Overview

Workload Types
- What is a Workload?
- Instruction Workloads
- Synthetic Workloads
- Standard Benchmarks
- Exercisers and Drivers

Workload Selection
- Considerations
- Example
What is a Workload?

- **Workload**: anything a computer is asked to do
- **Test workload**: any workload used to analyze performance
- **Real workload**: any observed during normal operations
- **Synthetic workload**: created for controlled testing
Real Workloads

- Advantage: represent reality
- Disadvantage: uncontrolled
  - Can't be repeated
  - Can't be described simply
  - Difficult to analyze
- Nevertheless, often useful for “final analysis” papers
  - E.g., “We ran system foo and it works well”
Synthetic Workloads

- Advantages:
  - Controllable
  - Repeatable
  - Portable to other systems
  - Easily modified
- Disadvantage: can never be sure real world will be the same
Instruction Workloads

- Useful only for CPU performance
  - But teach useful lessons for other situations

- Development over decades
  - "Typical" instruction (ADD)
  - Instruction mix (by frequency of use)
    - Sensitive to compiler, application, architecture
    - Still used today (GFLOPS)

- Processor clock rate
  - Only valid within processor family
Modern complexity makes mixes invalid

- Pipelining
- Data/instruction caching
- Prefetching

*Kernel* is inner loop that does useful work:

- Sieve, matrix inversion, sort, etc.
- Ignores setup, I/O, so can be timed by analysis if desired (at least in theory)
Synthetic Workloads

- Complete programs
  - Designed specifically for measurement
  - May do real or “fake” work
  - May be adjustable (parameterized)
- Two major classes:
  - Real-world benchmarks
  - Purpose-written exercisers

Concern is that real-world benchmarks represent only a specific problem.
Concern is that exercisers may not stress the system the same way as real programs (e.g., page faults).
Real-World Benchmarks

- Pick a representative application
- Pick sample data
- Run it on system to be tested
- Modified Andrew Benchmark, MAB, is a real-world benchmark
- Easy to do, accurate for that sample data
- Fails to consider other applications, data
Application Benchmarks

- Variation on real-world benchmarks
- Choose most important subset of functions
- Write benchmark to test those functions
- Tests what computer will be used for
- Need to be sure important characteristics aren’t missed
- Mix of functions must reflect reality
“Standard” Benchmarks

- Often need to compare general-purpose computer systems for general-purpose use
  - E.g., should I buy an AMD or Intel CPU?
  - Tougher: Mac or PC?
- Desire for an easy, comprehensive answer
- People writing articles may need to compare tens of machines
- Often need to make comparisons over time
  - Is this year’s PowerPC faster than last year’s Pentium?
    - Probably yes, but by how much?
- Don’t want to spend time writing own code
  - Could be buggy or not representative
  - Need to compare against other people’s results
- “Standard” benchmarks offer solution
Popular “Standard” Benchmarks

- Sieve, 8 queens, etc.
- Whetstone
- Linpack
- Dhrystone
- Debit/credit
- TPC
- SPEC
- MAB
- Winstone, webstone, etc.
- Postmark, IOzone, FileBench
- ...
Prime number sieve (Erastothenes)
  - Nested for loops
  - Often such small array that it's silly
8 queens
  - Recursive
Many others
Generally not representative of real problems
Whetstone

- Dates way back (can compare against 70’s)
- Based on real observed instruction frequencies
- Entirely synthetic (no useful result)
  - Modern optimizers may delete code
- Mixed data types, but best for floating-point
- Be careful of incomparable variants!
LINPACK

- Based on real programs and data
- Developed by supercomputer users
- Great if you’re doing serious numerical computation
Dhrystone

- Bad pun on “Whetstone”
- Motivated by Whetstone’s perceived excessive emphasis on floating point
- Dates to when µp’s were integer-only
- Still somewhat popular in PC world
- Again, watch out for version mismatches
Debit/Credit Benchmark

- Developed for transaction-processing environments
  - CPU processing is usually trivial
  - Remarkably demanding I/O, scheduling requirements
- Models real TPS workloads synthetically
- Modern version is TPC benchmark
TPC Benchmark

- Initiated by anonymous paper
- Now controlled by Transaction Processing Council
  - Work very hard to be fair & prevent gaming
- Audited
  - Expensive to run
- Requires publishing system cost
  - Including 5-year maintenance costs
- Evolving versions to keep up with technology

Jim Gray and 24 others: “A Measure of Transaction Processing Power.”
SPEC Suite

- Result of multi-manufacturer consortium
  - Results are audited
  - Can be very expensive to run
- Addresses flaws in existing benchmarks
- Uses real applications, trying to characterize specific real environments
- Considers multiple CPUs
- Geometric mean gives SPECmark for system
- Accepted standard comparison method
  - Regular updates, like TPC

SPEC Suite
- Result of multi-manufacturer consortium
- Results are audited
- Can be very expensive to run
- Addresses flaws in existing benchmarks
- Uses real applications, trying to characterize specific real environments
- Considers multiple CPUs
- Geometric mean gives SPECmark for system
- Accepted standard comparison method
- Regular updates, like TPC
Modified Andrew Benchmark

- Used in research to compare file system, operating system designs
- Based on software-engineering workload
- Exercises copying, compiling, linking
- Ill-designed, but common use makes it important
- Needs scaling up for modern systems
  - Common alternates: compile `ssh` or Linux kernel
Winstone, Webstone, etc.

