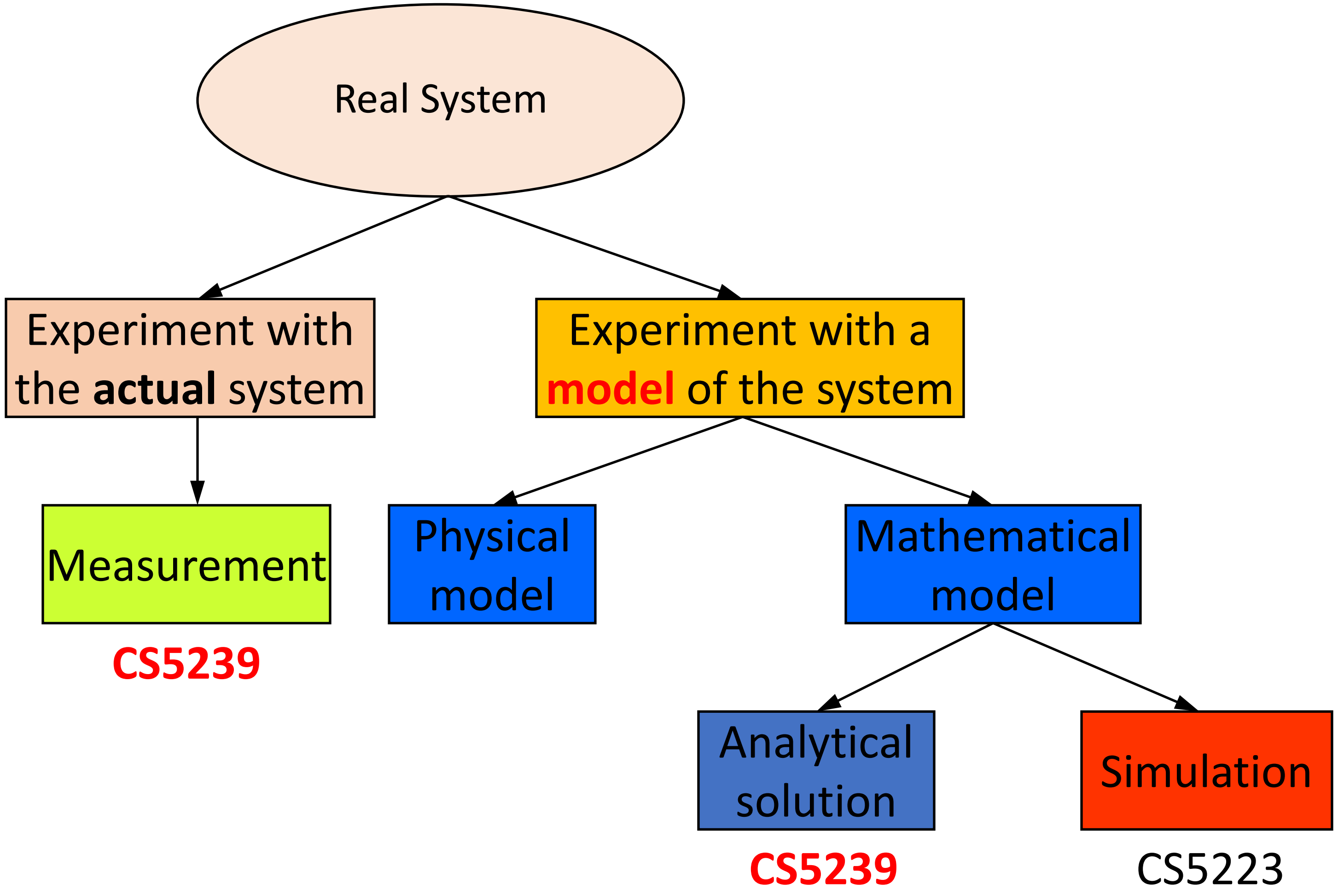
CS5239 Computer System Performance Analysis

Lecture 1 - Introduction

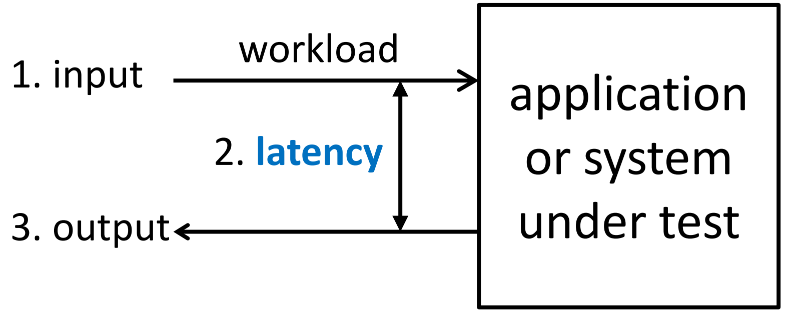
L01: Introduction

- What is performance?
- Performance scope and challenges
- Activities
- Latency
- Observability

Ways to Study a System



What is Performance?



- Shows how well a computer system performs a given job or activity
  - Latency (time)
  - Bandwidth (rate)

- Performance is important to achieve:
  - Time deadline
  - Cost (and efficiency) of computing resources
  - Energy cost

Performance of a computer system

- Performance of a computer system is multidimensional
  - complex component interactions
  - hard to predict how it will scale
  - …
- Sharing of resource -> congestion/queueing

Performance Evaluation

- Performance evaluation is of interest to computer system users, administrators, developers, and designers

