

JikesNODE and PearColator: A Jikes RVM Operating System and Legacy Code Execution Environment

Dr. Ian Rogers and Dr. Chris Kirkham

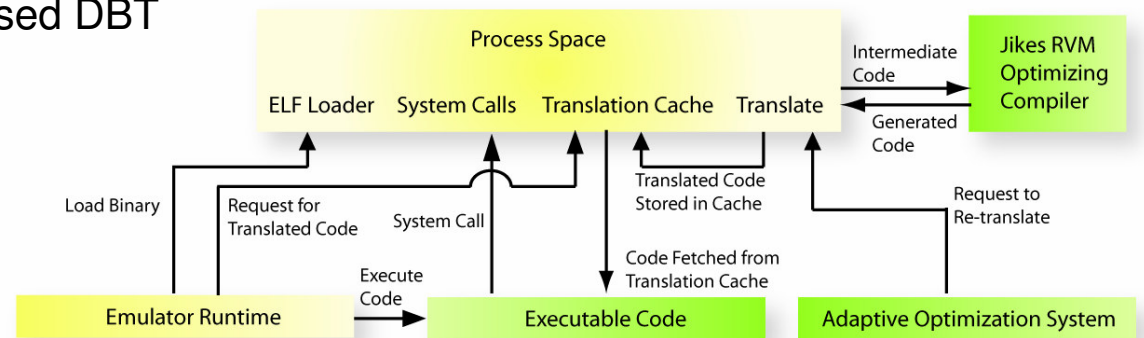
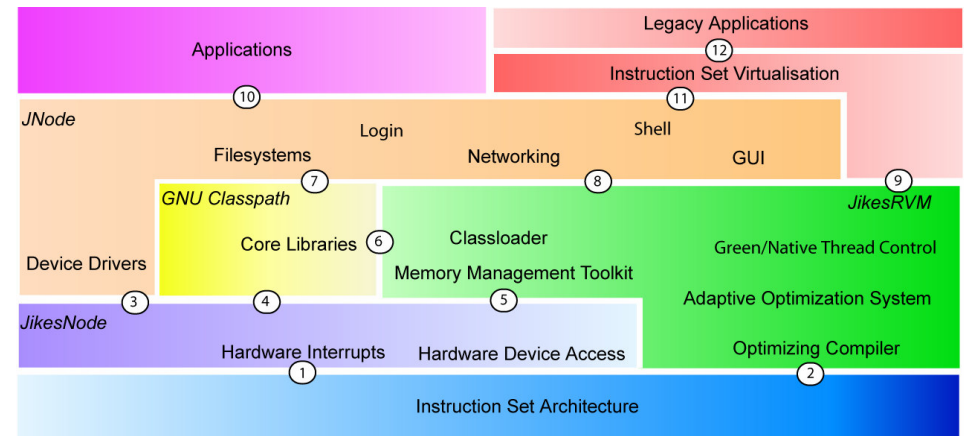
The JAMAICA Project

The Advanced Processor Technologies group

<http://www.cs.manchester.ac.uk/apt>

Presentation outline:

- Motivation
 - virtualization
 - the JAMAICA architecture
- The Jikes RVM optimizing compiler
- JikesNODE – Jikes RVM based OS
 - structure
 - status
- PearColator – Jikes RVM based DBT
 - structure
 - performance
 - future work
- Summary



Virtualization

- Platform independence
 - Operating system virtualization
 - Run multiple operating systems simultaneously on virtualized hardware
 - Application virtualization
 - Standard application formats such as ELF can run on a multitude of operating systems as binary format and system call interface are standardized.
 - Wine allows windows applications to run on FreeBSD, Linux and Solaris
 - Instruction set virtualization
 - Dynamic binary translators ...
- Hardware flexibility
 - Transmeta - 4-way VLIW TM3000 and TM5000 processors, 8-way VLIW TM8000 processor all run IA32 code
- New compiler optimizations
 - e.g. dynamic parallelization (ECOOP-POOSC presentation from yesterday)

