CSE 590XYZ - Lecture 2

Today's schedule

- High-volume web serving is harder than you think, Brian Pinkerton, Vice President, Engineering, Excite, Inc.
- Project Overview
- Introduction to Probability (Continued)
- Introduction to Simulation

CSE 597 - Performance Analysis

Project Overview

- In a nutshell, distributed web servers
- Basic problem is how to do load balancing

[Diagram of load assignment to web servers]
**Project Overview**

- **Load Balancing**
  - Per request - good load balancing, high overhead
  - DNS Round-Robin - problematic because of caching

- **Vague Project Goal:** How should load balancing be done?

- **More specific immediate goal:** Is the Dias, Kish, Mukherjee, and Tewari right?

- **You’re free (and encouraged!) to pose your own questions!**

**Project Problems**

- **Dias et al.** don’t provide anywhere near enough information to try to duplicate their work. (This is very common.)

- **We have measurement data from www.webcrawler.com.** The data has (a) a timestamp, (b) the IP address of the client, (c) the requested URL, and (d) sometimes the byte count of response.

- **It doesn’t have any indication of how much CPU/disk/etc. resource was consumed.**
Project Vision

- This week: work on defining the problem and a model structure
- Next week: trace driven simulation
- Then, a stochastic simulation
- Then, a queueing model
- Then, a Petri net model (maybe)

Cumulative Distribution Function

- The cumulative distribution function, \textit{cdf}, is defined by 
  \[ F(x) = \sum_{i=x}^{\infty} f(i) \]
- \( F(x) = \text{Prob}[X \geq x] \)
- Clearly, \( 0 \leq F(x) \leq 1 \), and \( F(\infty) = 1 \)
Properties of the CDF

- \( F(x) - F(x-1) = f(x) \)
- A less than obvious property

\[
E[X] = \sum_{i=0}^{\infty} i f(i) = f(1) + f(2) + f(3) + ... \\
+ f(2) + f(3) + ... \\
+ f(3) + ... \\
+ ...
\]

\[
= \sum_{i=0}^{\infty} (1 - F(i))
\]

Some Common Distributions

Uniform: \( U(A,B) \)

\( f(x) = \frac{1}{B-A+1} \)

- \( \mu = (A+B)/2 \)
- \( \sigma^2 = (B-A)^2/12 + (B-A)/6 \)

“chosen at random”
Bernoulli (Selection)

- Outcome 1 with probability p, 0 with probability 1-p

\[ f(x) \]

- \[ \mu = p \]
- \[ \sigma^2 = p(1-p) \]

Geometric

- Example: Number of tosses until a heads is tossed
- If p is the probability “of success”, then \[ f(x) = p(1-p)^{x-1} \]

- \[ \mu = \frac{1}{p} \]
- \[ \sigma^2 = \frac{(1-p)}{p^2} \]
- The geometric is “memoryless” -- distn of number of future throws is unaffected by how many have taken place so far
**Binomial**

- Number of heads in $n$ flips
- In general, given $n$ trials and probability $p$ of outcome 1 (and 1-$p$ of outcome 0), $f_{n,p}(x) = \binom{n}{x}p^x(1-p)^{n-x}$

![Graph of Binomial Distribution]

- $\mu = np$
- $\sigma^2 = np(1-p)$

**Continuous Random Variables**

- Take on a non-countable set of values
- For us, this will mean the interval $[0, \infty)$
- **Examples:**
  - The time between successive arrivals (inter-arrival time)
  - The time in service (service time)
  - The total time in system (response time)

- All the concepts are “more or less the same” as in the discrete case, except...
(Continuous) Probability Distributions

• Suppose \( X \) has the following distribution

\[
\begin{align*}
\text{f(x)} & \quad 0 & 0.2 & 0.4 & 0.6 & 0.8 & 1 & 1.2 & 1.4 & 1.6 \\
0 & 0.5 & 1 & 1.5 & 2 & 2.5
\end{align*}
\]

• What does it mean?