- Goal: obtain or provide highest performance at the lowest cost

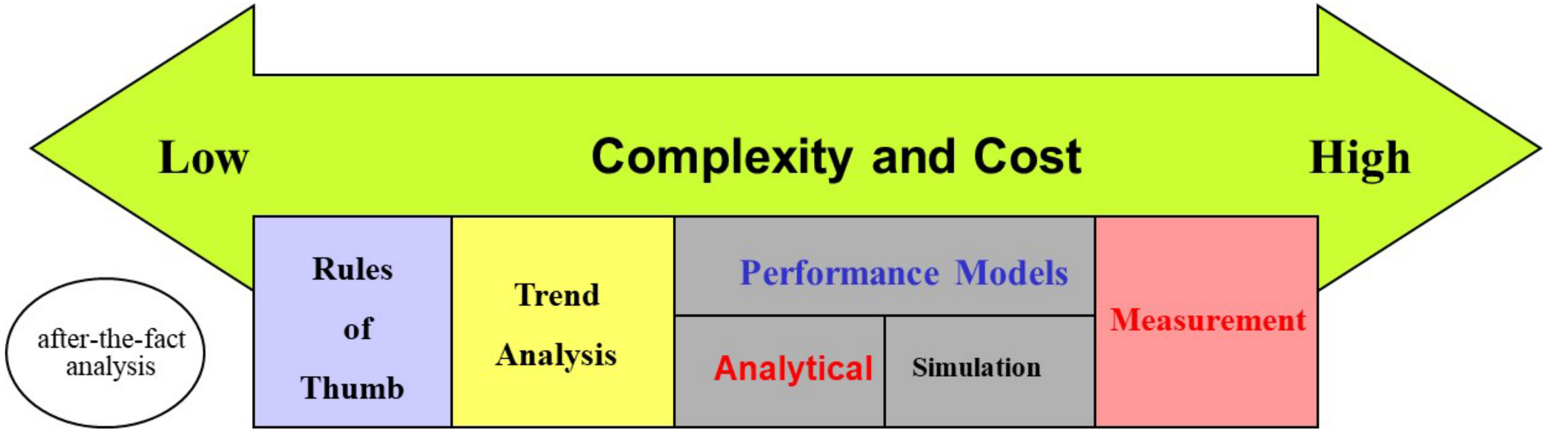
performance-cost ratio

Trade-offs

- The good/fast/cheap "pick two" trade-off

- Many IT projects choose on-time and inexpensive, leaving performance to be fixed later

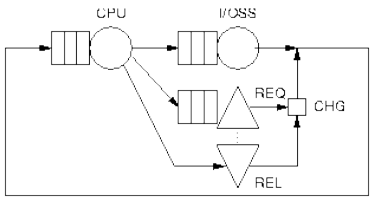
Performance Evaluation: How



- Measurements of actual systems
- Modeling
  - Simulation using software models
  - Analytical modeling using techniques such as queueing analysis

Performance Evaluation Techniques

- Analytical modelling
  - Mathematical representation of the system
  - Model assumptions may not be met in practice - lower accuracy
  - Ease of changing parameters
  - Key insights – validation of simulations / measurements



- Simulation
  - Program written to model important features of the system
  - Can be easily modified to study impact of changes
  - Cost – writing and running the program
  - Impossible to model every small detail – sometimes low accuracy

Models are abstraction of the system

Performance Evaluation Techniques

- Measurement
  - No simplifying assumptions
  - Highest credibility
  - Only works if systems / applications exist
  - Information only on specific system being measured
  - Harder to change system parameters in a real system
  - May be intrusive and perturb operation of systems
  - Difficult and time consuming
  - Need for software tools

Goals of Performance Evaluation

- Set expectations → service level
  - Provide information for users
- System tuning & performance debugging
  - Find parameters that produce best overall performance
  - Identify bottlenecks
  - Determine impact of a new feature
- Compare alternative designs
  - Which configurations are best under which conditions
  - Which program/algorithm is faster
- …

