lecd # 1

Date: CH # 1

M T W T F S S

COMP SYSTEMS PERFORMANCE MODELING & EVALUATION.

* Performance Evaluation is about quantifying the service delivered by a computer system.

Eg:
- comparing the power consumption of several server farm configurations.
- knowing the response time experienced by a customer performing a reservation over the internet.
- comparing compilers for a multiprocessors machine.

* Common goals of Performance Evaluation:

1) compare alternatives : when purchasing a new comp. sys. you may be consented with several diff. systems from which to choose. you may've several diff. options within each system that may impact both cost of performance such as size of MM, no. of processors, type of n/w interface, etc In this case, goal of performance analysis task is to provide quantitative info. about which configuration is best

2) impact analysis

↳ In designing new systems or in upgrading existing systems, you often need to determine the impact of adding or removing the specific features of the system. For instance, the designers of a new processor may want to understand wether it make sense to add an additional sorting point execution unit to the microarchitecture or wether the size of the on-chip cache be increased instead. This type of analysis is often refered to as 'Before & After Comparison' since only one well defined component of the system is changed.

Date: M T W T F S S

3) System Tuning :- The goal of performance analysis of system tuning is to find set of parameter values that produces the best overall performance. In time shared OS, it is possible to control the no. of processes that allowed to actively shared the process-x. The overall performance perceived by the system users can be substantially impacted both by this number & by the time quantum allocated to each process.

Many other system parameters such as disk & n/w buffer sizes for eg. can also significantly impact the system performance. Since the performance impacts of these various parameters can be closely interconnected finding the best set of parameter value can be a very difficult task.

4) Identifying relative Performance

The performance of a comp. sys. typically has meaning only relative to something else such as another sys. or another configuration of the same sys. The goal in this situation may be to quantify the change in performance relative to history i.e. relative to previous generation of the system. Another goal may be to quantify the performance relative to customers expectations or to a competitor system.

5) Performance Debugging Debugging the pgm for correct exec. is a fundamental pre-requisite for any application pgm. Once the pgm is functionally correct however the performance analysis task becomes one of finding performance probs i.e. the pgm not produces the correct result but it may be much slower than desired. The goal of perf. analysis at this point is to apply the appropriate tools & analysis

techniques to determine why the pgm is not meeting pe
Date: M T W T F S S
expectations.

6) Setting Expectations

Users of comp systems may've some ideas of what the capabilities of the next generation of a line of comp systems should be. The task of the perf. analysis in this case is to set the appropriate expectations for what a sys. is actually capable of doing

↳ In all of these cases, the effort involved in the performance analysis task should be proportional to the cost of making a wrong decision.

Eg: Suppose you are comparing diff. manufacturing systems to determine which best satisfies the requirements for a large purchasing decision.

The financial cost of making the wrong decision could be the quite substantial both in term of the cost of the sys itself & in terms of the subsequent impacts on the various parts of a large project or organization.

In this particular case you'll probably want to perform a very detail throughly analysis.

However, if you are simply trying to choose a sys for your own personal use. The cost of choosing the wrong one is minimal. Your performance analysis in this case may be correctly limited to reading a few reviews from a trade magazines.

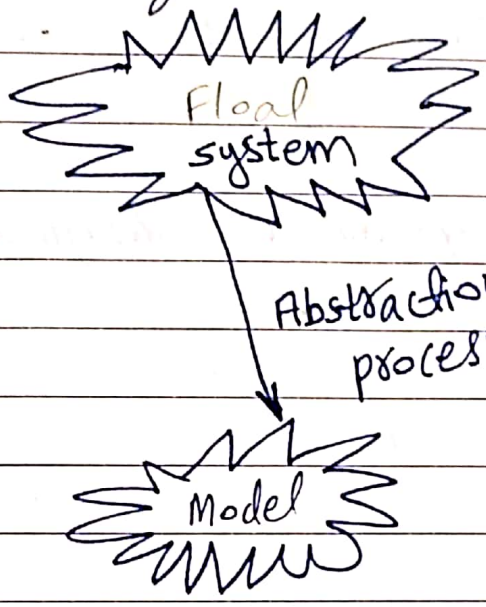
* MODEL

Modeling is the enterprise of device-ing a simplified representation of a complex system with the goal to capture certain behavioural aspects of the no sys.

Date: M T W T F S S

for Eg. providing predictions of the system's performance measures matrices of interest. Such a simplified representation is called a Model.

- A model is an abstraction of a real-world system used to gain insight into some critical aspect/s) of a system.



Modeling cause for abstraction & simplification.

Infact if each & every detail of the sys. under study were to be reproduce then the model's cost may approach that of the model system. There by nullifating against creating a model in first place.

They may have various forms,

- Physical models (scaled replicas)
- mathematical equations & relations (abstractions)
- graphical representations.

While modeling is ultimately motivated by economic considerations. Hence, the other motivational factors are,

* Motivation for Modeling.

- Economic Considerations

Evaluating systems performance under unusual scenarios.

A model may be a necessity if the routine operation of the real life system under study can not be disturbted w/o severe consiquences.

Date:
M T W T F S S

For Eg: attempting an upgrade of a production line in the middle of filling customer orders with tight deadlines. In other cases, the extreme scenarios modeled is to be avoided at all cost

- **Evaluating system performance under unusual scenarios**
  - upgrading a plant during operation
  - modelled scenario to be avoided at any cost.

Example: Think of modeling crash of mid air aircraft or core meltdown in a nuclear reactor

- **Predicting the performance of experimental design decisions**

When the underline system doesn't yet exist model construction & manipulation is far cheaper & safer than building the real life system or even its prototype.

Fundamental design decisions may effect performance of a system. A model can be used as part of the design process to avoid that decisions & to help quantify a cost benefit analysis.

* F stories appear periodically in the media on projects that'll rush to the implementation phase w/o proper verification that there design is adequate only to discovered that system was flawed to 1 degree or another

- **Ranking multiple designs & analysing their trade-offs.**

This case is related to the previous one except that the economic motivation is even greater.

A manufacturer may have a range of compatible systems with diff. characteristics i.e. which config. would be best for a particular application.

