Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

SEMESTER 2 CRA EXAM 2019/2020

MODULE: CA4006 - Concurrent and Distributed Programming

PROGRAMME(S):

CASE	BSc in Computer Applications (Sft.Eng.)
ECSAO	Study Abroad (Engineering & Computing)
CPSSD	BSc in ComputationalProblem Solv&SW Dev.

YEAR OF STUDY: 4,O

EXAMINER(S):

Dr. Rob Brennan	(Internal)	6008

TIME ALLOWED: 3 Hours and 30 minutes

INSTRUCTIONS: Answer 4 questions from the 5 provided. All questions carry equal marks.

Please type all code and pseudo-code rather than using images/screenshots.

This is an open book exam so it is expected that wherever possible you state things in your own words (i.e. do not just cut and paste). Please reference all your sources of images or quotes (supply a URL or academic reference).

Please submit by the “due date” in Loop. There is up to an extra 1 hour for people who have technical difficulties, in which case you must contact DESC (DCU Examination Support Centre):
examsupport@dcu.ie or 01 700 6151

Rob is online for clarifications during the exam: rob.brennan@dcu.ie but all serious issues should go to DESC.

Please monitor your email during the exam in case we need to make an announcement.

OTHER RESTRICTIONS.
All answers will automatically be checked for similarity with internet sources and other students. Exams submitted late for technical difficulties without contacting DESC may have marks deducted. Please note that where a candidate answers more than the required number of questions, the examiner will mark all questions attempted and then select the highest scoring ones.

QUESTION 1 ***[TOTAL MARKS: 25]***

Q 1(a) **[16 Marks]**

Define and distinguish clearly between HPC and HTP systems and give an example of a typical application for each type of system. Explain the relative importance of latency for these systems, e.g. using Gustafson-Barsis' Law. Hence briefly describe the types of hardware architectures each system is likely to employ and explain why they are the most suitable.

Q 1(b) **[5 Marks]**

How many tasks must be buffered in a system if it receives 10,000 transactions per minute and the average task latency is 100ms?

Q 1(c) **[4 Marks]**

How would you decompose the work done by the transaction processing system described in Q1(b) to support concurrency? Explain why you picked this strategy.

[End of Question 1]

QUESTION 2 ***[TOTAL MARKS: 25]***

Q 2(a) **[10 Marks]**

What are the requirements for deadlock to occur in concurrent code? Which requirement has been the most important in your studies or projects to date? Explain, using an example, why this is the case.

Q 2(b) **[8 Marks]**

With reference to the Concurrent Code below, answer the questions below:

```
01 public class Account {
02    private AtomicInteger balance = new AtomicInteger(0);
03
04    public int withdraw (int amount){
05          int balance = this.balance.get();
06          balance = balance – amount;
07          this.balance.set(balance};
08          return balance;
09    }
10 }
```

Figure Q2. Bank Account

i) Where is the critical section in this code?
ii) Does the AtomicInteger on line 02 protect the account balance effectively? Explain your answer.
iii) Is this code threadsafe? Explain your answer.
iv) Re-write the code to be more efficient and concurrently correct.

Q 2(c) **[7 Marks]**

Modify the code in fig Q2 above to give a correct concurrent transaction class capable of transferring money from one account to another by checking the available balance first and ensuring that no funds are lost mid-transaction due to the activity of many threads in the system.

[End of Question 2]

QUESTION 3 ***[TOTAL MARKS: 25]***

Q 3(a) **[8 Marks]**

Explain the operation of a monitor and how it can be used to control access to a shared resource. Include references to the diagram in fig. Q3 below. Describe a scenario, e.g. using pseudo-code, where the threads A, B and C manipulate the shared data in that order due to the operation of the monitor.

