THE BCS PROFESSIONAL EXAMINATIONS
BCS Level 6 Professional Graduate Diploma in IT

April 2008

EXAMINERS REPORT

Distributed & Parallel Systems

General Comments

Despite the steadily increasing importance of distributed and parallel systems in science, education and commerce, the number of students taking this paper continues to be very modest indeed. Looking at the performance, several students appear to be poorly prepared to tackle the examination, which seems to be the trend. This may be a consequence of the broad range of topics with which the student must become familiar in order to confidently tackle the paper. Given the above, there were some very good answers which indicates that good preparation is the key.

Question 1

a) Distinguish between heavyweight and lightweight processes.

(5 marks)

b) Distinguish between blocking and non-blocking inter-process communication mechanisms.

(5 marks)

c) Distinguish between pre-emptive and non pre-emptive approaches to thread scheduling.

(5 marks)

d) What is the purpose of the distributed garbage collector in RMI? How is it implemented?

(10 marks)

Answer Pointers

a) Both heavyweight processes (HWP) and lightweight processes (LWP), both found in the Linux kernel, are used to achieve multitasking. In contrast to a HWP, LWP share their address space and system resources with other processes. However, unlike threads, LWPs have private process identifiers and have parent-child relationships with other processes. Whilst threads may be managed at the application level or by the kernel, LWPs are managed by the kernel, operating within the same scheduling framework as HWPs. LWPs differ from HWPs in that they posses a minimal execution context and only sufficient accounting information to enable scheduling. The term “process” usually refers to a running program, whilst LWP typically represents a thread of execution within a program (indeed, LWPs can be conveniently used to implement threads).

b) In non-blocking communication framework (e.g., in Linux systems programming in C, one that uses the IPC_NOWAIT setting), a process does not have its execution suspended waiting for a data item arrive. Conversely, a blocking communication mechanism will suspend the receiving process until the expected data arrives. Using blocking communication, deadlock can occur if the data item does not arrive. With non-blocking communication, it is necessary to build contingency into the program to deal with the scenario that the data has yet to arrive (e.g., periodically re-checking, and performing other duties in lieu of processing the absent data item). However, blocking is a useful mechanism for synchronizing processes.

c) In preemptive scheduling, processes can be forcibly removed from the CPU when the scheduler decides that a different process should be furnished with CPU time. This allows processes that are technically runnable to be suspended. Conversely, in non-preemptive scheduling, processes must yield the CPU, and cannot be forcibly removed. The responsibility for this to be done lies with the programmer. Typically, shorter jobs have a lower priority than longer jobs, though strategies are used to promote overall fairness (e.g., ensuring that processes do not wait indefinitely). Where a process is does not yield the CPU, and it is not suspended for other reasons (e.g., is set to wait for an unavailable resource), it can run to completion.
d) Distributed garbage collection is accomplished by collaboration between local garbage collection and a specialised module to carry out distributed garbage collection that counts references to remote objects. The purpose is to ensure that objects which are no longer referenced either locally or remotely are destroyed, and the memory they occupied is returned to the system for re-use. Details: 1. Each server maintains a set of names of processes than hold remote object references for each of its remote objects (e.g., A.holders is a set of client processes having proxies for object A). This is held in a column in the remote object table. 2. When a client receives a remote reference to a remote object, A, it makes an addRef(A) invocation to the server of that remote object and then creates a proxy. The server then adds C to A.holders. 3. When a the garbage collector of a client, C, notices that a proxy to a remote object is unreachable, it invokes removeRef(A) to the corresponding server and then deletes the proxy. The server removes C from A.holders. 4. When A.holders is empty, the server's local garbage collector will reclaim the memory occupied by A unless local holders exist. [See recommended text by Colouris, Dollimore & Kindberg].

Examiner's Guidance Notes

41% of candidates attempted this question, which tested understanding of common dichotomies in terminology (question A to C), and RMI memory management (D). Whilst one student attained a near perfect answer of 23/25 marks, 3 of the candidates achieved 0/25, 1/25 and 2/25 marks respectively, meaning that overall this question was poorly answered (mean mark 7/25).

