

EEC 686/785

Modeling & Performance Evaluation of Computer Systems

Lecture 2

Wenbing Zhao

Department of Electrical and Computer Engineering
Cleveland State University

wenbing@ieee.org

http://academic.csuohio.edu/zhao_w/teaching/EEC685-F05/eec685.htm

(based on Dr. Raj jain's lecture notes)

A Systematic Approach to Performance Evaluation

1. State goals and define the system
2. List services and outcomes
3. Select metrics
4. List parameters
5. Select factors to study
6. Select evaluation technique
7. Select workload
8. Design experiments
9. Analyze and interpret data
10. Present results

Outline

- Review of lecture 1
- Part I: An overview of performance evaluation
 - A Systematic Approach to Performance Evaluation
 - Selection of techniques and metrics

Case Study: Remote Pipes vs. RPC

- **RPC (remote procedure call)** is an extension of local blocking procedure call, i.e., a program on one computer system calls a procedure object on another system
 - The calling program waits until the procedure is complete and the results is returned
- **Remote Pipes** are also procedure-like, but when called, the caller is not blocked
 - The execution of the pipe occurs concurrently with the continued execution of the caller. The results, if any, are later returned asynchronously

Case Study: Remote Pipes vs. RPC

■ Goal and system definition:

- The goal of the case study is to compare the performance of applications using remote pipes to those of similar applications using remote procedure calls
- The key component under study is the **channel**: a channel can be either a procedure or a pipe
- The system consists of two computers connected via a network. The requests are sent via the channel from the client computer to the server computer

Figure 2.1 omitted

Case Study: Remote Pipes vs. RPC

■ Metrics (continued):

- This leads to
 - Elapsed time per call
 - Maximum call rate per unit of time, or equivalently, the time required to complete a block of n successive calls
 - Local CPU time per call
 - Remote CPU time per call
 - Number of bytes sent on the link per call

Case Study: Remote Pipes vs. RPC

■ Services: the services offered by the system are the two types of channel calls – procedure call and remote pipe

- Data transfer is chosen as the application
- The calls are classified as small data transfer or large data transfer

■ Metrics:

- No errors and failures. Correct operation only
- Rate, time, resource per service
- Resource = client, server, network

Case Study: Remote Pipes vs. RPC

■ System parameters that affect the performance of a given application and data size are the following:

- Speed of the local CPU
- Speed of the remote CPU
- Speed of the network
- Operating system overhead for interfacing with the channels
- Operating system overhead for interfacing with the networks
- Reliability of the network affecting the number of retransmissions required

Case Study: Remote Pipes vs. RPC

- **Workload parameters** that affect the performance are the following:
 - Time between successive calls
 - Number of sizes of the call parameters
 - Number and sizes of the results
 - Type of channel
 - Other loads on the local and remote CPUs
 - Other loads on the network

Case Study: Remote Pipes vs. RPC

- Note:
 - Fixed: type of CPUs and operating systems
 - Ignore retransmissions due to network errors
 - Measure under no other load on the hosts and the network

Case Study: Remote Pipes vs. RPC

- **Factors:** The key factors chosen for this study
 - **Type of channel:** remote pipes and remote procedure calls
 - **Size of the network:** short distance and long distance
 - **Sizes of the call parameters:** small and large
 - **Number n of consecutive calls:** 1, 2, 4, 8, 16, ..., 1024

Case Study: Remote Pipes vs. RPC

- **Evaluation techniques:**
 - Prototypes implemented => measurements
 - Use analytical modeling for validation
- **Workload:**
 - Synthetic program generating the specified types of channel requests
 - Null channel requests => resources used in monitoring and logging

Case Study: Remote Pipes vs. RPC

■ Experimental design

- A full factorial experimental design with $2^3 \times 11 = 88$ experiments will be used

■ Data analysis

- Analysis of variance for the first three factors
- Regression for number n of successive calls

