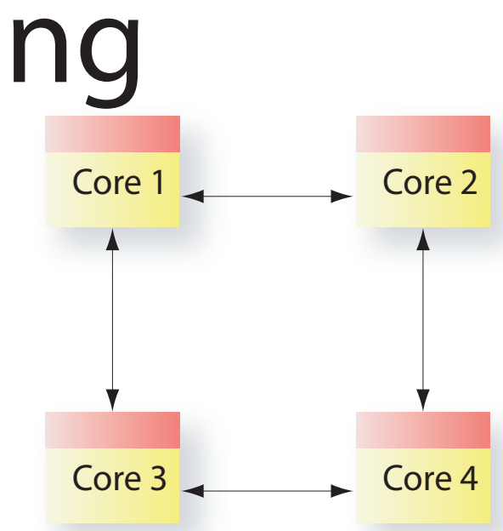
Chip Multithreaded

Long memory latencies can be hidden by scheduling other hardware contexts.



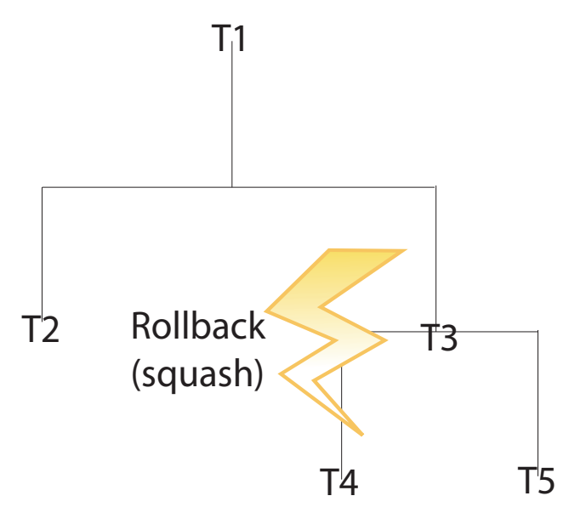
Non-Uniform Memory Access

Local memories are inherently faster and lower power. Non-uniformity exposes the need for optimizing memory accesses.



Transactional Memory

Hardware support for nested transactions and rollback to support lock-free parallel execution.



Object-Oriented Memory

Addressing to support objects, legacy support, non-local memories and transactions. TLB caches data keeping memory accesses cheap.

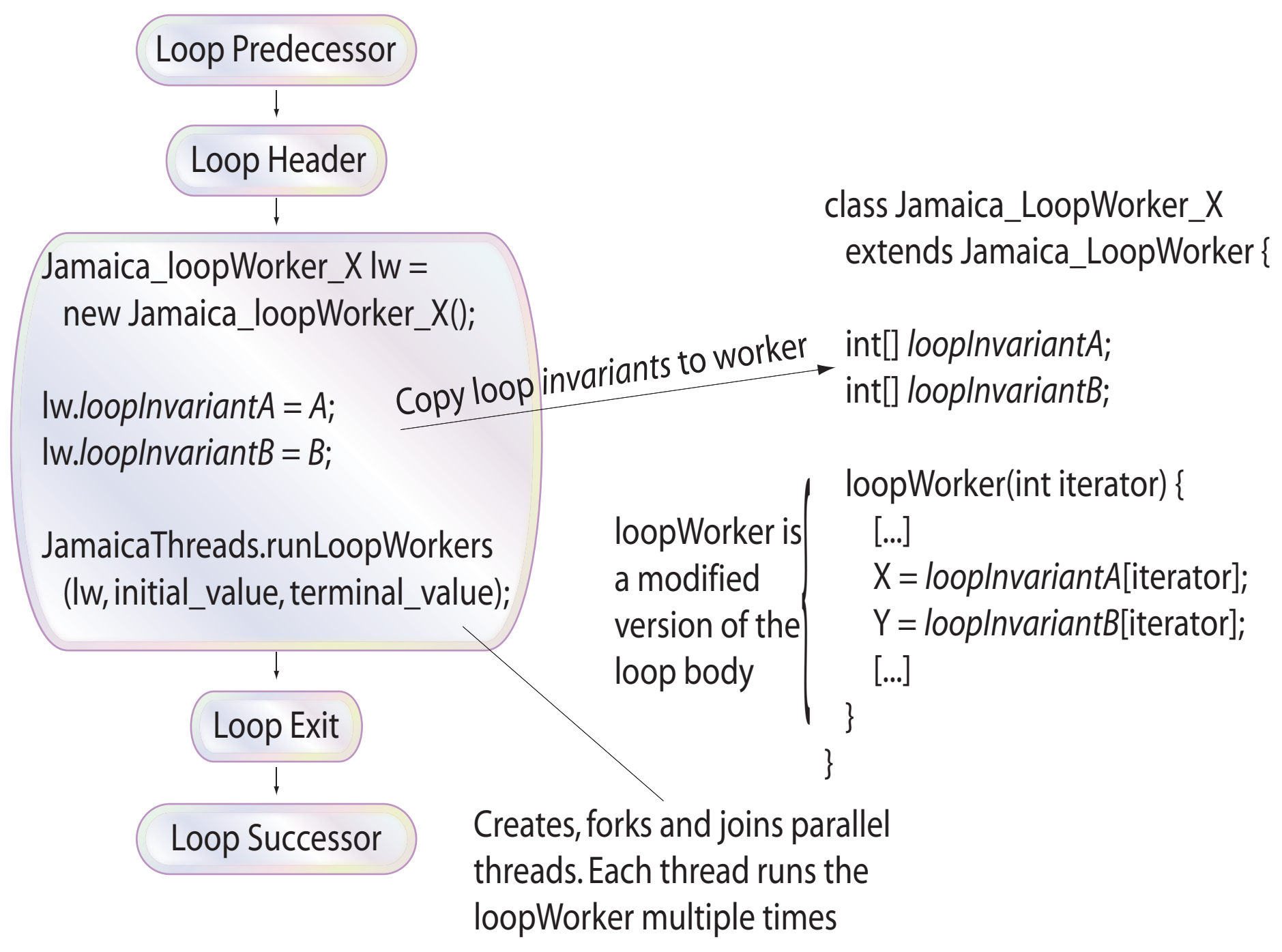
Shared or Local | Transaction Epoch | Object ID | Offset

