PAPER C339

PERFORMANCE ENGINEERING

Wednesday 18 March 2020, 10:00
Duration: 120 minutes
Post-processing time: 30 minutes
Answer THREE questions

Paper contains 4 questions

1 a i) Using SPECjbb2015 as a reference example, describe the typical execution phases of a load testing experiment for a distributed application.

ii) Present the notion of bottleneck switch. State why this effect is important and why it is challenging to monitor in real systems.

b A small blog consists of three web pages: P (Posts), H (History), and C (Comments). Consider the following user behaviour graph, where E and X respectively denote entry and exit states:

$$P = [p_{ij}] = \begin{array}{c} \\ E \\ P \\ H \\ C \\ X \end{array} \begin{array}{c} \begin{array}{ccccc} E & P & H & C & X \end{array} \\ \begin{bmatrix} & 0.60 & 0.20 & 0.20 & \\ & & 0.10 & 0.10 & 0.80 \\ & 0.60 & 0.30 & & 0.10 \\ & 0.10 & & & 0.90 \\ & & & & 1.00 \end{bmatrix} \end{array}$$

i) Determine the mean user session length.

ii) Assume that every page is served by a front server with mean service time $S_f = 1.0ms$. The History page additionally issues $k = 3$ calls to a database server, each with mean service time $S_d = 1.5ms$. Find the utilization of the front server and of the database server. The session arrival rate is $\lambda = 300$ sessions/s.

iii) Explain how would you use the transition probability matrix P to determine the probability that a session terminates after visiting n pages.

c Consider the following demand matrix $\mathbf{D}$ showing resources as rows, with labels $i = A, B, C, D$, and showing classes as columns, with labels $c = 1, 2, 3$:

$$\mathbf{D} = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 2 & 5 & 9 \\ 4 & 7 & 1 \\ 5 & 8 & 12 \\ 0 & 7 & 0 \end{bmatrix} \end{array}$$

State whether or not resource C can be a bottleneck resource. Justify.

The three parts carry, respectively, 25%, 55%, and 20% of the marks.

2 a You are asked to quantify the impact of two compiler optimization options, called A and B, on a software system. Suppose that a 2^2 design is carried out turning ON and OFF the two configuration options, simultaneously or one at a time. The measurements collected for the system latency are as follows:

A/B	OFF	ON
OFF	6	10
ON	8	18

i) Propose a response model for this design. Explain the qualitative interpretation of the effects in the proposed model.

ii) Allocate the experimental variation across factors and their interaction.

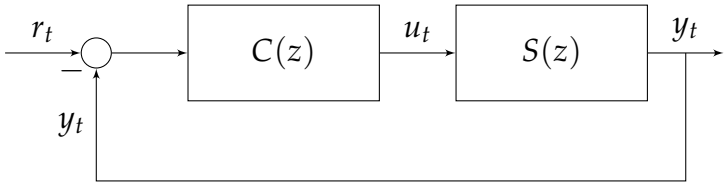
b i) Explain what are the trend, seasonal, and random components of a time series. Then, state briefly how would you proceed to apply an autoregressive model to a measured time series.

ii) A controller predicts the number of request arrivals a_t in timeslot t using an autoregressive AR(1) process. Assume a_t to be stationary. Suppose that the monitoring system reports the following statistics for a_t: mean $E[a_t] = 1$, variance $Var[a_t] = 3$, and lag-1 auto-covariance $K_1 = 0.9$. Fit the AR(1) process parameters to the data.

iii) A closed-loop controller can periodically adjust a server configuration option u_t, at discrete time instants $t = 1, 2, \ldots$, in order to control a server queue-length. Let r_t be the target queue-length level at time t and let y_t be the queue-length level in the server at time t. Denote by $S(z)$ the transfer function of the server and by $C(z)$ the transfer function of the controller. Assume the following transfer functions

$$S(z) = \frac{2}{z - 0.5} \qquad C(z) = K\frac{z}{z - 0.5}$$

and the control architecture



Determine the transfer function for the entire system. For $K = 1$, estimate the settling time and the steady-state gain.

The two parts carry, respectively, 35% and 65% of the marks.