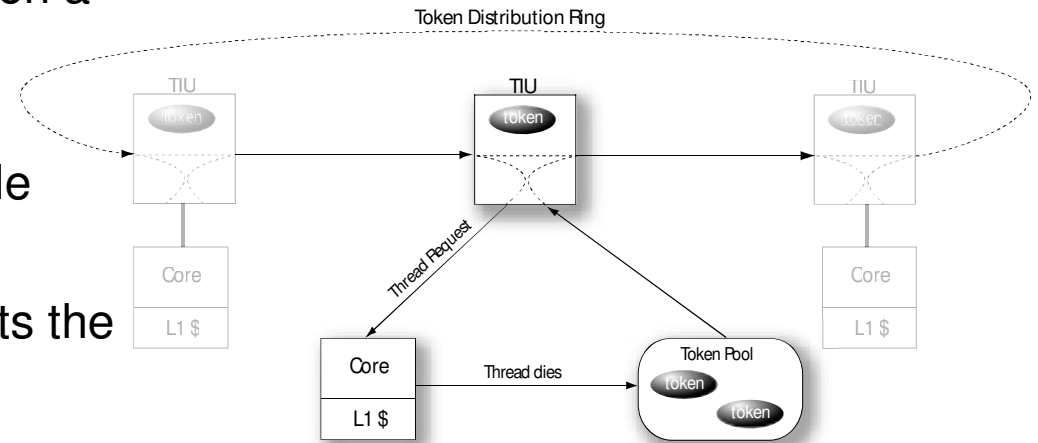
SMT and CMP Architectures

- Simultaneous MultiThreading (SMT):
 - performance gap between processor and memory is growing
 - threads can be scheduled on cache misses to hide memory access time
- Chip MultiProcessors (CMP):
 - instruction level parallelism reaching limits
 - reduce design complexity
 - local clocks aid clock distribution
- Threaded code necessary to expose parallelism
- New mechanisms to help expose threaded parallelism
 - thread scheduling and work distribution
 - speculative threading (transactional commit mechanism)
- Need to expose maximum amount of code to these mechanisms, hopefully in a flexible manner

Overview of the JAMAICA architecture

Work distribution

- Idle threads distribute tokens on a token ring bus
- Executing context on a core requests to ship work to an idle context or core and context
- Taking a token from ring grants the use of a particular context
- Shipping of work between cores occurs over data bus
- Gives lightweight thread creation
- When token is redistributed, work has been completed
- Thread unit monitors for completion of forked work