■ Data presentation

- The final results will be plotted as a function of the block size n

Three Rules of Validation

- Do not trust the results of a **simulation model** until they have been validated by **analytical modeling** or **measurements**
- Do not trust the results of an **analytical model** until they have been validated by a **simulation model** or **measurements**
- Do not trust the results of a **measurement** until they have been validated by **simulation** or **analytical modeling**

Selecting an Evaluation Technique

Criterion	Analytical Modeling	Simulation	Measurement
Stage	Any	Any	Post-Prototype
Time Required	Small	Medium	Varies
Tools	Analysts	Computer languages	Instrumentation
Accuracy	Low	Moderate	Varies
Trade-off evaluation	Easy	Moderate	Difficult
Cost	Small	Medium	High
Scalability	Low	Medium	High

Selecting Metrics

- For each performance study, a set of performance criteria or metrics must be chosen
- One way to prepare this is to list the services offered by the system
- For each service request made to the system, there are several possible outcomes:
 - Service provided correctly
 - Service provided incorrectly
 - No service is provided
- Examples: database server, gateway in a computer network

Three Possible Outcomes

Figure 3.1 omitted

Metrics for Successful Performance

- The three metrics related to **time-rate-resource** for successful performance are also called **responsiveness, productivity, and utilization** metrics
- The resource with highest utilization is called the **bottleneck**
 - Performance optimizations at the bottleneck offer the highest payoff

Metrics for Successful Performance

- If the system performs the service correctly, its performance is measured by:
 - The **time** taken to perform the service
 - The **rate** at which the service is performed, and
 - The **resources** consumed while performing the service

Metrics for Incorrect Outcome

- If the system performs the service incorrectly, an error is said to have occurred
- It is helpful to classify errors and to determine the probabilities of each class of errors
- Example: for a gateway, we want to find the probability of
 - Single-bit errors,
 - Two-bit errors, and
 - Partial packet delivery

Metrics for No-Service Outcome

- If the system does not perform the service, it is said to be down, failed, or unavailable
- It is helpful to classify the failure modes and to determine the probabilities of each class
- Example: a gateway may be
 - Unavailable 0.01% of time due to processor failure, and
 - Unavailable 0.03% due to software failure

Selecting Metrics

- In computer systems shared by many users, two types of performance metrics need to be considered:
 - Individual – reflects the utility of each users
 - Response time
 - Global – reflect the system-wide utility
 - Resource utilization, reliability, availability
- There are cases when the decision that optimizes individual metrics is different from the one that optimizes the system metric
 - Throughput in a computer network

Selecting Metrics

- The metrics associated with the three outcomes are also called **speed**, **reliability**, and **availability** metrics
- For many metrics, the *mean value* is all that is important. However, the effect of *variability* might also be important

Metrics Selection Criteria

- **Low-variability** – helps reduce the number of repetitions required to obtain a given level of statistical confidence
- **Non-redundancy** – if two metrics give essentially the same information, it is less confusing to study only one
 - For example, in computer networks, studying the average queue lengths in addition to average waiting time in a queue may not provide any additional insights
- **Completeness** – the set of metrics included in the study should be complete. All possible outcomes should be reflected in the set

Case Study:

25

Two Congestion Control Algorithms

- A computer network consists of a number of end systems interconnected via a number of intermediate systems
 - The end systems send packets to other end systems on the network
 - The intermediate systems forward the packets along the right path
- The problem of congestion occurs when the number of packets waiting at an intermediate system **exceeds** the system's buffering capacity and some of the packets have to be dropped

2 September 2005

EEEC686/785

Wenbing Zhao

Performance Metrics in Case Study

27

- For packets delivered in order, time-rate-resource =>
 - Response time: the delay inside the network
 - Throughput: number of packets per unit of time
 - Processor time per packet on the source end system
 - Processor time per packet on the destination end systems
 - Processor time per packet on the intermediate systems

2 September 2005

EEEC686/785

Wenbing Zhao

Case Study:

