6.033 Lecture 7
Operating Systems Part VII

Americo De Filippo

June 5, 2023

References

[1] Saltzer, Jerome H. and M. Frans Kaashoek. Principles of Computer System Design: An Introduction (2009): **Sections: 6.1 / .3**

Today we are going to look a set of tequinque to make a computer system more efficients, there will be 2 main ones. In this lecture we are going to understead about Concurrency and Caching.

Performance matrices For understeading what performance is we have to define some performance matrices, one of those could be the capacity (number of instruction per second or size of memory), utilization (the percent of capacity we are using), latency (time for req. to complete), throughput (req/sec).

Performance bottlenecks When we are doing an analysis about performance of a system we want to find for performance bottlenecks, one famous is I/0 bottlenecks which is the performance bottlenecks of accessing all the computer storage devices.

1 Concurrecy

We are going to look a 2 types of concurrency:

1.1 Between modules

Pipeling We want to have each of module works on a different request, in the pipeline, in this way all the time response is given to the one which have the worst latency. In this way we can optimize the system by optimizing the bottlneck. The way that we are going to do that is with withing a module cuncurrency

1.2 Within a module

This paradigm of concurrency **has to be reviewed from the lecture**. Let's have multiple threads send requests at the same time, they all will talk to the same queue etc... In this way we can run multiple threads a the same time creating a complete system that tries to make use of all is power. But with this comes also the problem of race condition that we are gonna face here.

in order to fix race condition In this way we are going at the same time reading the page and dequing. This is called atomic (we are going to make this atomic) isolating the threads with an isolationg routine called locks. lock is a variable which can be set or unset, it has two routine ACQUIRE, RELEASE. Most modern microcessor have an instruction called RSL (read-set-lock). This special instruction has the proprety of read and set a new state of a variable.

2 Caching

The bottleneck of our system is the disk stage, cause read on a large disk is very slow, if we add a cache stage, we are gonna add some kind of middle layer part, so that we are gonna look at the disk only if we get a cache miss.

2.1 Cost advantage

the basic equation is: $C(cache) + P(miss)C(disk)$, the P is the probability of the data not being in the cache (the cache is very little), so very little probability ¡=1

locality of reference What this means is that if we look at which pages a program access, there will be some small set of files that the user/program is working with regularity. So in this case we store them in the cache and we will have a very low probability of a miss and a very fast execution as a consequence.

2.2 Page addition

When we call add in the cache, and it is full. We need some kind of **replacement policy**: FIFO (first in, first out), LRU (last unread in the cache).