COMP 190-088: Systems Performance Analysis

Workloads

Jasleen Kaur

Department of Computer Science
The University of North Carolina at Chapel Hill

Spring 2005

Workloads: Outline

- **Popular Workload Types**
- Guidelines for Selecting Workload
- Workload Characterization

Real vs Synthetic Workloads

- Real workloads
  - Those observed on a real system
  - Not suitable for analysis
    - Can not be repeated or modified
    - May contain sensitive information
- Synthetic workloads
  - Generated synthetically
    - With characteristics similar to real workloads
  - Suitable for analysis
    - Can be repeatedly applied in a controlled manner
    - Workload can be easily modified without affecting operation
  - No real-world data files need be involved
    - May be too large to handle
    - May contain sensitive information
  - Can be easily ported to other systems
    - Usually small in size
  - May have built-in measurement capabilities

Instruction Mixes

- Addition instruction
- Instruction mixes
  - Designed by observing relative frequencies of various instructions on real systems
  - Example: Gibson mix
    - Measured memory cycle times, addition and multiplication times
    - Developed in 1959

Load and Store	31.2
Fixed-point Add and Subtract	6.1
Compares	3.8
Bracnhes	16.6
Floating Add and Subtract	6.9
Floating Multiply	3.8
Floating Divide	1.5
Fixed-point Multiply	0.6
Fixed-point Divide	0.2
Shifting	4.4
Logical, And, Or	1.6
Instructions not using registers	5.3
Indexing	18.0
TOTAL	100.0

Problems with Instruction Mixes

- Today's computers provide many complex classes of instructions
  - Not reflected in traditional mixes
- Instruction times are highly variable
  - Depend on addressing modes, cache hit rates, pipeline efficiency, interference from other devices
  - Mixes do not reflect virtual addressing facilities (page transition tables
- Instruction times also vary according to parameter values
  - Frequency of 0's, distribution of 0's in a multiplier, number of times a conditional branch is taken
- Only measure processor speed
  - May not reflect system performance if processor is not bottleneck

Utility of Instruction Mixes

- Useful in estimating time required to execute key algorithms in application and system programs
- Provide a single number for comparison of similar architectures
  - MIPS, MFLOPS

Kernels

- A set of instructions which constitute a higher-level functions/ services provided by the processors
- Popularized by researchers
  - Not based on actual measurements of systems
  - eg, Sieve, Ackermann's function, Tree searching, Matrix Inversion, Sorting
- Limitations:
  - Similar disadvantages as instruction mixes
    - Variable instruction times, newer functions not reflected
  - Do not typically make use of I/O devices
    - Do not reflect total system performance

Synthetic Programs

- Synthetic workloads that incorporate I/O
  - Simple loops that make a specifiable number of service requests interspersed with computation
    - I/O requests, process creation, forking, memory allocation, …
  - Typically written in high-level languages

```
GetTime(CPU1, Elapsed1)
Do N1 file reads
Do N2 computations
Do N3 file writes
GetTime(CPU2, Elapsed2)
CPUTime=CPU2 - CPU1
ElapsedTime = Elapsed2 - Elapsed1
```

- Pros:
  - Incorporate I/O
  - Can be easily developed and distributed
  - Have built-in measurement capability
- Cons:
  - Not representative and too small
  - Not suitable for multi-user environments
    - Loops may create synchronization and impact observed performance

Application-specific Benchmarks

- Consist of a representative subset of functions to be performance by the application
- Make use of all system resources
  - Processors, I/O devices, networks, databases

Popular Benchmarks

- Seive
  - Used to compare microprocessors, PCs, and high-level languages
  - Find all prime numbers below N
  - Written in high-level languages
- Ackermann's function
  - Used to assess efficiency of procedure-calling mechanisms
  - Ackermann(3,n) = $2^{n+3} - 3$
    - Computed recursively
  - Metrics:
    - Average execution time
    - Number of instructions executed per call
    - Amount of stack space required for each call
      - Depends on maximum depth of procedure call

