The Mono JIT optimizations and evolution

Massimiliano Mantione
massi@ximian.com

October 24, 2006
Current status

✔ Five years old and fairly mature subsystem

✔ Supports various optimizations and AOT compilation

✔ Ported to x86, PowerPC, Sparc, AMD64, s390, s390x, ARM, IA64, while Alpha and MIPS are underway

✔ During these five years has already been rewritten once
Inline and basic options on by default

✔ Cprop, together with deadce, work in synergy and are needed to make inline effective

✔ Also a “tree propagation” hack is needed

✔ Results

✔ XMLMark/SAX improved by 6%
✔ Fast Fourier Transformation improved by 21% on x86
✔ SciMark improved by 5% on x86 and 2.8% on amd64
✔ Mono bootstrap improved by 2.5%
Partial Redundancy Elimination

✔ Includes loop invariant code motion

✔ Not enabled by default because it needs tuning and slows the JIT down

✔ Results
  ✔ XMLMark improved by 5% on x86
  ✔ Fast Fourier Transformation improved by 21% on x86
  ✔ SciMark improved by 22% on x86 (-7% on amd64)
  ✔ Mcs bootstrap improved by 3% on hot run (-6% cold)
Intermediate Representation (IR)

✔ Tree based IR
  ✔ CIL arguments and locals correspond to local variables
  ✔ CIL “homeless values” (stack slots) correspond to tree nodes

✔ Opcodes are lower level than CIL ones

✔ BURG is used for instruction selection and linearization of the instruction trees
Current issues

✔ The regalloc split uses registers suboptimally

✔ Callee saved registers are never used for global variables (and vice-versa)

✔ The “treemover” is needed only because of this

✔ Complex optimizations (SSAPRE) need tuning, and interact badly with the regalloc

✔ SSA based optimizations are not used by default because they make the JIT slow
Ongoing work: linear IR

✔ No more trees of instructions
  ✔ All CIL values go into virtual registers (vregs)
  ✔ BURG is no longer used
  ✔ All opcodes are decomposed early (low level IR)

✔ Vregs are handled uniformly by all passes

✔ This makes a unified regalloc possible
Ongoing work: GREG (Global REGalloc)

- Unifies the current global and local ones
- Is more accurate
  - Live ranges are exact, taking holes into account
  - Can easily split live ranges at any point
  - Uses "second chance binpacking" to exploit registers as much as possible
  - The information to tune it (weight number of uses, spill costs...) is easily available
- Works on code in SSA form
What's this SSA thing?

✓ A “refined” form of IR, where each variable use can be reached by exactly one definition

✓ Is generally considered expensive to build, but...

✓ ...makes everything easier and faster!
How does SSA help GREG?

✔ In SSA use-definition relations are very natural

✔ Representing each move (also spills) as an SSA definition takes advantage of this simplicity

✔ With SSA, high level information on values (think register rematerialization) is readily available

✔ A register allocator already does the job of “undoing SSA form”, and actually benefits from SSA while doing this job
Profiling the SSA code on which GREG is based (mono –compile-all mscorlib.dll)

✔ With callgrind
  ✔ SSA, liveness computation and deadce add 15.68%
  ✔ Old liveness 7.86%, local deadce 6.83%, local cprop 6.16%

✔ With oprofile
  ✔ SSA, liveness computation and deadce add 7.23%
  ✔ Old liveness 3.96%, local deadce 3.69%, local cprop 3.77%

✔ With 'time' (wall clock measurement)
  ✔ SSA, liveness computation and deadce add 11.84%
Advantages of going fully SSA

✔ Global optimizations available by default (instead of the local versions we have now)

✔ No more separate pilelines when more powerful optimizations are enabled

✔ Every data flow analysis pass gets faster (also liveness computation)


Multilevel IR

- Alias analysis becomes feasible
- High level reasoning becomes faster
- JIT code becomes cleaner
- Eventually, the results of high level analysis passes will be used for interprocedural optimizations
Wrapping up:

☑ Lots of nice things to do!
☑ In little time... (we want them all and now)
☑ Which means: *business as usual*
☑ For discussion:

☑ Post on “mono-devel-list@ximian.com”
☑ Or to me, at “massi@ximian.com”
☑ ...or let's just talk now!