Computer Performance Evaluation and Benchmarking

EE 382M

Dr. Lizy Kurian John

Desirable features for modeling/evaluation techniques

- Accurate
- Not expensive
- Non-invasive
- User-friendly
- Fast
- Easy to change or extend
- Must not need application source
- Measure all activity (OS, DLL..)
- Provide control over aspects mesured (selective measurement)

Tradeoffs in modeling/evaluation techniques

- Analytical Models
- +
- -

- Tools with GUI
- +
- -

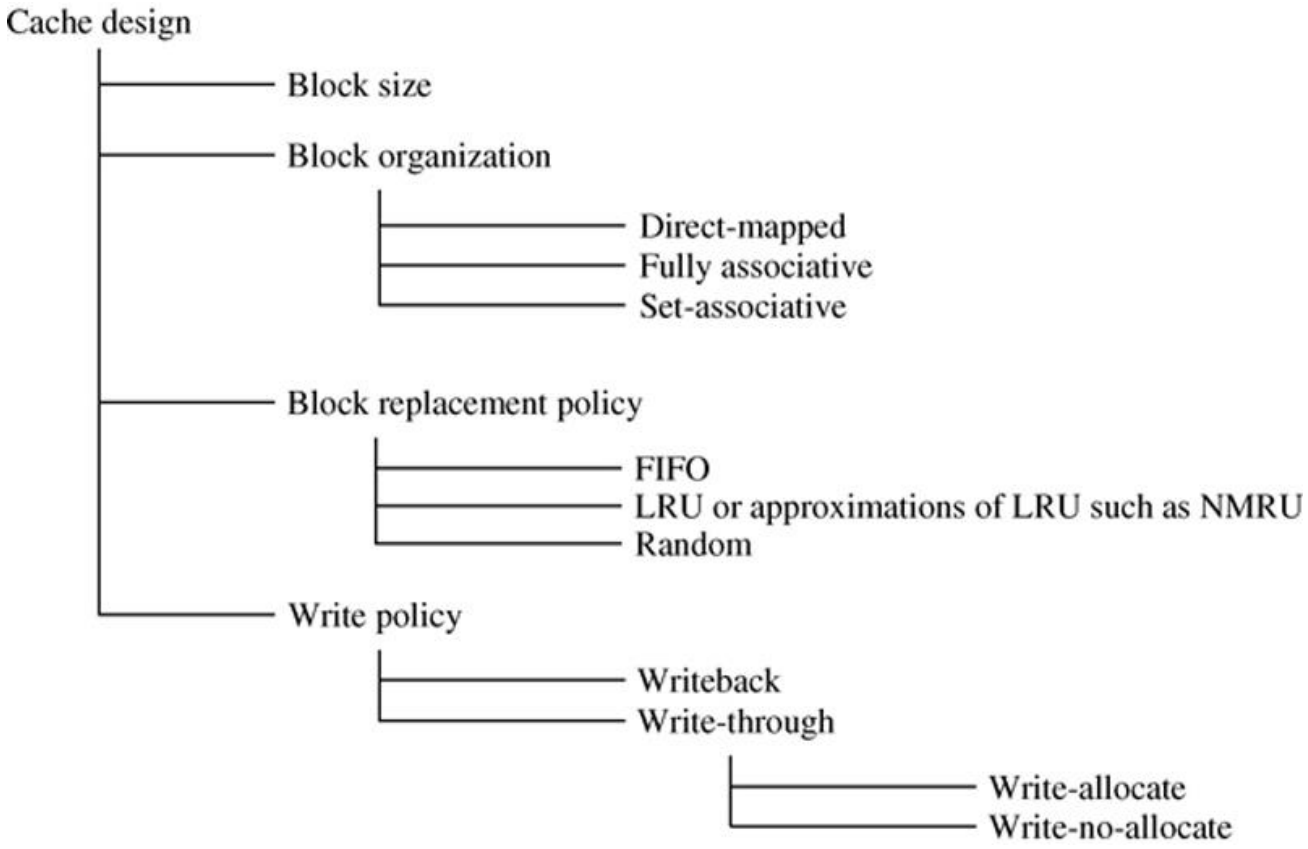
Simulation

- Defacto performance modeling method in industry and academia
  - (currently and for more than 1.5 decades)
- Cycle accurate simulators
- Accuracy better than analytic models
- Simulators written in C, C++, or Java – run on an existing machine called the host
- The machine being modeled is called the target
- Functional simulators
- Timing simulators

