Computer Architecture
计 算 机 体 系 结 构

Lecture 7. Performance Evaluation
第七讲、性能分析与评测简介

Chao Li, PhD.
李超 博士

SJTU-SE346, Spring 2019

Review

- Physical components of a disk
- Design good drive interfaces
- ATA/SCSI
- Data striping and data mirroring
- DAS, NAS, SAN
- Flash memory, SLC/MLC

Outlines

- Quantitative Analysis
- Analytical Evaluation
- Architecture Simulation
- Workload Design

Evaluation Metrics

Typical metrics

- Events frequency
- Interval durations
- Parameter sizes

Characteristics of good metrics

- Allows unambiguous comparison
- Allows one to develop models
- Meaningful and easy to estimate

A general classification

- Higher is better metrics , e.g., productivity, speed
- Lower is better metrics, e.g., responsiveness, cost
- Nominal is better metrics, e.g., utilization

Amdahl's Law

- Suppose that
  - The fraction f of a program is parallelizable
  - The other $1\text{-}f$ is purely sequential
  - Parallelizable part has linear speedup
- Then the effective speed with n processors, is given by:

$$S(n) = \frac{S(1)}{1 - f + f / n} \qquad \lim_{n\to\infty} \frac{S(n)}{S(1)} = \frac{1}{1 - f}$$

The theoretical speedup is always limited by the part of the task that cannot benefit from the improvement

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Calculating CPI

$$CPI = \sum \frac{IC_i \times CPI_i}{Instruction\ count}$$

$$CPU\ time = \left(\sum IC_i \times CPI_i\right) \times Clock\ cycle\ time$$

Example:

Suppose we have made the following measures:

Frequency of FP operations = 25%

Average CPI of FP operations = 4.0

Average CPI of other instructions = 1.33

Then:

CPI original = $(4\times25\%)+(1.33\times75\%) = 2.0$

Memory Performance Analysis

$$\frac{Misses}{Instruction} = Miss\ rate \times \frac{Memory\ accesses}{Instruction}$$

$$Average\ memory\ access\ time = Hit\ time + Miss\ rate \times Miss\ Penalty$$

$$Average\ memory\ access\ time = Hit\ time_{L1} + Miss\ rate_{L1} \times Miss\ Penalty_{L1}$$

$$Miss\ Penalty_{L1} = Hit\ time_{L2} + Miss\ rate_{L2} \times Miss\ Penalty_{L2}$$

Outlines

- Quantitative Analysis
- Analytical Modeling
- Architecture Simulation
- Workload Design

Little's Law

- Suppose that in a stable system:
  - λ = the average job arrival rate (number per unit time)
  - W =The average waiting time in the system for an item
- Then the average length in the queuing system is:

$$L = \lambda W$$

Example:

Assume requests arrive at 100 per minute and stay on average of 30 seconds. This means the average number of requests in the buffer is L= 100 × 0.5 = 50

Another Example: BW Estimation

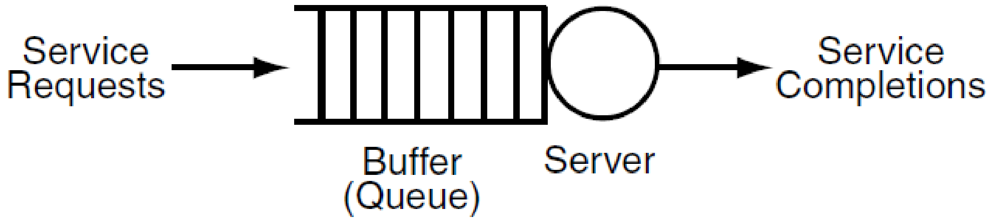
- Factors that affect BW
  - Cache miss
  - Data prefetch

$$\text{BW} = \frac{outstanding\ requests}{average\ \text{latency}\ of\ a\ single\ request}$$

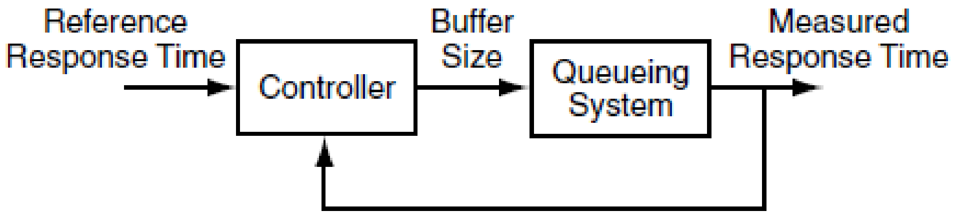
$$= \frac{(LLC\ misses + prefetches) * \#threads * cache\ line\ size}{access\ latency + \text{contention}\ latency}$$

