PJ2 For Clusters

Important links:

1. Link to the book: [https://www.cs.rit.edu/~ark/bcbd_2/](https://www.cs.rit.edu/~ark/bcbd_2/) (Chapters 18 - 28 deal with clusters).
2. Link to setting up pj2: [https://www.cs.rit.edu/~ark/runningpj2.shtml](https://www.cs.rit.edu/~ark/runningpj2.shtml) (The link above has a typo when setting up the path for JDK 1.7. Use one of the statements used below instead:

   For bash, $ export PATH=/usr/local/dcs/versions/jdk1.7.0_51/bin:$PATH
   For csh, $ setenv PATH /usr/local/dcs/versions/jdk1.7.0_51/bin:$PATH)
4. Link to examples from the book: [https://www.cs.rit.edu/~ark/bcbd_2/#sources](https://www.cs.rit.edu/~ark/bcbd_2/#sources)

Important points:

1. Set up pj2 in your programming environment and test it using one of the example programs to ensure it works as expected.
2. When running a program on one of RIT's cluster computers, use the tracker interfaces mentioned in [https://www.cs.rit.edu/~ark/runningpj2.shtml](https://www.cs.rit.edu/~ark/runningpj2.shtml) to check node usage.
3. Every cluster parallel program in pj2 is expressed as a ‘Job’ which can consist of one or more ‘Task’(s). Each task runs on a separate node in the cluster.
4. Usually, the main method is used to just sets up the tasks but does not actually execute them. After the main method is done, the tasks are then created and executed by pj2 by running tasks across nodes as specified.
5. There are ways to configure when tasks need to run using the ‘rule’ method.
6. Hybrid cluster programs can be used to combine multithreaded and cluster programming approaches so that every core in each node of a cluster is used for a computation.
7. PJ2 uses the concept of tuple space in order to share objects between different nodes in the cluster. (For communication between tasks). Rules can be used to react to situations where a tuple of a specific format appears in tuple space.
Tasks can listen for tuples, read tuples (with and without blocking). Tuples are immutable.

8. When multiple tasks need to interact with each other, task groups can be used.
9. Use the master-worker pattern if you need to loop across a program and the outermost loop is the parallel loop.
10. Never do any cumbersome processing in the tuple listener. This quickly reduces the efficiency of the parallelism.
11. If you need to implement the reduction pattern, compute the first set of results within each task (in shared memory) and then subsequently the final result by reducing the results from each task. Use tuple space to pass around the results.
12. The main drawback of tuple space is that communication can be expensive in terms of time. Minimizing the number of tuples sent around is key.
13. Cluster programming with pj2 allows 2 levels of load balancing - high level balancing at the task level as well as more low level balancing at the thread level. Use combinations of both to try different load balancing schemes based on the task that needs to be solved.
14. When a problem required some kind of output to a file or other I/O based activity, consider doing this activity in a separate task while using other tasks to perform the actual computation.