Question 2

a) A sequential program has three principal sections. The input section takes 10% of the total time. Processing section takes 70% of the total time. The output section takes the remaining 20%. What is the maximum attainable speedup if only the processing part can be parallelized?

(5 marks)

b) Briefly describe a scenario that might lead to super-linear speedup.

(5 marks)

c) Distinguish between the terms *scalability* and *granularity* in the context of parallel applications.

(5 marks)

d) Outline the advantages and disadvantages of cluster computing in contrast to conventional high performance supercomputing.

(10 marks)

Answer Pointers

a) Given that only 70% of the program can be parallelized, working on the assumption that a perfect theoretical algorithm has reduced execution of this portion of the program to take no time at all, the remaining portion of the program takes 30% of the original time (i.e., 10% for input + 20% for output). Expressed as speedup, this yields 3.33 since $3.33 \times 30 = 100$.

b) Super-linear speedup can occur where the parallel version an algorithm operates better than N×, where N is the number of processing elements added. This situation is unusual. One scenario in which this might occur is when the original data set is so large than (slow) virtual memory is needed when running on one processing element. Increasing the number of processing elements may mean that, because only a fraction of the data is processed by each element, no virtual memory is required. This can also happen when the data to be processed can all be held in (fast) cache memory with the parallel implementation, but not in the serial implementation.

c) Scalability: This can refer to the ability of the algorithm to exploit additional processing resources made available to it (e.g. increasing the number of processing elements), or to continue to work is a predictable fashion as the volume of data to be processed grows. Granularity: This refers to the ratio of computation to communication. In fine-grain parallelism, individual tasks are small (in code size and execution time). Data are transmitted to processing elements frequently. Conversely, in coarse-grain parallelism, data is transmitted infrequently, and after much computation. Fine granularity increases the potential for parallelism (thus speed-up), but also increases synchronization and communication overheads.

d) Cluster Computing: involves a loosely coupled collection of computers, usually appearing to the programmer as a single computing resource. Individual machines are typically networked together. They are cheap, using commercial off-the-shelf hardware, and (potentially) free software. They are resilient to failure: a fault on a single processing element should not affect the ability of the virtual machine to function. It is possible to use any standard network of computers to perform cluster duties (e.g., overnight, or when idle), or two use a stack of dedicated networked PC, such as in a Beowulf cluster.

Supercomputing: typically employs custom-built CPUs that operate faster than conventional CPUs since they employ cutting edge designs that permit parallel execution of instructions (among other things). Supercomputers are often designed for specific computational tasks (such as numerical calculations), and may perform considerable less well at more generalised tasks. Memory hierarchies are designed to ensure the processor is continuously supplied with data and instructions (i.e., CPU idle time is minimized). I/O systems provide high bandwidth to maximize the speed that data can be moved around the system. Amdahl's law continues to apply: supercomputer design attempts to eliminate the serial portion of programs (see question a) as far as possible to maximize speedup. Where a multiple CPUs are used, they are more tightly coupled than a cluster implementation, with increased communication bandwidth and speed.

Examiner’s Guidance Notes

67% of candidates attempted this question, an achieved a mean of 11/25 marks.

Question 3

a) Briefly describe the role of NFS in a distributed system.

(5 marks)

b) Briefly describe the role of NIS in a distributed system.

(5 marks)

c) A colour videoconferencing application requires 3 bytes per pixel (RGB), operates at a spatial resolution of 320×200 pixels, and a temporal resolution of 10 frames per second. What are the data rate requirements for this item of traffic if transmitted in a raw (uncompressed) format?

(5 marks)

d) What are the three quality-of-service (QoS) parameters? Explain how these will be configured differently for videoconferencing application in contrast to a file transfer application?