Date: M T W T F S S

In a nutshell, a model can be used to investigate a wide variety of what if questions about real world system. Potential changes to the system can be stimulated & their impact on the system can be predicted & thus we find adequate parameters before implementation. Hence, modeling can be used as an analysis tool for predicting the effect of changes & a design tool to predict the performance of new system.

* Modeling Tools (Techniques of Performance Evaluation)

The major classes of modeling tools we today are

1) Analytical
2) Simulation
3) Testbed
4) Operational Analysis

1) Analytical Modeling Tools

- involves constructing a mathematical model of the system behaviour (at the desired level of detail) and solving it.
- analytical modeling tools works for simpler systems.
- For example,
  - queuing models
  - Petri nets for problem solving.
- A simple analytical model of the overall average memory-access time observed by an executing pgm is:

$$t_{avg} = h.t_c + (1-h).t_m$$

i.e. the avg mem access time is the sum of time delay observed by the mem. reference in cache multiplied by the hit ratio & the corresponding

Date:
M T W T F S S

delay if the reference location is not in the cache multiplied by miss ratio.

So, In order to apply this simple model to a specific application pgm, we would need to know the hit ratio (h) for the pgm & the values of tc and tm.

The memory access parameters tc & tm may often be found in the manufacturer's specifications of the system.

The hit ratio (h) for an application pgm is often found through a simulation of the application.

This model can provide us with some insights into the relative effects of increasing the hit ratio or changing the memory timing parameters for instance.

Advantages

-highly flexible & low in cost & it generates a good insights into the workings of the system but the major drawback is that the results of analytical model tend to be much less believable & much less accurate.

2) Simulation Modeling Tools

- It is a dynamic tool that uses a computer to imitate the operation of an entire process or system.
- A computer model with random elements and an underlying timeline is called Monte-Carlo simulation. For eg. the operation of a manufacturing process over a period of time.
- It involves development of model of systems, then experiments are conducted with an appropriate abstraction of workload.

Date: M T W T F S S

- Simulation is a powerful technique for studying memory-system behaviour due to its high degree of flexibility. For e.g. to study the impact on performance.
  - the sizes of cache & memory
  - the relative cache & memory delays
  - to study the effectiveness of pipelining in CPU.
- Advantages:
  they are highly flexible & there cost is slightly higher than that of analytical modeling tool but the results are less believable & less accurate.

~~COMPARE~~ Comparison b/w Analytical Modeling & Simulation

- Suppose the example of distance travelled in particular time duration.
- Making simplifying assumptions, then the given analytical model is appropriate,

$$S = V * t$$

- Taking friction, acceleration of other factors into account, it will involve complex differential equations.
- Such equations are hard to solve analytically therefore, simulation is preferred in this case.

3) Testbeds.

- A realistic hardware-software environment for testing components without having the ultimate system.
- For e.g. New aircraft engines are fitted to a testbed aircraft for flight testing.
  here, the environment for testing the new aircraft ~~system~~ engine is actually a testbed.

Date:
M T W T F S S

* the thing to be tested is not a testbed offcourse.
- Its important feature is that it only focuses on a subset of the total system.
* The other aspects are just the simulated pieces that provide their stimulous or input.
- Examples of Testbed in the content of software development & computer networks.

In s/w development, Testbeding is a method of testing a particular ~~model~~ module function or class in an isolated fashion. i.e apart from the system it will later be added to.

So, a skeleton framework is implemented around the module so that it behaves as the part of a the larger system.

Testbeds in the content of computer n/ws are used to analyze a wide range of components For Eg. Testing the developed protocols & applications on simulators give inaccurate results. So solution is to develop testbed for execution of new protocols & applications.

- Testbed is made up of three components
  - Experimental subsystem → collection of real world system components & prototype that we which to model
  - Monitoring subsystem
  - Simulation- Stimulation Subsystem → it allows the experimentor to submit i/ps & get the o/ps for experimentation.

to ← collode & analyse the collected information

Date: M T W T F S S

The testbed approach provides a method to investigate system aspects that are complimentary to simulation & analytical model.

- the advantage of testbeds is they improve the understanding of functional requirements & operational behaviour of elements of the system. But there cost is more & they are limited in applications.

For e.g. we should'nt model a complex distributed computing system in a testbed. Instead we'll use analytical or simulation model at the first as the first pass & use a testbed b/w the initial concept & the final design.

4) Operational Analysis (Direct Measurement)

- It is the measurement & evaluation of an actual system in operation.
- It is also used to develop projections about the system's future operations.
- It involves instrumenting the system to extract the information.
- In an actual system, it may be very difficult (or impossible) to change the parameters.

For Eg. Evaluating the performance impact of varying the speed of main memory is simply not possible in most real systems. So,

- It can be very time consuming and costly. but they have high believability & very accurate results.

Date: M T W T F S S

System.

Operational Analysis

Experiment with the actual system (Direct measurement)
↳ may be too costly or disruptive
↳ Not appropriate for the design phase.

Experiment with the model of the system

Mathematical Model

Physical Model

Analytical Model (solution)
↳ if the model is simple enough; e.g. calculus, algebra, probability theory.

Simulation
↳ Highly Complex Systems

* Intentionally omitted the testbed part bcz they actually focuses on the subsystem of the larger system.

Key Differences among the various Modeling Techniques:

Characteristics	Analytical model	Simulation	Testbed	Direct measurement
Flexibility	High	High/Medium	Medium/Low	Low
Cost	Low	Medium	" /high	High
Believability	Low	"	" / "	High
Accuracy	Low	"	Varies	High
Time Required	Small	"	Varies	Varies

~~Amdhal~~
Amdahl's law

1) linearity:

Date: M T W T F S S

* Emulation

- An emulator is hardware or software that enables one computer system (called the host) to behave like another computer system (called the guest). An emulator typically enables the host system to run software or use peripheral devices designed for the guest system.
- Emulation allows program to run on the platform other than the one for which they were originally developed.
- For example:
  - Many printers are designed to emulate the laser jet printers because so much softwares written for hp printers.
  - Android Emulator
  - DOSBox emulates the command-line interface of DOS.