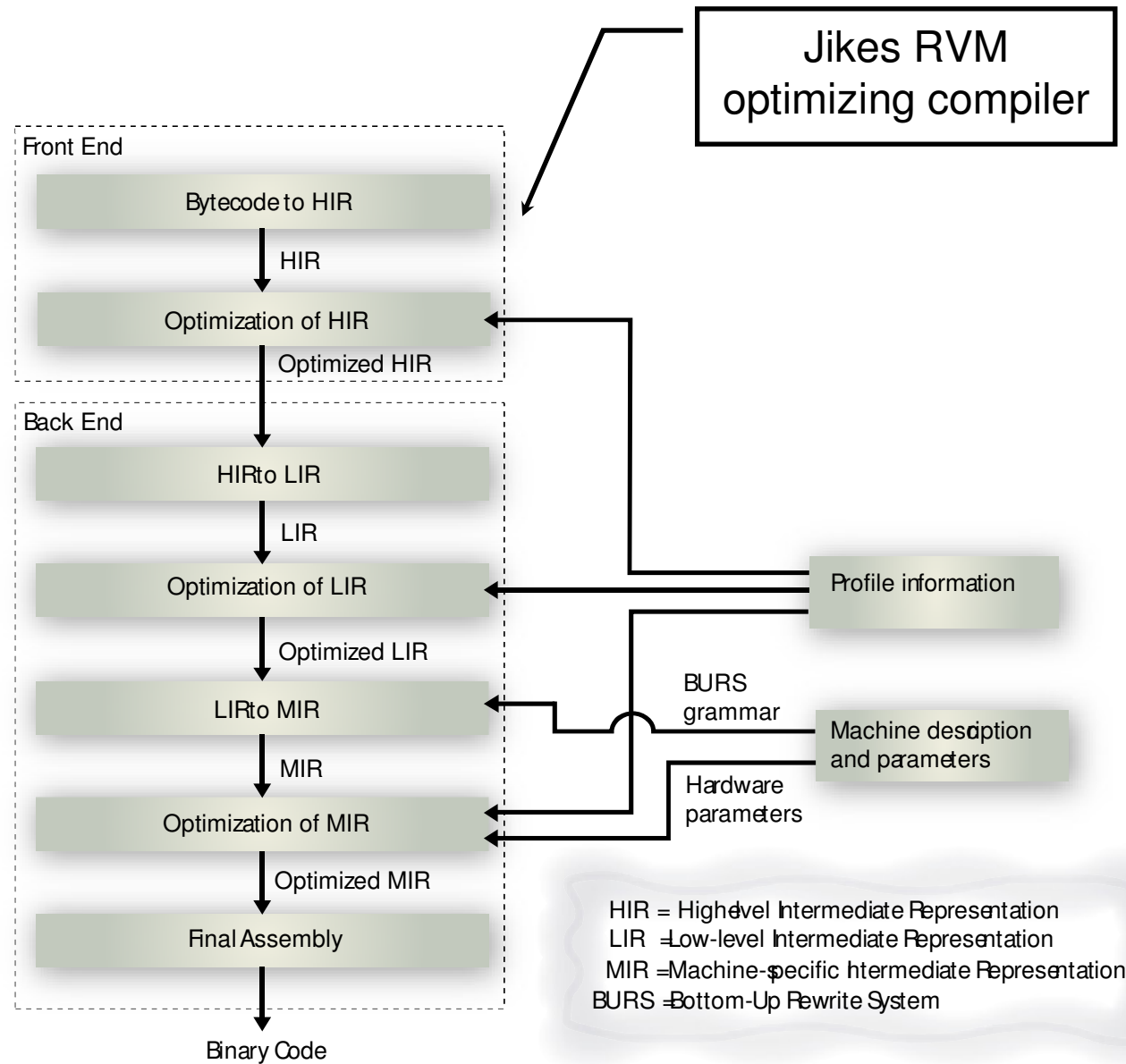


Software support for the JAMAICA architecture

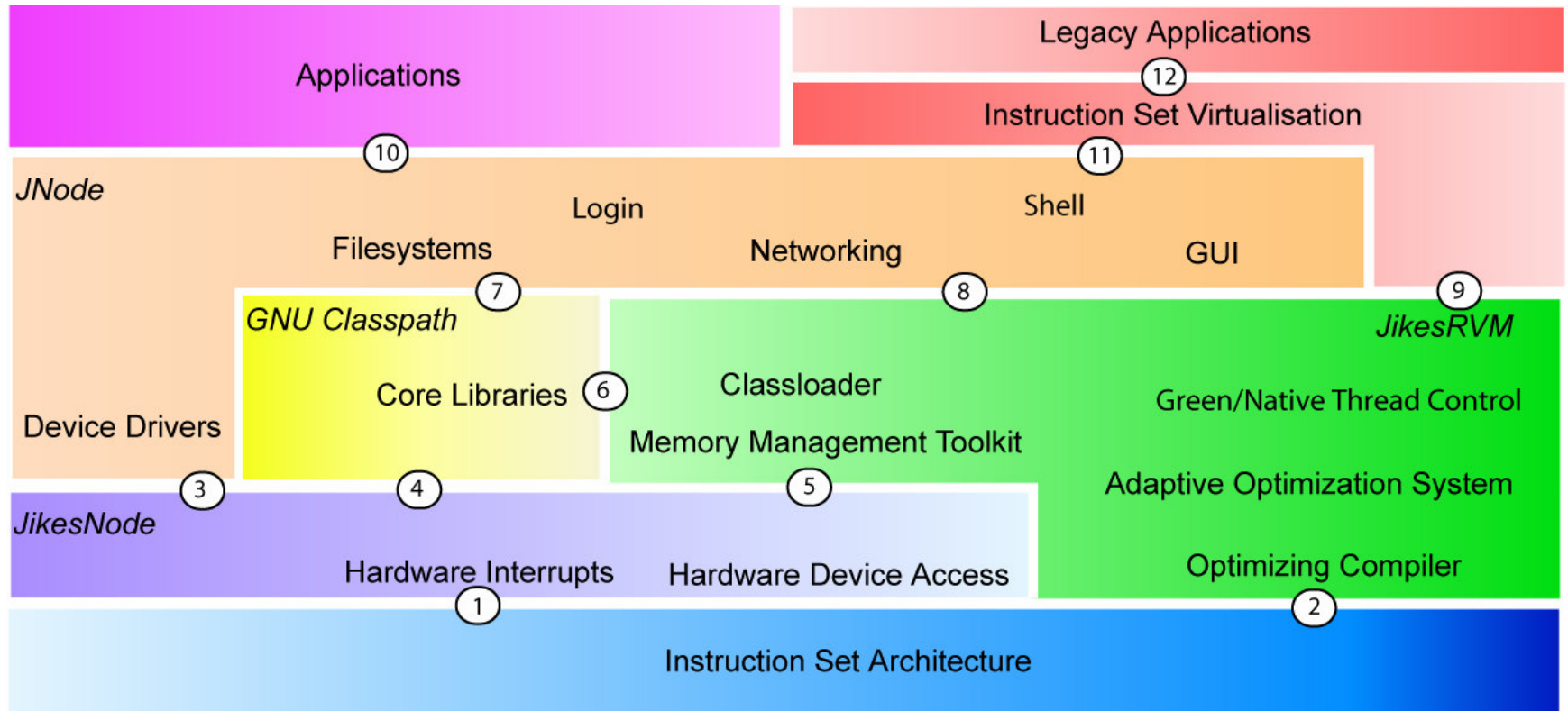
- Tools
 - C compiler – based on Princeton's LCC
 - jtrans – Java class file to assembler
 - javar – modified to generate jtrans parallel constructs
 - sim-idbg – interactive debugger and simulator in C
 - SIMPA – threaded, interactive, cycle accurate and fast simulator in Java
 - Jikes RVM – JAMAICA back-end and runtime