Usefulness of Little's Law in Practice

- Queueing systems are widely used to model the performance of computing systems

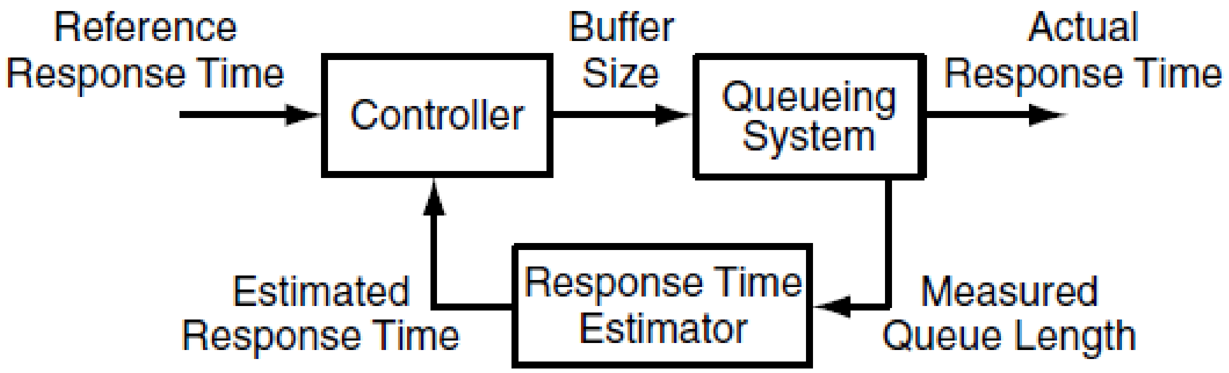


- Measurement of response time can be challenging
  - Could be impractical or very time-consuming



Usefulness of Little's Law in Practice (Cont'd)

- Instantaneously evaluate system response
  - An indirect estimation approach
  - The queue length can be measured instantaneously
  - Estimate response time using the Little's Law



Estimate of response time using a transducer

On The Evaluation of Energy and Power

- Dynamic Power
  - C: load capacitance
  - V: supply voltage (typically less than 1.5 V)
  - A: activity factor (a fraction between 0 and 1)
  - f : operating frequency

$$\boldsymbol{P} = \mathbf{a} \times \boldsymbol{CV^2Af}$$

- Frequency typically scales linearly with voltage
  - *i.e.*, $f = kV$, Thus, the combined V^2f portion of the dynamic power equation has a cubic impact on power dissipation

Server Speed vs. Server Power

- The CPU power-to-frequency relationship is cubic
- Server speed typically depends on frequency
- A cubic relationship between speed and power?
- Experimental results show that the power-to-speed relationship is almost linear
  - Due to limited voltage/frequency levels
  - Not applied to many components in the server

Estimating Computer Power

- A liner model of computer power is widely used:
  - P_{dyn} : dynamic power
  - P_{idle} : idle power (can be seen as static power)
  - U: system utilization, an indicator of system activity