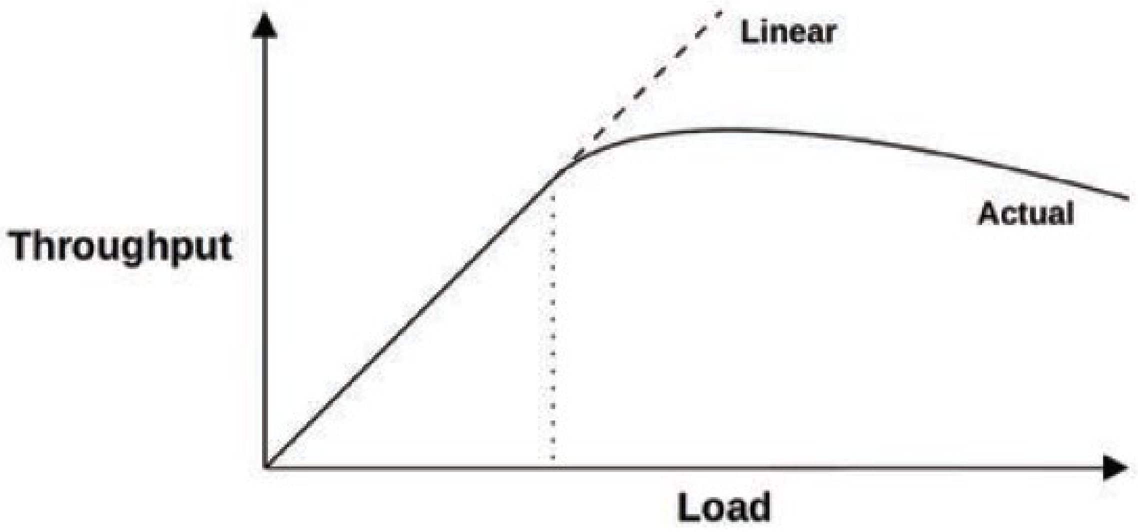
Applications of Performance Evaluation

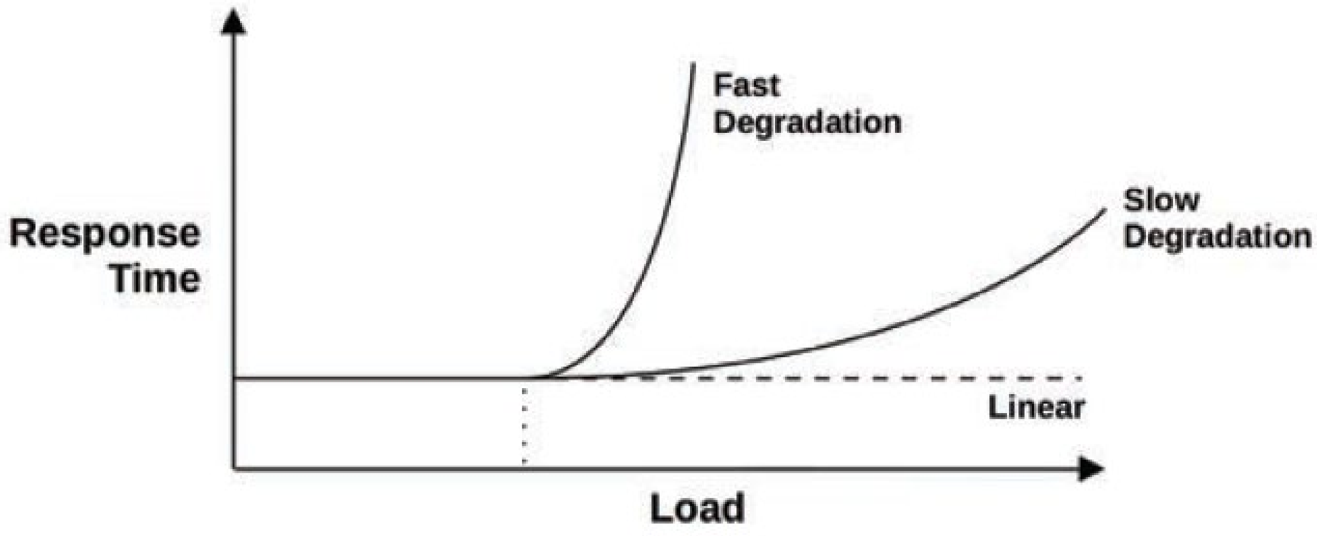
- Selection/configuration of computer systems:
  - System upgrade
  - System design: design of applications and equipment
- Analysis of existing systems: capacity planning predicts when future load levels will saturate the system and determines cost-effective way of delaying system saturation

Global Internet Traffic

Scalability

- The performance of the system under increasing load
  - Linear scalability until the knee point
  - A component reaches 100% utilization: saturation point
    - Queueing is frequent





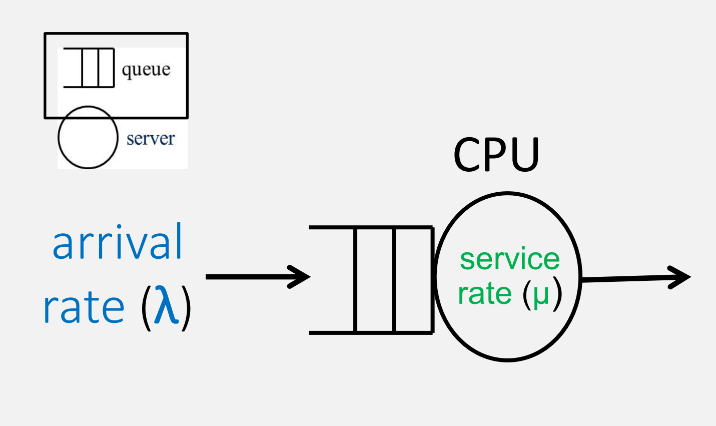
Known and Unknowns

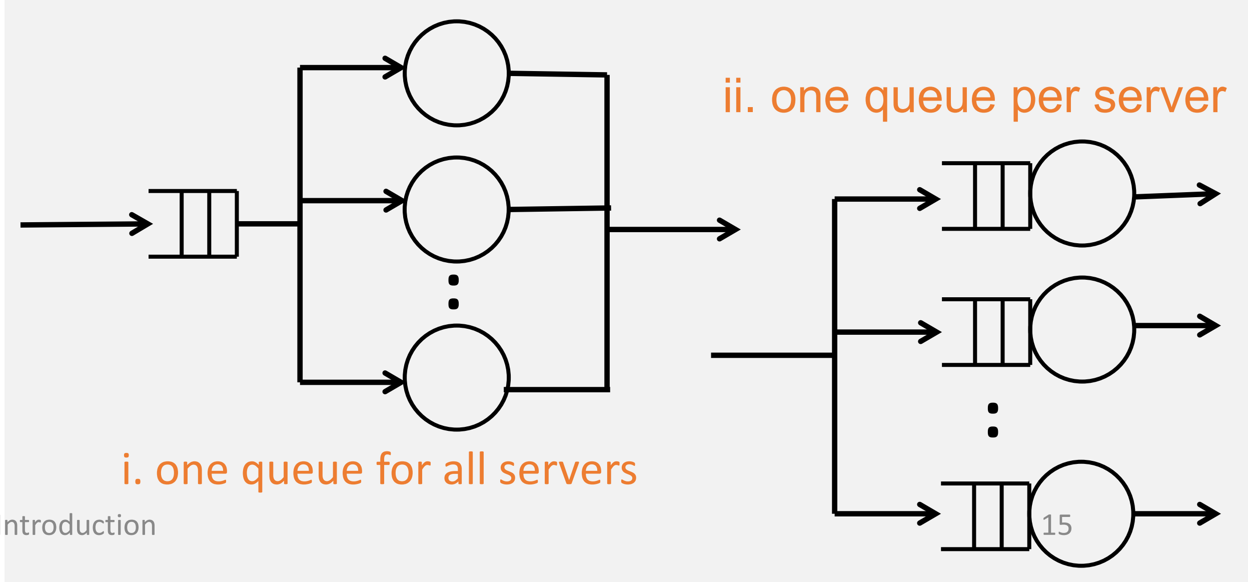
US Secretary of Defence Donald Rumsfeld, 2002

- There are known knowns; there are things we know we know
  - Example: you know that you should check the CPU utilization and you know utilization is 10%
- We also know there are known unknowns; that is to say we know there are some things we do not know.
  - Example: you know that you could be checking what is making the CPU busy but have yet to do so
- But there are also unknown unknowns – there are things we do not know we don't know.
  - Example: you may not know what device interrupts are making the CPU busy and thus you are not checking them
- Performance issues can originate from anywhere, including areas of the system that you know nothing about and therefore not checking (unknown unknowns)