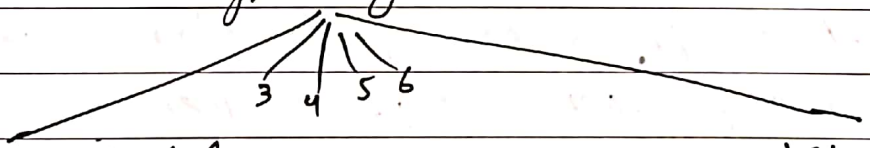
~~Since, 1990s~~

many vedio games ethusiates have used emulator to play classic arcade games.

~~lect #2~~

week 1

Types of models



1) Dynamic Model
↳ represents systems that evolve with time.
Eg: A model to gauge performance of a search engine.

2) Static Model
↳ Those systems that don't evolve with time.
Example: DC voltage-current relationship.

Date:
M T W T F S S

3) Deterministic Model
↳ These are no probabilistic components in the systems.
↳ Example : The worst case analysis of any algorithm is always predictable.

4) Stochastic Model
↳ At least one component has probabilistic behaviours.
↳ Example : Queuing systems.

5) Continous Model
↳ System's state changes continously.
Eg : A chemical process

6) Discrete Model
↳ State changes only at discrete points in time.
Eg : Inventory Model.

* Performance Metric

While measuring the cost of a system is often relatively straight forward determining the performance of a computer system can often time seem like a difficult task.

One of the pb is that people often disagrees strongly on how performance should be measured or interpreted & even on what performance actually means.

'Metric is an actual value that is used to describe performance'

There are two types of performance metric:
- system Oriented
- User Oriented

i) System Oriented Measures: These are the metrics imp. from the system administrator's perspective.

i.1 • Throughput → no. of jobs completed in unit time. Its nature of job depends on the content or application. This metric gives the productivity of the system.

Date: M T W T F S S

* Representatives Examples:
↳ the types of various systems & their associates metrices:

System	Metric
a) Online transaction Processing Sys. (OLTP)	Transactions per second (tps)
b) Website	HTTP requests/second Page view per second Bytes per second.
c) E-Commerce Site	Web Interactions Per Second (WIPS) sessions per second searches per second.
d) Router	Packets Per second → associative with router
e) CPU	? Millions of Instructions Per second (MIPS)
f) Disk	IIOs Per second & KB transfered/sec
g) E-mail server	messages sent per second.

i.2 • Resource Utilization (ρ)
↳ It indicates the fraction of time the resource is busy serving requests.

System managers often want to balance resources to have same utilization. Eg: the load balancing on CPUs bit often this is difficult to achieve when i/p o/p becomes bottleneck.
So, ↳ Resource Utilization can be defined in a diff. matter as,

Date:
M T W T F S S

$e(\%) = \frac{\#\text{ of busy processors}}{\text{Total \# of processors}}$, $p(\%) = \frac{\text{Memory used}}{\text{Total Memory}}$.

↳ Resources with the highest utilization (e) values are considered as bottleneck. Performance optimization at this resource offers the highest payoff.

↳ Finding the utilization of various resources inside the system is an important part of performance evaluation.

ii) User-Oriented Measures

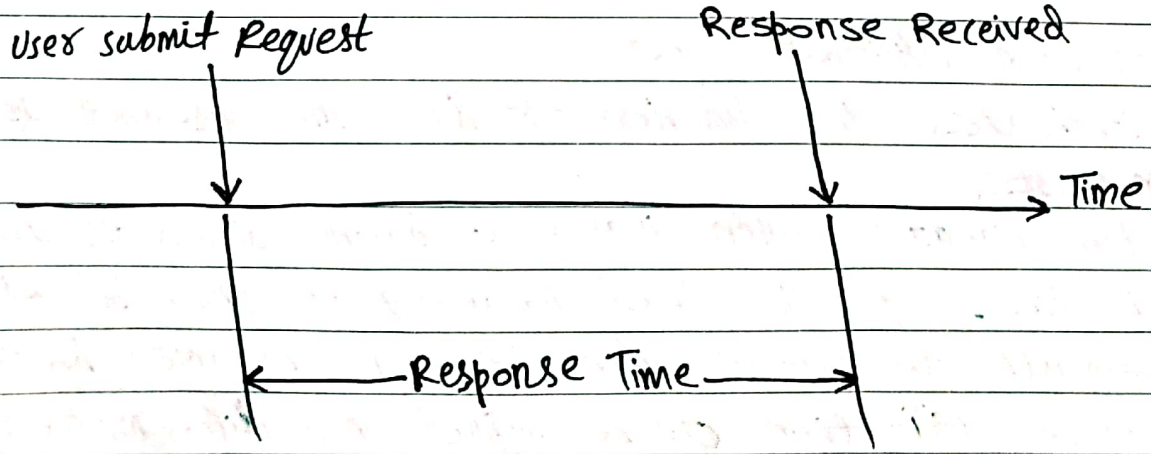
This is the metric important from the user's perspective.

Example is,

ii.1) Response time

↳ The time elapsed b/w the request submitted & system's response received by the user.

↳ For a batch stream, responsiveness is measured by "turnaround time." i.e. the time b/w the submission of a batch job & the completion of its o/p.



~~Amdhal~~
Amdahl's law

Date: M T W T F S S

* workload or load (WL)
  - ↳ The quantity & nature of requests submitted to system during some given period of time to model or drive the system.
  - ↳ Eg. the no. of requests submitted to database server per second i.e. the intensity of workload.
  - ↳ WL also depends on nature of request.

  Eg: All database request are not equivalent

  Eg: No. of instructions per second & the mix of instructions types presented for execution per second.

Duration of the load

- ↳ Duration may be all at once, requiring the system to queue up the requests & perform them as resources become available.

In comp. systems environment, all of the resources are not available all the time so, we queue up all the workload & then as the resources become available we serve them.

- ↳ Duration could be endless, with the load continually refreshed to provide a constant saturation load to the system.