$$P_{total} = P_{dyn} \cdot U + P_{idle}$$

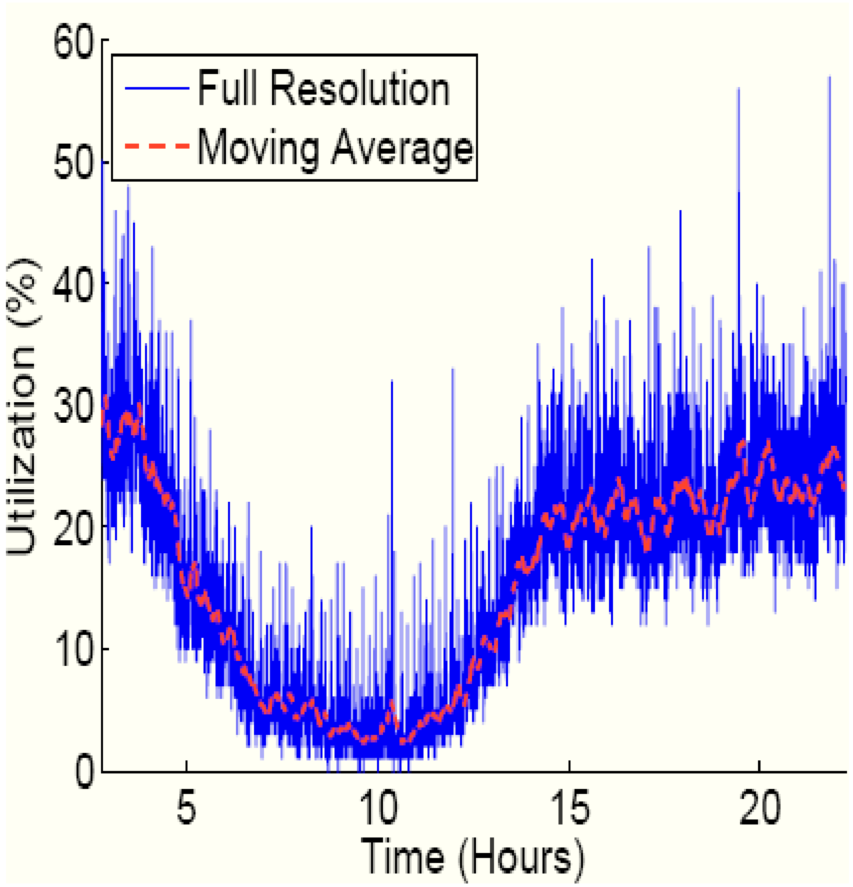
Example:

Assume the average CPU utilization is 30%, the nameplate power (maximum power demand) is 180W and the idle power is 100W. This means the average power is $P = 100 + (180-100) \times 0.3 = 124W$

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

Estimating Computer Power (Cont'd)

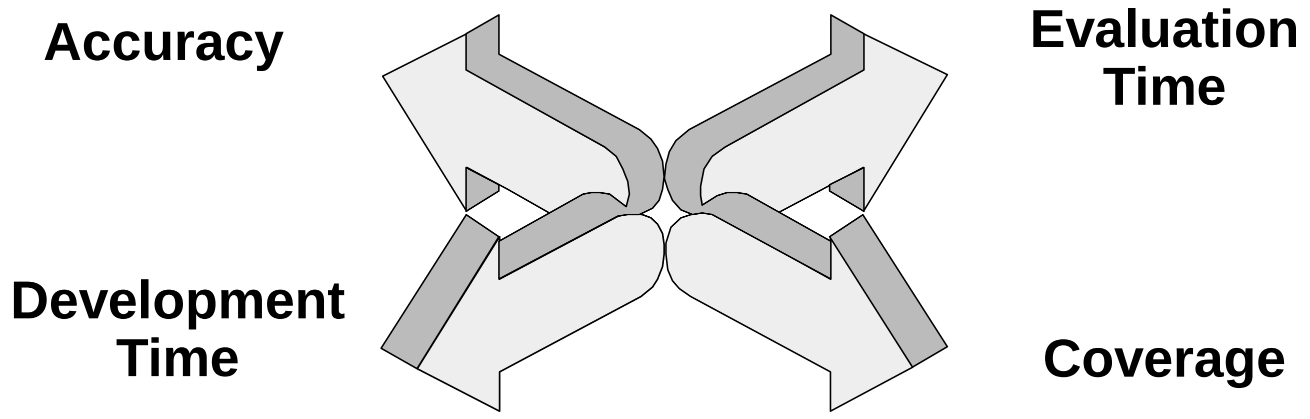
- **Pros:**
  - Gives full system average power evaluation
  - Use server/CPU utilization as machine-level activities
  - Fairly good estimation when server numbers are large
- **Cons:**
  - Fail to predict peak power



Outlines

- Quantitative Analysis
- Analytical Evaluation
- Architecture Simulation
- Workload Design

Computer Architect's Toolbox



- How accurate is the simulation model
- How long does it take to run a simulation
- How long does it take to develop the simulator
- What fraction of the design space can we explore

Functional Simulation

- Models only the functional characteristics of ISA
  - No timing issue is considered
  - Basically a instruction-set emulator
- One important application is validation of the design
  - Rather than evaluating system performance characteristics
- It can also generate instruction and address traces
  - Can be used as inputs to other simulation tools !

