Reducing Performance Non-determinism via Cache-aware Page Allocation Strategies

Michal Hocko, Tomas Kalibera

Distributed Systems Research Group
http://dsrg.mff.cuni.cz
Department of Software Engineering
Charles University in Prague
Faculty of Mathematics and Physics
Czech Republic
Performance Non-determinism

Response time

Request index
Non-determinism in execution is particularly bad for benchmarking.
Main() {
    initialize();
    warm-up();

    for (i=0; i<nmeasurements, i++) {
        before = getCurrentTime();
        doOperation();
        after = getCurrentTime();

        results[i] = after - before;
    }

    print(results);
}

Cmd-line> ./benchmark
Cmd-line> ./benchmark
Cmd-line> ./benchmark
Non-Determinism in Measurement and Execution

Time [ms]

1st execution

2nd execution

average

measurements

measurements
Non-Determinism in Measurement and Execution
Non-Determinism in Measurement and Execution
Non-determinism in Execution is Costly

Main() {
    initialize();
    warm-up();

    for (i=0; i<nmeasurements; i++) {
        before = getCurrentTime();
        doOperation();
        after = getCurrentTime();

        results[i] = after - before;
    }

    print(results);
}

Cmd-line> ./benchmark
Cmd-line> ./benchmark
Cmd-line> ./benchmark

Repeated with every execution

One measurement

One execution
Non-determinism in execution is caused by cache & virtual memory.
Application Memory Layout (Linux)

- Program code
- Initialized data
- Non-initialized data
- Stack
- Cmd-line arguments
- Environment

Low virtual addresses

High virtual addresses
Cache and Addressing on Typical System

Page address

Page offset

Virtual address

Decided by OS

Page address

Page offset

Physical address

Cache set

Line offset

Cache lines

data

data

data
Page/Cache Color

- Operating system assigns colors to pages
- Data from pages of different colors do not collide in the cache
Could a cache-aware strategy for selecting page colors reduce non-determinism in execution?
Good Old Cache-aware Strategies

- **Page Coloring**
  - Heuristic for “spatial locality”
  - Adjacent pages have different color – do not collide
  - Solaris, Windows, Free BSD

- **Bin Hopping**
  - Heuristic for “temporal locality”
  - Pages first accessed in sequence have different color
  - Digital Unix

- **No Support in Linux**
Our Contribution

• Linux Kernel extension for strategies
  – Supports bin hopping and page coloring as modules
  – Supports more: other strategies, application layer control, etc

• Large empirical study in Linux
  – 4500 benchmark experiments
  – Evaluation based on statistical methods
Benchmarks

• Mono (C#)
  – SciMark2 – FFT (numerical)
  – TCP/HTTP Ping (remote communication)
  – Rijndael (cryptography)

• SciMark2 (C, numerical)
  – FFT, Matrix Factorization, Monte Carlo, ...

• Csibe (C/C++)
  – JPEG (multimedia compression)
  – GZIP, BZIP2, PNG (lossless compression)
  – Lexical analysis, abstract machine simulator, ...
Evaluation Methodology

• Executed about 4500 experiments
• Question for evaluation:
  – “Does page coloring or bin hopping provide lower response time/non-determinism than the default kernel strategy?”
• Metrics
  – Mean response time, impact factor of non-determinism
• Quantitative Summary
• Qualitative Summary
Non-det. in Execution: Quantitative Summary

- **Least Non-det.**
  - Default Kernel: 50%
  - Page Coloring: 60%
  - Bin Hopping: 60%

- **2nd Least Non-det.**
  - Default Kernel: 40%
  - Page Coloring: 30%
  - Bin Hopping: 40%

- **3rd Least Non-det.**
  - Default Kernel: 20%
  - Page Coloring: 10%
  - Bin Hopping: 30%
Summary

• Response time
  – Cache-aware strategies don’t help
  – Page coloring performs like default, bin hopping is sometimes slightly slower

• Non-determinism
  – Cache-aware strategies reduce non-determinism
  – Bin hopping sometimes reduces a bit more than page coloring

• Our kernel extension allows to select a strategy on application basis