Example – Cache Simulator

- Functional simulator
- Timing/Performance simulators
- Cache – data array and tag array
- A C program that simulates the caching operation
- Eg: Takes a program or address sequences used by the program
- Simulates putting the instructions/data according to the specified mapping strategy, replacement strategy, write policy etc.
- One needs to consider all of the design parameters

Cache Design Parameters



MIPS example code

```
for i=1,100, i++                ; repeat 100 times
y(i) = x(i) + y(i)              ; add ith element of the arrays
```

Assume that the x and y arrays start at locations 4000 and 8000 (decimal).
Assume code located at 2000

Answer:

```
            andi    $3, $3, 0           ; initialize index register
            andi    $2, $2, 0           ; clear register for loop bound
            addi    $2, $2, 400         ; counter for loop bound
$label:     lw      $15, 4000($3)       ;load x(i) to R15
            lw      $14, 8000($3)       ;load y(i) to R14
            add     $24, $15, $14       ;x(i) + y(i)
            sw      $24, 8000($3)       ;save new y(i)
            addu    $3, $3, 4           ;update address register, address= address + 4
            bne     $3, $2, label
```

Refer to the MIPS Handout on Blackboard to learn MIPS ISA details

Cache address sequence

- I-cache
  - 2000, 2004, 2008, (2012, ..2032, 2012)…
- D-cache
  - 4000, 8000, 8000, 4004, 8004, 8004,
  - ……4396, 8396, 8396

Cache Performance Simulator

- Estimate number of hits, misses, total memory access time
- How to make the simulator accurate?
- One needs to have good assumptions on timings for various events
- Ideally the cache simulator must be linked to a CPU simulator that gives total execution time
- When caching strategy changes, you can tell the impact on processor's overall execution time

Trace Driven Simulation – Pros and Cons

+ Simple (Easy to understand)

+ Easy to debug

+ Experiments repeatable easily (traces can be distributed to others to validate/debunk)

- Traces often prohibitively long (proportional to dynamic instrn size)

- Mispredicted path missing in speculative microarchitectures

Solving Cons of Trace Driven Simulation

For trace size

-Trace Sampling (Crowley & Baer paper)

-Trace Reduction/Trace Compression (eg: QPT's trace regeneration)

For speculative path

-Reconstruct the mispredicted path from a pass through the trace (construct memory image)

Functional simulator

- RTL Model (Verilog or VHDL)
- Is the design correctly caching as you expect?
- Performance Model
- What kind of performance benefits can you expect from this cache?
- Which block size, mapping strategy or replacement strategy should you adopt?

