2014 October 2
Discussion of “Automatically Characterizing Large Scale Program Behavior”

This is one of the "simpoint" papers.

- The authors wrote many simpoint papers and this one lies in the middle.
- Simpoints are very popular because simulation takes too long; simpoints extract the most relevant simulation points for more rapid simulation.
- Designed for the Alpha ISA.

Brad Calder

- Was at UC San Diego at the time of the paper, but he moved to Microsoft later.

Greg Hamerly

- Was a student, now a professor.

Timothy Sherwood

- Now at UC Santa Barbara.

The paper captures the idea that programs go through phases.

- There are phases within a program--large chunks--with approximately similar behavior.
    - These can lead to architecture, dynamic, and compiler optimizations and power management.
    - It is not necessary or realistic to try to examine or optimize for a program at an instruction granularity. It would be too costly to adapt at too fine a granularity.
- Phases in programs:
    - Fine-grain: 1-10 instructions.
    - Coarse grain: 1000-10000 instructions.
    - Large scale: 100 million instruction chunks.
- This paper focuses on using relatively few large chunks.
    - It was later found that using smaller chunks allows you to capture more information using fewer instructions.
- Simulation has to be long enough to warm up the machine.
    - Simpoints can be long enough to include their own warmup periods.
    - According to information theory, in general, using smaller chunks is a better approach, but these large simpoints are very convenient.

BBV

- Basic Block Vector: Used to identify and compare the phases.
- A basic block is a continuous sequence of instructions with one entry point and one exit point.
    - In general, branches delineate basic blocks.
    - In general, the more frequent the branches, the smaller the basic blocks.
- Basic blocks are found to determine overall program behavior.
    - In general, all modern microarchitectural techniques work better with larger basic blocks.
- Static versus dynamic basic blocks.
    - A static analysis of the code may reveal 500 basic blocks in a program.
    - But any given basic block may or may not be executed at all. These are dynamic basic blocks.
    - Many profiling tools just identify dynamic basic blocks.

    - Dynamic and static basic blocks may even be different.
- The basic block vector is a count associated with each basic block that captures the behavior of the run of a set number of dynamic instructions.
    - After creating vectors for each 100 million instruction chunk, the BBVs are compared to each other to find similarity.
        - They chose to use Manhattan distance to determine similarity. The distance is normalized so that the minimum distance is 0 and the maximum is 2.
- Basic block similarity matrix: the darker the area, the more similar it is.
    - Different programs show very distinct similarity matrices.
    - These allow us to reduce the number of things to simulate while still representing the overall program.
- What they found that, at the most, any program will have 20 different types of chunks.
    - PCA or other linear projections can be used to reduce the number of chunk types.
    - The paper reduces the dimensions down to 15.
    - GCC only needs 4 clusters.

Phase Finding Algorithm

- Profile the basic blocks.
- Reduce dimension of BBV data to 15.

As the number of dimensions increases, the amount of information increases.

Bayesian Information Criterion - A penalized likelihood.

- The criterion tells the likelihood that a cluster is good.
- It also attempts to fit the clusters onto a sphere.
- It allows them to formally establish how close to the optimal they have managed to get.
- To reach 90% accuracy, very few clusters are needed.

Validation

- Comparing the IPCs of simpoint simulations, fast forwarding, no fast-forwarding, and full simulation…
    - Simpoints are shown to be fairly close to the full simulation. The fast-forwarding approach (or even avoiding fast-forwarding and just simulating a limited number of instructions) is shown to be pretty poor.
    - Simpoints are not always the best approximation approach. Fast forwarding sometimes win.
        - This is to be expected because the simpoints technique is so coarse.
    - When using multiple simpoints, the simpoints get different weights.
        - The paper presents data showing the weights of the different simpoints; if you can only run one or two simpoints, always choose those with the highest weights.
        - Though the SPEC binaries cannot be released, by giving enough details on compilation and simpoints, the authors allow anyone to reproduce their results and use their conclusions.
    - Long simpoints show what happens when you run 10x as long as the original (single) simpoint.
        - LongSP does pretty well.
        - But using multiple simpoints does the best.

It would be completely unrealistic to execute full benchmark suites; they are simply too long and simulators are simply too slow. Even on SimpleScalar, SPEC CPU2006 would take years to run to completion.

How we can use this in our research:

- CPU2000: Alpha simpoints from UCSD.
- CPU2006: PINPOINTS tool from Intel.
- CPU2006: Pin Points from UT LCA (ICCD 2008 paper, Nair and John) (x86 binaries).
- CPU2006: 22 Alpha binaries - K. Ganesan (SPEC workshop 2009).
- PARSEC: ROI (Region of Interest). This is a similar idea to simpoints.
- CPU2006: for SIMICS--based on Ultra SPARC binaries--being generated in LCA now.

Available simpoint tools:

- PINPOINTS tool from Intel (PIN based).
- Valgrind BBV generation tool (Open source).
- Qemu BBV generation (Open source).
- PinPlay: fast forwards to the simulation point.

Even going from SPEC CPU2000 to SPEC CPU2006, where the programs grew by about 10x, the number of simpoints needed is essentially unchanged (to reach 90% accuracy). Simpoints just grab the essential information and SPEC 2006 is found not to be that different than SPEC 2000.

However, for SPEC CPU2006, the L2 MPKI error sometimes exceeds 120%. However, the actual number is itself very small, so a small change leads to a big percentage change. It's not that bad of a change going from 1 miss to 2 miss.

- There are times when things look really bad, but closer examination reveals it's not that meaningful--not that bad.