Performance and Optimization

There will be a lab (which does not count towards a grade - you either pass or fail) and a written exam. The exam has predictable questions. Assignments will be made available as soon as possible. Assignments handed in will have to be able to be reproducible. No standard for the reports. Labs and projects can be done alone or in pairs (or even groups of three). There will also be two projecs. Project one is "painful" - it's theorietical. A solution is available in the textbook, but it will have to be slightly extended. The final grade will be the average of the projects and exam. There are bonus points for additional points (such as in lab 3) which will be counted towards a better grade for the exam. There will be a midterm test online with a deadline of 25th of September. Lab 1 has a deadline of 6th of October. Assignment 1 is due 23th of October and assignment 2 31th of October. The exam is 28th of October. If you won't be in the labs - show them to the teacher before handing in.

The course book is Software Performance and Scalability : A Quantitative Approach. Available online via O'Reilly.

Hardware platforms

Topics

Performance is the degree to which a software system or compoennt meets its objective for timeliness - *response time*, *throughput* (number of requests that can be processed in some specific time) in seconds. *Note: this has been an exam question.*

Quantative characteriation - measures in numbers.

Qualitive - measures in function.

Moore's law - the density of transistors on a computer chip is increasing exponentially, doubling approximately every two years.

Scalability is the ability of the system to continue to meet its response time or throughput objectives as the demand for the software functions increases.

Processor implies a seperate chip package or socket.

A system with one, two or n processors are called **one-way** (UP), **two-way** (DP) and **N-way** (MP) **systems**.

A processor can be a **dual-core**, **quad-core** etc.

Cores are called "Execution engines"

You can have hyper-threading enabled on each core.

Factors

1. Hardware

2. Software platform
3. Design and Implementation

Test *qualitatively* in a *benchmark*, which is at least:

- repeatable
- representative
- easy to run
- verifiable

Dependence on hardware platform

- Largley determined by CPU
- Scalability of a software system depends on the scalability of the hardware platform

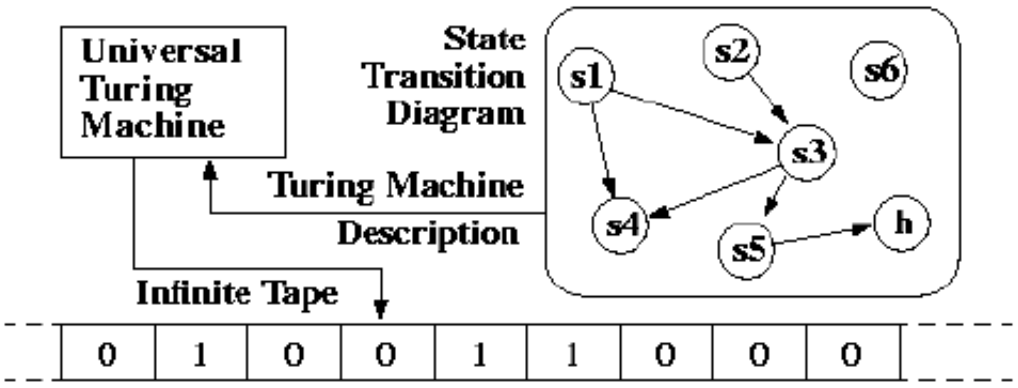
~~Runs great on my computer~~ - test on a wide range of computers or give developers slower machines.

Turing machine

Was introduced as a theoretical model in 1936. It sets a barrier to what is computable. The *recursively numerable languages*.

A *Turing Machine* has a tape, a head, a table of instructions (state + symbol) and a state register.

The model has the assumptions that the disk space is unlimited and that completing a task is doable regardles of the amount of time it takes.



Today, computers work the same. There tape is represented by RAM and SSDs, heads are I/O controllers and the table is the CPU.

Von Neuman Machine

1946 the convept of a stored program was introduced. Previously the computer was built to execute a particular. The new machine made the **hardware independent from the software**.

It has memory, a control unit and a Arithmetic Logic Unit (ALU) and a accumulator (I/O).

The term *von Neuman bottleneck* means the latency for data transfer between CPU and memory. It resulted in cache units in modern day CPUs.

Intel machine

The machine is from 1971.

