**CS 6378 – Advanced OS**

**Project 3 – Preventing Useless Checkpoints**

Viswam Nathan (vxn078000)

Rahul Ratthen Dhanendran

Lakshmi Shanti Duddu

6 May 2014

**Introduction**

For this project we were to implement a distributed communication system, in which each of the nodes periodically took checkpoints. Then, on top of this we were to implement *Helary et al’s* algorithm for preventing these checkpoints becoming useless. In order to do this the algorithm makes some receiving nodes take additional ‘forced checkpoints’ as and when appropriate. A measure of the performance of the algorithm is the number of forced checkpoints it takes with respect to the total number of independent checkpoints that it takes normally. There are two factors that affect this performance: the message transmission time (MTT) and independent checkpoint time (ICT). We varied these two parameters and observed the performance changes of the implemented algorithm.

**Experimental Setup**

Our system consisted of 10 nodes in a completely connected graph. The distributed system communication was established through TCP protocol.

The program takes as input the desired mean MTT and desired ICT. These two times were to be randomly generated and updated dynamically during the course of the run. The MTT was exponentially distributed and the ICT was distributed uniformly between [0.5\*ICT to 1.5\*ICT].

Each node consists of 4 threads running in parallel. The main thread was responsible for initializing and running the algorithm module for preventing useless checkpoints when necessary. An MTT thread was responsible for initiating message transmissions at random times distributed exponentially, and similarly an ICT thread initiated independent checkpoints at random time intervals as dictated by the uniform distribution. Finally, a receiver thread was responsible for listening to the socket, receiving messages from other nodes and indicating to the main thread that algorithm processing was necessary. Each message transmission would involve piggy-backing all the information required by Helary’s algorithm and then picking a random destination to which to unicast.

All nodes are started in quick succession using a script, and each node automatically terminates once it has taken 50 independent checkpoints as part of its run.

**Algorithm implementation**

We implemented the protocol described in the provided paper. The algorithm makes a decision on whether or not to take a forced checkpoint each time a message is received on a node. Each node uses its own information such as the number of checkpoints taken as well as the information piggy-backed onto the incoming message which provides information on the sending node. It uses these to check if a zig-zag cycle will result, which means one of the taken checkpoints will become useless. If this is the case it will take a forced checkpoint.

**Method of Analysis**

Each node writes to a text file the number of forced checkpoints it took as part of the last run. The number of independent checkpoints is always fixed at 50 since the node terminates after this. The total sum of all forced checkpoints on all 10 nodes is taken, and divided by 500, which is the total number of independent checkpoints in the system. This gives the *checkpoint ratio* which indicates the performance of the algorithm: a lower ratio indicates fewer forced checkpoints and hence better performance.

Each run had a given input for MTT and ICT and produced a checkpoint ratio. This was repeated 10 times and the checkpoint ratio averaged to obtain a figure of merit for this configuration of MTT and ICT. Subsequently either the MTT or ICT value was changed, and the process was repeated. So we obtained an averaged checkpoint ratio for each MTT-ICT pair. These averaged checkpoint ratio points were plotted against either a fixed MTT and varying ICT, or a fixed ICT and varying MTT.

**Varying MTT with fixed ICT**

For these simulations, the mean for the exponential distribution of MTT times was varied from 10ms to 100ms. The mean of the uniform distribution of ICT was fixed at 1 second.

The following graph shows the results of the simulation.

**Analysis**

We can see a clear trend that shows that as the inter-message transmission time increases, the number of forced checkpoints relative to independent checkpoints decreases. This is understandable since a higher MTT means fewer messages in a given period of time. This in turn means lower probability of z-cycles being formed to create useless checkpoints that would cause the algorithm to take forced ones.

**Varying ICT with fixed MTT**

For these simulations, the mean for the exponential distribution of MTT times was fixed at 50ms. The mean of the uniform distribution of ICT was varied between 0.5 seconds and 1.5 seconds.

The following graph shows the results of the simulation:

**Analysis**

We can see a clear increasing trend in the number of forced checkpoints as the ICT is increased. A low mean ICT means the independent checkpointing is more frequent. This means that, in a sense, the independent checkpoints themselves do the job of the forced checkpoints in preventing z-cycles. Conversely, if the ICT is increased while keeping the message frequency the same, then the algorithm would need to step in and take more forced checkpoints to prevent any checkpoint from becoming useless.

**Conclusion**

In this project we implemented *Helary et al’s* algorithm to prevent useless checkpoints by taking some forced checkpoints at appropriate times to prevent z-cycles. We tested this algorithm in a distributed system with 10 nodes that took independent checkpoints periodically. We then observed the behavior of the algorithm as a function of the variation in MTT and ICT. We found that in general the number of forced checkpoints (relative to the number of independent checkpoints) increased if either the MTT was decreased or the ICT was increased.