Performance Modeling and Design of Computer Systems- Ch 2 Queueing Theory Terminology

Debobroto Das Robin

Kent State University

Spring 2020

drobin@kent.edu

Overview

Queueing Theory
Theory of Queues

- The theory behind what happens when you have lots of jobs
- what makes queues appear and how to make them go away
- Queueing theory applies anywhere that queues come up
- Example
    - CPU uses a time-sharing scheduler to serve a queue of jobs waiting for CPU time
    - Router in a network serves a queue of packets waiting to be routed.
- Queueing theory is built on **stochastic modeling and analysis**
    - Model and analyze service demands of jobs and the interarrival times of jobs as random variables.

Goal of Queueing Theory
2 Goals

- Predicting the system perfor- mance. Ex.
  - predicting mean delay or delay variability in service
  - number of jobs that will be in queue
  - mean number of servers being utilized
- Developing design of improved system
- Example
  - Can we build a better system from 1 slow discs or one faster disc
  -

Power of Queueing Theory
Design Example 1

- Consider a system of a single CPU that serves a queue of jobs in First-Come- First-Served (FCFS) order
- Assume any distrbution for the arraival
  - $\lambda =$ arrrival rate $=$ No of jobs arrives per second
  - $\mu =$ service rate $=$ No of jobs served per second
  - **response time**$=$ time diff. bet. job arrives until it completes service
  - $E[T] =$ mean response time $\sum(x.P(x))$. Can you serve customer within old time length?

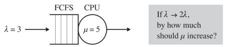

FCFS
CPU
$\lambda = 3$
$\mu = 5$
If $\lambda \to 2\lambda$, by how much should μ increase?

Figure 1.2. A system with a single CPU that serves jobs in FCFS order.

- **Question**: What if λ is doubled?
  - How the $E[T]$ changes? increases or decreases?
  - If decreases, can using powerful cpu solve the problem?
  - How much powerful CPU is necessary

Power of Queueing Theory
Design Example 2

- Consider the closed system of figure

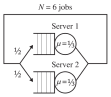


Figure 1.3. A closed batch system.

- Replace server 1 with a server that is twice as fast (the new server services jobs at an average rate of 2 jobs every 3 seconds).
  - Does this improvement affect the average response time in the system?
  - Does it affect the throughput?
  - Both cases improvoment is no or negligible
- If the system is converted to open system? where arrival times are independent of service com- pletions.
  - Absolutely possible

Power of Queueing Theory
Design Example 3

- one fast CPU of speed s,or n slow CPUs each of speed s/n (see Figure).
- Goal is to minimize mean response time.

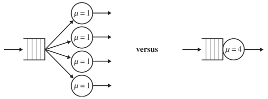


Figure 1.5. Which is better for minimizing mean response time: many slow servers or one fast server?

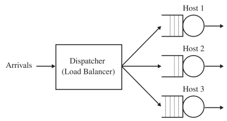
- one fast CPU of speed s,or n slow CPUs each of speed s/n (see Figure 1.5). Your goal is to minimize mean response time.
    - Choice depends on **the variability of the job size**
        - **Question**: when job size variability is high? Answer: we prefer many slow servers because we do not want short jobs getting stuck behind long ones.
        - Question: Which system do you prefer when load is low? Answer: When load is low, not all servers will be utilized, so it seems better to go with one fast server.

Power of Queueing Theory
Design Example 3..cont..

- If jobs are preemptible; Jobs can be stopped and restarted where they left off.
- **Question** do you prefer many slow machines as compared to a single fast machine? Answer: If your jobs are preemptible, you could always use a single fast machine to simulate the effect of n slow machines.
- Resources can vary. CPU, GPU, MEMORY etc.
- Complexity and variation grows

Power of Queueing Theory

Design Example 4



- Assume that all hosts are identical (homogeneous)
- all jobs only use a single resource.
- once jobs are assigned to a host, they are processed there in FCFS order and are non-preemptible.
- Which task assignment policies yields the lowest mean response time?
- Some options Random, Round-Robin, Size-Interval-Task-Assignment (SITA), Central- Queue:
- More possible. Answer depends on various parameters
  - If job size variability is low, then the LWL policy is best.
  - If job size variability is high, then it is important to keep short jobs from getting stuck behind long ones, so a SITA-like policy,

Power of Queueing Theory
Design Example 4

- A single server. Jobs arrive according to a Poisson process. Arbitrary Job size
- Question : Which Scheduling policy is best w.r.t. mean response time if **non- preemptive service orders**?
- Scheduling policies are: First-Come-First-Served (FCFS), Non-Preemptive Last-Come- First-Served (LCFS), etc
- Ans: All are same for **non-preemptive service orders**
- now if **Non Preemptive-LCFS policy (PLCFS)**- Whenever a new arrival enters the system, it immediately preempts the job in service
- Mean resposne time depends on the variability of the job size distribution.
    - job size distribution is at least moderately variable, then PLCFS will be a huge improvement.

  If the job size distribution is hardly variable (basically constant), then PLCFS policy will be up to a factor of 2 worse