26

Two Congestion Control Algorithms

- **System:** the network
- **Service:** send packets from specified source to specified destination in order
- **Possible outcomes:**
 - Some packets are delivered in order to the correct destination
 - Some packets are delivered out-of-order to the destination
 - Some packets are delivered more than once (duplicates)
 - Some packets are dropped on the way (lost packets)

2 September 2005

EEEC686/785

Wenbing Zhao

Performance Metrics in Case Study

28

- A highly variant response might trigger a retransmission => Variability of the response time
- Out-of-order packets consume buffers => probability of out-of-order arrivals
- Duplicate packets consume the network resources => probability of duplicate packets
- Lost packets require retransmission => probability of lost packets
- Too much loss cause disconnection => probability of disconnect

2 September 2005

EEEC686/785

Wenbing Zhao

Performance Metrics in Case Study

- Shared resource => *fairness*

$$f(x_1, x_2, \dots, x_n) = \frac{(\sum_{i=1}^n x_i)^2}{n \sum_{i=1}^n x_i^2}$$

Fairness properties:

- Always lies between 0 and 1
- Equal throughput => fairness = 1
- If k of n users receive x and $n-k$ users receive 0 throughput: the fairness index is k/n

Commonly Used Performance Metrics

- We will introduce a number of commonly used performance metrics
- For each one, the definition provided is only one of many possibilities

Performance Metrics in Case Study

- Throughput and delay were found redundant => user **power**: power = throughput / response time
- Variance in response time redundant with the **probability of duplication** and the **probability of disconnection**
- Total nine metrics

Response Time

- *Simple definition*: the interval between a user's request and the system response
- *More realistic definition* – two alternatives
 - Interval between the end of a request submission and the beginning of the corresponding response from the system
 - Interval between the end of a request submission and the end of the corresponding response from the system

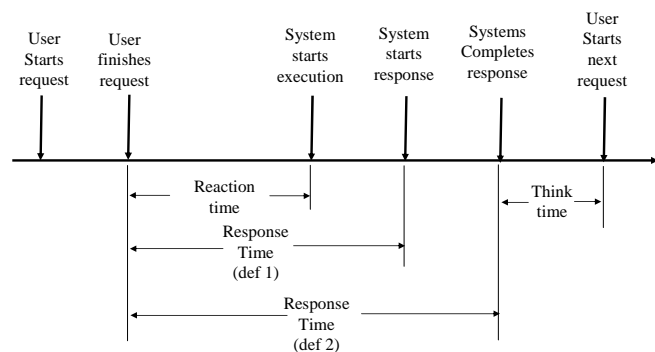
Reaction Time

- **Reaction time:** the time between submission of a request and the beginning of its execution by the system
 - To measure the reaction time, one has to be able to monitor the actions inside a system
 - The beginning of the execution may not correspond to any externally visible event
 - Example: the interval between a user's last key stroke and the user's process receiving the first CPU quantum would be called **reaction time**

Turnaround Time and Stretch Factor

- **Turnaround time** = the time between the submission of a batch job and the completion of its output
- **Stretch factor:** the ratio of the response time at a particular load to that at the minimum load

Response Time and Reaction Time



Throughput

- **Throughput** is the rate (requests per unit of time) at which the request can be serviced by the system
- Examples:
 - Jobs per second
 - Requests per second
 - Millions of Instructions Per Second (MIPS)
 - Millions of Floating Point Operations Per Second (MFLOPS)
 - Packets Per Second (PPS)
 - Bits per second (bps)
 - Transactions Per Second (TPS)

Capacity

Figure 3.3 omitted

Efficiency

- **Efficiency:** ratio usable capacity to nominal capacity
 - E.g., if the max throughput from a 100-Mbps LAN is only 95 Mbps, its efficiency is 85%

Capacity

- **Nominal capacity:** maximum achievable throughput under ideal workload conditions.
 - E.g., bandwidth in bits per second
 - The response time at maximum throughput is too high
- **Usable capacity:** maximum throughput achievable without exceeding a prespecified response-time limit
- **Knee capacity:** knee = low response time and high throughput
 - It is considered as the optimal operation point

