FIT3142-Tutorial 7

2.1 Question 1 (25%)

Explain the difference between the two most common scenarios in the development (design) of a Distributed Application. Which is better?

Adaptation and Systematic design

- Many early and existing distributed applications were adaptations and ports of existing applications written for individual machines or clusters;
- As a result some adaptations may be cumbersome or not deliver their potential in performance or functionality;
- For an application to perform well in a distributed environment its should be architected from the outset to exploit the strengths of a distributed environment and avoid its weaknesses;
- Therefore a systematic design approach is essential for good results.

- **Scenario A:** "Clean Sheet of Paper" – permits optimised design at every stage and best possible design choices, but can be very expensive.
- **Scenario B:** "Grid Enabling Existing Application"– may present difficulties dependent on application behaviour; design must accept limitations of existing code; also known as "wrappering".
- • In Scenario B it is usually industry practice to perform a "qualification"of the application to determine whether it is a good candidate for use in a grid environment. If the application fails the qualification, Scenario A may be the only choice.

2.2 Question 2 (25%)

Explain Ferreira's seven qualification failure scenarios for a Distributed Application. What do all have in common?

1. High inter-process communication between jobs without high speed switch connection (for example, MPI); in general, multi-threaded applications need to be checked for their need of inter-process communication.
2. Strict job scheduling requirements depending on data provisioning by uncontrolled data producers.
3. Unresolved obstacles to establish sufficient bandwidth on the network.
4. Strongly limiting system environment dependencies for the jobs.
5. Requirements for safe business transactions (commit and roll-back) via a grid. At the moment, there are no standards for transactions on grids.
6. High interdependencies between the jobs, which expose complex job flow management to the grid server and cause high rates of inter-process communication.
7. Unsupported network protocols used by jobs may be prohibited to perform their tasks due to firewall rules.

2.3 Question 3 (25%)

Explain the relationship between messaging and scalability in a Distributed Application. What are other causes of scalability problems?

The higher the number of message exchange in a distributed application, the lower the scalability of the application due to higher probability of issues can happen to the application because of the scaling action itself.

Other causes of scalability problems:

- Algorithm does not scale well, and there are no good alternative algorithms.
- high data dependencies
- unmet network performance and bandwidth
- lack of financial resources
- application algorithm have bad support for distributed environment

2.4 Question 4 (25%)

Explain the differences between compressing and compacting messages. Which would you employ where?

Compacting Message Data

- If single precision is acceptable for messaging, then how much do we save by using 32-bit single precision vs 64-bit double precision, given N^2data elements in a message?
- How many bytes are required if we convert from double to single precision, per message?
- Given message size is 8 bytes for each "double" and 4 bytes for each "float", then 8/4 = 2; 2^2= 4.
- By using a single precision rather than double precision value in the message, we have reduced the size of a N^2data element message four fold.
- In some computations the messages which are exchanged may involve sparse matrices, or repeated patterns of numbers.
- Such behaviours, if repeated over and over again, can be exploited to reduce message size.
- In such situations, it is feasible to set up the messaging so that what is transmitted comprises the parts of the message which change message by message.
- This idea is used for instance in VJ TCP header compression, for low speed PPP links.

Compressing Message Data

- Another technique for reducing message sizes is that of compressing messages before transmission, and uncompressing them after receipt.
- Compression and decompression is usually more expensive computationally, compared to compacting data.
- However, compute power is relatively cheap, and in a grid application where communications delay is very expensive, compression may work well.
- There are many good compression algorithms and mature well tested code available.