3 a In your own words, define the concepts *Tracing* and *Profiling*. Describe how they are related and when either is applied.

b Assume the following program:

```
int f(int* input, int inputSize) {
  int sum = 0;
  for(size_t i = 0; i < inputSize; i++) {
    if(input[i] > 50)
      sum += input[i];
  }
  return sum;
}
```

assume that `input` contains values uniform-randomly distributed between 0 and 100.

i) What sequence of input values would constitute the worst case for a regular/counter-based branch predictor with four states (assuming a very large input array)? *Explain and justify your answer.*

ii) How does the number of predictor states impact your answer?

c You are given the following piece of code:

```
bool findValueInTreeArray(int* tree, int size, int needle,
                          int currentPosition = 0) {
  if(currentPosition >= size)
    return false;
  else if(tree[currentPosition] == needle)
    return true;
  else if(tree[currentPosition] < needle)
    return findValueInTreeArray(tree, size, needle,
                                currentPosition * 2 + 1);
  else
    return findValueInTreeArray(tree, size, needle,
                                currentPosition * 2 + 2);
}

int countMatches(int* input1, int input1Size, int* input2,
                 int input2Size) {
  int sum = 0;
  for(size_t i = 0; i < input1Size; i++) {
    if(findValueInTreeArray(input2, input2Size, input1[i]))
      sum++;
  }
  return sum;
}
```

i) Describe the access pattern of this program (`countMatches` is the entry-point) in the *generic cost model access pattern algebra* discussed in class. Discuss aspects of the program that cannot be modeled accurately and how the model falls short.

ii) Form hypotheses regarding the microarchitectural bottlenecks of this program. Since you do not know the hardware parameters of the computer the code runs on, you need to entertain multiple possibilities. Describe how you would verify or falsify each of your hypotheses.

iii) How would you optimize this code?

d As discussed in class, there are three dimensions to consider when measuring and improving system performance. However, two of these are under the control of the system developer. Either of the two can often be manipulated such that the overall system is resilient to varying parameters in the other dimensions. For each of the dimensions,

i) provide a specific example of a parameter and how it can vary

ii) briefly describe how the variation in your example impacts performance and

iii) describe how parameters of the other dimensions can be modified to mitigate the performance impact.

The four parts carry, respectively, 15%, 20%, 40%, and 25% of the marks.

4 a In your own words, define and distinguish performance prediction through interpolation of a numerical model and the fitting of the parameters of the characteristic equation of an analytical model to empirical data. Discuss the advantages and disadvantages of either approach.

b Each of the following snippets exposes one of the four main pipeline hazards. Name each of the hazards and map it to **exactly** one of the snippets. Briefly justify your response.

```
////////////////////////////////////////
void worker(int* input, int size, int* sum, int v) {
  for(int i = 0; i < size; i++)
    *sum += input[i] * (input[i] < v);
}
int f1(int* input, int size, int v) {
  int sums[2];
  thread workerThread(worker, input, size / 2, sums, v);
  worker(input + size / 2, size / 2, sums + 1, v);
  workerThread.join();
  return sums[0] + sums[1];
}

////////////////////////////////////////
int f2(int* input, int size, int v) {
  int sum = 0;
  for(int i = 0; i < size; i++)
    if(input[i] < v)
      sum += input[i];
  return sum;
}


////////////////////////////////////////
struct t {
  int a;
  int b;
  int c;
  int d;
};
int f3(t* input, int size, int v) {
  int sum = 0;
  for(int i = 0; i < size; i++)
    sum += input[i].a * (input[i].a < v);
  return sum;
}

////////////////////////////////////////
int f4(int* input, int size, int v) {
  if(size == 0)
    return 0;
```

```
  int sum = f4(input + 1, size - 1, v);
  if(*input < v)
    sum += *input;
  return sum;
}
```

c You are given two profiles of runs of the same (unknown) application – same input, same binary, same output. The only difference of the two runs is one was run in multi-threaded, the other in single-threaded mode (the application has a switch to control the threading behaviour). The following table summarizes the behaviour of the two runs and as well as the difference between the two runs (as reported by VTune):

Metric	Multi-threaded	Single-threaded	Difference
Elapsed Time	2.112s	25.499s	-23.386s
Clockticks	86,437,000,000	82,550,000,000	3,887,000,000
Instructions Retired	116,246,000,000	111,592,000,000	4,654,000,000
CPI Rate	0.744	0.740	0.004
Retiring	33.1%	31.2%	1.9%
Front-End Bound	8.0%	7.0%	1.0%
Bad Speculation	20.5%	21.0%	-0.5%
Back-End Bound	38.3%	40.7%	-2.4%
↪ Memory Bound	49.3%	50.9%	-1.6%
↪ Core Bound	0.0%	0.0%	Not changed
Average CPU Frequency	2.9 GHz	3.3 GHz	-421.7 MHz
Total Thread Count	22	1	21

i) What microarchitectural component limits this application's performance? Justify your answer.

ii) What kinds of applications expose behavior like this? Give at least two different examples, either in the form of code or a description of a real application you are familiar with.

d Despite recent developments, the predominant language for high-performance programming remains C++. Discuss what language features make C++ a good choice for high-performance programming and what features are, in your justified opinion, lacking.

The four parts carry, respectively, 15%, 25%, 35%, and 25% of the marks.