Pipelined Execution Simulation

Number of Stages

Simulate each stage

Delays associated with pipeline stalls

Even cache hits can take multiple cycles

Branch Prediction

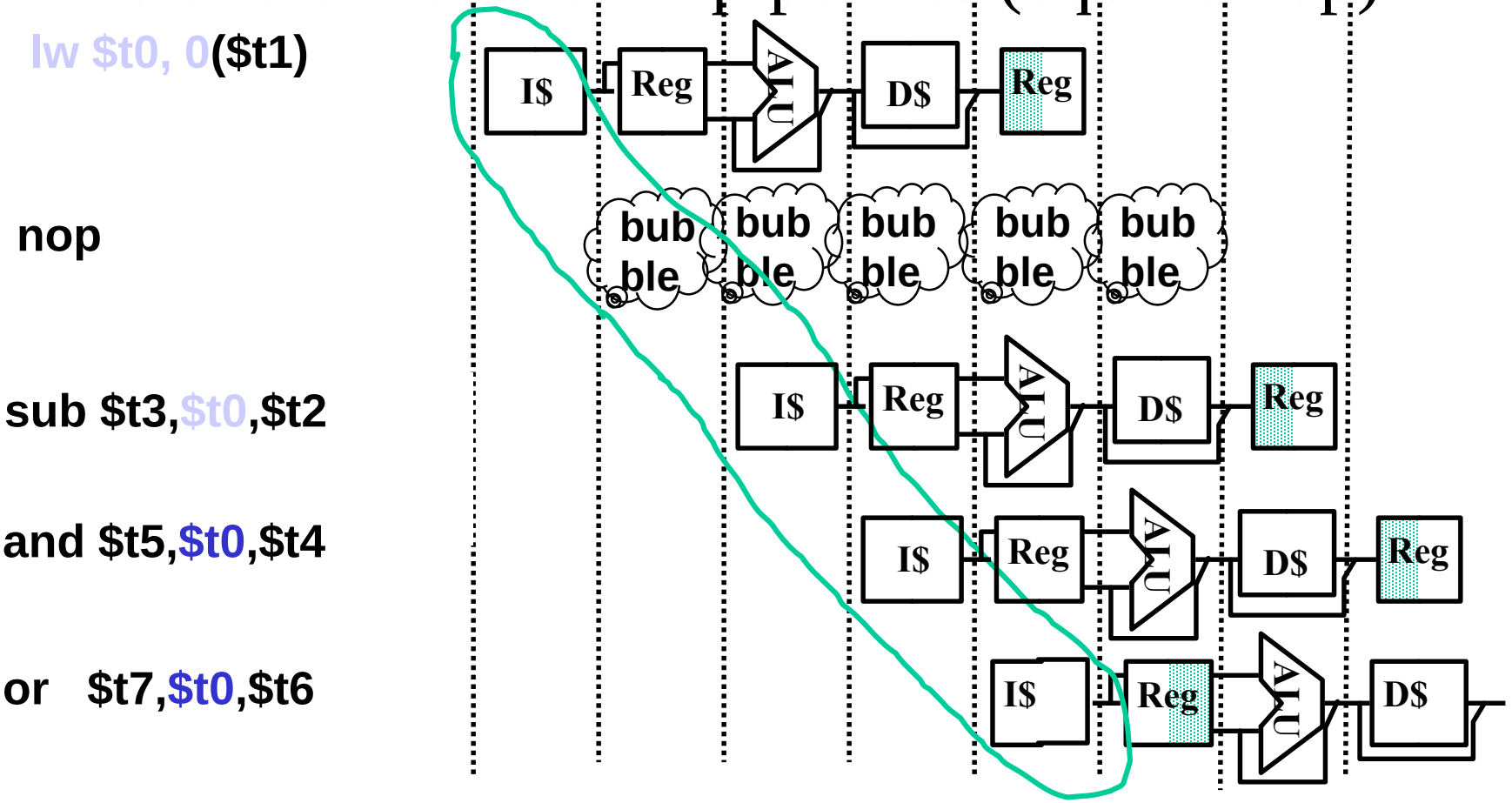
Input executable or trace to the simulator

Input configuration parameters

Cycle by cycle execution of the instructions

Pipelined Execution – Stalls

- Stalls or bubbles in pipelines (eqvt to nop)



