Exam #2 for Capacity Planning (CIS 4930/6930)

Fall 2001

>>>*SOLUTIONS* <<<

Welcome to exam #2 in *Capacity Planning* (CIS 4930/6930). You have 75 minutes. Read each problem carefully. There are six required problems (each worth 16 points) and one extra credit problem worth 10 points. You may have with you a calculator, pencils, erasers, blank paper, lucky rabbit's foot, and one 8.5 x 11 inch "formula sheet". On this formula sheet you may have anything you want (definitions, formulas, etc.) *handwritten by you*. You may use both sides. Computer generated text, photocopies, and scans are not allowed on this sheet. Please submit your formula sheet with your exam. Please start each numbered problem on a new sheet of paper and do not write on the back of the sheets (I really do not care about saving paper!). Submit everything in problem order. No sharing of calculators. Good luck and be sure to show your work!

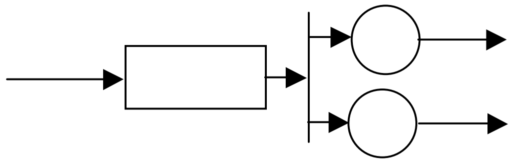
Problem #1 (10 minutes)

Answer the following questions about queueing.

a) Why is queueing theory of interest to capacity planners?

Queueing behavior dictates the performance of most computer systems. Queueing results in waiting, and waiting is a primary cause of response time (service time also contributes to response time).

b) Sketch a D/M/2/10 queue. Carefully label your sketch.



Arrivals are deterministic. Service is Markovian. The buffer capacity is 10 customers. There are an infinite number of customers in the universe.

c) State Little's Law.

$L = \lambda \cdot W$ (or queue length = arrival rate multiplied by wait (or response) time)

d) Given λ (arrival rate), μ (service rate), W (mean wait time in system), and L (mean number in system), solve for W_q (mean wait in queue) and L_q (mean number in queue). You may assume that this is a single server queue.

$W_q = W - (1/\mu)$ and $L_q = L - (\lambda/\mu)$

Problem #2 (10 minutes)

Answer the following questions about workloads.

a) What is a workload? What are the key attributes of a workload?

Workload is the input to a system. A workload is a time series, so the key attributes are interarrival times of jobs and job size.

b) Give four different methods of generating a synthetic workload.

1) Trace, 2) a "reasonable" probability distribution, 3) a fitted probability distribution, and 4) an empirical probability distribution.

c) Give three attributes of a good workload model.

```
Here are several attributes… Accurate, parsimonious, tunable, and simple to
generate
```

d) What are the pluses and minuses of using a trace as a workload (i.e., a trace-driven workload)?

```
"Plus" is that it is accurate for an existing user population, "minuses"
include that it is difficult to tune, may require very large amounts of
storage, and may not - in general - exist for new systems (i.e., systems with
no existing user group to trace).
```

<u>Problem #3</u> (15 minutes)

You are given a trace with records of format `<time_stamp, host_name, transaction_type>`. The time stamp format is `hours:minutes:seconds`. You may assume that `12:59:30` is "time 0" for these measurements. <u>The trace is found at the end of the exam</u>.

a) Compute the mean and standard deviation of all interarrival times (interarrival time is the same as "delta" time)

Mean = (1 + 2 + 4 + 1 + + 1+ 4 + 5 + 14) / 8 = 4 seconds

Standard Deviation = $\sqrt{\frac{1}{8}\left((1-4)^2+(2-4)^2+\cdots+(14-8)^2\right)}$ = 4.17 seconds

b) Plot a histogram for the interarrival times

```
(%) |
3/8 |   |
    |   |
2/8 |   |           |
    |   |           |
1/8 |   |   |       |   |                                   |
    |   |   |       |   |                                   |
    +---|---|---|---|---|---|---|---|---|---|---|---|---|---|----
        1   2   3   4   5   6   7   8   9  10  11  12  13  14
                                               interarrival time (sec)
```

c) Write a C function that will return a value from an empirical distribution based on the above measurements.

```
double emp(void)
{
  double z;
  z = (double) rand() / RAND_MAX;
  if (z <= (3 / 8.0)) return(1.0);
  else if (z <= (4 / 8.0)) return(2.0);
  else if (z <= (6 / 8.0)) return(4.0);
  else if (z <= (7 / 8.0)) return(5.0);
  else return(14.0);
}
```

d) Give the interarrival times for the giga1 events.

```
Filtering-in the giga1 events we have: 3, 4, 1, 10, 14 seconds
```

Problem #4 (15 minutes)

Design a benchmark for a web server. Describe the workload, measurements to be taken, and how the benchmark can be implemented. Carefully describe what your benchmark measures. Overall, the WHY is much more important than the WHAT. This question tests if you know what a benchmark is and what aspects are of interest to benchmark in a web server.

