Research Planning & Methodological Approaches from a Computer Architect’s Perspective

Per Stenström

Department of Computer Engineering, Chalmers, Göteborg, Sweden

http://www.ce.chalmers.se/~pers

Some Biographic Data…

• Who Am I?
  – Professor of Computer Architecture since 1995

• What is Computer Architecture?
  – Engineering discipline about computer design
  – Was Art but became an engineering discipline when the field developed sound engineering methodologies (more about this later)
Agenda

- Research philosophy
- Research planning issues
- Formulation of research questions: Pitfalls
- Methodology issues
- Illustrative example: performance prediction of complex systems

My Research Philosophy

At least two driving forces:

- Issues to tackle: High intellectual appeal and societal relevance
- Methodology: Inspiration from experimental physics
Research Planning: Route

- Defining a Methodology (Experimental plan)
- Analysis of Results
- Framing the Reality
- Formulation of Research Question

Example of a Framework

In this example, the framework is set up to reason about properties of different architectures.
Example Research Question

Which architecture is best?
Leads to a number of issues
  – What does best mean (performance, power, …)?
  – Under what assumptions?
    • Applications, middleware, compiler etc
    • Hardware technology
    • Organizational parameters (microarch, cache hierarchy etc.)
  – Generalization of results

Note: research results must be *reproducible*

Formulation of Research Questions: Some Pitfalls

• Is it scientifically relevant enough? (incremental vs ground-breaking research)
• What is the long-term impact?
• What are the risks involved?
• What is the risk of getting scooped?
• What are the methodological implications?
Example Contributions

- Innovative design principles
  (microarchitectural mechanisms, cache algorithms etc.)
- Quantitative characterizations
- Quantitative principles
- Classification methodologies
- Design methodologies
- Frameworks and models

Understanding Performance Bottlenecks in Complex Computer Systems
**Transaction-Oriented Systems**

- Short response time and high throughput are important
- Technologies involved:
  - Multiprocessors
  - Commodity OS
  - Database engines
  - User apps
- Complex interaction between several layers of hw and sw

*How do we identify performance bottlenecks in this complex system?*

**Outline**

- Complete System Level Simulation
- Methodology overview
- One case study in detail:
  - Cache memory bottlenecks in Decision Support Systems (DSS)
- Other case studies in brief
- Concluding remarks
System Simulation Approach

- **Pros:** Complete observability w.r.t performance bottlenecks
- **Cons:** Slowdown of >50; often has to scale down workload

Methodology Overview

Goal: Understanding how system parameters affect performance -- not prediction of absolute performance!

Key methodology steps:
- Design of system model (leverage on existing bldg blocks)
- Verification
- Validation: if system exists! Note: possible to study future non-existing systems
- “What if” experiments
- Measurements
- Analysis
Case Study:

Understanding Memory System Bottlenecks in Decision Support Systems

System Timing Model

• Processor execution is accurately delayed

• Correct interleaving of events in the system!!!
The Workload: A Decision-Support System

Compiles a list of matching entries in several database relations

Question: Will moderately sized caches suffice for huge databases?

Simulation Results

- Miss rates for instr., priv. and meta data rapidly decay (128 Kbytes)
- Miss rate component for database data is low

What about Tera-byte sized databases?
Methodological Approach

Challenges:
- Not feasible to simulate huge databases
- Need source code: we used PostgreSQL and MySQL

Approach:
- Analytical model using
  - parameters that describe the query
  - parameters measured on downscaled query executions
  - system parameters

Footprints and Reuse Characteristics in DSS

- MWS: instructions, private, and meta data
  - can be measured on a downscaled simulation
- DWS: all tuples accessed at lower levels
  - can be computed based on query composition and prob. of match
Analytical model—an overview

- **Goal:** Predicts miss rate versus cache size for fully-assoc. caches with a LRU replacement policy for single-proc. systems

  Number of cold misses:
  - size of footprint/block size
    - $|\text{MWS}|$ is measured
    - $|\text{DWS}|$ computed based on parameters describing the query (size of relations, probability of matching a search criterion, index versus sequential scan, etc)

  Number of capacity misses for tuple access at level $i$:
  - $C_M(i, \frac{1-C_0}{1-C_0})$ if $C_0 < \text{Cache size} < |\text{MWS}|$
  - $|\text{MWS}| - C_0$ if $|\text{MWS}| \leq \text{Cache size} < |\text{MWS}| + |\text{DWS}|$

  Number of accesses per tuple access: measured
  Total number of misses and accesses: computed

Model Validation

- **Goal:**
  - Prediction accuracy for queries with different compositions
    - Q3, Q6, and Q10 from TPC-D
  - Prediction accuracy when scaling up the database
    - parameters at 5Mbyte used to predict at 200 Mbytes databases
  - Robustness across database engines
    - Two engines: PostgreSQL and MySQL

![Graph](chart.png)
Our Findings

- **MWS**: footprint of instructions and private data to access a single tuple
  - typically small (< 1 Mbyte) and not affected by database size
- **DWS**: footprint of database data (tuples) accessed across consecutive invocations of same scan node
  - typically small impact (~0.1%) on overall miss rate

---

Model Predictions: Miss rates for Huge Databases

- The main qualitative conclusion holds:
  - Cache performance is good for DSS even for huge databases!
Other Case Studies in Brief

- Performance tuning of software
  - Esp. understanding perf. bottlenecks in parallel software
- Performance tuning of compilers
  - Esp. w.r.t interaction with memory system
- Performance bottlenecks in OS
  - Lock performance and interaction with cache coherence
- Analysis of energy-efficiency of computer systems

Concluding Remarks

- Performance is a function of a complex interaction between several layers of hw and sw
- System simulation is a viable methodology
- Important application areas:
  - Performance analysis/tuning of software (app, OS, compilers)
  - Comparative perf. analysis of various design alternatives at all levels (app, OS, hw-platform)