Popular Benchmarks

- LINPACK
  - Used to gauge performance of scientific applications
  - Programs that solve dense systems of linear equations
    - High percentage of floating-point additions and multiplications
  - Metric: MFLOPS, execution rate
- Dhrystone
  - Represents system programming env with many procedure calls
    - Low nesting depth of function calls
    - Low number of instructions per function call
    - Large fraction of time spent in string copying and comparison
  - Popular measure of integer performance
    - No floating-point or I/O operations

Popular Benchmarks

- Debit-credit Benchmark
  - Standard for comparing transaction processing systems
    - Represents a distributed banking network
- Metric: price-performance ratio
- Newer variants: TPC-A, TPC-B
  - http://www.tpc.org/information/benchmarks.asp

Popular Benchmarks

- SPEC Benchmark Suite
  - Systems Performance Evaluation Cooperative
    - Non-profit corporation that develops a standardized set of benchmarks
  - http://www.spec.org/benchmarks.html
  - Benchmarks available for:
    - CPU
    - Graphics/Applications
    - Java Client/Server
    - Mail Servers
    - Network File System
    - Web Servers

Workloads: Outline

- Popular Workload Types
- **Guidelines for Selecting Workload**
- Workload Characterization

Common Problems with Workload Selection

- Unrepresentative workload
  - Workload should represent actual usage of the systems
  - Wrong workload can lead to inaccurate conclusions
    - Benchmarking games
- Ignoring corner-cases
  - Workload should incorporate average-case as well as infrequent request types

Guidelines for Workload Selection

- Should be complete
  - eg, include floating-point operations for studying ALUs
- Should serve the purpose of study
  - eg, text-editing workloads not suitable for mainly-graphics editor
- Should be representative
  - Should match real usage in arrival rate, resource demands, usage profile, …
- Should be timely
- Should be repeatable
  - Should not make too many random choices
- Should allow little impact of external components
  - eg, programs with I/O not suitable for comparing processors
- Should represent appropriate loading level
  - Best-case, worst-case, average-case
- Should use the right level of detail

Level of Detail

- Most frequent request
  - Eg, addition instruction for ALUs
  - May not provide enough information about the system
- Frequency of request types
  - Eg, instruction mixes
  - Not context-sensitive
- Time-stamped sequence of requests
  - Too detailed
- Average resource demand
  - Not representative if variability is high
- Distribution of resource demands
  - Useful if distribution is different from that used in modeling

Workloads: Outline

- Popular Workload Types
- Guidelines for Selecting Workload
- **Workload Characterization**

Workload Characterization

- Characterizing workload:
  - Studying real-user environments
  - Observing key characteristics
    - Services requested, user resource demands
  - Developing a workload model that can be used repeatedly
    - Describing a typical workload component
- Selecting workload components:
  - Each component should represent a homogeneous group
  - Selection should be guided by purpose of study
  - Components should occur at the SUT interface
    - eg, use CPU instructions for characterizing ALU workloads

Selecting Workload Parameters

- Parameters should depend on workload, rather than the system
  - eg, response time depends on the system!
  - "Number of service request" preferable to "amount of resource demanded"
    - eg, network mail session parameters:
      - ☺: Size of message, number of recipients
      - ☹: CPU time, number of network messages
- Performance-impacting parameters should be included
  - eg, protocol type may have no impact on packet forwarding time at a router
- Typical service-request parameters:
  - Arrival time, Type of request/resource demanded, Duration of request, Quantity of resource demanded

Summarizing Workload Parameters: Techniques

- Averaging
- Specifying Dispersions
- Single-parameter Histograms
- Multi-parameter Histograms
- Markov Models

