7.0 Compilation based on dynamic (runtime) information

Motivation: decide when a Partial redundancy should be turned into a full redundancy...

7.1 Compiler models
   Compiler structure to exploit dynamic information

7.2 Techniques to obtain dynamic information

7.3 Classification of dyn. information

7.4 Types of profiles
7.1 Compiler Structure

So far we used a simple compiler model.

"ahead of time" compiler

To include dynamic information, the model must change.

(2) Using dynamic information requires re-compilation.
why use dynamic info only once?

"continuous compilation"
- compile works in parallel to program execution

many systems delay compilation until (just before) execution
"just-in-time" Compiler extend to use dynamic information
JIT (just in time) usually offers different optimization levels.

- JIT compilation delays execution (first compilation must be fast).
- Optimization requires (re)compilations.

Boundary blurring:
- Simple AOT compiler
- Sophisticated JIT compilers
Issues (for the design of the Compiler Structure)

- Sophisticated infrastructure
  - must be able to modify executing binary
  "Self modifying code"

- Efficiency of Compiler
  - time & space
  - Compilation is part of execution
    - Multiprocessor/core systems may have extra cycles ...
      Still potential for indirect influence
    - Focus on partial compilation (focus on important parts)
      - What and when to optimize?
    - Predictable output binary difficult or impossible

5
• When are past properties (runtime information from past executions) a good (visible?) predictor of the future?

• Cross-compilation difficult
7.2 Obtaining dynamic program information

two approaches

1) Instrument program

- add instructions to obtain desired information

may be subject to optimization

problems

- extra instructions take time
- perturb measurements
  - primary effects
  - secondary (more insidious)

larger code segment
  - diffread cache misses
- Use hardware (platform) features to obtain information.
  Most processors contain a program-monitoring unit (PMU) for performance monitoring.
  - May report branches taken.
  - Can be set up to record data addresses.
  Based on sampling:
  - Every k-th instruction is monitored (recorded, counted, ...)
  \( (k-1) \) instructions are not inspected.
  Bookkeeping incurs overhead. PMU no impact; bookkeeping has...
  Limit the overhead by choosing appropriate \( k \).
sampling powerful idea - can be applied to instrumentation-based monitoring.

7.3 classification of dynamic information

- more an idea generator than a strict (mutually exclusive) classification

1. program properties vs platform properties
   - does not change if program is executed on a different platform
   - captures details about program processing input
   1) depends on hardware platform moving execution from platform $X$ to platform $Y$
"profile" information

e.g. frequency of invoking a method

may change data
e.g. cache miss rate

for \( x, y \) with caches of different size

must understand hardware... esp.
PMMU of your platform

Remarks

- Assume deterministic programs
  
  Same input \( \rightarrow \) same output

  (may have to build harness to capture the environment)

- PMU is (often) not part of the architecture specification [PMU is not constant over processor generations]
2) granularity of information

whole program metrics vs fine-grained metrics

- method
- basic block
- individual instructions

4 dimensions / questions to think about

1) what kind of information do you want to record

- relationship between dynamic information and compiler optimization/ transformation is not (always) obvious
2) How accurate does the information appear at what cost?

6) Perturbation ...

All approaches to exploit PPM information must trade off accuracy, cost, perturbation!

3) Must bridge gap between information obtained by PMU or instrumentation and compiler's data structures.

- may obtain absolute addresses in data space
- compiler works with symbol table
4) How stable is information? How does it scale?

What size program can be handled?

Always remember: cost of getting & processing dynamic information must be covered by the optimization!