Performance Questions

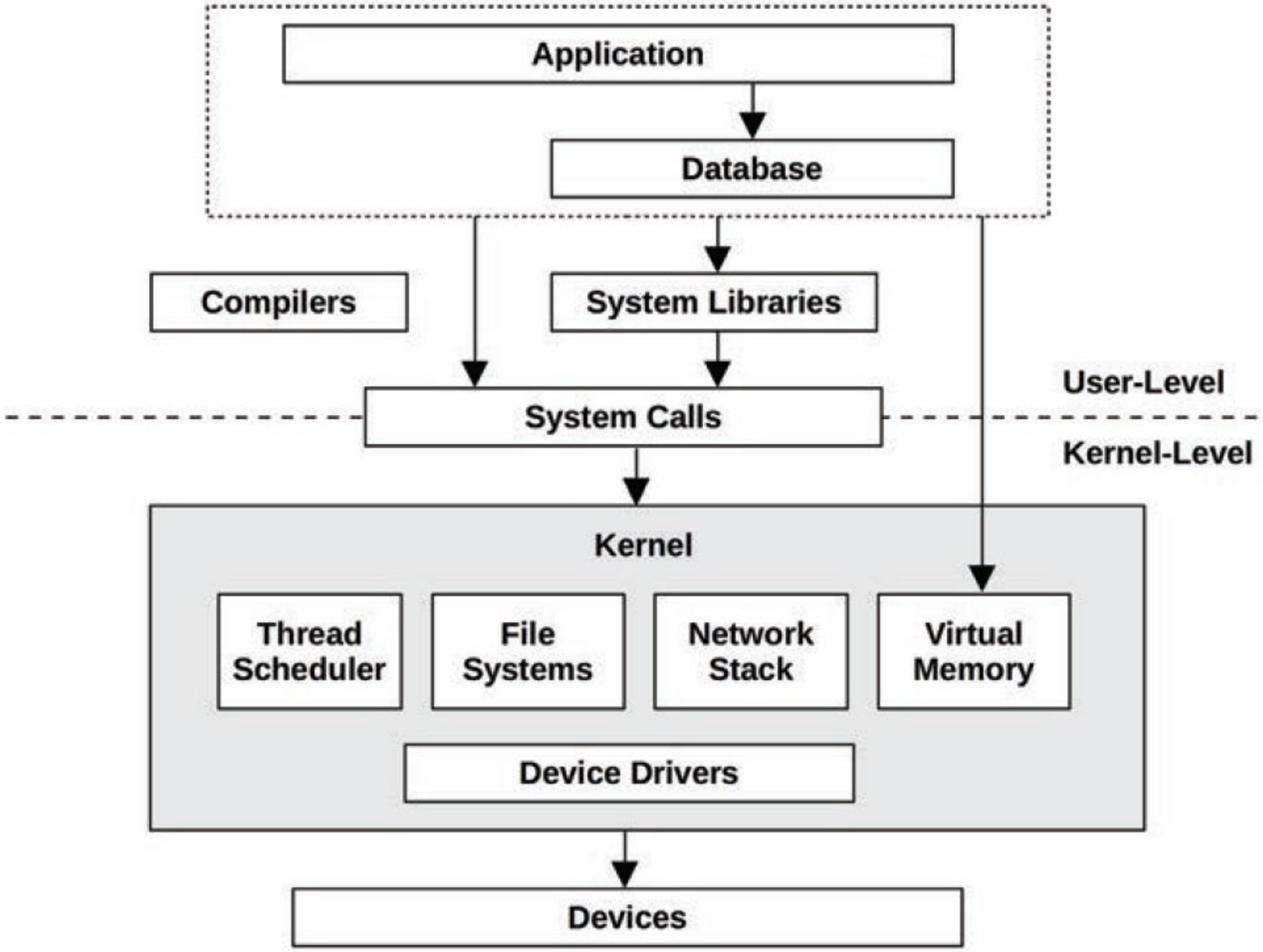
1. What is the average time it takes a job to complete service? [latency]
2. What is the throughput of the system (number of jobs completed per unit time)? [bandwidth]
3. If arrival rate is doubled ($\lambda \rightarrow 2\lambda$), do we do nothing or how much should the service rate (μ) increase?
4. If we need more server (CPU) capacity, what are our options?
   a. Buy a new server with the needed capacity
   b. Buy a few smaller servers that adds up to the required capacity
   c. How do we organize these servers?
      i. One queue for all servers
      ii. One queue for each server
      iii. Does it matter?