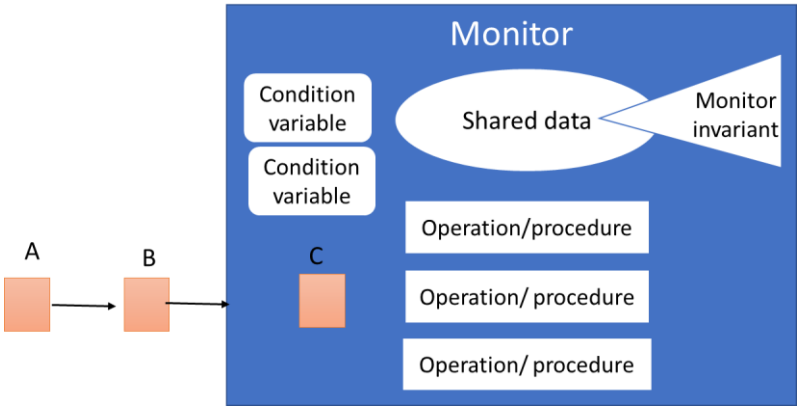

Figure Q3: A Monitor

Q 3(b) **[6 Marks]**

Describe at least three features (or pitfalls or limitations) of `synchronized` keyword monitors in Java. What is the most common problem you encountered with the `synchronized` keyword in your project work? (Explain your reasons, with an example).

Q 3(c) **[11 Marks]**

Write code for the sleeping barber problem using `lock` and `condition` objects in java to implement a `Barbershop` class. The barber and customers are interacting processes, and the barber shop is the monitor in which they interact. You are only required to implement the body of the monitor, not the main code that uses it.

[End of Question 3]

QUESTION 4 ***[TOTAL MARKS: 25]***

Q 4(a) **[10 Marks]**

What parallel processing design pattern is illustrated in figure Q4 below? Describe the main features of this pattern. In figure Q4 what would be the optimum number of processors for the tasks illustrated? Explain the rationale for your answer. Describe an example use case for this pattern explaining why it is especially applicable in your opinion.

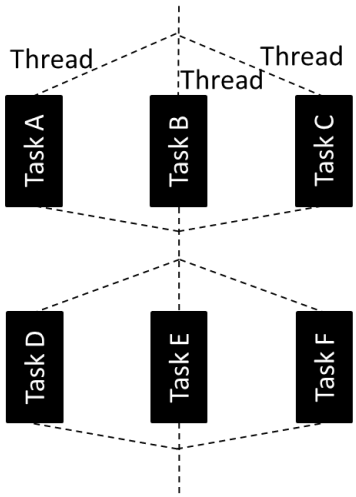

Figure Q4: Parallel Pattern

Q 4(b) **[10 Marks]**

Explain with the aid of a worked example the operation of the following distributed MPI code for calculation of Pi. Describe each MPI command used and why it is important for parallel execution of the program.

[continued on next page]

```
int main(int argc, char *argv[]) {
MPI_Init(&argc, &argv);
#define INT_MAX_ 1000000000
int myid, size, inside=0, outside=0, points=10000;
double x,y, Pi_comp, Pi_real=3.14159265358979323846264 3;
MPI_Comm_rank(MPI_COMM_WORLD, &myid);
MPI_Comm_size(MPI_COMM_WORLD, &size);
MPI_Bcast(&points,1,MPI_INT, 0, MPI_COMM_WORLD);
rands=new double[2*points];
for (int i=0; i<2*points; i++ ){
      rands[i]=random();
      }
for (int i=0; i<points;i++ ){
      x=rands[2*i]/INT_MAX_;
      y=rands[2*i+1]/INT_MAX_;
      if((x*x+y*y)<1) inside++      /* point is inside unit circle so incr
var inside */
}
delete[] rands;
if (myid == 0) {
      for (int i=1; i<size; i++) {
            int temp;
            MPI_Recv(&temp, 1, MPI_INT, i, i, MPI_COMM_WORLD,
                MPI_STATUS_IGNORE);
            inside+=temp;     }
}
else
      MPI_Send(&inside, 1, MPI_INT, 0, i, MPI_COMM_WORLD);

MPI_Reduce(&inside,&total,1,MPI_INT,MPI_SUM,0, MPI_COMM_WORLD);
if (myid == 0) {
      Pi_comp = 4 * (double) inside / (double)(size*points);
      cout << "Value obtained: " << Pi_comp << endl << "Pi:" << Pi_real <<
endl;}
MPI_Finalize(); return 0;
}
```

Figure 4 MPI Code for Calculation of Pi

Q 4(c) **[5 Marks]**

How could the code in part (b) be made to scale more efficiently for compute-intensive problems with high communication costs and extreme locality? Explain your assumptions.

[End of Question 4]

QUESTION 5 ***[TOTAL MARKS: 25]***

Q 5(a) **[11 Marks]**

Give an example of a commercial distributed system. Describe two standard types of system design choices for distributed systems in terms of: the choices available, complications/issues of distributed systems, and design problems to be addressed. Give an example of a distributed design choice you made in your project work and explain what worked (or did not) and why.