Trace-Driven Simulation

- Takes program instruction and address traces into a detailed microarchitecture timing simulator

- Separates functional simulation from the timing simulation

- Disadvantages:
  - The need to store the trace files (can be huge)
  - Will not accurately model the effects along mis-predicted paths

Mis-predicted instructions are **nullified** – they do not show up in a trace file generated via function simulation, although they may affect cache and/or predictor contents

Execution-Driven Simulation

- The de-facto simulation approach today
- Combines functional with timing simulation
- Achieves higher accuracy than trace-driven simulation
- At the cost of increased development/evaluation time
- Examples:
  – SimpleScalar, GEMS, Simics, M5, PTLSim …

Simulation Approach Comparison

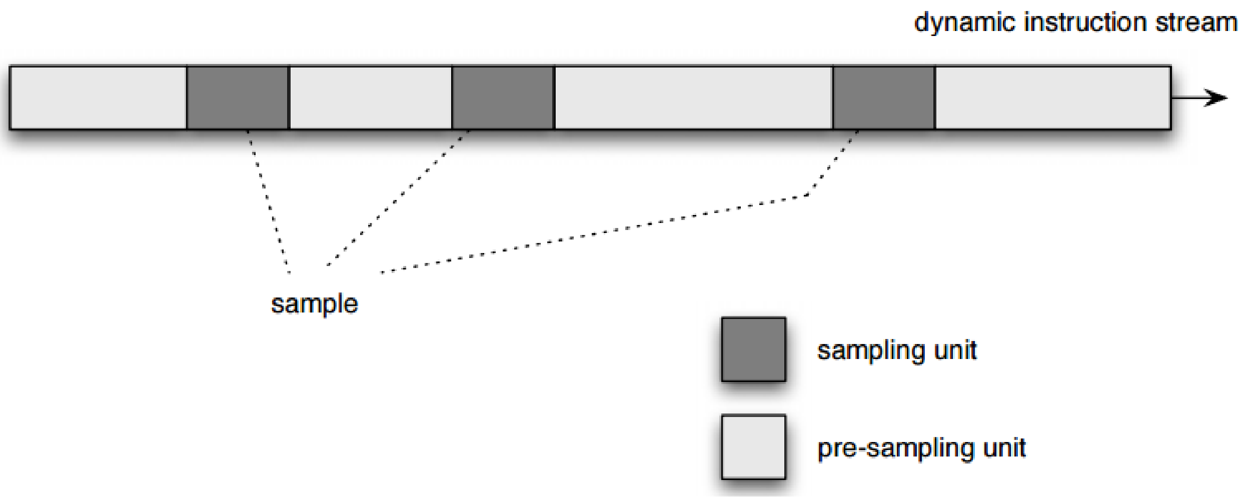
	Function Simulation	**Trace-Driven Simulation**	**Execution-Driven Simulation**
Development Time	Excellent	Poor	Very Poor
Evaluation Time	Good	Poor	Very Poor
Accuracy	Excellent	Very good	Excellent
Coverage	Poor	Excellent	Excellent

- Full system simulation
  - Trace-driven simulation and execution-driven simulation

- More simulator/tools?
  - Check: **http://pages.cs.wisc.edu/~arch/www/tools.html**

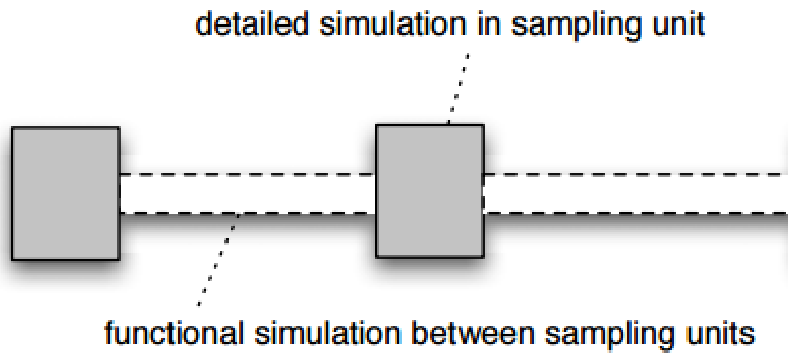
Simulation Acceleration

- Cycle-accurate simulation is slow
  - Simulates a microarchitecture on a cycle-by-cycle basis
- Sampled Simulation
  - Only a small fraction of the total dynamic instruction count, are simulated in a cycle-accurate manner.
  - requires accurately provide a sampling unit's architecture starting image (ASI) , i.e., register and memory states, etc.