It is also possible that we are always recieving some inputs from the outside world this situation is called the "Endless duration of load"

- ↳ loads can be periodic, where the load is reentered after the prescribed period of time.

load is constructed based on the focus of the analysis. for eg: If we are interested in comparing the h/w of ~~two~~ comp. systems, our focus may be on the lower instructions.

Date:
M T W T F S S

Another example is to measure transaction throughput through some database system. So, this workload would have transactional that read & write data items from database system & do some additional computational work the real system.

* Controllable Vs Uncontrollable parameters of workload

Controllable parameters are those that can be controlled by the system designer/administrator. Eg. the scheduling disciplines, inter-connections b/w devices & resource allocation policies.

Uncontrollable parameters, e.g., the inter-arrival times & service demands of incoming jobs.

These i/ps can be use for driving the real system or its simulation model & for determining distributions for analytic or simulation modeling. All these inputs are refered to as 'workload'.

There is a science to workload development & selection for the comp systems modelar. For eg. the database community has developed a s of transaction workload tp benchmarks for testing a variety of database system configuration.

Likewise the PC industry has developed a set of systems workload allowing customers to access the performance of one comp. architecture against another.

Date: M T W T F S S

Workload Characterization

↳ It involves deciding about
- different aspects of the workload (controllable & uncontrollable parameters etc),
- level of detail for recording the workload, and
- representation of the workload.

The workload characterization pb becomes more difficult when designing or evaluating a system for an anticipated need. In this case, we must generate the represented workload artificially.

↳ Workload characterization only builds a model of real workload, since not every aspect of real workload may be captured or is relevant.

Analytical modeling usually requires a rather abstract workload model whereas simulation can use a very detail model but for the reasons of cost effectiveness & robustness. It is desirable to keep the model & hence the workload as abstract as possible.

↳ For reasons of cost-effectiveness & robustness, it is desirable to keep model (hence the WL) as abstract as possible.

For Eg: when evaluating the ~~effiliveness of~~ effectiveness of an instruction pipeline or cache we need information of instruction sequences in addition to the relative frequency of various instructions. However in this particular case, the actual instruction traces are not necessary for the effective workload characterization. It may be adequate to capture only the first order sequencing effects represented by the probabilities or the relative frequencies of the form. lets suppose P_{ij} which gives the probability that the j^{th} instr. will ~~be~~ immediately follow the i^{th} instr.

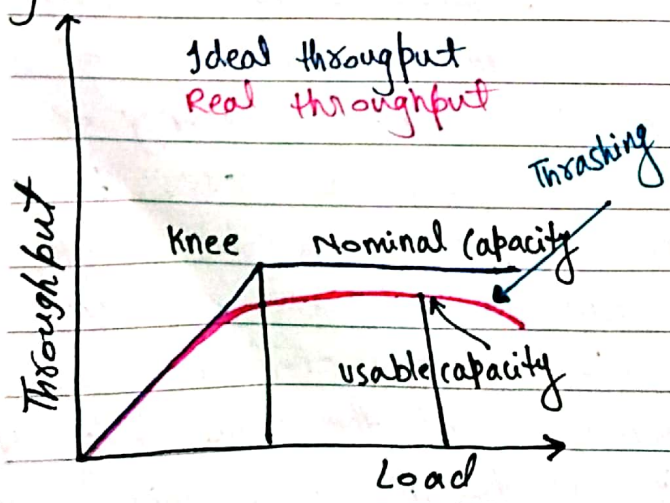
Date:
M T W T F S S

So, the first order sequencing effects can be represented by the probabilities in the form of Pij which gives the probability that the jth instr. will immediately follow the ith instr. This info. is enough for the workload characterization & we dont need the actual instruction traces. Such info. is not only less expensive to collect but is also less sensitive to minor changes in the workload

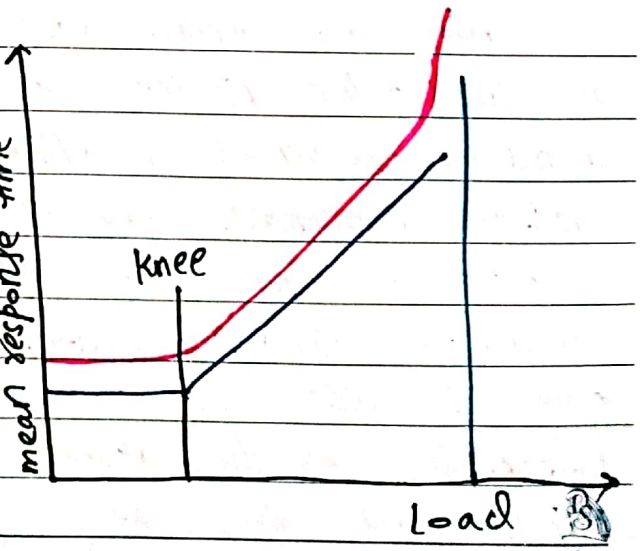
↳ A workload model may be executable or non-executable

- E.g. recording the arrival instants & service durations of jobs creates an executable model
- whereas only determining the distributions creates a non-executable model.
- An executable model need not be a record of i/ps, it can also be a program that generates the inputs
- Executable workloads are useful in direct measurements & trace-driven simulations.
- Non-executable WLs are useful for analytic modeling & distribution-driven simulation.

Throughput Vs. workload



Mean response time Vs workload



Date: M T W T F S S

* the set of all i/ps recieved during a specific interval of the is called its WL. For eg. database queries, http requests & so on.
* the throughput of the sys. generally increases as the load on the sys. initially increases.
* Graph: After certain load, the throughput stops increasing in most cases it would even start decreasing & this phenomena is called 'Thrashing'.
* The maximum achievable throughput under ideal workload conditions is called 'Nominal capacity of the system.

often the response time at maximum throughput is too high to be acceptable. In such cases, it is more interesting to know the maximum throughput achievable w/o exceeding a pre-specified response time limit. This may be called the 'usuable capacity' of the system.

In many applications, the 'knee' of the throughput or the response time curve is considered as the 'optimal operating point'. Knee is the point beyond which response time increases rapidly as the function of load. but on the other hand the gain in throughput is small.

