Performance tuning tools

Performance tuning techniques

- Optimizing the entire program is unnecessary work
  - 90:10 or even 99:1 rule
- Optimize only the hotspots
  - Pragmatic definition:
    - Hotspot is the code where optimization has the greatest impact wrt. its cost
  - Problems:
    - The cost of optimization (the human effort needed) may be wildly variable
      - But it is probably proportional to the size of the hotspot code
    - The effect of optimization (the time saved) is difficult to predict
      - The upper bound of the effect is the total time spent in the hotspot
  - Approximate definition:
    - Hotspot is the code where total time divided by code size is largest
  - Total or self time?
    - Do we include the time spent in the procedures called from the hotspot?
      - If we do, the hotspot itself must be extended to these procedures too
      - We can hardly shrink the time spent in a procedure without changing its code
    - Procedure integration by compiler will often decide

Techniques for measuring program behavior

- Instrumentation
  - Modifying the program to measure itself
  - Performed by a compiler on intermediate code or by a tool on binary code
  - The additional code significantly disrupts the program
    - It makes sense to measure only unaffected quantities
  - Profile: number of passes through important points in the program
    - Basic blocks (transitions between them)
    - Procedures
    - Procedures including mutual calls
  - Profile driven optimization
    - The compiler uses the previously measured profile
      - to determine which parts of the program to optimize
      - to estimate the effect of some optimizations

Techniques for measuring program behavior

- Sampling
  - The unmodified program is launched
  - At appropriately selected moments, the current position is recorded
    - the instruction pointer
    - optionally, the calling procedure or a part of the call stack
  - The sampling moments must be
    - Sparse enough to not affect program execution
    - Dense enough to produce statistically significant data
    - Correlated with the program execution in a well-defined way
      - Independent - random sampling (approximation: periodic sampling)
      - Dependent on selected events (number of executed instructions, memory accesses, etc.)
  - Some parts (maybe a majority) of the code will never be hit by sampling
    - Sampling naturally prefers frequently executed code - the hotspots
    - Sampling is not accurate enough to pinpoint individual instructions
      - But averaging across a loop will work

Techniques for measuring program behavior

- Sampling
  - Event generation techniques
    - "Software" - timer interrupt
      - It requires more frequent interrupts than the usual OS timer setting
      - Periodic interrupts may not be statistically independent of program execution
    - "Hardware" - profiling support in the CPU
      - The CPU generates an internal interrupt when the preset number of events is reached
      - Events: Clock ticks, instructions, memory accesses, branch misprediction, ...
      - Only few types of events may be measured simultaneously
      - The program may be rerun with different event setting
      - The profiling software may frequently change the setup during one execution
  - Sample recording techniques
    - "Software" - the record is created by the interrupt handler
    - "Hardware" - the record is created by the CPU (by writing into memory)
      - Allows for more frequent sampling
      - Does not allow call-stack exploration
      - Does not allow randomization of the sampling period
    - In both cases, the record may be misplaced by few instructions

Techniques for measuring program behavior

- Instrumentation
  - Accurate measurement of (somewhat) distorted execution
  - Some compiler support usually required
  - No HW or OS support needed
- Sampling
  - Approximate measurement of (almost) true behavior
  - Compiler support not required
    - Debugging information needed to understand results
  - Possible without any HW support
    - CPU support improves accuracy and adds new events
    - Understanding CPU-specific events is difficult
  - OS kernel support always required
    - Manipulating timer interrupt and/or setting-up the CPU support
    - Handling the timer/sampling interrupt