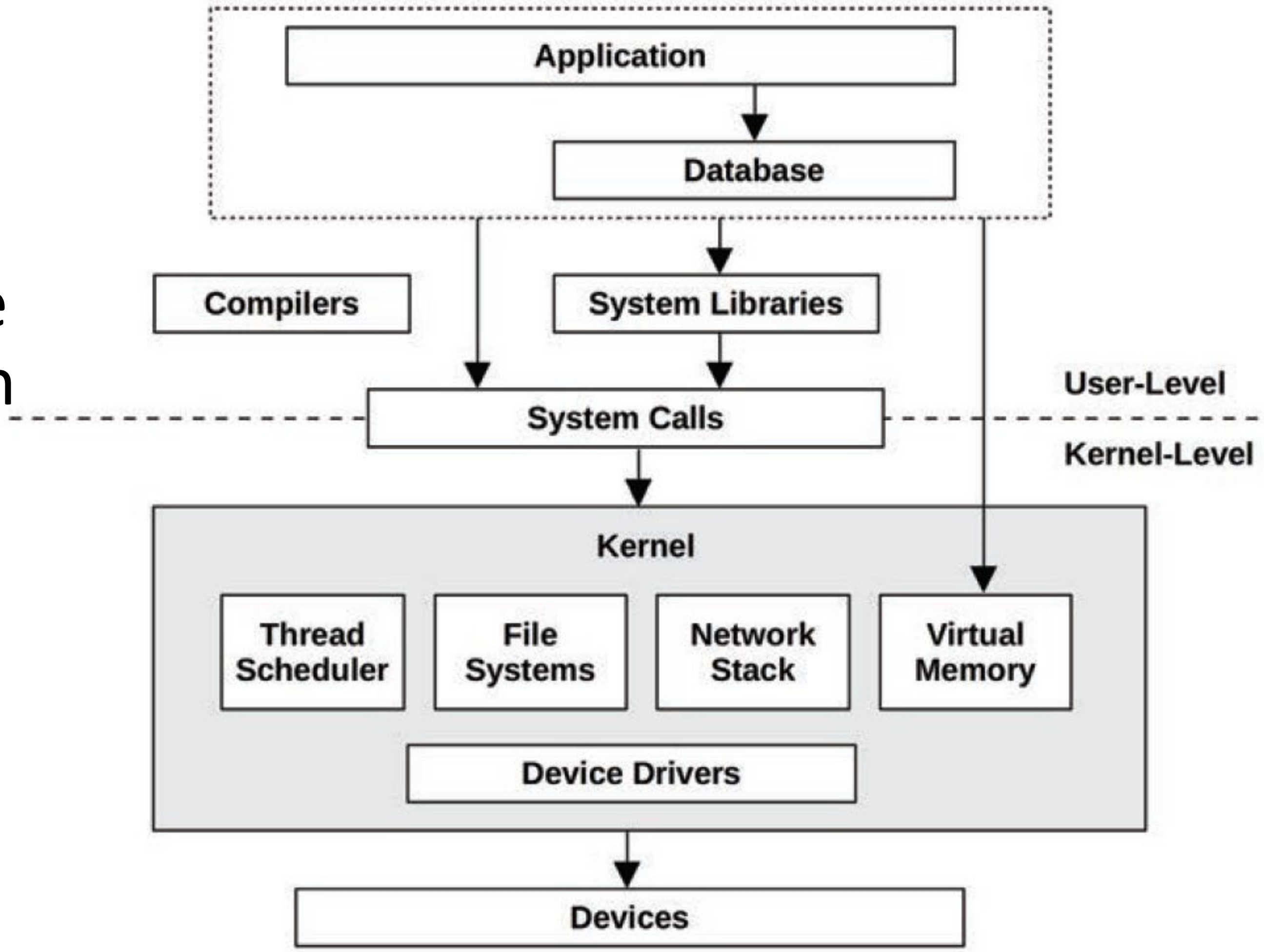
Systems Performance

- Study the performance of an entire computer system
  - All major software and hardware components
    - Anything in the data path, from storage devices to application software
  - For distributed systems, multiple servers and applications



Systems Performance

- Study the performance of the full system stack
- Full stack → the entire software stack from the application down to metal (the hardware)
  - Includes system libraries, the kernel, and the hardware
  - Includes the compiler

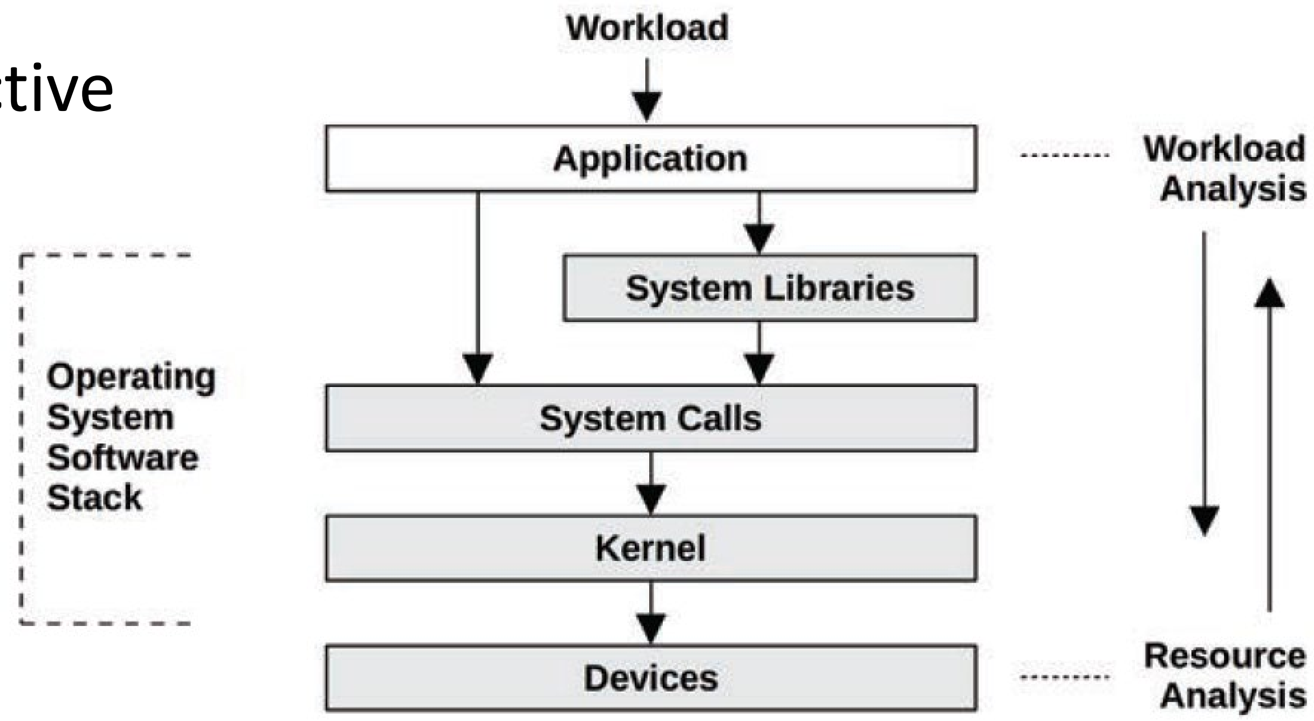


Performance Activities

- Should follow the life cycle of a software project from conception through development to production deployment
- Set objectives and create a performance model as a first step in product development
  - Don't defer performance engineering work to a later time, after a problem arises
- Capacity planning – study the resource footprint of development software to see how well the design can meet the target needs
- Cloud applications performance
  - *Canary testing*
  - *Blue-green development*