Before the knee point, the response time doesn't increase significantly but the throughput rises as the load increases.

* The throughput at the knee is called 'knee capacity' of the system. It is also common to measure capacity in terms of load for eg: the no. of users rather than the throughput.

Date:
M T W T F S S

* Efficiency

↳ It is defined as the ratio of usable performance to maximum (theoretical/nominal) performance.

* Stretch factor

The ratio of response time at a particular load to that at the minimum load is called 'stretch factor'.

For a time sharing system, for example, stretch factor is defined as the ratio of response time with multiprogramming to that w/o multiprogramming.

Speedup & relative change are also termed as normalisation measures, & are particularly imp. for comparing alternatives.

* Execution Time.

This is the ultimate performance measure. & this is the measure of response time type.

The usage of term 'execution time' is privelent & rather reserve for processor's performance. Since, we are ultimately interested in how quickly a given pgm is executed. the fundamental performance metric of any comp sys. is the time required to execute a given application pgm. However w/o a precise & accurate measure of time it is impossible to analyse or compare system performance characteristics.

Date: M T W T F S S

Consequently, it is imp. to know how to measure the execution time of a pgm or a portion of pgm & to understand the limitations of the measuring tool.

i) The Wall Clock Time:

The wall clock time measures the total time that a user would have to wait to obtain the results produced by the pgm. That is, the measurement includes the time spent waiting for input/output operations to complete, memory paging & other system operations performed on behalf of this application, all of which are integral components of the program's execution.

when a pgm is executed it doesn't spend whole time writing on the processor instead it also spends time waiting for some system's task. for eg. paging or I/O to finish. The total time ellapsed is known as wall clock time. In a time sharing system it also includes waiting on ~~waiting on~~ other pgms as well.

The sum analyst calls it an unfair game to include time sharing overhead in total pgm execution time. It is argued that rather CPU time must be used in its place which is the measure of time a process spends just executing on processor.

ii) CPU time:

This is the time CPU spends on process execution.

However, balance approach is ~~used~~ to report both the wall clock time & CPU time to reflect the interference of time sharing.

- Both are reported generally.
- Measured a programs total elapsed execution time several times & report atleast the mean & variance of the times.

Date:
M T W T F S S

However, a balance approach is to report both the wall clock time & the CPU time to reflect the interference of time sharing.

In addition to system overhead effects, the measured execution time of an application pgm can vary significantly from one run to another. since the pgm must attend with random events such as the execution of background operating system task, diff. virtual to pysical page mapping & cache mappings, variable system load in a time shared systems & so forth.

As a result the pgm execution time is non-deterministic. Its imp. than to measure a pgm's total elapsed execution time several times & report atleast the mean & variance of the times.

As execution time satisfies all of the characteristics of a good performance metric. It is linear, reliable, repeatable, easy to measure, consistent. across system & independant of outside influences.

Therefore Execution time is one of the best metrics to use when analyzing comp. sys. performance.

Normalization Measures

* Speedup:

The speedup of system 2 w.r.t system 1 is denoted as $S_{2,1}$ & defined as,

$$S_{2,1} = R_2/R_1$$

where R is the rate/throughput metric.

If you want to define the speedup in terms of execution or response time. we can define as rate is the ratio of work done over time

Date: M T W T F S S

If 'W' is the work done then, $R = WT$ W/T

let $W_1 = W_2 = W$

$$S_{2,1} = \frac{W/T_2}{W/T_1} = \frac{T_1}{T_2}$$

If sys 2 is faster than sys 1, then the execution time of sys 2 will be less than system 1 & the speedup of sys 2 w.r.t sys 1 will be greater than sys 1.

- If '2' is faster than '1', $T_2 < T_1$ and $S_{2,1} > 1$
- If '2' is slower than '1', $T_2 > T_1$ and $S_{2,1} < 1$

speedup tells us how much better or worse a system is w.r.t another system.

* Relative change :

The relative change in performance of system 2 w.r.t sys 1 is defined as, $\Delta_{2,1} = \frac{R_2 - R_1}{R_1}$

In terms of response time & execution time,

$$\Delta_{2,1} = \frac{W/T_2 - W/T_1}{W/T_1}$$

$$\Delta_{2,1} = \frac{T_1 - T_2}{T_2}$$

$\Delta_{2,1}$ is positive if system 2 is faster.

$\Delta_{2,1}$ is negative if system 2 is slower.

Date:
M T W T F S S

Re-visiting Throughput Metrics

Millions of Instructions Per second (MIPS)

- It is an attempt to develop a rate metric from computer systems that allows a direct comparison of their speeds.
- If a machine executes IC instructions in time ET then MIPS metric can be calculated as

$$MIPS = \frac{IC}{ET * 10^6}$$

The pb with MIPS as a performance metric is that diff. processors can do substantially diff. amounts of computation with a single instruction. Eg. One processor may have a branch instruction that branches after checking the state of a specified condition code bit. Another processor on the other hand, may have a branch instruction that first decrements a specified count register, & the branches after

Millions of Floating-point operations per second (MFLOPS)

- It is an attempt to correct the primary shortcoming of the MIPS metric by more precisely defining the unit of 'computation' performed by the computer system when executing a program.
- If a machine executes OP floating point operations in time ET then MFLOPS metric can be calculated as:

$$MFLOPS = \frac{OP}{ET * 10^6}$$

Date: M T W T F S S

comparing the resulting value in the register with zero.

In both cases the branch instruction is executed but in the first case, a single instruction does one simple operation whereas, in the second case, one instruction actually performs several operations. The failure of MIPS metric is that each inst. corresponds to one unit of computation even though in this eg the scnd inst. actually performs more real computation.

* The MFLOPs metric is a definite improvement over the MIPS metric since the results of a floating-point computation are more clearly comparable across computer systems than is the execution of a single instruction. An imp. pb with this metric, however is that the MFLOPs rating for a system executing a program that performs no floating-point calculations is exactly zero. This program may actually be performing very useful operations, though such as searching a database or sorting a large set of records.

So, both of these metrics are unreliable & inconsistent.