What It Doesn’t Mean

• \( f(x) \) is NOT the probability that \( x \) occurs…
• because \( f(x) \) is not a probability at all
• To talk about probabilities, you must integrate

\[
\text{Prob}[0 \leq X \leq 2] = \int_0^2 f(x) \, dx
\]
Other Properties/Peculiarities

- We require (only) that \( f(x) \geq 0 \), for all \( x \)
  \[ \int_{-\infty}^{\infty} f(x) \, dx = 1 \]
- \( \text{Prob}[X = x] = \int_{x}^{\infty} f(y) \, dy = 0 \) (at least for the distributions we’ll consider)
- Note that \( \text{Prob}[X=\infty] = 0 \) does NOT mean \( x \) can’t occur
- \( \text{Prob}[X \in S] = 0 \) for any countable set \( S \)
- \( \text{Prob}[X \in S] = 1 \) means an outcome in \( S \) occurs almost surely

The CDF

- \( F(x) = \text{Prob}[X \leq x] \)
- \( f(x) = \frac{dF(y)}{dy} \bigg|_{y=x} \) (almost always)
Why the CDF?

- Any function that has domain $(-\infty, \infty)$, has range $[0,1]$, and is monotone non-decreasing is a CDF.
Expectation / Moments

- Mean

\[ E(X) = \mu = \int_{0}^{\infty} x f(x) \, dx = \int_{0}^{\infty} (1 - F(x)) \, dx \]

- All properties of means we saw for discrete distributions still hold

- All definitions we saw for moments still hold, e.g.,

\[ \text{Variance} = \sigma^2 = E[(X - \mu)^2] = \int_{0}^{\infty} (x - \mu)^2 f(x) \, dx \]

Some Common Distributions

Uniform: \( U(A, B) \)

\[ f(x) = \frac{1}{B - A} \]

\[ \mu = \frac{A + B}{2} \]

\[ \sigma^2 = \frac{(B - A)^2}{12} \]

"chosen at random"
(Negative) Exponential

- $f(x) = \lambda e^{-\lambda x}$
- $F(x) = 1 - e^{-\lambda x}$

- $\mu = 1/\lambda$
- $\sigma^2 = 1/\lambda^2$
- $CV^2 = 1$

Poisson

- Number of events with exponential inter-event times during an interval of length $t$
- If $\lambda$ is the mean rate at which events take place (so $1/\lambda$ is the mean time between events), then

$$f(x, t) = \frac{(\lambda t)^x}{x!} e^{-\lambda t}$$

- $\mu = \lambda t$
- $\sigma^2 = \lambda t$
Discrete/Continuous

<table>
<thead>
<tr>
<th>Discrete Time</th>
<th>Continuous Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-arrival time distribution</td>
<td>Geometric</td>
</tr>
<tr>
<td>Distribution of Number of Arrivals in (0,t)</td>
<td>Binomial</td>
</tr>
</tbody>
</table>

Simulation

- Types of simulation
  - static vs. dynamic
    - Does time play a role?
  - deterministic vs. stochastic
    - Is there randomness in the inputs or the operation of the model?
  - continuous vs. discrete
    - Do “state changes” take place continuously, or only instantaneously at certain points in time?
Static Simulation - Monte Carlo Methods

- Typically useful when we need to estimate the value of a function too complicated to deal with mathematically
- Make repeated random trials
- Each trial is independent of the others
- Each trial simply computes some value
- Average the outcomes to estimate the result
- More trials → “more accurate” answer

Classical Example: Integration

- Suppose we want to compute $I = \int_{a}^{b} g(x) \, dx$ for some function $g(x)$
- Method 1
  * Create a bounding box
  * Throw $N$ points in box at random: $(x_i, y_i) = (a + (b-a)U_{x_i}, hU_{y_i})$
  * $X_i = 1$ if point is below $g(x)$, 0 otherwise
  * Estimate $I$ as $\frac{1}{N} \sum_{i=1}^{N} X_i$
Monte Carlo Integration II

- Sample \( g(x) \) at values of \( x \) chosen randomly in \([a,b]\)

\[
E[X] = E[g(y)] = \int_{a}^{b} g(y) f(y) dy = \frac{1}{b-a} \int_{a}^{b} g(y) dy = \frac{1}{b-a} I
\]

- \( X_i = g(a + (b - a) U_i) \)
- Estimate \( I \) as \( (b - a) \frac{1}{N} \sum_{i=1}^{N} X_i \)

More Typical Computer System Example

- Computer system problems for which Monte Carlo is used are often combinatorial problems
- **Example:** What is the probability that three distinct link failures will disconnect a given network topology?
Deterministic Simulations

- Probability doesn’t play a role
- Typically trace driven evaluations of design proposals
  - memory cache design
  - processor design
  - distributed file system design
  - etc.

