IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2019

BEng Honours Degree in Computing Part III
BEng Honours Degree in Electronic and Information Engineering Part III
MEng Honours Degree in Electronic and Information Engineering Part III
BEng Honours Degree in Mathematics and Computer Science Part III
MEng Honours Degree in Mathematics and Computer Science Part III
MEng Honours Degrees in Computing Part III
MSc in Advanced Computing
MSc in Computing Science (Specialist)
for Internal Students of the Imperial College of Science, Technology and Medicine

*This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute*

# PAPER C339

# PERFORMANCE ENGINEERING

Friday 22nd March 2019, 10:00
Duration: 120 minutes

*Answer THREE questions*

Paper contains 4 questions
Calculators required

1 a For each of the following statements, state if it is true or false. For each of your answers, give a brief justification.

i) A benchmark should ensure that experimental results are comparable across multiple systems.

ii) Whenever possible, a benchmark should exploit vendor-specific features available in the system under test.

iii) A user behaviour graph can be fitted automatically from log data.

iv) A user behaviour graph can only generate valid navigation paths.

b A small department runs a website hosting four web pages: $H$ (Home), $R$ (Research), $P$ (People), and $T$ (Teaching). Assume that during load testing the following transition probability matrix is used to model user sessions

$$P = [p_{ij}] = \begin{array}{c} \\ E \\ H \\ R \\ P \\ T \\ X \end{array} \begin{array}{c} \begin{array}{cccccc} E & H & R & P & T & X \end{array} \\ \begin{bmatrix} 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0.7 & 0.2 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.1 & 0 & 0.9 \\ 0 & 0 & 0 & 0 & 0 & 1.0 \end{bmatrix} \end{array}$$

where $E$ and $X$ respectively denote entry and exit states.

i) Determine the mean user session length.

ii) Assume that all pages are served from a front server with mean service time $S_f = 1.0ms$. The Home page additionally requires $k = 2$ calls to a database server to refresh latest news and events, each with mean service time $S_d = 1.6ms$. Find the maximum session throughput.

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

c Present the architecture of the SPECjbb2015 benchmark. Then give an example of a situation where you would recommend a company to use this benchmark.

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

2 a A performance engineer is asked to quantify the impact of enabling dynamic resource allocation ($A$) and buffering ($B$) in a stream processing 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 throughput (messages/ms) are as follows:

| A/B | OFF | ON |
|---|---|---|
| OFF | 12 | 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.

iii) Give the sign table for a $2^{3-1}$ fractional factorial design with generator $D = BC$. Indicate the resolution of this design and all its confoundings.

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

$$\mathbf{D} = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 10 & 5 & 9 \\ 4 & 7 & 1 \\ 3 & 0 & 12 \\ 0 & 10 & 0 \end{bmatrix} \end{array}$$

Write a linear program (LP) to establish if resource $B$ is a potential bottleneck or not. Explain how to interpret the optimal value of the LP.

c i) Give an example to illustrate the general form of an autoscaling rule based on static thresholds.

ii) An autoscaling 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 $Var[A_t] = 2$, and lag-1 auto-covariance $K_1 = 0.8$. Fit the AR(1) process parameters to the data.

*The three parts carry, respectively, 50%, 15%, and 35% of the marks.*

3 Preliminaries:
Given the following utility function

```
void* getColumn(std::string);
```

you are given the following three pieces of code (these are artificial pieces of code that do not necessarily have a specific practical use):

Snippet 1

```
void snippet1() {
  int* columnA = (int*)getColumn("A");
  int* columnB = (int*)getColumn("B");
  int* columnC = (int*)getColumn("C");
  for(size_t i = 0; i < inputSize; i++) {
    if(columnA[i] + columnB[i] < 5)
      columnC[i] = columnA[i] * columnB[i];
  }
}
```

Snippet 2

```
struct Point {
  int A; int B; int C; int D; int E; int F; int G; int H;
  int I; int J; int K; int L; int M; int N; int O; int P;
};
void snippet2() {
  Point* columns = (Point*)getColumn("ABCDEFGHIJKLMNOP");
  for(size_t i = 0; i < inputSize; i++) {
    if(columns[i].A + columns[i].B < 5)
      columns[i].C = columns[i].A * columns[i].B;
  }
}
```

Snippet 3

```
void r(int* columnA, int* columnB, int* columnC, //
       size_t inputSize, size_t i) {
  if(i >= inputSize)
    return;
  else
    r(columnA, columnB, columnC, inputSize, i + 1);
  if(columnA[i] + columnB[i] < 5)
    columnC[i] = columnA[i] * columnB[i];
}
void snippet3() {
```

```
  r((int*)getColumn("A"), (int*)getColumn("B"), //
    (int*)getColumn("C"), inputSize, 0);
}
```

a What is false sharing? Under what circumstances is false sharing a performance problem and how can it be addressed?

b Describe the access pattern of each of the code snippets using the access pattern algebra (as accurately as possible). Discuss memory access aspects that cannot be modeled accurately.

c Through profiling, you have obtained the following three performance breakdowns for each of the code snippets

Profile 1

```
Retiring:  37.1%
Front-End Bound:  3.9%
Bad Speculation:  4.6%
Back-End Bound:  54.3%
Memory Bound:  24.2%
Core Bound:  30.2%
```

Profile 2

```
Retiring:  72.4%
Front-End Bound:  3.9%
Bad Speculation:  0.9%
Back-End Bound:  22.8%
Memory Bound:  4.0%
Core Bound:  18.8%
```

Profile 3

```
Retiring:  40.0%
Front-End Bound:  20.4%
Bad Speculation:  3.6%
Back-End Bound:  36.0%
Memory Bound:  6.5%
Core Bound:  29.5%
```

Assign each of the profiles to the code snippet that induced it. Justify your decision.

d Just-in-Time (JiT) compilation of code means to perform parts or all of the compilation process during program execution, i.e., at runtime. Many modern

languages apply JiT compilation to achieve a "compile once, run anywhere" programming experience. However, JiT compilation can also be applied as a means to achieve partial evaluation. Discuss the arising opportunities and the challenges when using JiT compilation to partially evaluate a program.