Characteristics of a Good Performance Metric

i) Linearity: a metric is linear if its value is proportional to actual performance. (Book 1 pg 33...)

ii) Reliability: a metric is reliable if system A outperforms system B always when the corresponding values of the metric indicated so. (book 1)

Date:
M T W T F S S

iii) Repeatability: reproducible and deterministic. (book 1)

iv) Ease of Measurement: if not easily measurable; chances are that it will be measured incorrectly & analyst will not use it. (book 1)

v) Consistency: a metric is consistent if its definition is same across all system configuration. (book 1)

vi) Independence: a metric should be insensitive to tweaking by vendors to be used in their benefit in order to befool the customer. (book 1)

1. Linearity::
Since humans intuitively tend to think in linear terms, the value of the metric should be linearly proportional to the actual performance of the machine. That is, if the value of the metric changes by a certain ratio, the actual performance of the machine should change by the same ratio.
This proportionality characteristic makes the metric intuitively appealing to most people.
For example, suppose that you are upgrading your system to a system whose speed metric (i.e. execution-rate metric) is twice as large as the same metric on your current system. You then would expect the new system to be able to run your application programs in half the time taken by your old system. Similarly, if the metric for the new system were three times larger than that of your current system, you would expect to see the execution times reduced to one-third of the original values.
Not all types of metrics satisfy this proportionally requirement. Logarithmic metrics, such as the dB scale used to describe the intensity of sound, for example, are nonlinear metrics in which an increase of one in the value of the metric corresponds to a factor of ten increase in the magnitude of the observed phenomenon. There is nothing inherently wrong with these types of nonlinear metrics, it is just that linear metrics tend to be more intuitively appealing when interpreting the performance of computer systems.

2. Reliability.::
A performance metric is considered to be reliable if system A always outperforms system B

While this requirement would seem to be so obvious as to be unnecessary to state explicitly, several commonly used performance metrics do not in fact satisfy this requirement.
examples are MIPS and MFLOPS ratings
it is not unusual for one processor to have a higher MIPS rating than another processor while the second processor actually executes a specific program in less time than does the processor with the higher value of the metric. Such a metric is essentially useless for summarizing performance, and we say that it is unreliable.

3. Repeatability.::
A performance metric is repeatable if the same value of the metric is measured each time the same experiment is performed. Note that this also implies that a good metric is deterministic.

4. Easiness of measurement.::
If a metric is not easy to measure, it is unlikely that anyone will actually use it. Furthermore, the more difficult a metric is to measure

4. Easiness of measurement.::
If a metric is not easy to measure, it is unlikely that anyone will actually use it. Furthermore, the more difficult a metric is to measure directly, or to derive from other measured values, the more likely it is that the metric will be determined incorrectly. The only thing worse than a bad metric is a metric whose value is measured incorrectly.

5. Consistency.::
A consistent performance metric is one for which the units of the metric and its precise definition are the same across different systems and different configurations of the same system. If the units of a metric are not consistent, it is impossible to use the metric to compare the performances of the different systems. While the necessity for this characteristic would also seem obvious, it is not satisfied by many popular metrics, such as MIPS and MFLOPS.

6. Independence.::
Many purchasers of computer systems decide which system to buy by comparing the values of some commonly used performance metric. As a result, there is a great deal of pressure on manufacturers to design their machines to optimize the value obtained for that particular metric, and to influence the composition of the metric to their benefit. To prevent corruption of its meaning, a good metric should be independent of such outside influences.

lec #3.

Date: M T W T F S S

Amdahl's law

Amdahl's law states that in parallelization, if f is the proportion of a system or program that can be made parallel, and 1-f is the proportion that remains serial, then the maximum speedup that can be achieved using 'n' number of processors is given by,

$$S = \frac{1}{1 - f + f/n}$$

$$S_{(max)} = \lim_{n \to \infty} \frac{1}{1 - f}$$

It is true that only part of any program can be made parallelizable & the serial part will remain uneffected that has to be executed serially. If we are able to parallelize any part of pgm then we can exploit the advantages of multiprocessing environment. Thus, the overall speedup of that particular application will be enhanced.

Q/ suppose that enhancement is achieved which runs 10 times faster than the original machine but is only usable 40% of time. Determine the speedup.

lec #3.

Date: M T W T F S S

Amdahl's law

Amdahl's law states that in parallelization, if f is the proportion of a system or program that can be made parallel, and 1-f is the proportion that remains serial, then the maximum speedup that can be achieved using 'n' number of processors is given by,

$$S = \frac{1}{1 - f + f/n}$$

$$S_{ideal} = \lim_{n \to \infty} \frac{1}{1 - f}$$

It is true that only part of any program can be made parallelizable & the serial part will remain uneffected that has to be executed serially. If we are able to parallelize any part of pgm then we can exploit the advantages of multiprocessing environment. Thus, the overall speedup of that particular application will be enhanced.

Q/ suppose that enhancement is achieved which runs 10 times faster than the original machine but is only usable 40% of time. Determine the speedup.

Date:
M T W T F S S

Amdahl's law – Pitfall

Improving an aspect of machine's performance and expecting a proportional improvement in overall performance.

However, it is not possible. The time can only be improved by a certain improvement factor & the remaining portion will always remain unaffected. It's the same portion that will remain serializable & can not be made parallelizable.

$$T_{improved} = \frac{T_{affected}}{\text{Improvement factor}} + T_{unaffected} \quad \text{— (1)}$$

Example: Suppose that a program runs in 100 seconds on a machine with multiply responsible of 80 seconds of this time.
↓ instructions

↳ Q/ How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?

↳ Qb) How about making it 5 times faster? State your conclusion as well.

Solution (a)

To be 4 times faster the program should run in

$100/4 = 25$ seconds

$\therefore T_{improved} = 25$ seconds

Date: M T W T F S S

$T_{affected}$ = 80 seconds → portion responsible for the multiplication instr. part.

$T_{unaffected}$ = 20 seconds → the remaining portion

By substituting all the values in the given eq:

① ⇒ $25 = \frac{80}{\text{Imp. Factor}} + 20$

$25 = \frac{80}{IF} + 20$

$5 = \frac{80}{\text{Imp. factor}}$

$5 = \frac{80}{IF}$

$IF = 80/5 = 16$.