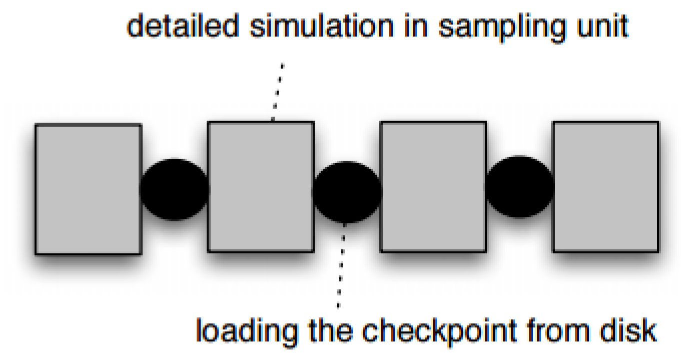


Simulation Acceleration (Cont'd)

- Fast-Forwarding
  - Constructs the architecture state through functional simulation
  - Switch between functional and execution-driven simulation
- Checkpointing
  - Stores the register/memory state prior to a sampling unit
  - Just need to load the data and update simulator during runtime


detailed simulation in sampling unit
functional simulation between sampling units

Fast-forwarding through functional simulation


detailed simulation in sampling unit
loading the checkpoint from disk

Checkpointing

Outlines

- Quantitative Analysis
- Analytical Evaluation
- Architecture Simulation
- Workload Design

Workload

- **Workload**: the mixture of programs and operating system commands that users run on a machine

- Workloads are preferably to be
  - Real Applications
  - Modified Applications

- Naive programs can be used in some circumstances
  - Kernels
  - Toy Benchmarks
  - Synthetic Benchmarks

Benchmark Suite

- Benchmark Suite:
  - put together collections of benchmarks to quantitatively measure the performance of systems with a variety of applications

- Popular benchmark suites in the past
  - SPEC 95, SPEC CPU 2000, SPEC CPU 2006 …
  - TPC-A, TPC-C, TPC-W…
  - LINPACK

- Example of emerging benchmarks suites today
  - CloudSuite 3.0 (http://cloudsuite.ch/)
  - Rodinia (http://www.cs.virginia.edu/~skadron/wiki/rodinia/)
  - Hibench (https://github.com/intel-hadoop/HiBench)

Example: CloudSuite Overview

- **Data Analytics**
  - Naive Bayes classifier on a Wikimedia dataset
- **Data Caching**
  - A Twitter caching server using a twitter dataset
- **Data Serving**
  - Yahoo! Cloud Serving Benchmark (data store application)
- **Graph Analytics**
  - Perform graph analytics on large-scale datasets
- **In-Memory Analytics**
  - collaborative filtering algorithm in-memory with user-movie rating data
- **Media Streaming**
  - A streaming server for hosted videos of various lengths and qualities
- **Web Search**
  - relies on the Apache Solr search engine framework
- **Web Serving**
  - Traditional web services with dynamic and static content delivery