(10 marks)

Answer Pointers

a) NFS is an acronym for Network File System. It was developed by Sun Microsystems, Inc, and is a client/server system, typically found in Unix/Linux, that permits users to access files across a network and treat them as if they resided in a local file directory. It is implemented by sharing directories on a server (i.e., exporting), and then mounting these directories on a local file system of a client onto a specific mount point.

b) NIS is an acronym for Network Information System. NIS is the current name for what was previously known as *yp* (yellow pages), and is typically found in Unix/Linux systems. NIS allows individual machines on a networked computer system to share configuration information, particularly password data.

c) The total sum of data requires for each second is video is simply the data required for a single frame multiplied by the temporal resolution. This yields 3×320×200×10 = 9606000 bytes/s (9.6Mb)

d) Three applicable QoS parameters are bandwidth, latency and loss rate. For a file transfer application, bandwidth and latency are do not *need* to have specific minimums (though, clearly, higher bandwidth will decrease the amount of time taken to transmit the file), but loss rate needs to be 0, since data loss cannot be tolerated in the transmission of a file. For a videoconferencing application, bandwidth will need to have a minimum tolerable level in order that the video stream is serviceable. Also, to avoid delay/jitter, latency should be minimized. In videoconferencing, a non-zero loss rate may be tolerated, since data corresponding to a particular video frame is only useful in if that frame is still current to the client. Where data that has not already been transmitted has been lost or is unlikely to be displayed at the client's machine because of its age, it can be abandoned.

Examiner's Guidance Notes

41% of candidates attempted this question, achieving a mean of 8/25 marks. Most candidates made a reasonable attempt at this question. The mean is dragged down by one candidate (who scored poorly in all questions) scoring only 2 marks, and would otherwise have been 9 (i.e., within the pass range, 40%).

Question 4

a) Briefly describe two methods of avoiding deadlock in a transaction and concurrency control system.

(5 marks)

b) Describe the *lost update problem*?

(5 marks)

c) Briefly describe the central server mutex algorithm.

(5 marks)

d) Briefly describe the role of *names* in distributed systems, using three examples to illustrate their usage.

(10 marks)

Answer Pointers

a)

1. Make it a requirement to release all locks held prior to switching contexts. For example, change the code so that context of process A always performs its commit before switching context to process B.

2. Make it a requirement that a given object may not be accessed from more than one context at the same time. For example change the code so that both the update and the select are done from the same context.

b) A lost update is a scenario in which two or more transactions update the same data item (e.g. in a database), but neither transaction is aware of the modification made by the other. Consequently the second change overwrites the first.

c) The simplest mutual exclusion algorithm: the central server grants permission to enter a critical section of code. Processes send a request to the server, and await a reply. A reply is accepted as a token that signifies permission to enter the critical section. The server will reply with the token immediately if no other process currently has the token. If another process does have the token, requests are queued. Once the process has completed execution of the critical section, a message is sent to the central server that signifies a return of the token for allocation to the next process in the queue.

d) Names are used in a distributed system to refer to a variety of resources. Names are needed to request a computer system to act upon a given resource chosen from many to enable the sharing of processes and to enable users to identify themselves. There are instances of names supplied in the form of addresses (physical network addresses and local inter-network addresses), of identifiers (port, process or group identifiers), of text (human-readable names), and of files (usually using human readable text). For example, a textual file might look like this: /users/smith/prog.c , a port or service-specific ID might look like this: 164997-9 , and a network address might look like this: 2:60:8c:2:b0:5a

Examiner's Guidance Notes

50% of candidates attempted this question, achieving a mean of 9/25 marks. The mean is reduced by one candidate scoring 0, and would otherwise have been 11 (i.e., within the pass range, 44%).

Question 5

a) A parallel algorithm is to be implemented to perform a brute-force search for prime numbers. Describe how data partitioning would be undertaken to promote load balancing.

(7 marks)

b) We have implemented the algorithm above on a computer cluster. Sketch a graph to show the expected performance (speedup) as the number of processing elements is increased, assuming that the workload is held constant.