Virtual Machine

Parallelization

Supercompiler

Supercompilers typically target mathematical languages extracting parallelism. We use the same approaches to dynamically create threads that can take advantage of lightweight threads and idle contexts.


Loop Predecessor
Loop Header
Copy loop invariants to worker
loopWorker is a modified version of the loop body
Loop Exit
Loop Successor
Creates, forks and joins parallel threads. Each thread runs the loopWorker multiple times

```
Jamaica_loopWorker_X lw =
  new Jamaica_loopWorker_X();

lw.loopInvariantA = A;
lw.loopInvariantB = B;

JamaicaThreads.runLoopWorkers
  (lw, initial_value, terminal_value);
```

```
class Jamaica_LoopWorker_X
  extends Jamaica_LoopWorker {

  int[] loopInvariantA;
  int[] loopInvariantB;

  loopWorker(int iterator) {
    [...]
    X = loopInvariantA[iterator];
    Y = loopInvariantB[iterator];
    [...]
  }
}
```

Speculation

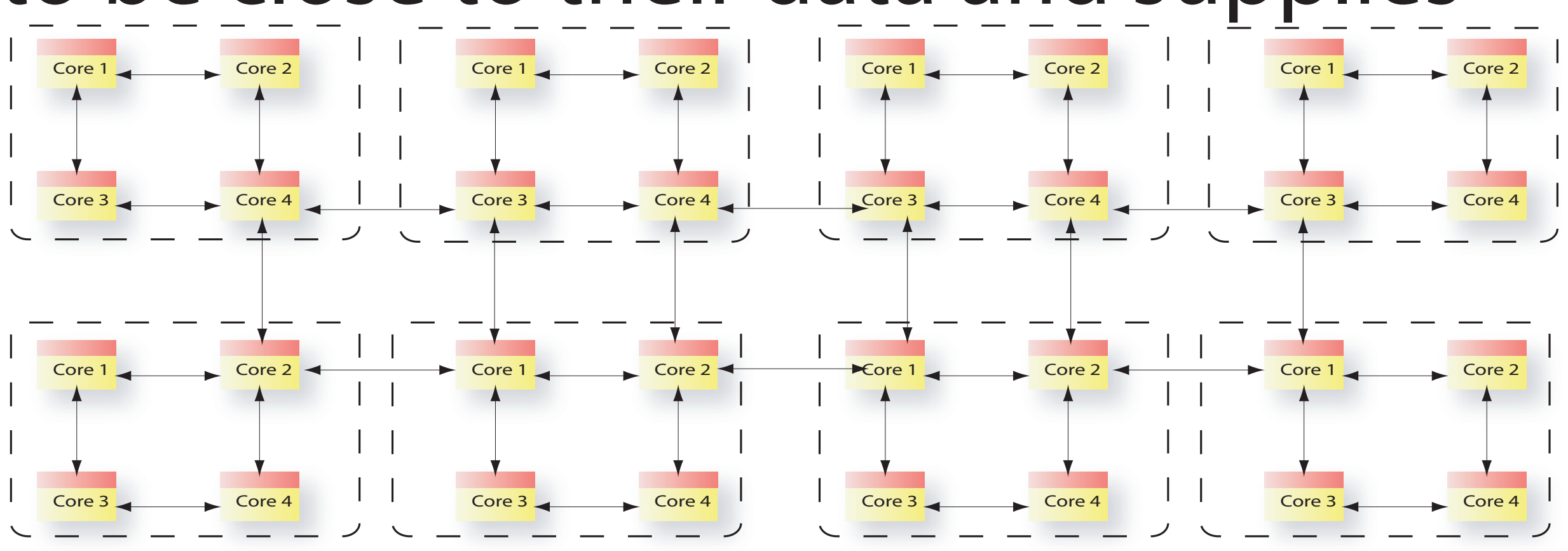
Use transactions to create work that can be discarded if a loop breaks-out early, or if two pieces of code interfere through side-effects.

```
for (i=0; i < A.length; i++) {
  if (A[i] == -1) break;
    A[i]++;
}
```

```
a = computeValue1();
b = computeValue2();
```