Averaging

- Present a single number that summarizes the observed values of a parameter

  If x1, x2, …, xn are n observed values of a parameter,

  Arithmetic Mean = $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$

- Other averaging quantities
  - Mean, Mode, Geometric Mean, Harmonic Mean
- ☹:
  - Not suitable in the presence of large variability

Specifying Dispersions

- Variance = $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$
- Standard deviation = σ
- Others:
  - Range, 10- and 90-percentiles, Interquartile range (IQR), Mean Absolute Deviation (MAD)
- Coefficient of variance (COV) = σ/μ
  - Zero COV: μ describes the complete set
  - High COV: μ insufficient
    - Divide requests into low-variance classes
    - Use histograms

Using Low-variance Sub-classes

	All programs		Editor program	
Data	Average	COV	Average	COV
CPU Time	2.16 s	40.23	2.57 s	3.54
# of writes	8.20	53.59	19.74	4.33
Write Bytes	10.21 KB	82.41	13.46 KB	3.87
# of reads	22.64	25.65	37.77	3.73
Read Bytes	49.70 KB	21.01	36.93 KB	3.16

- Resource demands of several programs executed at several sites for several months
  - COV after combining programs is high
  - COV of "editor" program is low
- Clustering techniques useful for identifying low-variance sub-classes

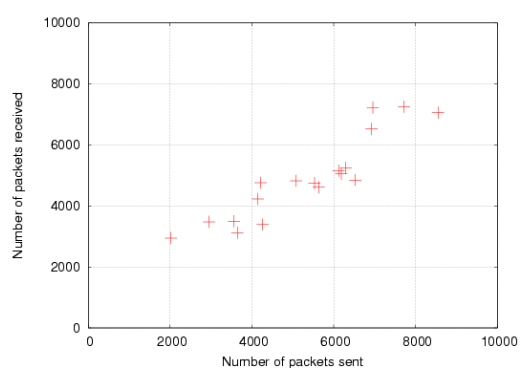
Single-parameter Histograms

- Histogram:
  - Depiction of the relative frequencies of various values of a parameter
  - Sub-ranges (buckets) used for continuous-valued parameters
- ☺:
  - Useful if variance is high
- ☹:
  - Too much detail
    - (# of buckets)*(# of parameters)
  - Ignores correlation among different parameters
    - eg, short jobs may use small CPU time and small number of disk I/O
      - Use of independent histograms for CPU time and I/Os may generate a job with short CPU time and large number of disk I/O
  - Ignores time-order of values
    - eg, analysis of cache-replacement policies

Multi-parameter Histograms

- N-dimensional histogram is used to describe the distribution of N workload parameters

Captures high correlation between the two parameters

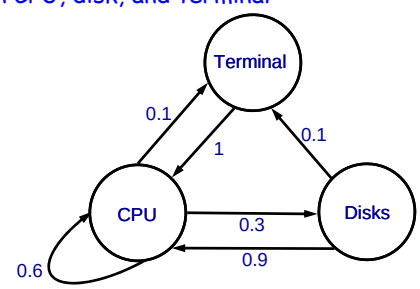


- ☹:
  - Difficult to plot (and visualize) joint histograms for more than two parameters
  - Too detailed
  - Ignores time-order of values

Markov Models

- Encodes number of requests of each type, as well as their order
  - If next request type depends ONLY on the last request, then requests follow a Markov Model
- Described in terms of a transition matrix
  - Probability of next state (or request type), given the current state
  - Eg, program's transition between CPU, disk, and terminal

From/To	CPU	Disk	Terminal
CPU	0.6	0.3	0.1
Disk	0.9	0	0.1
Terminal	1	0	0



- Used commonly in queuing analysis
- Characterization involves measuring transition probabilities on real system

Course Outline

- Selection of metrics
- Performance Evaluation Methodologies
- Workload selection
- **Measurements tools**
- Analysis and visualization of measured data
- System Modeling
- Simulations
- Case studies
- Distributed monitoring infrastructures
- PA in the Research and Industrial communities