The Jikes RVM

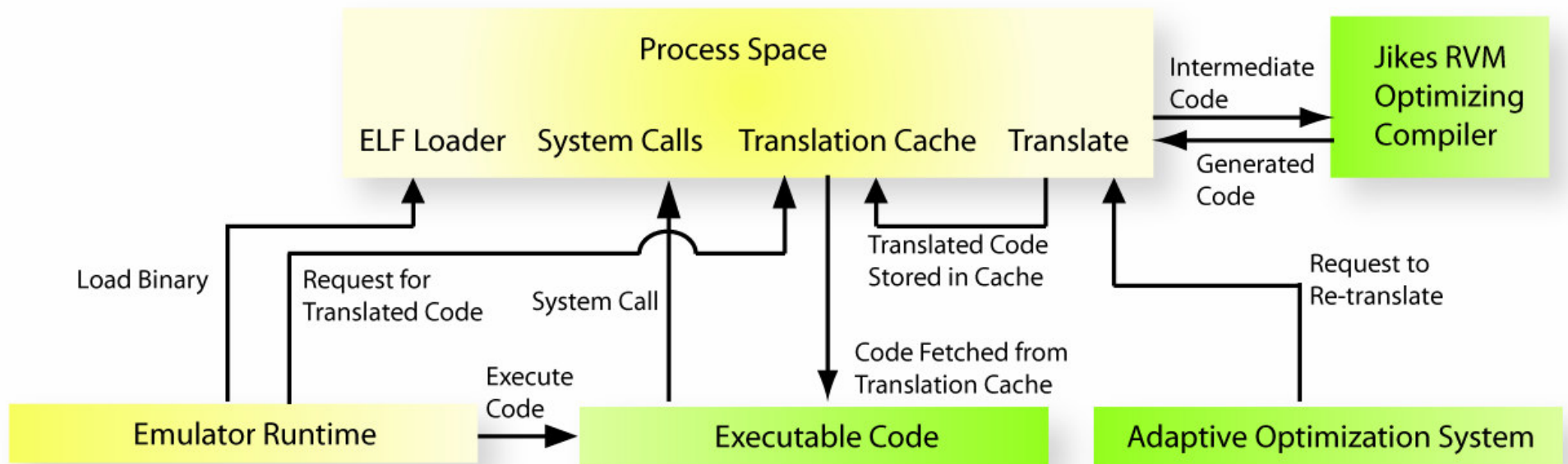
- JVM written in Java
- Support for IA32, PowerPC and JAMAICA
- Baseline (quick) and optimizing compilers
- Adaptive optimization and feedback system
- Extended array SSA form sub-stages in HIR and LIR optimization