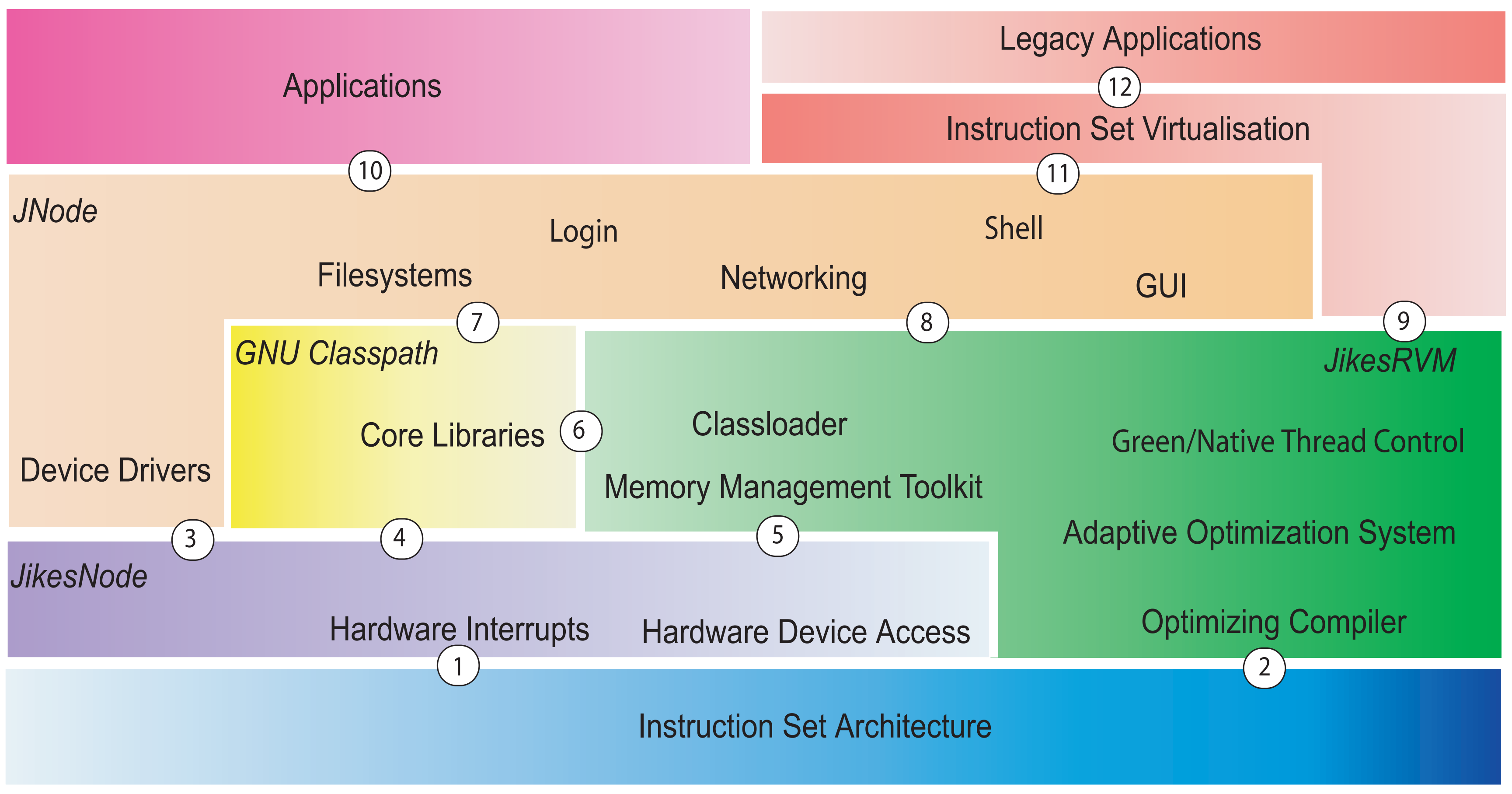
Locally Distributed VMs

Present the programmer with a single virtual machine, internally objects are distributed around memories, threads migrate to be close to their data and supplies of idle contexts.



Operating Systems

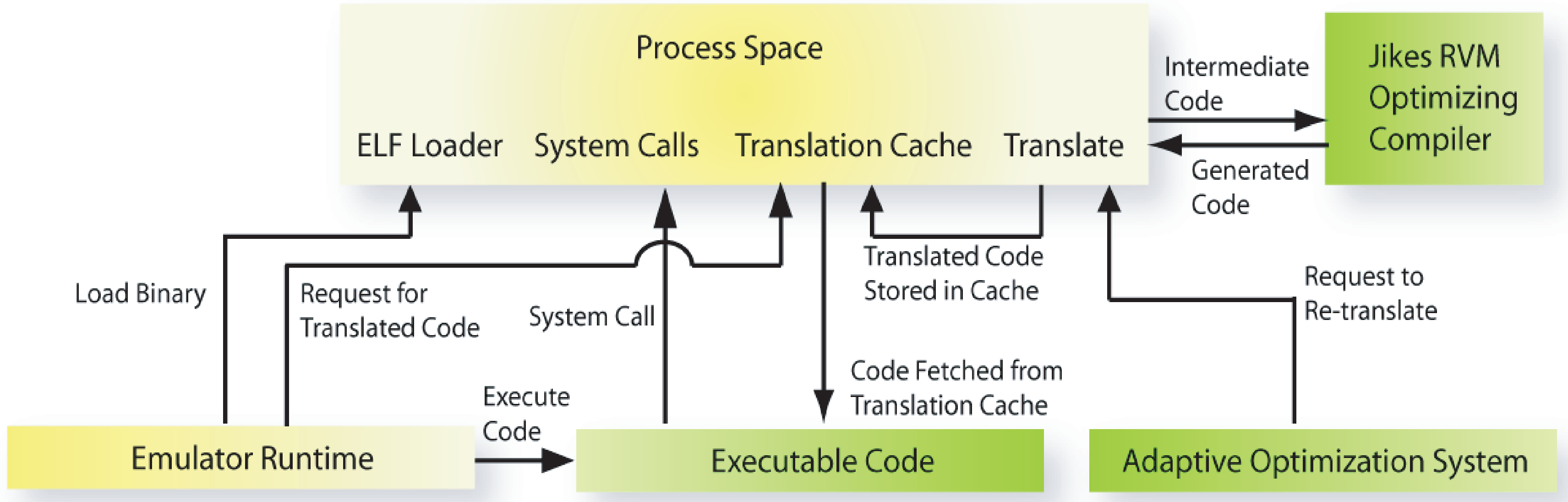
Integrating the VM and OS exposes optimisations across library and OS barriers. Having a "safe" language allows memory management to be simpler and faster. Scheduling improvements are exposed in both the VM and OS.



Legacy Support

Having a new architecture requires support if users are to migrate to it. FX!32 and Rosetta demonstrate the ability to migrate using dynamic binary translation. PearColator advances the approaches of FX!32 and Rosetta by utilizing the compiler performance of a VM optimizing compiler.

Hardware support for emulated virtual memory allows pages to be emulated as objects at consecutive addresses.



The University of Manchester

The Advanced Processor Technolgoies Group

JAMAICA Project

BinaryTranslator.org (PearColator)

The Jikes RVM