Solutions of Question5 part a

Example commercial distributed system [1 mark]

The types of design choices we must make are based on i) architecture, ii) implementation, iii) performance, iv) fault tolerance and v) consistency.

Q 5(b) **[14 Marks]**

For the RPC system illustrated in the UML activity diagram in fig Q5 below:

(i) Describe what is happening in the diagram.
(ii) What type of delivery semantics is illustrated? Explain why.
(iii) Write carefully commented pseudo code for the RPC client and server to support this semantics.

[continued on next page]

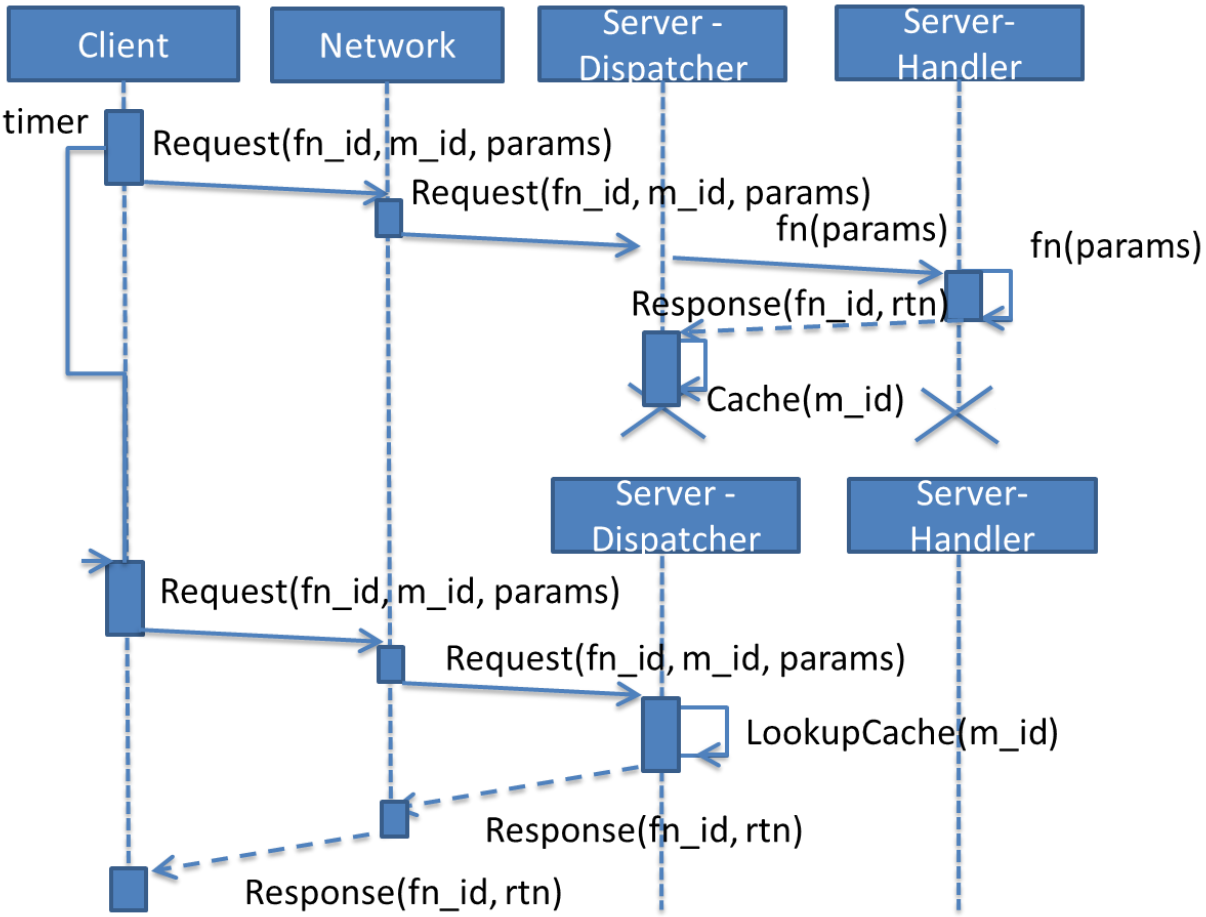

Figure Q5 An Asynchronous RPC System

[End of Question 5]

[END OF EXAM]