JikesNODE structure




PearColator



PearColator – Instruction decoders

- Each instruction has an instruction decoder with a disassemble, interpret and translate method
- Object-oriented dispatch enables profile guided re-optimization to approximate threaded interpreter. i.e.:

```
Class AddDecoder extends InstructionDecoder {  
    void interpret (ProcessSpace ps) {  
        // interpret instruction – decoded fields held in process space  
        return getInstructionDecoder(getNextInstructionAddress())  
    }  
}
```



Inlinable calls to superclass

PearColator – trace generation

- Monitor branches and try to construct call graph
- Global and local (within a procedure) graphs created
- Return instructions turned into branches or switches, which can be eliminated by constant propagation of the link register value

PearColator – lazy evaluation

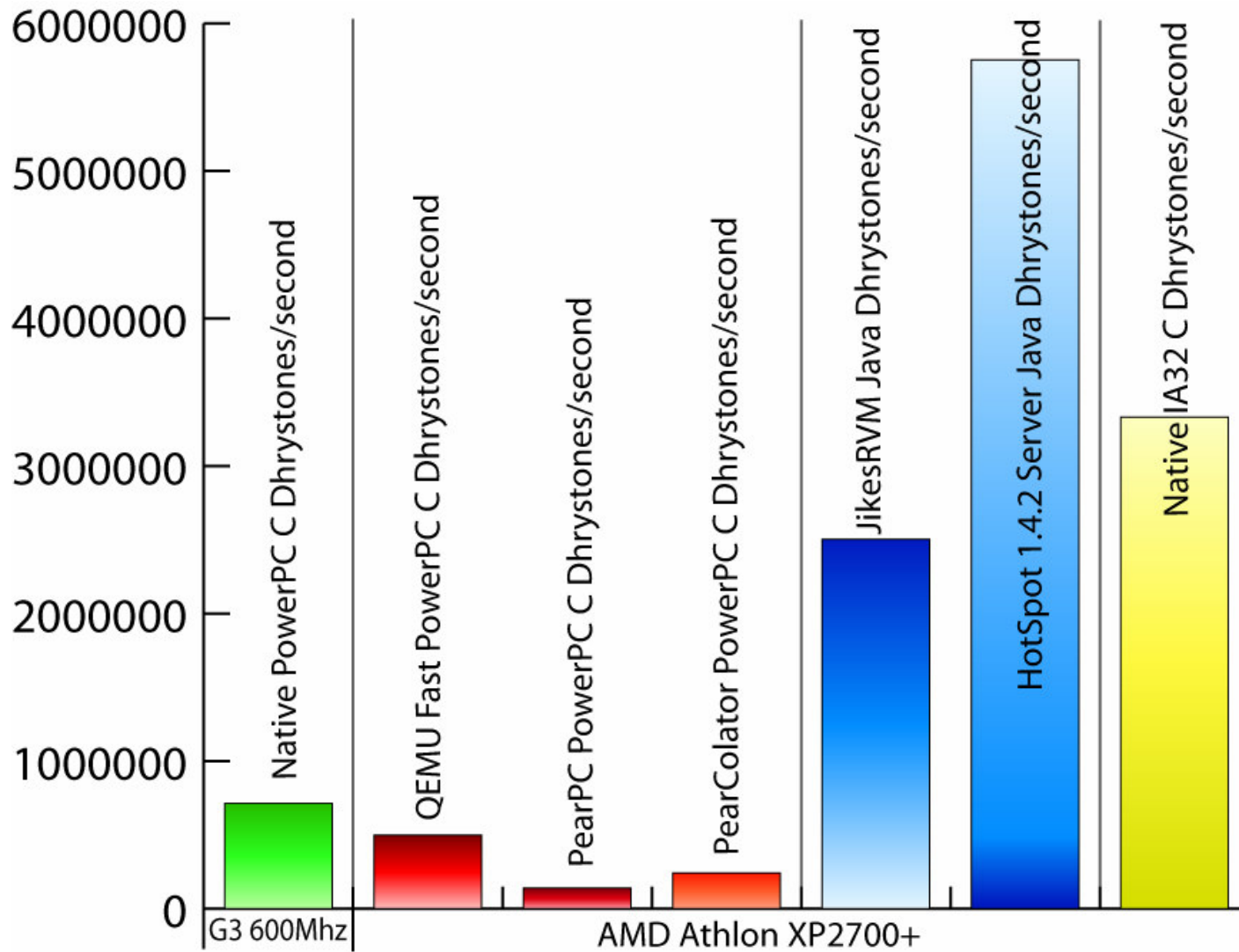
- Instructions in trace are recorded for backward branch patch up
- Key to translated instruction is a combination of address and 'lazy' state
- Lazy state used to avoid evaluation of all flag values in common compare-branch cases or to avoid sub-register recombination on X86 (ala FX!32)

