Workloads and Workload Selection

Dr. John Mellor-Crummey

Department of Computer Science
Rice University

johnmc@cs.rice.edu

Goals for Today

Understand

- **Different types of workloads**
- **What workloads are commonly used**
- **How to select appropriate workload types**

Workloads

Terms

- **Real workload**
  - —one observed during normal system operations
  - —non-repeatable
- **Synthetic workload**
  - —approximation of real workload
  - —can be applied repeatedly in a controlled manner
  - —no large data files; no sensitive data
  - —easily modified and ported
  - —easily measured
- **Test workload**
  - —any workload used in performance studies
  - —real or synthetic

Workload Classes

- **Non-executable**
  - —e.g. 125 packets per second, 2ms service time
  - —commonly used for analytical modeling and simulation

- **Executable**
  - —can be run and measured on system under test
  - —e.g. benchmark program, trace of commands to drive simulation

Types of Workloads

- **Single instruction**
- **Instruction mix**
- **Application kernels**
- **Synthetic programs**
- **Application benchmarks**

Workloads: Single Instruction

- **Single instruction throughput for addition**
  - —addition is most common instruction
  - —historical metric
    - – used when processor performance = system performance

Workloads: Instruction Mix

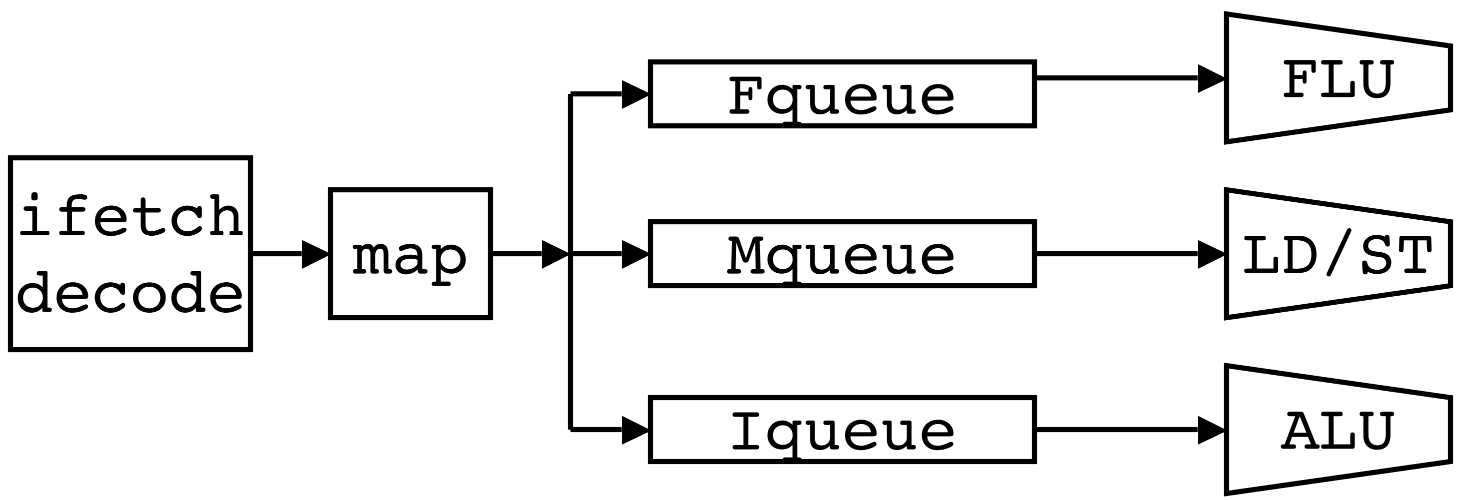
Instruction types + usage frequency

- **Purpose**
  - —obtain basic understanding of processor capabilities when applied to an instruction stream
  - —choose mix representative of code found in real workloads
- **Limitations**
  - —may not reflect factors affecting performance
    - – interactions, e.g. data dependences
    - – branch predictability
    - – pipelining and instruction level parallelism
  - —misses memory hierarchy and virtual address translation
  - —measures only processor performance
  - —measurements with an instruction mix may not reflect system performance if the processor is not the bottleneck
- **Example: Gibson mix (1959) - 13 instr. types + frequencies**

