Transactional System: Memo

8th December 2014

TO: Dr. Otte

FROM: Salvatore, Ryan, Chris, Andrew

SUBJECT: Transactional System Design

As it turns out, Python is quite picky about allowing multiprocessing.Process objects to interact with global memory.

Specifically, they can't.

Therefore:

Our transactional system uses a simple model, whereby the integers to be modified are stored in a multiprocessing list contained in a Manager object, and accessed via indices. Multiple multiprocessing.Process objects can interact with this list, and if locking is enabled, concurrency is managed via a global lock which indicates whether any one Process object is working on the values of the list at a time.

Multiple instances of the Manager class share a multiprocessing.List object, which allows them to act on the same data as the other instances. (In other circumstances, Python will deep-copy the data contained in an object for each mutliprocessing object, making a system such as this one impossible.) When a Manager wants to modify the data in the list, it requests the global write lock, waits until it is available if necessary, then randomly selects two values to modify and a delta-value by which to modify them. The delta-value for the second item is equal to the delta-value for the first item multiplied by negative 1, so that the sum of all the values in the list always remains the same.

The program terminates either after some specified number of seconds has elapsed, or after a specified number of transactions (value modifications) have been processed. These parameters are specified in the constructor for the Manager object and passed in by main.py, which controls building them.