- "Stone" has become suffix meaning "benchmark"
- Many specialized suites to test specialized applications
  - Too many to review here
  - Important to understand strengths & drawbacks
  - Bias toward certain workloads
  - Assumptions about system under test
Exercisers and Drivers

- For I/O, network, non-CPU measurements
- Generate a workload, feed to internal or external measured system
  - I/O on local OS
  - Network
- Sometimes uses dedicated system & interface hardware
Advantages & Disadvantages of Exercisers

+ Easy to develop, port
+ Can incorporate measurement
+ Easy to parameterize, adjust
  - High cost if external
  - Often too small compared to real workloads
    ▶ Thus not representative
    ▶ E.g., may use caches “incorrectly”
  - Internal exercisers often don’t have real CPU activity
    ▶ Affects overlap of CPU and I/O
  - Synchronization effects caused by loops

- High cost if external
- Often too small compared to real workloads
  ▶ Thus not representative
  ▶ E.g., may use caches “incorrectly”
- Internal exercisers often don’t have real CPU activity
  ▶ Affects overlap of CPU and I/O
- Synchronization effects caused by loops
▲ Considerations in selecting a workload
▲ Example of workload selection
What services does system actually use?
- Faster CPU won’t speed up CP
- Network performance useless for matrix work

What metrics measure these services?
- MIPS/GIPS for CPU speed
- Bandwidth/latency for network, I/O
- TPS for transaction processing
Completeness

- Computer systems are complex
  - Effect of interactions hard to predict
  - So must be sure to test entire system
- Important to understand balance between components
  - I.e., don’t use 90% CPU mix to evaluate I/O-bound application
Component Testing

- Sometimes only individual components are compared
  - Would a new CPU speed up our system?
  - How does IPV6 affect Web server performance?
- But component may not be directly related to performance
  - So be careful, do ANOVA, don’t extrapolate too much
Service Testing

- May be possible to isolate interfaces to just one component
  - E.g., instruction mix for CPU
- Consider services provided and used by that component
- System often has layers of services
  - Can cut at any point and insert workload
Characterizing a Service

- Identify *service* provided by major subsystem
- List *factors* affecting performance
- List *metrics* that quantify demands and performance
- Identify *workload* provided to that service
Example: Web Server

Web Page Visits → TCP/IP Connections
Network
Web Server → Web Page Accesses
File System → Disk Transfers
Disk Drive

Examples:
- Web Client
- Network
- Web Server
- File System
- Disk Drive
- Web Page Visits
- TCP/IP Connections
- HTTP Requests
- Web Page Accesses
- Disk Transfers
Web Client Analysis

- Services: visit page, follow hyperlink, display page information
- Factors: page size, number of links, fonts required, embedded graphics, sound, JavaScript usage
- Metrics: response time (both definitions)
- Workload: a list of pages to be visited and links to be followed
Network Analysis

- Services: connect to server, transmit request, transfer data
- Factors: bandwidth, latency, protocol used
- Metrics: connection setup time, response latency, achieved bandwidth
- Workload: a series of connections to one or more servers, with data transfer
Web Server Analysis

- Services: accept and validate connection, fetch & send HTTP data
- Factors: Network performance, CPU speed, system load, disk subsystem performance
- Metrics: response time, connections served
- Workload: a stream of incoming HTTP connections and requests
File System Analysis

- Services: open file, read file (writing often doesn’t matter for Web server)
- Factors: disk drive characteristics, file system software, cache size, partition size
- Metrics: response time, transfer rate
- Workload: a series of file-transfer requests
Disk Drive Analysis

- Services: read sector, write sector
- Factors: seek time, transfer rate
- Metrics: response time
- Workload: a statistically-generated stream of read/write requests
Level of Detail

- Detail trades off accuracy vs. cost
- Highest detail is complete trace
- Lowest is one request, usually most common
- Intermediate approach: weight by frequency
- We will return to this when we discuss *workload characterization*
Obviously, workload should represent desired application
- Arrival rate of requests
- Resource demands of each request
- Resource usage profile of workload over time

Again, accuracy and cost trade off

Need to understand whether detail matters
Timeliness

- Usage patterns change over time
  - File size grows to match disk size
  - Web pages grow to match network bandwidth
- If using “old” workloads, must be sure user behavior hasn’t changed
- Even worse, behavior may change after test, as result of installing new system
  - “Latent demand” phenomenon
Other Considerations

- Loading levels
  - Full capacity
  - Beyond capacity
  - Actual usage

- External components not considered as parameters

- Repeatability of workload