Utilization

- **Utilization:** the fraction of time the resource is busy servicing requests
 - For some resources such as CPUs, they are either busy or idle
 - => $\text{utilization} = \text{busy time} / \text{total time}$
 - **Idle time:** The period during which a resource is not being used
 - For some other resources such as memory, a fraction of the resource may be used
 - => $\text{utilization} = \text{average fraction used over an interval}$
 - It is desirable to balance the load so that no one resource is utilized more than others

Reliability and Availability

- **Reliability**
 - Probability of errors
 - Mean time between errors (error-free seconds)
- **Availability**: the fraction of the time the system is available to service users' requests
 - **Uptime**: the time during which the system is available
 - **Downtime**: the time during which the system is not available
 - Mean Time to Failure (**MTTF**): mean uptime, a better indicator
 - Mean Time to Repair (**MTTR**)
 - Availability = $MTTF / (MTTF + MTTR)$

Setting Performance Requirements

- **Examples**:
 - “The system should be both processing and memory efficient. It should not create excessive overhead.”
 - “There should be an extremely low probability that the network will duplicate a packet, deliver a packet to the wrong destination, or change the data in a packet.”
- **Problems**: non-specific, non-measurable, non-acceptable, non-realizable, non-thorough

Utility Classification of Performance Metrics

- Higher is Better or
HB: Throughput
- Lower is Better or
LB: Response time
- Normal is best or
NB: Utilization

Figure 3.5 omitted

Setting Performance Requirements

- **Solution**: **SMART** – the requirements must be Specific, Measurable, Acceptable, Realizable, and Thorough
- **Specific**: precludes the use of words like “low probability” and “rare”
- **Measurable**: requires verification that a given system meets the requirements
- **Acceptable**: requirements are high enough to be acceptable
- **Realizable**: requirements are low enough to be achievable
- **Thorough**: includes all possible outcomes and failure modes

Case Study: Local Area Networks

- **Service:** Send frame to destination station D
- **Outcomes:**
 - Frame is correctly delivered to D
 - Incorrectly delivered
 - Not delivered at all

Case Study: Local Area Networks

- **Reliability Requirements:** five different error modes. Different amount of damage => different level of acceptability
 - The probability of any bit being in error must be less than $1E-7$
 - The probability of any frame being in error (with error indication set) must be less than 1%
 - The probability of a frame in error being delivered without error indication must be less than $1E-15$
 - The probability of a frame being misdelivered due to an undetected error in the destination address must be less than $1E-18$
 - The probability of a frame being delivered more than once (duplicate) must be less than $1E-5$
 - The probability of losing a frame on the LAN (due to all sorts of errors) must be less than 1%

Case Study: Local Area Networks

- **Speed Requirements:**
 - The access delay at any station should be less than one second
 - Sustained throughput must be at least 80 Mbits/sec

Case Study: Local Area Networks

- **Availability Requirements:** the fault modes. Network re-initializations and permanent failures
 - The mean time to initialize the LAN must be less than 15 milliseconds
 - The mean time between LAN initialization must be at least one minute
 - The mean time to repair a LAN must be less than one hour (LAN partitions may be operational during this period).
 - The mean time between LAN partitioning must be at least one-half a week
- All of the numerical values specified above were checked for realizability by analytical modeling

Summary of Part I

■ Systematic approach:

- Define the system,
- List its services, metrics, parameters,
- Decide factors, evaluation technique, workload, experimental design,
- Analyze the data, and present results

Summary of Part I

■ Selecting metrics:

- For each service list time, rate, and resource consumption
- For each undesirable outcome, measure the frequency and duration of the outcome
- Check for low-variability, non-redundancy, and completeness

- **Performance requirements:** should be SMART – specific, measurable, acceptable, realization, and thorough

Summary of Part I

■ Selecting evaluation techniques:

- The life-cycle stage is the key.
- Other considerations are:
 - Time available, tools available,
 - Accuracy required,
 - Trade-offs to be evaluated,
 - Cost, and
 - Salability of results