*The four parts carry equal marks.*

4 a What is perturbation, what causes it and how can it be avoided?

b Describe (in text and/or a figure) the runtime of the following program depending on the parameter `s`. Are there alternative implementations that could perform better?

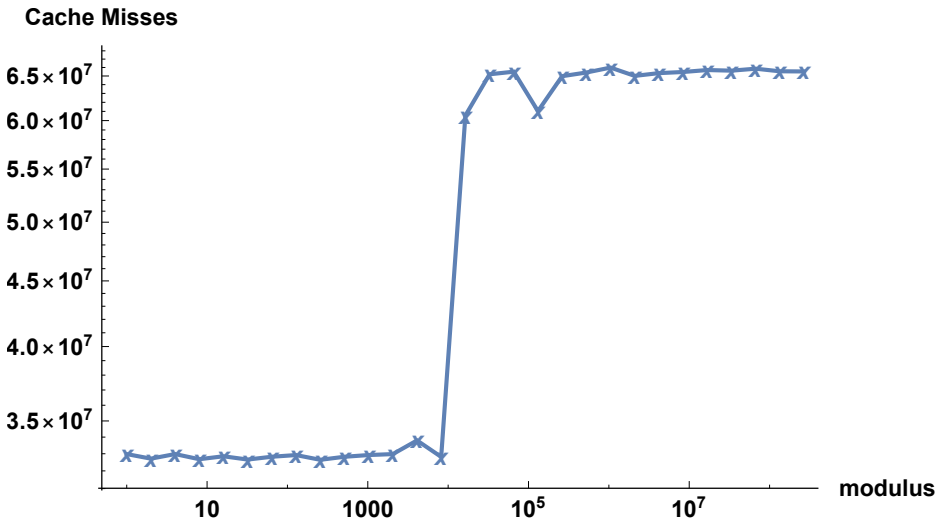
```
int f(int* a, int* b, int* c, int* d, int* d, int s) const {
  for(size_t i = 0; i < inputSize; i++) {
    if(a[i] > s) {
      if(b[i] < s) {
        count += c[i];
      }
    } else {
      if(d[i] < s)
        count += e[i];
    }
  }
};
```

Assume that the arrays `a` through `e` contain uniform random values in the domain `{1,200}`. The values of `a` are sorted while the values in all other columns follow no pattern.

c Assume the following piece of code

```
int doIt(int* input1, int* input2, int modulus) {
  int sum = 0;
  for(size_t i = 0; i < inputSize; i++)
    sum += input1[input2[i] % modulus];
  return sum;
};
```

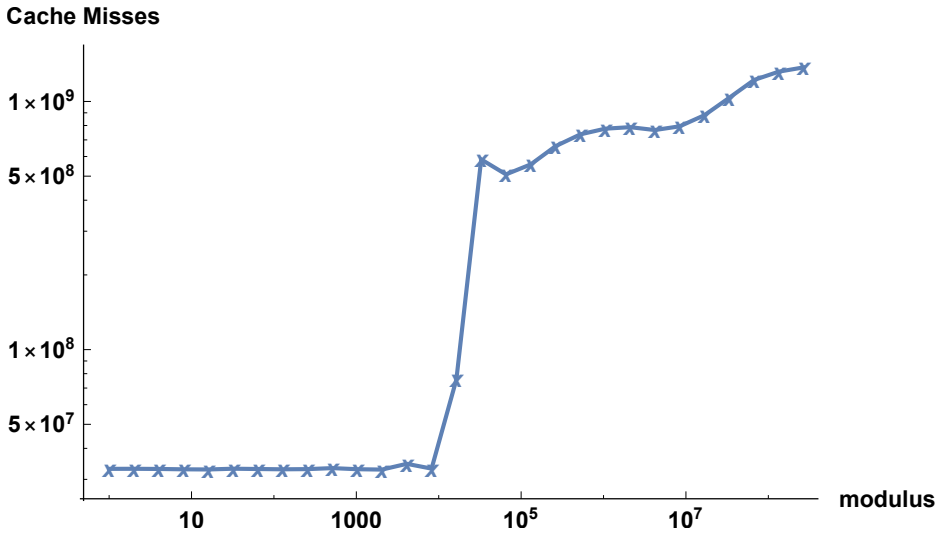
produces the following cache miss profile when varying the value of `modulus`




while

```
int doIt2(int* input1, int* input2, int modulus) {
  int sum = 0;
  for(size_t i = 0; i < inputSize; i++)
    sum += input2[input1[i] % modulus];
  return sum;
};
```

produces the following cache miss profile when varying the value of `modulus`




One of `input1` and `input2` is sorted, the other one shuffled. Which one is which? Justify your decision.

d CPU registers are usually dedicated to holding a certain type of value (int32, int64, float, double) and only support a subset of all operations to be performed on each register. To reduce the number of registers, modern CPUs "reuse" floating point registers for SIMD operations. Discuss the advantages and disadvantages of this design decision.

*The four parts carry equal marks.*