Continuous Simulations

- The “continuous” refers to when the state of the simulated system can change
- Examples:
  - evaluating strategies for regulating the flow of water out of the Hiram Chittendam locks to control the level of Lake Washington
  - evaluating the flight characteristics of the Boeing 787
  - evaluating robot motion planning algorithms
- These are not terribly common in modelling computer systems
Discrete Event Simulation

- Dynamic, stochastic, discrete event
- This is most common use of the term “simulation” in computer system performance evaluation

Multiprocessor Scheduling Policies

- Each parallel job has a speedup function, \( S(p) \)
- It also has maximum number of processors it can use, \( P' \)
- The job’s response time on \( p \) processors is \( T(1)/S(p) \)
- Basic question: How many processors should be allocated to a job when it begins execution?
  * Assumption: “run to completion”
  * Performance metric: average job response time
Multiprocessor Scheduling

- Basic idea: allocate many when there is little contention, few when contention is high
  - \#allocated = H(sum of P' over all jobs currently in system)
  - (H(n) is monotone non-increasing in n)
- Compare with FCFS
  - H(n) = P'

System State

- Arrival time
- P'
- Function S(p)
- #processors allocated
- Departure time

Job Queue
Number Free Processors

5
Arrival Process State

![Time of Next Arrival: 1102.3317](image)

Event Driver

- Finds time of next event in system. (In our case, an arrival or a departure.)
- Updates clock to that time
- Invokes routines to update state, depending on type of event
  - Arrival: place new customer in queue, invoke scheduler, and compute time of next arrival
  - Departure: invoke scheduler
Event Driver

- Often (always?) have a future events list structure to keep track of times of scheduled events
  - time of next arrival
  - departure times of all jobs that have been allocated processors
- The FEL is essentially a priority queue
- Details:
  - Can an event be cancelled (efficiently)?
  - Is ordering of simultaneous events guaranteed?
- We’ll come back to this topic briefly later

Event Driver

- It’s “natural” to write a simulation in an OO way
- Simula and C++ were both originally developed to support simulations
  - “The language was originally invented because the author wanted to write some event-driven simulations for which Simula67 would have been ideal, except for efficiency considerations.” Bjarne Stroustrup
  - “There never was a C++ paper design: design, documentation, and implementation went on simultaneously.”
Measurements

- The simulation must be instrumented to produce appropriate output measures
  - In our case, we need to know the average response time
- There are two kinds of outputs you might be interested in
  - transients
    - What is the state at time t?
    - What is the average value of some performance measure over \((t_i, t_f)\)?
  - equilibrium
    - What is the long term performance (averaged over an infinite interval)?
- We’ll generally be concerned with equilibrium measures

Measurements

- In general, measurements can be either event driven or taken by sampling
- Under event driven, the measurement module is invoked whenever a significant event occurs
  - arrival or departure
- Under sampling, “sampling events” are scheduled (put in the FEL)
- Sampling can be more efficient; event driven can be more natural and more accurate
  - the same considerations apply to measuring “real systems”
Measurements

- This structure is almost always a mistake
- It’s often much better for the simulation to produce a log file containing (nearly) raw data from which performance measures can be calculated

For our example:
- Could compute average response time in simulator
  - Let $R_i$ be the $i$th departure’s response time
  - Keep track of total response time of all departures:
    \[ R = R + R_i \]
    and total number of departures $D$
  - Print $R/D$ as final output
- But later you’re likely to want to see something you hadn’t thought of originally
  - What is the variance of response time?
  - How is response time related to $P’$?
Debugging

• Very difficult. Unlike normal coding, by definition you don’t know what output to expect.

• What to do?
  * Run degenerate cases you know the answers for
    • $P'=1$ for all jobs (and simple inter-arrival and service time distributions)
  * Modify the input parameters in a way that affects the output in a way you (think you) understand
    • Higher arrival rates should lead to longer response times

This Was So Easy...

• Even if the simulation is certifiably bug free and is a reasonable representation of the system, we still have a problem
  * Outputs are the results of a single random experiment. How representative are they?
• We have to generate “proper randomness.” How?
• Simulations can take a lot of time.
• Simulations can take a lot of space.