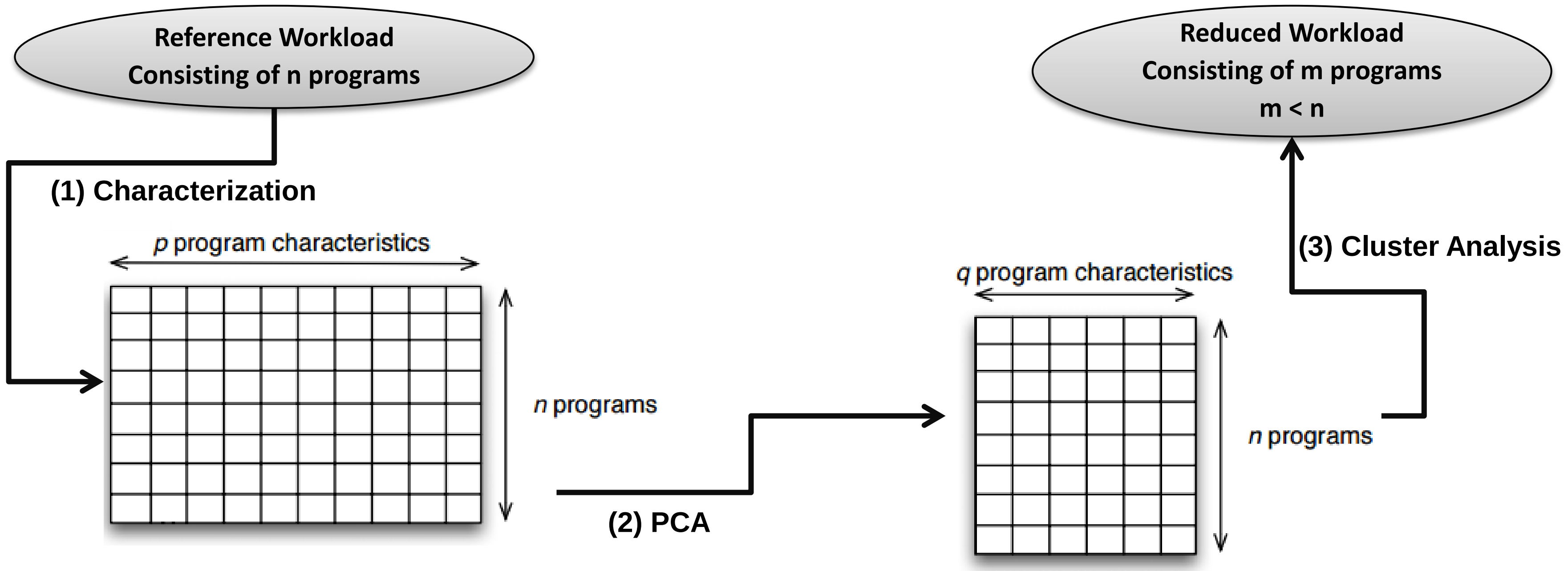
Workload Design

- Workload Characterization
  - To understand the behavior of the various benchmarks
  - Through various hardware monitors, simulators, etc.
- Principal Component Analysis (PCA)
  - A mature statistical data analysis technique
  - Transforms a number of possibly correlated variables into a small number of uncorrelated principal components
- Cluster analysis
  - Pick up most diverse benchmarks, e.g. through K-means

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

Workload Design (Cont'd)

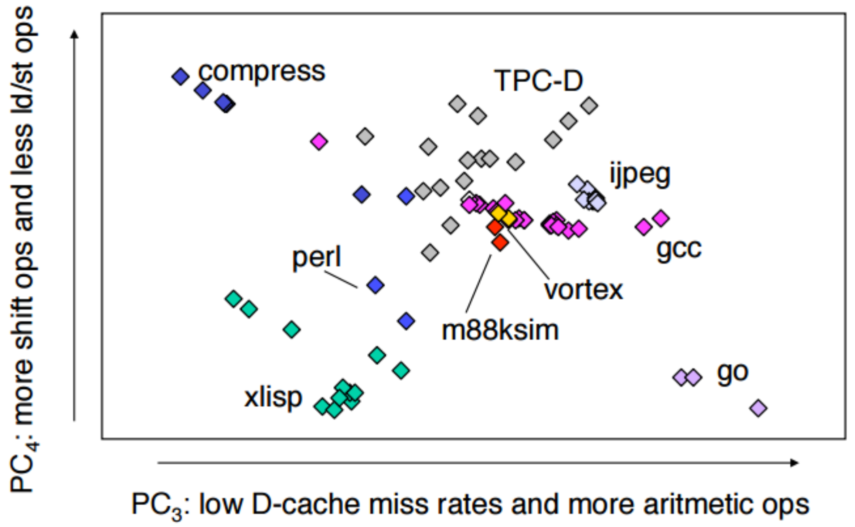
- Determine a reduced but representative workload



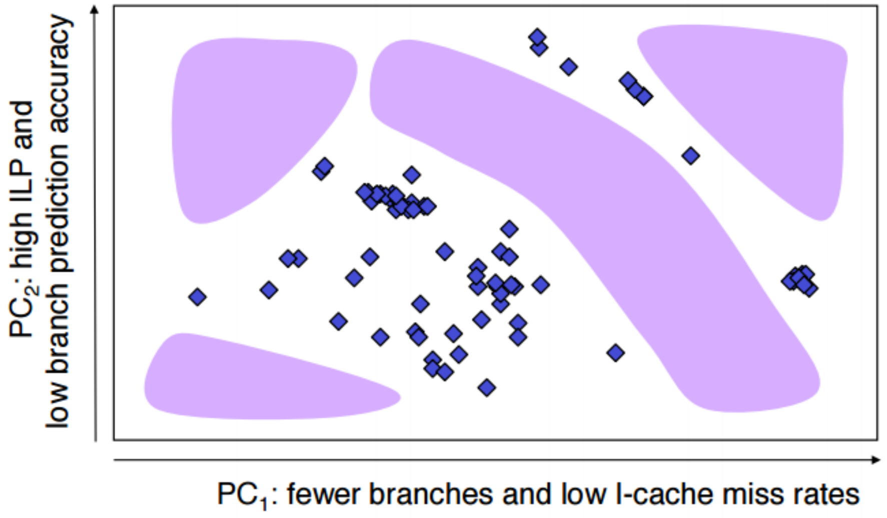
Schematic overview of the PCA-based workload reduction method