Pipelined Execution Example

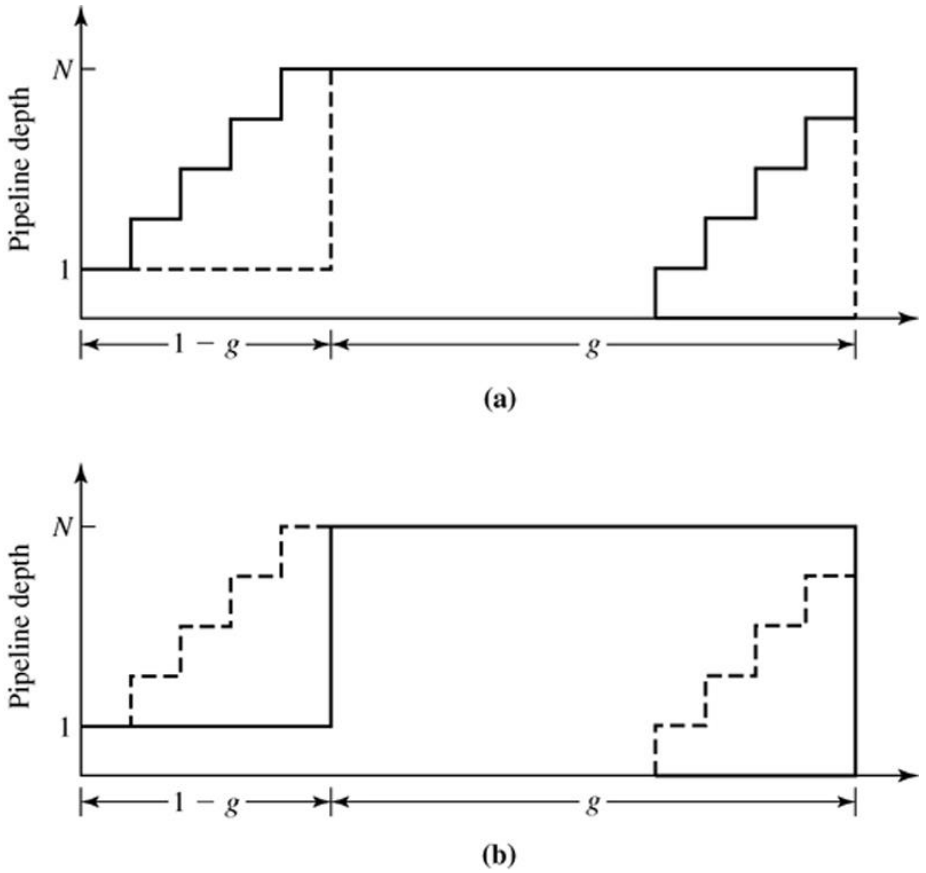
Assume 1 instr/clock, perfect branch prediction, 5 stage pipeline, data forwarding, 2 cycle cache hits, (10^3 iterations, assume pipeline full)

```
Loop:       lw      $t0, 0($s1)
            addu    $t0, $t0, $s2
            sw      $t0, 0($s1)
            addiu   $s1, $s1, -4
            bne     $s1, $zero, Loop
            nop
```

- How many pipeline stages (clock cycles) per loop iteration to execute this code ?
- How many pipeline stages (clock cycles) per loop iteration to execute this code if no data forwarding ?

1
2
3
4
5
6
7
8
9
10

Amdahl's Law for Pipelined Execution



On-chip Performance Monitoring Counters

- 2 to 18 counters on all modern processors
- Monitor hundreds of metrics
- Cycle count, I-counts at fetch, decode, retire
- Cache misses at each level of cache
- Branch mispredictions
- I- and D-TLB misses
- Eg: Bhandarkar paper

