cs703: Advanced Topics in Programming Languages and Compilers

Programming Languages and Compilers in Systems and Architectures

Lecture 6: Jikes RVM

Ras Bodik
University of Wisconsin
http://www.cs.wisc.edu/~bodik

Outline for today

- **Paper / guest talk announcements**
  - Monday, Feb 11: **Alex Mohr**, HijackGL, **room: CS2310** a graphics talk on intercepting and translating API calls
  - Wednesday, Feb 13: **Craig Zilles**, Master/Slave Speculative Parallelization with Distilled Pgms, **room: CS2310,** an architecture/compiler talk on speculative threads
- **Project 1**
  - possible topics
  - where to start
- **Background material for Wednesday paper (on run-time program specialization)**
  - program specialization (done on Friday)
  - Single Static Assignment form (SSA)

Project 1

- **Ideas:**
  - most of them exciting and cool or at least practical
  - many of them too complex for a warm-up project
  - missing some mini-reviews (?)
- **Attributes of a good project**
  - finish in a week -- 10 days
  - compare Jikes to other VMs; fun of making it better, tuning it
  - exposes more than one component of Jikes:
    - compiler intermediate representation (IR)
    - the run-time: start the program, libraries
    - garbage collector
    - lock implementation
    - adaptive system

Sample project suggestions:

- **evaluate, by playing with:**
  - allocators, adaptive system
  - find hot spots in an application, find optimization opportunities
- **profiling:**
  - edge / path profiling
  - call-graph profiling (object type profiling)
  - removable dynamic counters
- **classical optimizations:**
  - loop-invariant code motion
  - constant propagation
  - *adaptive* loop unrolling
  - *simple* register allocator
- **hardware optimizations:**
  - turn off some functional units

Sample projects, cont.

- **adaptive optimization:**
  - better adaptive optimization plan (what's hot?)
  - estimate optimization cost
  - reduce optimization overhead
  - persistent optimization
  - translate on basic-block granularity (rather than method gran.)
- **locking and scheduling**
  - how is it implemented?
  - implement other scheduling policies
- **new allocator / new garbage collector**
  - a cache sensitive allocator / copying garbage collector
- **performance evaluation (did optimizations do a good job?)**
- **security monitoring**
- **JNI**

Classify the ideas

- **green:** can be finished in a week
- **blue:** fun – improves/speeds up Jikes
- **red:** teaches features useful for project

Project 1-**a (part I)**

- **Due Wednesday, Feb 13**
  - it's lots time, but you'll have four papers to read, too.
  - start early
- **Form groups of two people**
  - compiler expert plus a non-compiler expert
  - meet, think, and divide work
  - work: mainly understanding code
  - use class mailing list for questions and suggestions (cs703-1list@cs)
- **What to do:**
  - install Jikes on linux, @tux, (ask me for disk space if needed
  - collect and run some benchmarks (I'll give you specjvm98)
  - compare speed with some other JVM on tux (w/, w/out JIT)
  - implement simple instrumentation, minimal requirements:
    - count how often some (small) set of procedures was called
    - output the profile to a file
  - prepare a write-up (one-page email)
- **Send me questions on what to cover in class** (missing background)

Installing Jikes (on tux)

- **We have a version running on IBM AIX**
- **Anybody succeeded on Linux?**
  - I could not download IBM JDK (Java libraries, run-time, etc) from IBM web site
- **Multi-processing / multi-threading**
  - Jikes on AIX/PowerPC supports multi-processor (MP) hardware (Linux does not)
  - if you want to do experiments with locking, MP garbage collection, ask me for account

Where to start

- **Read User's Guide**
  - covers installation, too.
  - pretty good, for a research prototype
- **Tutorials on the Jalapeno web page**
  - also useful
- **I am not an expert**
  - implemented some optimizations, with a myopic view of the RVM
  - but slowly learning remaining pieces
  - I can help with explaining compiler techniques, and help you find the desired functionality in the code