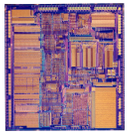
Discovering the Runtime Structure of Software with Probabilistic Generative Models

Scott Richardson, Michael Otte, Michael Mozer, Amer Diwan, Dan Conners

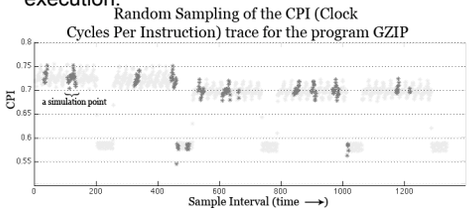
Designing a better microprocessor

- Accurate profiling tools, needed to design better microprocessors, are very slow.
  - Particularly if the microarchitectural design is emulated.
  - Benchmarking suites exist to profile a microprocessor (e.g., SPEC2000).
- Need to speed up microarchitectural analysis.



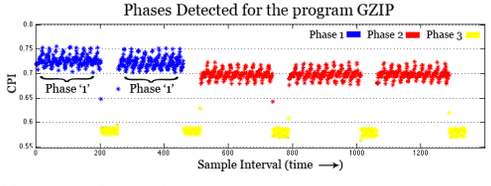
Simulation Points

- Perform cycle accurate emulation only for *simulation points*.
  - A **simulation point** is an interval of program execution.



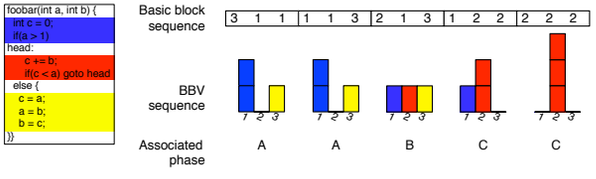
Can we do better?

- Strategically select simulation points.
  - Don't choose simulation points at random.
  - Exploit a program's *phase* behavior.
    - A **phase** is a distinct pattern of program behavior (e.g., CPI over time).



Discovering phases

- Infer phases from an executed basic block sequence.
  - Basic block trace is cheap to collect.
- Procedure
  - Periodically record Basic Block Vectors (BBV) as the program executes.
    - A **BBV** is a histogram summarizing each period of basic blocks.
  - Cluster BBVs into phases.



Related Work

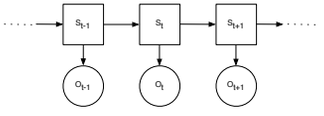
- SimPoint (Sherwood, Perelman, Hamerly, et al.)
  - *k*-means clustering of BBVs
- Multinomial mixture model (MMM)
  - Probabilistic interpretation of BBV distribution

Limitation with current models

- SimPoint
  - Why treat a BBV as a continuous vector?
    A BBV is composed of discrete blocks.
  - A better interpretation is multinomial.
- MMMs
  - BBV is randomly projected in a way that disrupts its interpretation as a count vector.
- The BBV's temporal structure is ignored.

Why care about time?

- Current phase constrains next phase.
  - i.e., if the observation is ambiguous, the temporal constraint can achieve greater certainty about the phase.
- We use Hidden Markov Models to explicitly capture sequential structure of the BBVs.



Applying an HMM

- Observation at each time step corresponds to a BBV.
- Hidden state corresponds to a program's latent phase.
- Alternative HMM emission distributions: multinomial and a Gaussian.

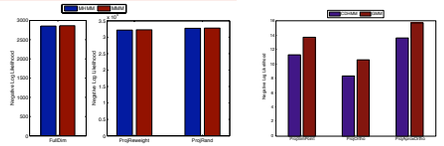
	temporal	atemporal
multinomial	multinomial HMM	MMM
Gaussian	CDHMM	GMM

Our approach

Our space of models varies over 5 dimensions

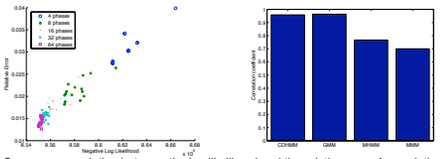
1) HMM vs. mixture model
   - Are temporal constraints beneficial?
2) Multinomial vs. Gaussian output
   - What is the better interpretation of a BBV?
3) BBVs vs. Reduced Dimensionality BBVs
   - Does random projection corrupt the data?
4) Techniques for selecting simulation points
5) Techniques for estimating runtime profile

Likelihood based model evaluation



The BBV sequence is not significantly more likely under the HMMs than the mixture models.

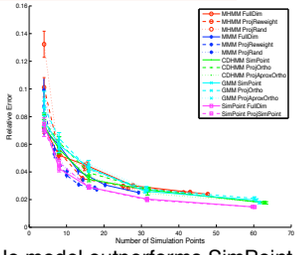
Model likelihood correlates with error in runtime profile estimates



Spearman correlation between the log likelihood and the relative error. A correlation of 1 indicates strong correlation. The model space is integrated over all dimensions except the model type.

The simulation point selection technique MaxLikelihood is most strongly correlated.

Estimating the runtime profile: A comparison of the algorithms



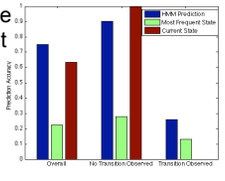
Relative CPI estimate error, averaged over the set of benchmarks, as a function of the number of simulation points in a model. The multinomial and Gaussian models are trained with their respective data representations.

No model outperforms SimPoint.

The phase must be unambiguous given the observation

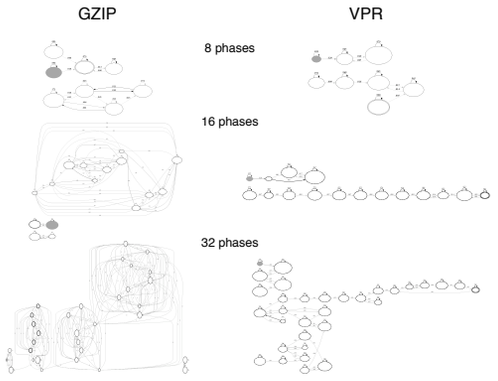
There *is* temporal structure in the data

- HMMs transition to a different phase after 36% of the observations.
- The HMM predicts the next phase more accurately than an algorithm that assumes:
  Next phase ← Current phase
  Next phase ← Most frequent



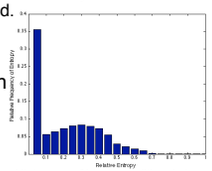
The temporal structure of programs

- The HMM's transition distribution allows us to analyze structure in programs.



Code vs. data dependent phases

- Code dependent phases
  - A given basic block is associated with a single phase.
- Data dependent phases
  - A given basic block may behave differently based on the context in which it is invoked.
- 36% of the phases are heavily code dependent.
- However, a significant portion of the basic blocks can appear in multiple phases.



Histogram of block-conditional phase entropy for all CDHMM models.

Conclusions

- SimPoint (*k*-means) does very well.
- To our surprise, the temporal constraints of the HMM did not improve the quality of the models.
- Dimensionality reduction does not have a significant impact on model performance.
- Phases are both code and data dependent.

Future work

- If observations are made more quickly, the temporal context may become more important.
  - Additionally, this requires that a smaller percentage of the program be functionally simulated.