Analytical Modeling example: A simple analytical model for pipelined processors

Evaluating Pipelining using laws of Parallel Processing

Amdahl's Law

Vectorizability

Amdahl's Law

In parallel processing, serial part limits total performance

T= original execution time

S= fraction of time in serial code, eg: 0.2

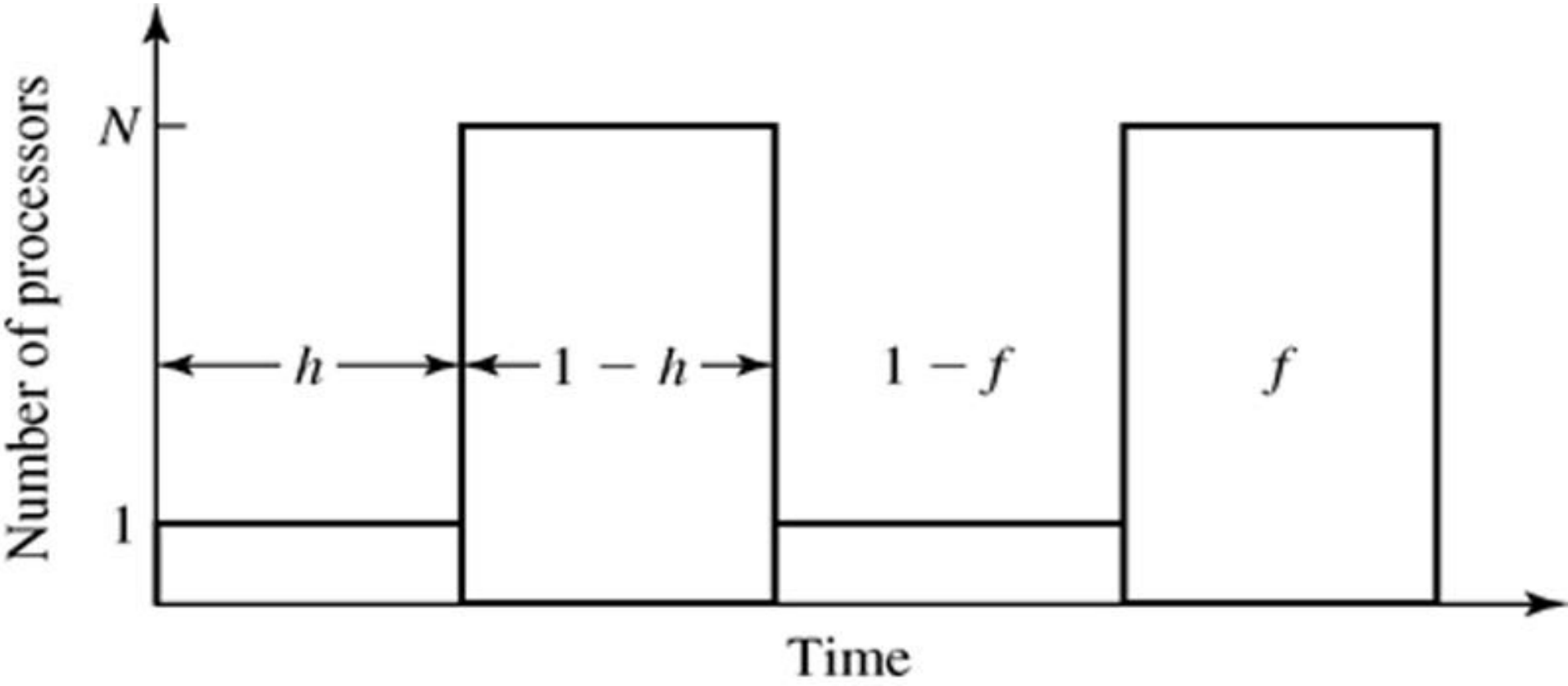
P= fraction of time in parallel code = 1-S