Perspectives

- The resource analysis perspective
  → system administrators
- The workload analysis perspective
  → application developers



Challenges

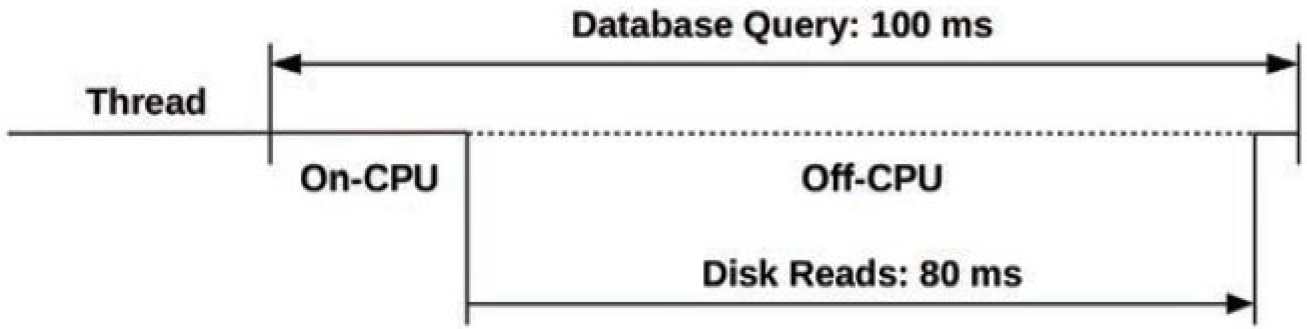
- Subjectivity
  - “The average disk I/O response time is 1 ms.” → is it good or bad?
- Complexity
  - Lack of an obvious starting point for analysis
    - Start with a hypothesis
  - Cascading failure – one failed component causes performance issues in others
  - Bottlenecks – complex and unexpectedly related to each other
- Multiple causes: issues do not have a single root cause
- Multiple performance issues
  - Quantify the magnitude of issues

Metrics

- Metrics are needed to quantify the performance of a system
- Latency - measure of time spent waiting
  - Broadly, it means the time for any operation to complete
    - Time taken by an application request, a database query, a file system operation, etc.
  - Example: the time needed for a website to load completely, from link click to screen paint
  - Allows maximum speedup to be estimated
  - Ambiguous term without qualifying terms
    - In networking, latency can mean the time for a connection to be established but not the data transfer time [connection latency]; or it can mean the total duration of a connection, including the data transfer [request latency]

Latency - Example

- A database query that takes 100 ms [latency] during which it spends 80 ms blocked waiting for disk reads
- The maximum performance improvement is obtained by eliminating disk reads (e.g., by caching) from 100 ms to 20 ms
  - 5x faster [speedup]



- What if you are looking at I/O operations per second instead?