Improvement factor = 16 → so, the multiplication portion will be improved by the factor of 16 to get desired response time.

Solution (b)

Date:
M T W T F S S

* Means Vs End Based Metric (Pg 51 (2.6))

One of the problems with many of the metrics that makes them unreliable is that they measure what was done whether or not it was useful. What makes a performance metric reliable, ~~however,~~ is that it accurately & consistently measures progress towards a goal.

Reliability is one of the most important characteristics of performance.

i) Means-based: Metrics that measures what was done, whether or not it was useful, have been called means-based metrics.
For Eg: NOP instructions, multiply by 0 etc. It results in an unreliable metric.

Date: M T W T F S S

whereas, ends-based metrics measure what is actually accomplished.

ii) End-based : they measures progress towards a goal. Only counts what is actually accomplished. It results in reliable metric.

To obtain a ~~feel~~ feel for the difference between these two types of metrics, consider the vector dot-product routine

Example (a)

Consider the given vector dot-product routine

```
s=0;
for (i=1; i<N ; i++)
    s = s + x[i] * y[i];
```

let,

t_A = # of clock cycles for one addition floating point operation.

t_M = # of clock cycles for one multiplication floating point operation.

Q i) Total time required to execute this pgm?

If t_1 is the time required to execute this pgm then,

$$t_1 = N(t_A + t_M)$$

* N is the total no. of operations

ii) Execution Rate? the

Execution Rate is ↑ amount of work done per unit time.

Date:
M T W T F S S

Here, total amount of work done?
we are actually performing two operations addition & multiplication. where # of operations or variables is N. so, the total amount of work done will be 2N. & we already know the execution time in part (i).

$$R_1 = \frac{2N}{N(t_A + t_m)} \text{ FLOPS/cycle.}$$

$$R_1 = \frac{2}{t_A + t_m}$$

Example (b)

Since there is no need to perform the addition or multiplication operations for elements whose value is zero, it may be possible to reduce the total execution time if many elements of the two vectors are zero

Consider the given vector dot-product routine

```
s = 0;
for (i=1; i<N; i++)
if (x[i] !=0 && y[i] !=0)
    s = s + x[i] * y[i];
```

let,

t_{if} = # of clock cycles to execute one "if" instruction

f = fraction of N for which both x[i] and y[i] are non-zero.

Date: M T W T F S S

i) Total time required to execute this program?

let t_2 be the total time required to execute this pgm.

$$t_2 = fN(t_A + t_m) + Nt_{if}$$

$$t_2 = \{f(t_A + t_m) + t_{if}\}N$$

ii) Execution Rate?

$$R_2 = \frac{2Nf}{\{f(t_A + t_m) + t_{if}\}N} \text{ Flops/cycle}$$

$$R_2 = \frac{2f}{f(t_A + t_m) + t_{if}} \text{ Flops/cycle}$$

Q/ Suppose that:

$t_{if} = 4$, $t_+ = 5$, $t_* = 10$, Clock Rate = 250 MHz & $f = 10\%$

i) calculate the execution time & execution Rate for both examples (a) & (b)

ii) compare the results using mean-based & end-based metric & verify that end-based metric is reliable.

ans

$t_1 = 60N$ ns

$t_2 = 22N$ ns

$S_{2,1} = 60N/22N = 2.73$

$R_1 = 33$ MFLOPS

$R_2 = 9.09$ MFLOPS

Date:
M T W T F S S

↳ On the basis of execution time, (or the end based metric), it actually shows that program 2 is ((60/22)-1)*100 = 172% faster than program 1.

↳ On the basis of execution rate (i.e. on the basis of ratio of performance or on the basis of values of R1 & R2) then program 2 is (1-9.09/33)*100 = 72% slower than program 1.

Thus even though we have reduced the total execution time from $t_1 = 60N$ ns to $t_2 = 22N$ ns, the means-based metric (MFLOPS) shows this result that pgm 2 is 72% slower than pgm 1.

From this examples, we have reached completely different conclusions when using these two different types of metrics because the means-based metric unfairly gives program 1 credit for all of the useless operations for multiplying & adding zero. This example highlights the dangers of using the wrong metric to reach a conclusion about computer-system performance.

↳ which one is end based?

Orio! the result that is being calculated on the basis of execution time is the result of end-based metric & is more reliable result.

Date: M T W T F S S

Performance Measures

Computer systems architect & designers look for configurations of computer systems for elements so that system performance meets desired measures of the user applications.

The interesting performance measure of a computer system depend on the domain of application. For eg. the requirements for a system to be used in a short term unmeant space mission are very different from those mission.

All performance measures deal with three basic issues:

- how quickly a given task can be accomplished,
- how well the system can deal with failures & other unusual situations, and
- how effectively the system uses the available resources.

Since a computer system will typically run several types of tasks the measures the measures in the first category should be evaluated several types of tasks.

The measures in the first category should be evaluated seperately for each workload type. Possible classifications for workload are real time control, transaction processing, interactive computing & batch.

Similarly, the faults a system differ in the damage to it.

Thus all measures in the second category should be evaluated for each fault type.

Date: M T W T F S S

Categories of Performance Measures

1) **Responsiveness:** These measures are intended to evaluate how quickly a given task can be accomplished by the system. The possible measures are processing time, response time, turnaround time etc. It is important to know that many variable are involve with response or turnaround time. For eg. I/O channel traffic may cause variation in the major for the same job as OS load or CPU loads. These user measures are all considered random & therefore typically discussed in terms of expected or average values as well as variances from these values.

2) ~~Usage~~ **Usage level:** It evaluate how well the various components of the system are being used. The possible measures are throughput & utilization of various resources. The objectives behind these measures conflict with those behind the responsiveness measure. since a well utilize system would generally respond more slowly than a less utilize system.
We can use throughput measures to determine system's capacity by observing the no. of waiting item is never zero. when

3) **Mission-ability:** These measures indicate if the system would remain continuously operational for the duration of a mission.