0x10000074.0x0.-1	lil R10,0x0	<i>BB3</i>	0
0x10000078.0x0.-1	lil R11,0x0	<i>BB5</i>	1
0x1000007c.0x0.-1	addi R10,0x1(R10)	<i>BB6</i>	2
0x10000080.0x0.-1	add R11,R11,R10	<i>BB7</i>	3
0x10000084.0x0.-1	cmpiW cr0,R10,0x5	<i>BB8</i>	4
0x10000088.0x10000000.-1	blt 1000007c	<i>BB9</i>	5
0x1000008c.0x10000000.-1	lil R0,0x1	<i>BB10</i>	6
0x10000090.0x10000000.-1	svca 0x0	<i>BB12</i>	7
0x10000094.0x0.-1	beq 10000098	<i>BB13</i>	8
0x10000098.0x0.-1	Invalid opcode	<i>BB14</i>	9
0x1000007c.0x10000000.-1	addi R10,0x1(R10)	<i>BB16</i>	10
0x10000080.0x10000000.-1	add R11,R11,R10	<i>BB17</i>	11
0x10000084.0x10000000.-1	cmpiW cr0,R10,0x5	<i>BB18</i>	12

PearColator – memory model

- Object-oriented structure with multiple different mechanisms for paging, endian transformation
- On stack replacement allows policy for memory model to be changed on the fly at the expense of throwing away translations
- Examples:
 - Can use 1gb segment as a Java array, if memory above 1gb is needed switch to paged mechanism (costs 1 load per load or store)
 - Can pre-swap integers on file/device accesses to avoid byteswap instructions. If majority of memory accesses are from file/device alter policy so that byteswapping is done in translation
- JAMAICA has separated address spaces for translated instructions and the data address space. Pages can be marked as containing a translation to detect self-modifying code. Mechanisms appear similar to ones patented by Transmeta.

PearColator - performance



Future work

- Full machine emulation
 - Addition of devices and supervisor instructions to PearColator to allow full PowerPC hardware to be emulated and boot an operating system
- Support for user mode execution of 32bit X86 applications
- Tidy up and moving out of Jikes RVM code tree to allow creation of its own open source project

Future work

- Speculative execution
 - Range of speculative and non-speculative execution states
 - tree rooted at non-speculative state with branches for every spawned speculative context
 - speculative contexts may spawn more speculative contexts
 - If speculation goes wrong squash speculative state
 - throw away values in cache or a buffer
 - Detect speculation problems:
 - in software: when a value isn't that expected explicitly squash
 - in hardware: when an address is loaded by a speculative context, ensure that stores to the same address from a less speculative context cause a squash
 - Problems with creating speculative threads and avoiding excessive squashing
 - Mechanism may aid virtual machines, e.g. handling of unaligned memory accesses

Summary

- We have presented a Java operating system implemented on the Jikes RVM that exposes a greater amount of bytecode to the adaptive compilation and optimization system
- We have presented a dynamic binary translator that can be incorporated with the operating system to allow migration to new computer architectures like JAMAICA
- Operating system needs work on device drivers and on JNODE to become more complete. JAMAICA and X86 builds being worked upon.
- DBT performance is already on a par with comparable DBTs
- Virtualization of the instruction set has been demonstrated to allow for hardware migration and new compiler optimizations, in particular parallelization

Thanks!

- ... and any questions?