Parallel Systems Practicum Exam
January 19th 2010

(1) Define the abstract interface of a generic parallel solution for a discrete optimization problem. It should solve the following puzzle problem plus a wide range of similar problems (as wide as possible without adding complexity to the algorithm). The puzzle problem is discussed in the chapter on DOP. You should find the shortest number of moves to arrive at the ordered state. The solution will do a best-first search: it will first search the moves that bring the state closer to the final state. Closer is defined as the number of pieces which are at the correct position. Note that we don't want to explore the same states twice. For the puzzle you can arrive at the same state via different sequences of moves.

So you have to define the interface from the library which is exposed to the user. Give a little explanation on the methods.

(2) Consider the problem of multiple threads needing a common resource and the resource can only be used by one thread at a time. See SpecificNotification.java. The resource here is the SpecificNotification object. Threads will do something with the object. For doing so they first have to acquire the resource and release it when finished.

(a) First, make sure that only one thread can do something with the object at the same time. If the resource is in use, threads will have to wait until it is released.

Attention: the synchronization should happen by the acquire and release, not by making DoSomething() a synchronized method! This is forbidden.

(b) Secondly, add priority rules to the mechanism. If multiple threads are waiting to acquire the resource, the thread with the highest priority gets the resource. You may solve it with two priority levels, but the best solution gives the possibility of N priority levels (e.g. N=5). Show that it works on the given example by giving the threads a different priority level.

(3) Consider a processing pipeline of 9 processes and 5 packages. See Pipeline.java. Implement a message-passing solution for 3 processors, in which the final data packages arrive back at the master. Next, implement the optimization of dynamically migrating processes in case of load imbalances. If out time, write the optimization in pseudocode.