Applications of PCA-Based Design

- Workload Analysis
  - To visualize the workload space
  - Reason about how benchmarks differ from each other
- Workload Reduction
  - Build a small set of representative benchmarks



Example PCA space as a function of two principle components



Finding regions not covered by a benchmark suite

Multi-Program Benchmark

- Single-Program Benchmark is relatively simple:
  - A benchmark is a program plus a specific input
- Multi-Program Benchmark can be complex
  - Each program may have a different runtime
  - Different interactions depending on how they align with each other
- Issues with Multi-program benchmark
  - Load imbalance
  - Resource contention (program alignment matters)

Multi-Program Benchmark Example:

A multi-program benchmark can be described as a 4-tuple

$$\langle \mathcal{B}, \mathcal{T}, \mathcal{F}, \mathcal{S}_0 \rangle.$$

- ***B* -** A set of single-program benchmarks
- ***T* -** The terminating condition for the benchmark
- ***F* -** The selection function with decides which member of ***B*** to run on a particular core when that core becomes idle
- ***S* -** the initial state for ***F***

Summarize Performance

- SPECRatio:
  - normalize execution times to a reference computer

$$SPECRatio = \frac{Execution\ Time_{reference}}{Execution\ Time_{rated}}$$

$$\frac{SPECRatio_A}{SPECRatio_B} = \frac{Execution\ Time_B}{Execution\ Time_A} = \frac{Performance_A}{Performance_B}$$

- The use of geometric mean to calculate average
  - SPECRatio is a ratio rather than absolute execution time

Summarize Performance

- Chooses the appropriate average
  - depending on how the metric is computed
- Harmonic mean is meaningful when:
  - The metric is obtained by dividing A by B
  - A is weighed equally among the benchmarks

$$\frac{\sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} B_i} = \frac{n \cdot A}{\sum_{i=1}^{n} B_i} = \frac{n}{\sum_{i=1}^{n} \frac{B_i}{A}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{A/B_i}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{A_i/B_i}} = HM(A_i/B_i)$$

- Arithmetic mean is meaningful when:
  - The metric is obtained by dividing A by B
  - B is weighed equally among the benchmarks

$$\frac{\sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} B_i} = \frac{\sum_{i=1}^{n} A_i}{n \cdot B} = \frac{1}{n}\sum_{i=1}^{n}\frac{A_i}{B} = \frac{1}{n}\sum_{i=1}^{n}\frac{A_i}{B_i} = AM(A_i/B_i)$$

Example: System Throughput

- Normalized Turnaround Time (NTT)