Date: M T W T F S S

The possible measures are 1) distribution of the work accomplished during the mission time. 2) interval availability: It is the probability that the system will keep performance satisfactorily throughout the mission time.
3) lifetime : It is the time when the probability of unacceptable behaviour increases beyond some threshold.
These measures are useful when the unacceptable behaviour may be catastrophic.

4) Dependability : It indicates how reliable the system is over the long run.
The possible measures are no. of failures per day mean time to failure (MTTF), mean time to repair (MTTR), long term availability & cost of a failure. These measures are useful when repairs are possible & failures are tolerable.
There is a major called MTBF 'Mean Time Between Failure'. It is commonly used & defined as the sum of mean time to failure & mean time to repair

5) Productivity: These measures indicate how effectively a user can get his or her work accomplished.
The possible measures are user friendlyness, maintainability & understandability.
As Performance measures falls into two catogories system oriented & user oriented. system oriented measures included include the throughput & utilization. and user oriented typically involves responsiveness & productivity.

Date:
M T W T F S S

Classification of Computer systems (chart)

Date: M T W T F S S

Quick Review of Performance Definition

For some program running on machine X,

- $\text{performance} = \frac{1}{\text{Execution time}_X}$

If you want to compare performances of two machines by the factor of n then,

"X is n times faster Y".

- $\frac{\text{Performance}_X}{\text{Performance}_Y} = n$

Computing CPU Time

- The time to execute a given program can be computed as.

$$\text{CPU time} = \text{CPU clock cycles} \times \text{clock cycle time} \quad \text{---(i)}$$

- since clock cycle time and clock rate are reciprocals

(i) → CPU time = CPU clock cycles / clock rate

- the no. of CPU clock cycles can be determined by

$$\text{CPU clock cycles} = (\text{instruction/program}) \times (\text{clock cycles/instruction})$$
$$= \frac{\text{Instruction count} \times \text{CPI}}{\text{clock rate}}.$$

Date: M T W T F S S

Computing CPI

- The CPI is the average numbers of cycles per instruction.
- If for each instruction type, we ~~can compute~~ know its frequency & number of cycles need to execute it, we can compute the oresall CPI as follows:

$$CPI = \sum CPI \times F$$

Eg/

Operation	Frequency	CPI	CPI * F
ALU	50%.	1	0.5
Load	20%.	5	1
Store	10%.	3	0.3
Branch	20%.	2	0.4
Total	100%.		2.2

Q/ you are lead designer for a company that produces microprocessors for a graphics accelator. The performance analysis department reported the following instruction ~~usoo usu~~ usage profile for the popular benchmark suite 'smartBench':

Instruction Class	CPI	Usage
Integer/logical	2	30%.
Load/Store	2	25%.
Floating Point	6	15%.
Branches	2	10%.
Others	4	20%.

Date: M T W T F S S

a) Given that SmartBench executes in 2 seconds on 500 MHz processor, how many machine instructions were executed?

b) How long will SmartBench take to execute if floating point instructions are made 4 times faster?

Ans a) 333.33 M instructions

b) 1.775 seconds

Date:
M T W T F S S

EVENTS

- Events is an action of interest in our system.

- Events of interest to computer engineers/architect are:
  - The beginning or end of a clock cycle.
  - The beginning or end of a computer's instruction execution cycle.
  - The reading of a memory location.
  - The initiation of a block data transfer from a secondary storage device.
  - The initiation of a process or task.

Each simple action from the clock ticks to more complex action such as an instruction execution all become part of larger system's action. For eg: the initiation of a direct memory transfer of data from secondary storage the DMA transfer being used as part of the system's memory management paging algorithm & the paging algorithm relationship to the movement of one running process being replaced by another due to a process switch handled by the OS.

- In terms of performance assessment, the system analyst must have understanding of the relationship of events to each other.

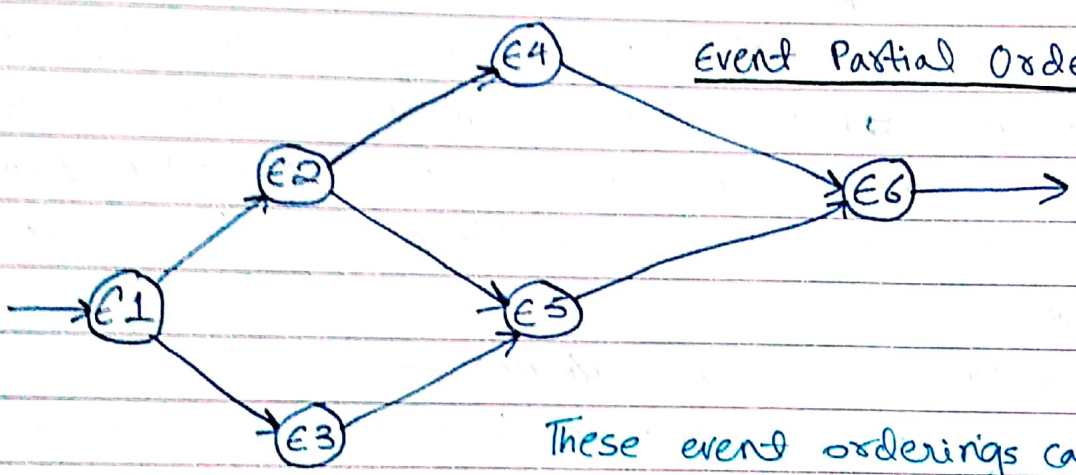
For instance, we need to know that a file access event is composed of

i) ~~page~~ disk access events, ii) the memory page replacement and iii) memory load & store algorithm event events.

Date:
M T W T F S S

In addition to knowing the events involved with a higher order event, events ordering must also be ~~Event~~ understood.

Event ordering means for eg. the ~~replacement~~ algorithm must be access first to determine which page to move before the new page can be loaded into main memory.



These event orderings can be represented by simple event list or by more complex Partial Ordering.

↳ circles are representing different events

↳ & the arrow head are determining the timing of these events.

Here; E1 is the first event to occur & before E4, for instance E1 & E2 are to be occured in the given sequence.