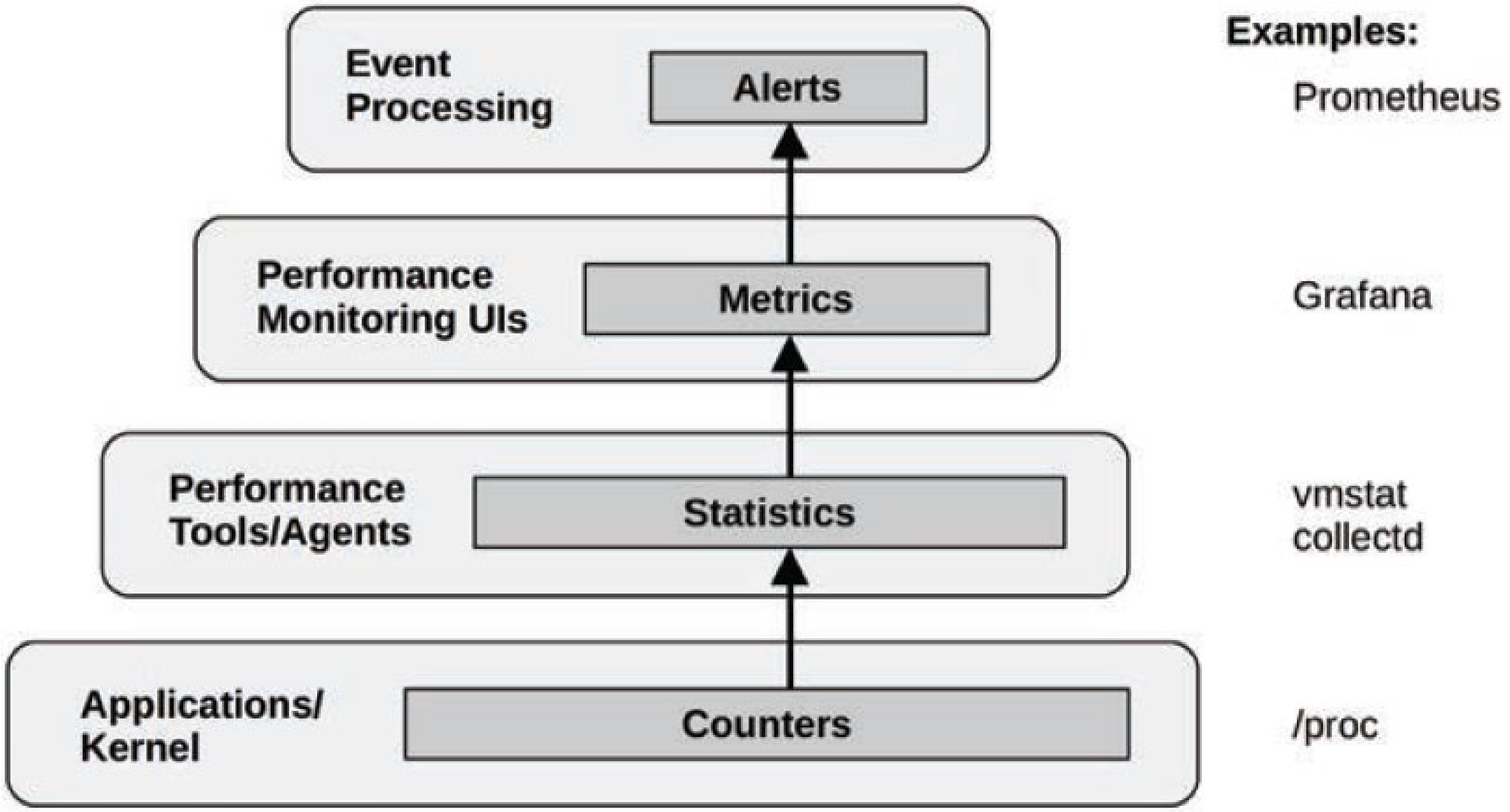
Observability

- Understanding a system through observation using different observability tools
- Observability tools use
  - Counters
  - Profiling
  - Tracing
- For production environments, observability tools may perturb production workloads through resource contention

Counters, Statistics, Metrics, and Alerts (1)

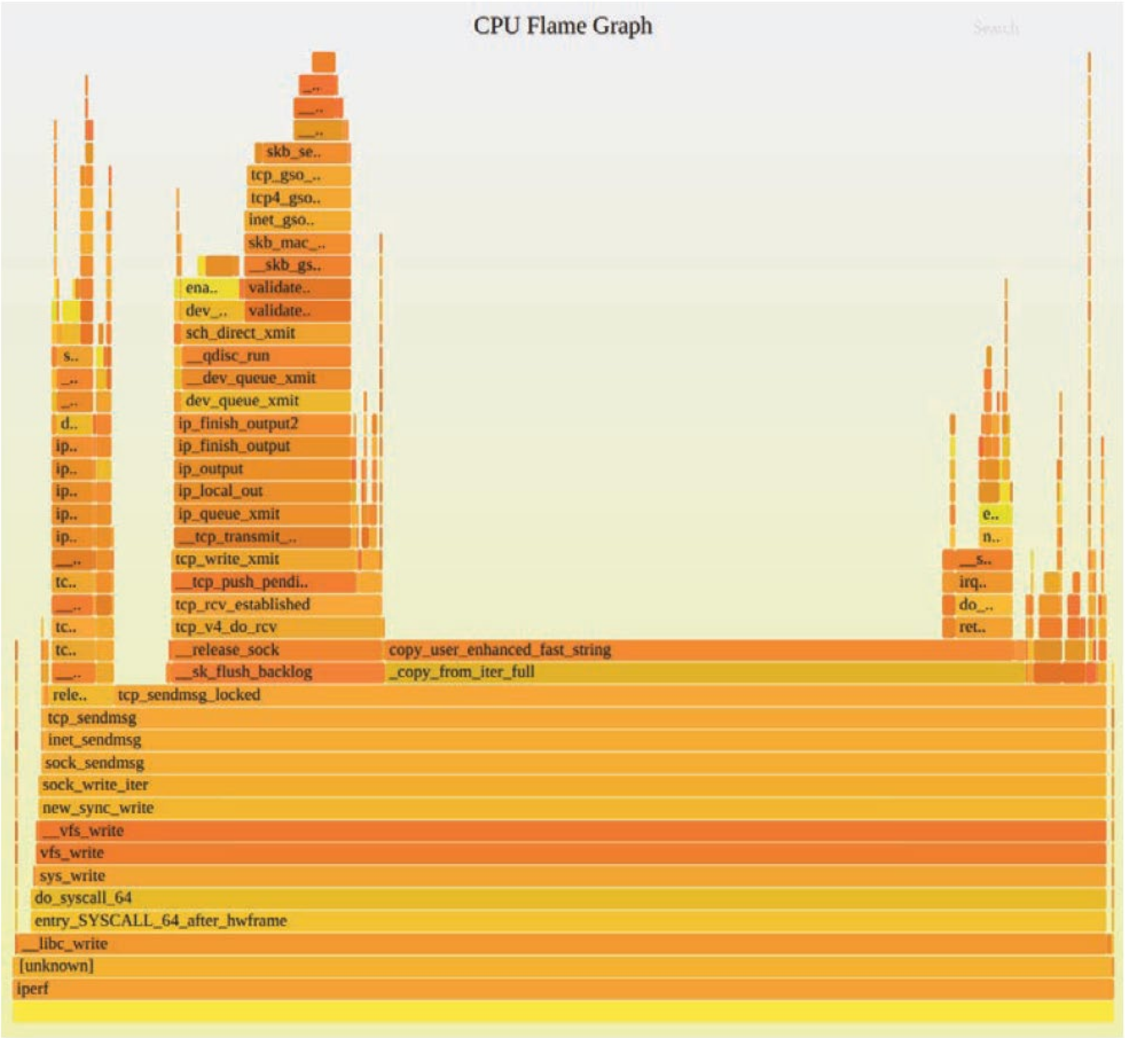
- Applications and the kernel typically provide data on their state and activity:
  - Example: operation counts byte counts, latency measurements, resource utilization, and error rates
  - Implemented as integer variables called counters
    - Hard-coded in the software
    - Some are cumulative, and they always increment
- Cumulative counters can be read at different times by performance tools for calculating statistics
  - Examples: the rate of change over time, the average, percentiles, etc.
  - Metric is a statistic that has been selected to evaluate or monitor a target
  - Monitoring software can also support creating custom alerts from these metrics

Performance Instrumentation Terminology



Profiling

- Refers to the use of tools that perform sampling: taking a subset (a sample) of measurements to paint a coarse picture of the target
  - CPUs are a common profiling target
    - Take timed-interval samples of the on-CPU code paths
- Visualizing the profiles is important
  - Example: CPU flame graphs