A benchmark should be representative of a real workload. For the case of a web server, there should be GET requests for both small and large files (small files evaluate the caching ability of the server and large files the file and network I/O). There should also be requests for dynamic content (this evaluates the CPU capability of the server). The distribution of file sizes, breakdown of requests for static and dynamic content, and time between request interarrivals should be based on trace logs, but should be tunable parameters (i.e., so that workloads not representative of any existing traces can also be generated to evaluate possible future users). Since real users request content in "parallel" with each other (and browsers also make parallel requests), the benchmark must be able to make parallel requests. This evaluates how well the server handles multiple, concurrent connections (and would evaluate the operating system and/or basic design of the web server program). Measurements should be made on request response time, throughput of responses per second, and CPU utilization of the server. Also, the numbr of dropped or failed requests should be counted. The benchmark would consist of a set of files and/or scripts to be installed on the server and then a script of requests to be made from one, or more, client machines. Each client machine should be able to emulate multiple real users so that it is not necessary to have one machine for each emulated user.

Problem #5 (10 minutes)

Answer the following questions about web servers.

a) When we say that a sockets function "blocks", what do we mean? Give several examples of sockets function calls that can block.

A function that blocks stops execution of the current thread until it returns. The functions that is blocking the calling thread is typically waiting on some external event (e.g., for a block of data to be retrieved from a disk). The connect() and recv() function calls in sockets are two examples of blocking functions.

b) Give the high-level steps (or flowchart) for how a web server works. Your description need only handle GET requests.

```
Repeat the following forever
    1) Wait for a connection
    2) Make the connection
    3) Receive and parse an incoming HTTP request
    4) Handle the HTTP request (if a GET, send the requested file)
    5) Close the connection
```

c) Describe the basic architecture (describe the key ideas for the design) of the Flash server and explain why the design choices were made.

The Flash server is a compromise between a multi-thread/process and single process architecture. The multi-tread architecture has thread/process overhead for each connection (but no one connection can block another). The single process architecture has blocking for file I/O (but no thread/process overhead). The Flash architecture has a single process that creates "helper threads" ONLY in the cases where blocking would otherwise occur. Thus, the Flash server has the "best of both worlds" of the multi-thread/process and single process architectures.

Problem #6 (10 minutes)

Answer the following questions about scaling.

a) Given a 10 processor system and an application in which 92% of the run-time can be parallelized, give the speed-up using Amdahl's Law.

$Speed-up = \frac{q}{(1+\alpha(q-1))}$ where α is the percentage of application that is serial by nature and q is the number of processors. So,

Speed-up = 10 / (1 + 0.08(10-1)) = 5.81

b) For an application that has 5% that must be serial, how many processor are required to get at least a 10 times speed-up. You may assume that Amdahl's Law applies.

We have 10 = q / (1 + 0.05(q - 1)). Solving for q we get 19 processors required.

Extra Credit: (5 minutes)

Consider a polling system with a master station (that does the polling) and N clients. When a client is polled it will with probability p send a data message (with transmission time t_{data}). It takes time t_{poll} to poll a client (t_{poll} is analogous to the token passing time for a token-based system). Derive the utilization U for this system as a function of N, t_{data}, t_{poll}, and p.

We know that $U = \frac{t_{useful}}{t_{useful} + t_{overhead}}$. We have that t_{useful} = $N * p * t_{data}$ and that $t_{overhead}$ = $N * (1 - p) * t_{poll}$. So, $U = \frac{p \cdot t_{data}}{p \cdot t_{data} + (1-p) \cdot t_{poll}}$. Easy (all you gotta do is THINK) ☺.

Trace for problem #3:

12:59:30	giga1	credit
12:59:31	giga2	credit
12:59:33	giga1	debit
12:59:37	giga1	credit
12:59:38	giga1	credit
12:59:39	giga2	debit
12:59:43	giga2	credit
12:59:48	giga1	debit
13:00:02	giga1	credit