Processor architecutre refers to the instruction set, registers and memory data-resident data structure that is **public to the programmer**.

Microarchitecture refers to the implementation of the processor architecture **in silicon** (not available to the programmer).

Hyper-threading

Likely to be in the exam.

Dual processor systems work with seperated sockets (physical processors).

Hyper-threading processors have shared ALUs and cache. The only seperated parts are the logical processors and registers.

There is a need to synchronize logical CPUs. Single-threaded applications can actually slow down single-threaded applications. It benefits multi-threaded applications.

When testing we need to know if hyper-threading is enabled in the *System under* Test (SUT).

Not the same as two physical processors.

The effectiveness of hyper-threading depends on how busy the SUT without hyper-threading is under the indended load.

The OS sees a physical core and a logical core (from hyper-threading) appears as the same unit.

TODO: insert diagram of CPU, ALU, Cache etc (hyper-threading).

Multicore

Likely to be in the exam.

Multicore is a lot more powerful since a dual-core processor is closer to two processors than a single-core hyper-threading processor.

These cores share nothing above the L2 cache. Both single and multi-core applications can benefit from multicore.

Increases front-side-bus speed (CPU and memory controller) and size of L2/L3 cache.

TODO: insert diagram of CPU, ALU, Cache etc (multicore).

System Under Test (SUT)

Think about these: ()

- **Processors**
- Motherboard
- Chipset
- **Storage**

- RAID
- Networking
- Operating Systems

Processors

Clock speed - the number of clock cycles a CPU can perform per second (in Hz).

Latency - the number of clocks it takes for an instruction to have its data available for use by another instruction.

Throughput - the number of clocks that the processor is required to wait before starting the execution of an identical instruction. How many instructions you can start per clock cycle.

$$\text{Latency} \geq \text{Throughput}$$

The equality does not tell you how well the instruction performs on a certain microprocessor, rather how well it performs in relation to another instruction.

Motherboard

All components are crucial to the performance.

Chipset

TODO:

Midterm

Questions from the midterm tend to appear in the exam.

Software performance

On Line Transcation Processing (OLTP) workload

Important

The user is involved - interactive user activities. Characterized by the type of user activitties. The number of users associated with each type of user activity. "Thinking" and "typing".

Use *real* workload (online) or *synthetic* workload. It is composed based on hypothetical use cases - usually described by stochastic processes. Useful when no data is available, no users are available yet or to answer various kinds of "what if" questions.

The response time ahs to be samll. The performance metric for OLTP is defined as $N_{\text{concurrent}} = N_{\text{active}}(\frac{\text{Response time}}{\text{Response time+think time}})$.

Response time = RT

Think Time = TT

Batch job workload

Specifies the amount of work to be processed without the necessity of intervention. Characterized by type of jobs - the number och jobs associated with each category.

Performance metric is *throughput* - the number of tasks completed within a given time period.

Amdahl's law

TODO

$$G = \frac{1}{1 - f + \frac{f}{n}}$$
$$f = \frac{T_1}{T_1 + T_2}$$

The smaller the impact factor of the system, the lesser todo

Queuing theory (QT)

Queuing theory was not developed specifically for software. It is generic for computing and optimizing the efficiency of any system that achieves its objectives by consuming multiple resources. In customer call centers, computer systems etc. we have a limited amount of resources that should be optimally used to achieve maximum possible efficiency.

Provides a way to unify all metrics with quantative performance laws.

A metaphor - visiting a banking center.

Server: banking center fulfilling the customer's service requests

Customer: the initiator of service requests

Wait time: the time duration a customer has to spend in line

Service time: the time duration from when the teller starts the service a customer to when the customer is leaving the teller and the next customer is called in for a service

Arrival rate: the rate at which customers arrive for service

Service rate: the rate at which customers are serviced

Utilization: the portion of a teller's time actually servicing customers rather than idling

Queue length: the total number of customer waiting or being serviced or both

Response time

Residence time

Throughput

Demand: Someone who comes often and sits a lot with the teller - number of visits $*$ service time

Objectives

- Minimizing
- Minimizing wait time
- Knowing average customer arrival rate
- Minimizing response time and residence time
- Striving for the highest possible throughput