N= number of parallel units

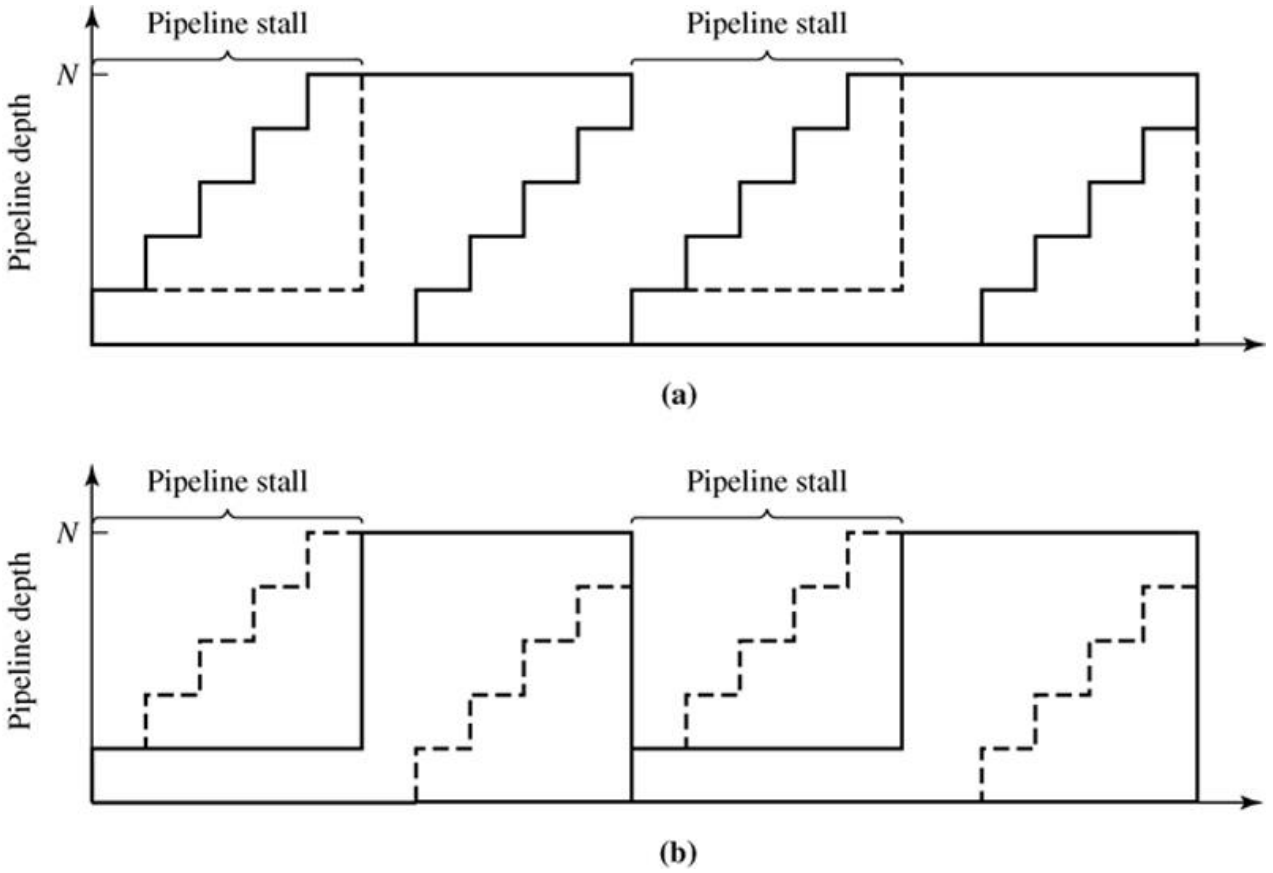
Speedup = 1/ (S +P/N)

Max speedup = 1/S

Amdahl's Law



Realistic Pipelining



Various Forms of Amdahl's Law

Eq 1 - Speedup, $S = 1/\{(1-f)+(f/N)\}$

Eq 2 – Speedup $S = 1 / \{(1-g)+(g/N)\}$

Eq 3 – $S= 1 / \{g1/1 + g2/2 + g3/3 +\ldots gN/N\}$

Amdahl's Law

Assume I-mix (load = 25%, branch = 20%,
taken branches = 66.6% of branches, hardware uses NT as policy,
Branch penalty is 4 cycles, load penalty is 1 cycle

Speedup of a 6-stage pipeline under these circumstances

Eq 1.6 – $S = 1 / \{g1/1 + g2/2 + g3/3 + \ldots gN/N\}$

$S = 1 / \{0.13/2 + 0.25/5 + 0.62/6 \} = 4.5$

Ideal S = 6

Difference between peak and actual pipelining improvement

Classification of Techniques

- Performance Modeling
  - Simulation
    - Trace-Driven Simulation
    - Execution Driven Simulation
    - Complete System Simulation
    - Event-Driven Simulation
    - Statistical Simulation
  - Analytical Modeling
    - Probabilistic Models
    - Queuing Models
    - Markov Models
    - PetriNet Models
- Performance Measurement
  - On-Chip Hardware Monitoring
  - Off-Chip Hardware Monitoring
  - Software Monitoring
  - Microcoded Instrumentation