$$NTT_i = \frac{T_i^{MP}}{T_i^{SP}}$$

Multiprogram execution time

Single-program execution time

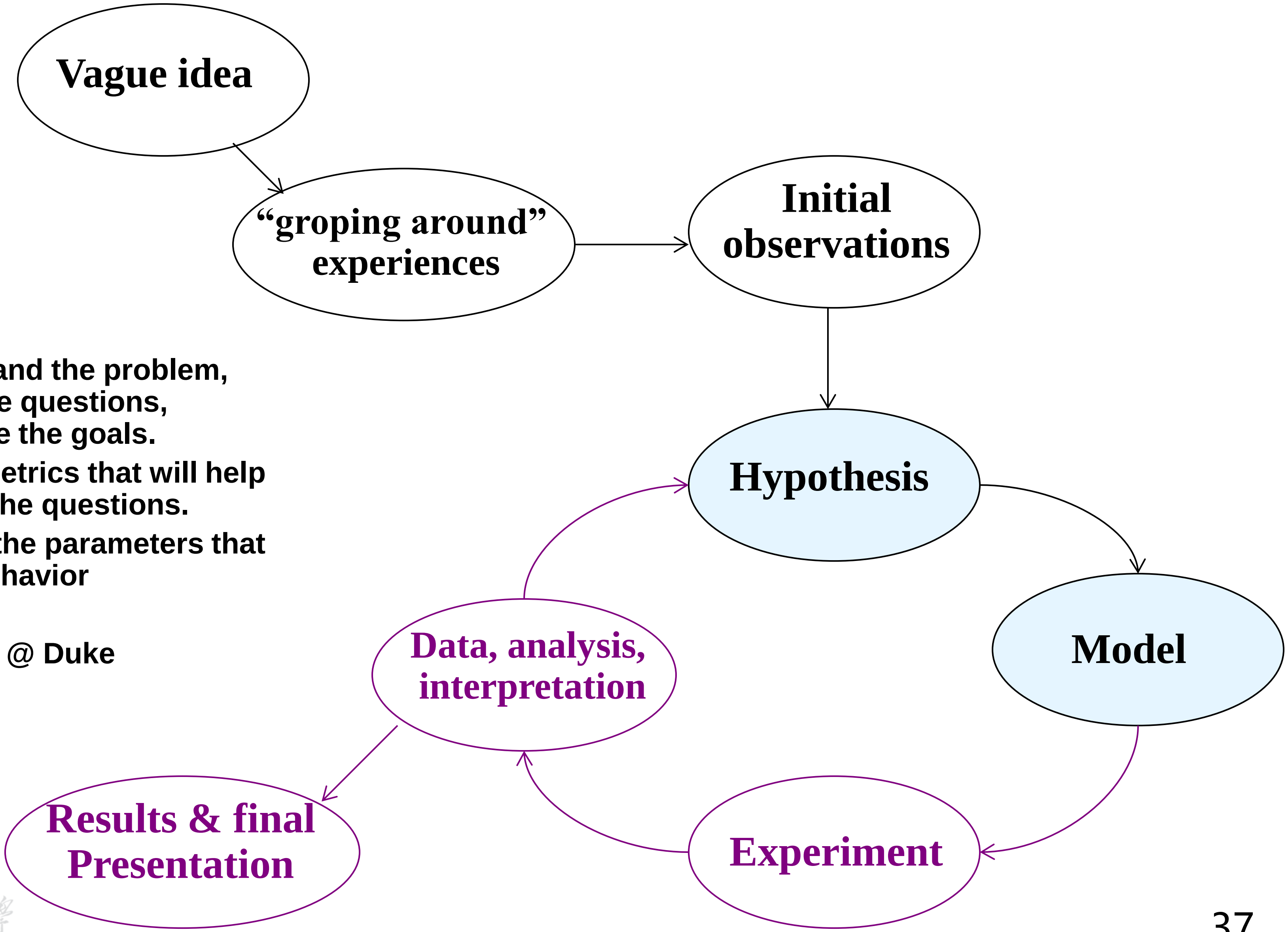
- Average Normalized Turnaround Time (ANTT)

$$ANTT = \frac{1}{n}\sum NTT_i$$

- IPC Throughput:

$$IPC\ Throughtput = \frac{1}{n}\sum IPC_i$$

Experimental Lifecycle


Vague idea
"groping around" experiences
Initial observations
Hypothesis
Model
Experiment
Data, analysis, interpretation
Results & final Presentation

1. Understand the problem, frame the questions, articulate the goals.
2. Select metrics that will help answer the questions.
3. Identify the parameters that affect behavior

By. Carla Ellis @ Duke

Summary

- Amdahl’s Law
- Calculating CPI
- Analyzing memory access time
- Little’s Law
- Estimating server power
- Trace-/Execution- driven simulation
- Simulation acceleration
- Concepts of workload characterization
- Multi-programmed workload

References

- 课本内容：J. Hennessy, D. Patterson. Computer Architecture, Fifth Edition: A Quantitative Approach.
  - Chapters: 1.5, 1.8, 1.9, 2.1, 2.2

- 其它参考：L. Eeckhout, 《Computer Architecture Performance Evaluation Methods 》，Synthesis Lectures on Computer Architecture.
  - Chapters: 2, 3, 5.2, 5.5, 5.6, 6.1, 6.2

Exercises

- Consider a processor that has the following property
  - ***F = 2.0GHz*** is the full frequency without frequency scaling
  - **U = 60%** is average utilization rate (the proportion of non-idle state)
  - It can enable Turbo Boost with a duty cycle of **D = 0.2** (the fraction of active period)
  - Under Turbo Boost, the processor could temporarily increase the frequency by **B = 10%.**

  Suppose **T = 2s** is the time the program would spend without frequency scaling. What is the estimated execution time when the Turbo Boost is enabled?

- Describe effective simulation method and metrics to evaluate the following systems
  - **a**. A 1000-processor massively parallel computer system
  - **b**. The performance of an ATM-based LAN system
  - **c**. A battlefield-communication system
  - **d**. A cellular network in a large city

- What are the emerging workloads that you know?