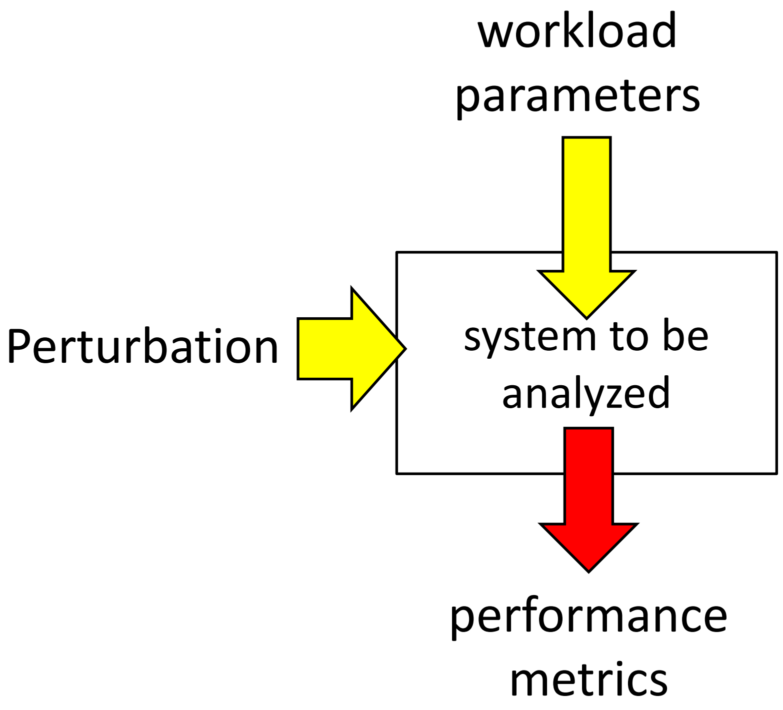
Tracing

- Event-based recording: event data is captured and saved for later analysis or consumed on-the-fly for custom summaries
  - Special-purpose tracing tools for system calls (Linux strace(1)) and network packets (e.g., Linux tcpdump(8))
  - General-purpose tracing tools that can analyze the execution of all software and hardware events (e.g., Linux Ftrace, BCC, and bpftrace).

- Static instrumentation: hard-coded software instrumentation points added to the source code
- Dynamic instrumentation creates instrumentation points after the software is running, by modifying in-memory instructions to insert instrumentation routines

Steps in Performance Evaluation Study

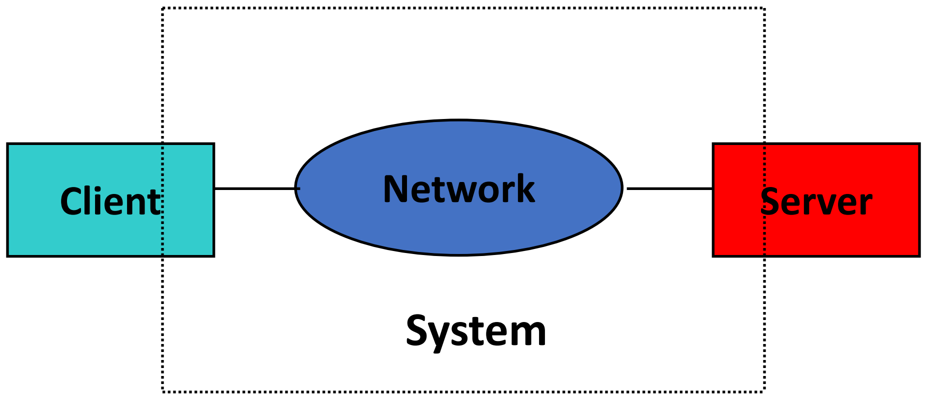
1. State the goals of the study and define the system boundaries
2. List system services and possible outcomes
3. Select performance metrics
4. List system and workload parameters
5. Select factors and their values
6. Select evaluation techniques
7. Select the workload
8. Design the experiments
9. Analyze and interpret the data
10. Present the results. Start over, if necessary



Example: Performance Evaluation Study

1. System definition:

- compares performance of applications using remote pipes (caller is not blocked) with remote procedure calls (calling program is blocked)



Example (2)

2. Service:

- two types of channel calls: remote procedure call and remote pipe

3. Metrics:

- resources are local computer (client), remote computer (server), and network link
- performance metrics:
  a. Elapsed time per call
  b. Maximum call rate per unit of time, or equivalently, the time required to complete a block of n successive calls
  c. Local CPU time per call
  d. Remote CPU time per call
  e. Number of bytes sent on the link per call

Example (3)

4. Parameters
   - i. workload parameters (resource demand)
     a. Time between successive calls
     b. Number and sizes of the call parameters
     c. Number and sizes of the results
     d. Types of channel
     e. Other loads on the local and remote CPUs
     f. Other loads on the network
   - ii. System parameters (resource capacity)
     a. Speed of the local CPU
     b. Speed of the remote CPU
     c. Speed of the network
     d. Operating system overhead for interfacing with the channels
     e. Operating system overhead for interfacing with the network
     f. Reliability of the network affecting the number of retransmissions required

Example (4)

5. Key Factors:
    a. Types of channel: 2 types – compares remote pipes and remote procedure calls
    b. Speed of the network: 2 remote host locations - short distance (in the campus) and long distance (across the country)
    c. Sizes of call parameters to be transferred: small and large
    d. Number n of consecutive calls: 11 values of n – 1, 2, 4, 8, 16, 32,…, 512, 1024

Factors are selected based on resource availability and objectives of evaluation

Example (5)

6. Evaluation Technique: Since both types of channels are available, measurements is used for evaluation. Analytical modeling is used to justify the consistency of measured values for different parameters.
7. Workload: a synthetic program generating the specified types of channel requests, monitor resources consumed and log the measured results.
8. Experimental Design: a full factorial experimental design consists of 88 (23 x 11) experiments
9. Data Analysis: Analysis of Variance to quantify the effects of the first three factors and regression to quantify the effects of the number n of successive calls
10. Data Presentation: final results plotted as a function of the block size n

Summary

- Motivation for performance analysis
- Tradeoffs
- Metrics
- Observability
- Steps in performance analysis

References

- [Gregg] – chapter 1
- [Jain] – chapters 2 & 3