(6 marks)

c) The efficiency of a parallel algorithm reflects how well it exploits the additional processing elements made available (i.e., speedup ÷ number of processing elements). Sketch a graph of expected efficiency as the number of processing elements is increased.

(6 marks)

d) Suggest three guaranteed ways that the speed of the prime-search algorithm could be improved.

(6 marks)

Answer Pointers

a) A brute force algorithm requires that each number in the range to be tested by dividing it by all numbers less than itself to check for remainders. This leads to the situation that lower numbers are faster to test than higher numbers, since there are fewer numbers to divide by. To promote load balancing, it would be necessary to supply an even workload to each processing element. This can be accomplished by avoiding (the more obvious approach of) sending sequential blocks of numbers to each processing element, but instead choosing a start point, and then an offset. For example, assuming 4 PEs, for PE1 start at 1, and jump by 4 (1, 5, 9, 13…). For PE2, start at 2 and jump by 4 (2, 6, 10, 14, …), for PE3, start at 3 and jump by 4 (3, 7, 11, 15, …), for PE4, start at 4 and jump by 4 (4, 8, 12, 16, …). This way, each process has an even share of high and low numbers to test, and computation is relatively equalized.

b) With speedup on the y axis, and processing elements on the x axis, speedup will initially increase from 1 (the original speed) upwards as the number of processing elements increases, but then fall as additional processing elements are added beyond an optimum. This will occur because, since the workload is held constant, one cannot add additional resources indefinitely and expect speed to improve. In an algorithm that shares the workload equally among the PEs, the effort of dividing, distributing and re-assembling the data will begin to outweigh the benefit of adding more computational resources, ultimately slowing the performance.

c) Efficiency is speedup(PE)/PE. The efficiency graph will reflect *sub*-linear but positive speedup <1 early on (i.e. utilization of the additional computing resources, but speedup for N resources will be <N times), followed by lower efficiency later on as additional resources are progressively less well exploited. It is possible to include negative efficiency, e.g., if speedup is -2 (i.e., the parallel version runs slower than the serial version, because of the issues outlined above, using 10 PE, then efficiency is -2/10 = -0.2).

d) Three guaranteed ways are i. increase the power of the processing elements (e.g. upgrade CPUs), ii., increase the efficiency of the algorithm (e.g. replace the brute force algorithm by the sieve of Eratosthenes), iii. increase the speed of the communication medium connecting the processing elements. Note that increasing the number of hosts does not guarantee an increase in speed (because of the issues outlined above).

Examiner's Guidance Notes

A very small number of candidates attempted this question (only 17%), and those that did achieved low marks (a mean of just 4/25 marks). This question required candidates to apply their learning to a specific case study, and would have required some degree of lateral thinking rather than being answerable from memorized bookwork.

Since a proportion of the course is dedicated to *parallel* systems, such poor performance on quite a fundamental question in parallel computing is worrisome.

Question 6

For a job interview, you have been asked to make a 30 minute presentation on the following topic:

Security in Distributed Systems: How is it accomplished?

Sketch out approximately 8 content-rich presentation slides, with associated notes, that you would use for your talk.

Please note: your answer will be assessed for its quality of approach, accuracy of content, clarity of expression, range of discussion, and depth of argument.

(25 marks)

Answer Pointers

This question format is used regularly in the distributed & parallel systems exam. On this occasion, the topic was security in distributed systems. The candidate would be expected to spend approx. 5 minutes on each slide, and to ensure that they are succinct, factual and informative. Credit is given for identifying relevant issues, illustrating these thoughtfully (with both words and diagrams), and having a logical structure to the presentation that leads the reader through the topics in a sensible manner.

Examiner's Guidance Notes

83% of candidates attempted this question, and it accrued the highest mean of any question at 15/25 marks. Some students assembled a larger number of less informative slides, and were only able to identify a small number of relevant topics. Selection of a suitable number of relevant topics, and a clear, detailed presentation of information within or near to the requested number of slides lead to the highest marks**.**