Analyzing a CPU using Instruction Mix

Cameron, Luo, Scharzmeier ISHPC 1999

- **Goal**
  - Improve understanding of superscalar microprocessors on scientific workloads
  - validate proposed queueing model of microprocessor core
    - model CPU as functional units + dispatch queues



Modeling a Microprocessor Core

$$\lambda_x = \frac{\text{total \# completed instructions}}{\text{\# type x instructions}}$$

Δ_x = execution rate of x-queue (instr/cycle)

β = ideal instruction dispatch rate (instr/cycle)

x = m(emory), i(nteger),f(loating point)

x-queue growth rate $$G_x = \frac{\beta}{\lambda_x} - \Delta_x$$

Cameron, Luo, Scharzmeier ISHPC 1999

Bottleneck Analysis of Microprocessor

- **Inputs: architectual constraints (R10K - 4-way ss)**
  - queue length = 16 per functional unit
  - max instructions in flight = 32
  - graduation rates: FP = 2/cycle; INT = 2/cycle; mem = 1/cycle
  - outstanding misses = 4 (Origin 2000)
- **Model**

—>$G_x > 0 \Longrightarrow$ x-queue will fill up and cause a stall

$$CPI_0 = \frac{\text{total \# cycles}}{\text{total \# instr}} = \frac{1}{\lambda_x \Delta_x}$$

—if multiple positive growth rates, must also consider threshold of max. # instructions in flight

- **Validation**

—synthesize representative instruction mixes

—analyze performance of synthetic instruction mix on processor

Cameron, Luo, Scharzmeier ISHPC 1999

R10K Model Validation using Instruction Mix

	Growth Rates					
Pattern	Gf	Gm	Gi	Meas CPI	Calc CPI	Rel Error
fff_*+*	1.96	-0.99	-1.98	0.66	0.66	0.004
ifff_+*+	0.98	-1.00	-0.99	0.52	0.50	0.045
ii	-2.00	-0.99	1.96	0.51	0.50	0.020
iiif	-1.01	-1.00	0.99	0.40	0.37	0.057
mfff_*+*	0.98	0.00	-1.99	0.50	0.50	0.006
miii	-2.00	0.00	0.99	0.40	0.37	0.056
mm	-2.00	2.95	-1.97	1.00	0.99	0.015
mmff_+*	-0.02	0.99	-1.99	0.50	0.50	0.015
mmif	-1.01	0.99	-0.99	0.51	0.50	0.020
mmii	-2.00	0.99	0.00	0.51	0.50	0.020
mmmf	-1.01	1.98	-1.99	0.76	0.75	0.014
mmmi	-2.00	1.98	-0.99	0.75	0.75	0.010

Cameron, Luo, Scharzmeier ISHPC 1999

Workloads: Application Kernels

- **Motivation**
  - —pipelining, instruction and data caching make instruction execution rates variable
  - —necessary to consider set of instructions that provides a service
- **Kernel examples**
  - —Eratosthenes' primality sieve, Ackermann's function, matrix multiplication, matrix inversion, sorting
- **Limitations**
  - —kernels are not based on measurements of real systems
  - —typically don't perform I/O and thus do not accurately characterize total system performance

Workloads: Synthetic Programs

- **Motivation**
  - —I/O and OS services are important part of real workloads
  - —kernels don't use OS services or I/O
- **Synthetic programs: loops containing I/O and/or OS calls**
  - —use them to compute CPU time per service call
    - – e.g. process creation, forking, memory allocation, …
- **Examples**
  - —LMBench: measure memory latency/bandwidth & core OS operations
  - —STREAM: sustainable memory bandwidth for vector kernels
  - —Livermore Loops: scientific FP-intensive loop nests
- **Advantages**
  - —quick to develop
  - —portable
  - —usually have built in measurement capabilities
- **Disadvantages**
  - —generally too small; unrepresentative disk or memory references
  - —typically unrepresentative CPU-I/O overlap