Problem solved by QT

For an average arrival rate λ and a service rate μ, how many parallel servers m are required in order for the system to operate under steady-state conditions (the load should not exceeed the system's processing capacity) (also called *equilibrium condition* - no customer is created or lost in the system).

$$\text{System load intensity} = \rho = \frac{\lambda}{m\mu}$$
$$\rho < 1$$
$$\iff$$
$$m > \frac{\lambda}{\mu}$$

Semantics

TODO: symbol and semantics (sida 10 L4)

Statistics

Markov process

The Markov Process is characterized by its memorylessness - the future states of the process are independent fo its past history and depend solely on the present state

$P(x \geq s + t | x \geq s) = P(x \geq t)$ - no matter how long you waited, each moment it is like starting from fresh.

A markov process implies that

1. The number of arrivals follows the Poisson distribution
2. The interarrival times follow the exponential distribution
3. The service times follow the exponential distribution

General (G) process

The process is not characterized by any single distribution because it is completely arbitrary.

Deterministic (D) process

The proces is predictable and characterized by various constants, for example, when the interarrival times are constant from one arrival to the next.

Notation for queuing nodes and systems

A queuing node is denoted with a subscript lowercase letter i.

A queuing system is denoted with a subscript sign of zero.

The same performance principes to characterize a node (CPU, disk etc.) and a whole system.

Kendall notation

A generic queue is representat as $\alpha/\sigma/m/\beta/N/Q$.

For example, a FIFO is noted as $M/M/m/\infty/\infty$.

TODO: insert semantics (page 15, L4)

Little's law

Throughput, response time and queuing length completely quantifies the performance of a queuing system.

$$N_i = X_i R_i$$

N - number of customers waiting or receiving service

X_i - throughput

R_i - response time

Open and Closed model

If we plot the response time R (y-axis) and the number of users N (x-axis). At 70% load we can see the system with an open model having a greatly increased response time. The closed model has a steady response time. Open models correspond to OLTP and closed models use batch jobs. Read book, section 4.4.10 and figure 4.18.

One can describe the system in Kendall notation and look at the notation and decision tree to understand whether it will be an open or closed model.

Server Oriented Architecture (SOA)

Exam question.

A reusable atomic service that does something indepentently or in a federate fashion - programming is about using services.

- Accounts receivable
- Accounts payable
- Management
- Purchase Orders
- Inventory

The user doesn't care about the *black box* - it sees the service.

Web Service Description Language (WSDL) is an XML file that describes the web service. XML - eXtensible Mark up Language.

Writing a new application is not coding from scratch - it takes a whole bunc of ready-made "black-box" services.

Analytical model

An acceptance critera regarding the errors of the analytical models are:

- Resource utilization: 10%
- Throughput: 10%
- Response time: 30%

There are system-level models and component-level models.

System-level models treat the actual system as a black box in which only the external entities such as arrival rate and throughput are considered.

Component-level models decomposes the system into finer granuarities of multiple hardware and software components - each component is treated as a queue (Queueing Network Models - QNM).

From a system's perspective, the slowest resource or mostly highly utilized resource should be optimized or tuned first, in order to see the largest effect on the performance of the overall system.

One can decrease the service demand by more efficient algorithms, better data access methods, simplified business logic, the underlayning hardware, heap allocation, garbage collection, configuration etc.

Optimization and tuning

Optimization is identifying and eliminating internal inefficient designs and implementations (code).

Tuning is establishing the optimal setting for every possbile external configuration parameter (configuration).

Profiling

Catch bugs at the developer level by using an API profiling.

Profiling consists of inserting probes in your code to collect the performance counters.

There are tools such as `gprof` which aids profiling.

If the program is found to be compute bound, it relies on the CPU. Make one change at a time and repeat the process.

1. Choose a better algorithm and data structure
2. Write clear, simpler code (complicated expressions are harder to optimize)
3. Have a perspective of what costs much
4. Understand the compiler options

Big O: $\mathcal{O}()$

$\mathcal{O}(1)$ is constant growth

$\mathcal{O}(N)$ is linear growth,

$\mathcal{O}(N^2)$ has a square growth, etc.

It can relate to time and space.

Cplex is an algorithm for optimization.