CMPSCI 691ST Systems **Fall 2011**

Lecture 12

Lecturer: Emery Berger *Scribe: Nicolas Scarrci*

12.1 Little's Law

In queueing theory Little's law relates the number of items processed by a queue to the average arrival rate of items in a queue and the average wait time of these items. Where L is defined as the number of jobs, λ is defined as the average arrival rate (or throughput) of new jobs, and W is defined as the average wait time (or latency) Little's law is:

$$L = \lambda W \tag{12.1}$$

Little's law is important for two reasons. Firstly, it is exceptional general because it only makes use of L, λ and W, which are defined (or easily definable) for all queues. Secondly, it allows us to calculate the third of our three measures whenever we are given the other two.

12.2 Flux

Flux is a coordination language while allows for the construction of servers by coordinating services in a way that is guarunteed to be deadlock free.

12.2.1 Simulation

There are two approaches to simulation. The first is to stochastically replay previously recorded data. The second is to generate events and then execute on this synthetic data. Flux takes the latter approach.
Flux simulation takes input in the form "arrival rate/service rate/number of servers".
Flux then uses the following relations to calculate performance of the server analytically.

$\rho = \lambda/\mu$

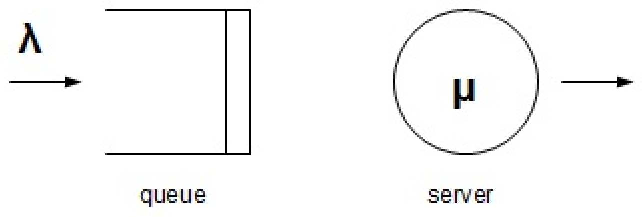
$L = \rho/(1 - \rho)$

$W = 1/(\mu - \lambda)$

12.3 Queueing Theory

12.3.1 Definition

A queue can be defined informally as a data structure which items are held in while they wait to be serviced. It is from this definition that the L, λ and W from Little's Law are defined. This definition also implies that if the rate at which items are serviced is less than the rate at which items arrive the size of the queue will increase without bound.



12.3.2 Discplines

The manner in which items are added to the queue and selected for service varies widely by queueing discipline.

FIFO
First In First Out. A FIFO queue removes items from the queue in the order in which they entered the queue.
FIFO is a fair policy, which is to say that every item that enters the queue eventually gets served (assuming no loops.)

LIFO
Last In First Out. A LIFO queue is effectively a stack in that it removes items from the "back" of the queue.

random
A random queue selects an item for service at random from those items that are waiting.

PRIORITY
A priority queue selects an item for service based on "priority," a flag set externally. Priority queues are not fair. An item may be starved if higher priority items keep entering the queue. Worse than this a high priority item may keep re-entering the queue because it is spinning on a lock that is held by the starved item, preventing the system from making progress on these two tasks. To combat this priority may be boosted or decreased on the fly. Despite its weaknesses priority queues are used widely in industry. Suffice to say academia and industry do not communicate regarding cpu schedulers.

SJF
Shortest Job First

EDF
Earliest Deadline First

12.3.2.1 CPU schedulers

CPU schedulers are effectively just a queue and queue management policy.

Proportional share is one scheduling policy where tasks are broken into ”high class” and ”low class”. High class tasks are permitted to take up 80% of CPU time, whereas low class tasks are permitted to take up only 20% of CPU time.
Other scheduling policies include lottery scheduling and stride scheduling.

12.4 Realtime Systems

A realtime system is a system for which new inputs arrive and enter the queue while the system is executing. In realtime systems there are two types of deadline: hard deadlines and soft deadlines.

12.4.1 Hard Deadlines

A hard deadline is a deadline that must be met 100% of the time. Alternatively hard deadlines can be thought of like this: if a hard deadline is missed, someone dies.

12.4.2 Soft Deadlines

A soft deadline is a deadline that does not need to be met 100% of the time. Arguably everything can be thought of as having a soft deadline (except for those things which has hard deadlines.)

12.5 Optimizations

Optimizations can be broken into two classes, common case optimizations and static optimizations.

12.5.1 Common Case Optimizations

Cache

Speculation

Branch prediction

Other history based optimizations

12.5.2 Static Optimizations

Scratchpad

Static branch prediction

12.6 Formal Verification

Formal Verification is the practice of proving things about code mathematically.

12.6.1 Problems

Programs are complex.

There are an exponential number of possible execution paths.

Program execution and/or control flow may be input dependent.

12.6.2 Solutions

12.6.2.1 Fix Maximums For Iteration

Fixing a maximum allows formal verification to enumerate the possibilities. Fixing a maximum will however limit the expressiveness of the code, and may make some calculations impossible.

12.6.2.2 Bound Operation Cost

Bounding operation costs allows for performance guarantees as well as a halting guarantee.
Issues:

This excludes dynamic memory allocation

fragmentation

asymptotic cost O(1) is possible

addresses are unpredictable

cache begins missing more frequently.

12.7 Tangents

12.7.1 Airbus

Airbus is a European air-travel company that is really strict about it's software requirements. While their practices may seem excessive they can boast that to date they have had no software errors. Additionally, in the event of an error they can reboot their system while the plane is flying.

12.7.2 Recovery Oriented Computing

Instead of avoiding errors (like Airbus) the idea behind recovery oriented computing is to deal will errors when they occur such that they do not create undesired behavior in the system. This is achieved by detecting errors early and rebooting quickly. After all is it really reasonable to attain to 100

12.7.3 Amazon's Vm Business

For a fee Amazon will vm your server for you. In order to turn a profit Amazon oversells it's server space knowing that whenever one server requires more resources these resources can be taken from another server that is not utilizing them. In order to optimize for multiple vms on the same physical machine Amazon implements "content based page sharing." This allows the physical machine to keep one copy of static data (for example a page of code) for all of the vms which use it, instead of keeping one identical copy for each vm.

12.7.4 Developers vs Researchers

Imagine that a mixed group of developers and researchers went for a walk on a mountain trail. Halfway through the walk the group comes upon a huge boulder obstructing the path. When the researchers see this they get very excited. "I didn't know there were boulders like this!" one researcher cries. "Quick let's measure how large it is!" shouts another. After a few minutes of taking notes about the boulder the researchers look up to notice that the developers have created a small path which goes around the boulder and connects up to the main trail later on.
The moral of this story is that developers are looking to solve an instance of a problem (ex the boulder on the path) whereas researchers are looking to understand and hopefully eliminate classes of problems instead.