STREAM Benchmark

Copy

```
DO J = 1, N
   C(J)= A(J)
END DO
```

Scale

```
DO J = 1, N
   B(J)= S*A(J)
END DO
```

Add

```
DO J = 1, N
   A(J)=B(J)+ C(J)
END DO
```

Triad

```
DO J = 1, N
   A(J)=B(J)+s*C(J)
END DO
```

www.streambench.org

Workloads: Application Benchmarks

- **Representative subset of functions for an application**
- **Typically make use of almost all resources**
  - CPU, I/O, networks, databases
- **Examples**
  - LINPACK (www.netlib.org/linpack)- solve dense linear equations
  - SPEC (www.spec.org)
    - CPU2000 - compute-intensive integer or FP performance
    - HPC2002 - parallel performance (qchem, weather, seismic)
    - OMP2002 -scientifc and engineering applications in OpenMP
    - jAppServer2004, JBB2000, JVM98 - Java
    - servers (network file, web, mail), graphics performance
  - TPC benchmarks (www.tpc.org)
    - TPC-C - on-line transaction processing benchmark
    - TPC-W - transactional web e-commerce benchmark
    - TPC-H - ad-hoc decision support benchmark
      - queries and concurrent data modifications

Key Parallel Benchmarks

- **NAS parallel benchmarks (www.nas.nasa.gov/Software/NPB)**
  - —widely used to benchmark parallel compilers
  - —serial, MPI, OpenMP, HPF versions
- **High Performance LINPACK (www.netlib.org/benchmark/hpl)**
  - —solves a dense linear system in double precision (64 bits) arithmetic on distributed-memory computers
  - —used to rate computers for the Top 500 list
- **HPC Challenge Benchmark (icl.cs.utk.edu/hpcc)**
  - —HPL - high performance LINPACK
  - —DGEMM - double precision real matrix-matrix multiplication
  - —STREAM - sustainable memory bandwidth for vector computation
  - —PTRANS - parallel matrix transpose (comm capacity)
  - —RandomAccess - rate of random updates of memory
  - —FFT - 1D discrete Fourier transform
  - —b_eff: effective communication bandwidth

Workload Selection

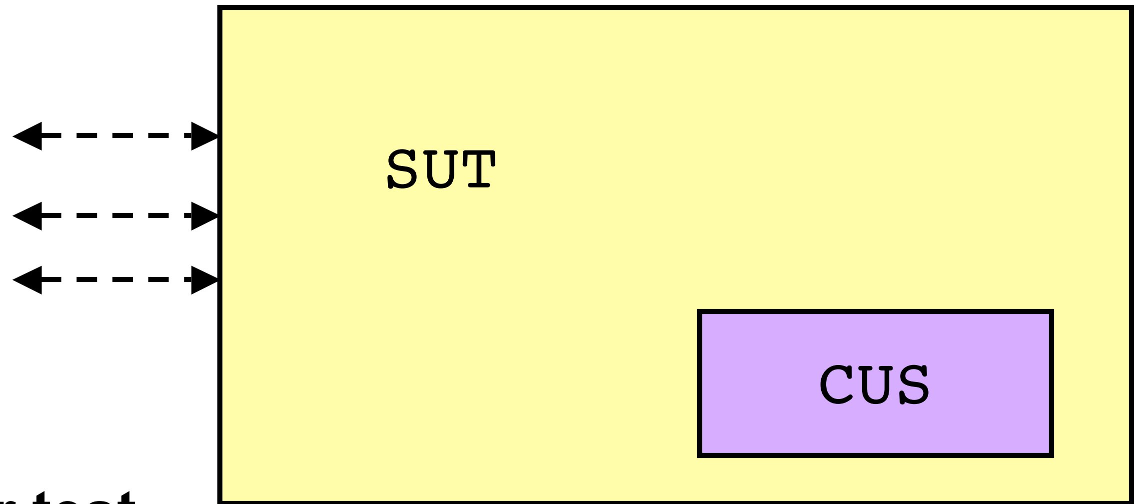
- most crucial part of performance evaluation project
- inappropriate workload $\Longrightarrow$ misleading conclusions

Principal Considerations

- Services exercised
- Level of detail
- Representativeness
- Timeliness
- Other considerations
  —loading level
  —impact of other components

Services Exercised

system services determine the workload and metrics



- **SUT = system under test**
- **CUS = component under study**
- **Metrics chosen should reflect system level performance**
- **Examples:**
  - —**SUT = CPU; CUS = ALU; metric = MIPS**
  - —**SUT = transaction proc. system; CUS = disk drive; metric = T/S**
- **Workload chosen should reflect SUT, <u>not CUS</u>**

Example

Comparing two banking systems differing only in CPU

- **Workload: transaction arrival frequencies**
- **Metric: transactions completed per second**

Workload Selection Rules of Thumb

- **For multiple services**
  - —workload should exercise as many services as possible
    - – e.g. analyze CPU with FP and INT workloads, not just one
- **For services exercised, consider purpose of study**
  - —text workload with graphics editor?
  - —graphics workload with text editor?

Level of Detail

- **Most frequent request**
  - —valid if one service is requested much more often than others
  - —examples: add instruction, kernels
- **Frequency of request types**
  - —example: instruction mix
  - —context sensitive services - must use a set (e.g. caching)
- **Time stamped sequence of requests (trace)**
  - —too much detail for analytical modeling
  - —may require exact reproduction of component behavior for timing
- **Average resource demand**
  - —analytical models use request rate rather than requests
  - —group similar services in classes; use avg. demand per class
- **Distribution of resource demands, used if**
  - —variance in resource demands is large
  - —distribution impacts performance

Representativeness

Test workload and real workload should have the same:

- **Arrival rate**
  - —should be the same or proportional to that of real application
- **Resource demands**
  - —total demand should be = or proportional to that of real application
- **Resource usage profile**
  - —amount and sequence in which resources are consumed
  - —especially important when forming a composite workload

Timeliness

- **Things always change, ignore change at your peril!**
- **Users change usage pattern based on**
  - —new services available
    - – e.g. WWW browsing as workload activity
  - —changes in system performance: users optimize demand
    - – e.g. slow multiplications led to FFT algorithms minimizing them
- **Anticipate changes**
  - —monitor user behavior on ongoing basis
  - —future may be different than past or present

Other Considerations

- **Load level**
  - —full capacity (best case)
  - —beyond capacity (worst case)
  - —real workload (average case)
  - —for procurement, consider typical case
  - —for design, consider best ⇔ worst cases
- **Impact of external components**
  - —don't use workload that makes external component bottleneck
    - – e.g. if studying CPU performance, don't use data so large that system is paging
  - —otherwise, all alternatives will give equally good performance
- **Repeatability**
  - —want to be able to repeat results without excessive variance
  - —highly random resource demands should be avoided

Example: Tape Backup System

- Characteristics
  - —Multiple tape systems, several tape drives each
  - —Drives have separate read and write subsystems
  - —each subsystem uses magnetic heads
- Services, factors, metrics, workloads
  - —backup system
    - – services: backup files, backup changed files,restore files, list catalog
    - – factors: file system size, foreground/background, incremental/full
    - – metrics: backup time, restore time
    - – workload: computer system with files to be backed up. vary frequency
  - —tape data system
    - – services: read/write to tape, read tape label, autoload tapes
    - – factors: type of tape drive
    - – metrics: speed, reliability, time between failures
    - – workload: synthetic program generating tape representative I/O requests
  - —tape drives
    - – services: read record, write record, rewind, find record, move to end of tape
    - – factors: cartridge or reel tapes, drive size
    - – metrics: time for each kind of service, requests/unit time, noise, power
    - – workload:synthetic program generating representative requests

Tape Backup System (Continued)

- **More services, factors, metrics, workloads**
  - **—read/write subsystem**
    - **services: read data/write data (as digital signals)**
    - **factors: data encoding technique, implementation technology (CMOS, etc)**
    - **metrics: coding density, I/O bandwidth, error rate**
    - **workload: read/write streams with varying bit patterns**
  - **—read/write heads**
    - **services: read signal, write signal (electrical signals)**
    - **factors: composition, inter head spacing, gap sizing, number of heads**
    - **metrics: magnetic field strength, hysteresis**
    - **workload: read/write currents of various amplitudes, different speed tapes**

Metrics and Workloads

- **What metrics and workload would you use to compare:**
  - **Mac Powerbook vs. Windows laptop**
  - **laptop vs. desktop system**