Concurrent processing with Blackboard

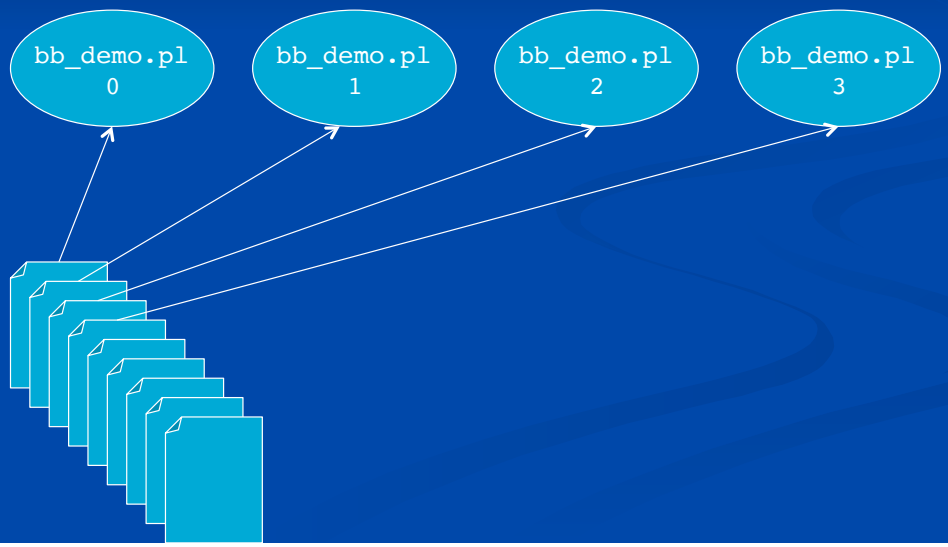
Blackboard strategy

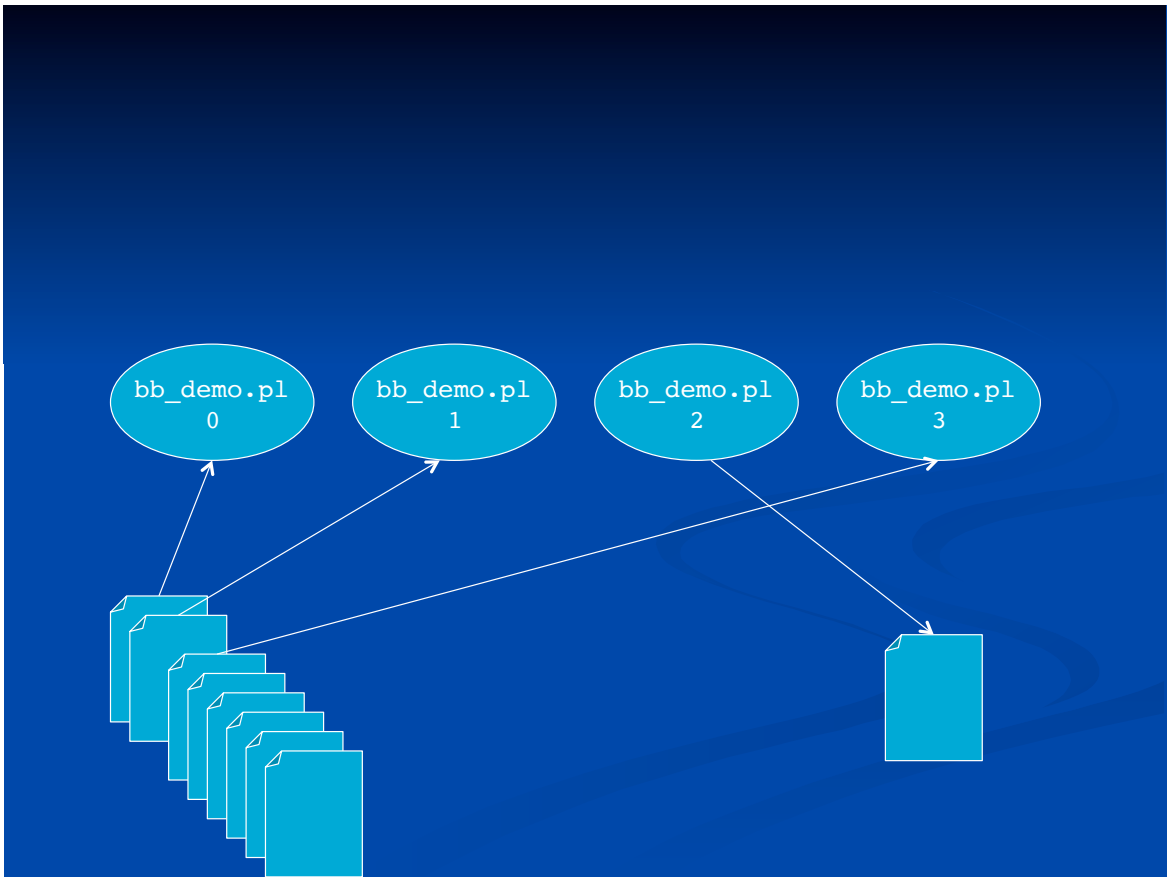
- You wrote a script that process data in a file
- But you have thousands of files to process
- Use a cluster
- Manually or automatically chunk the datasets subsets that can be processed concurrently by...
  - writing separate scripts
  - giving each script a list of input files to process
  - put in separate folders
  - use a database to organize the pipeline
  - Yuk! – there must be a better way...

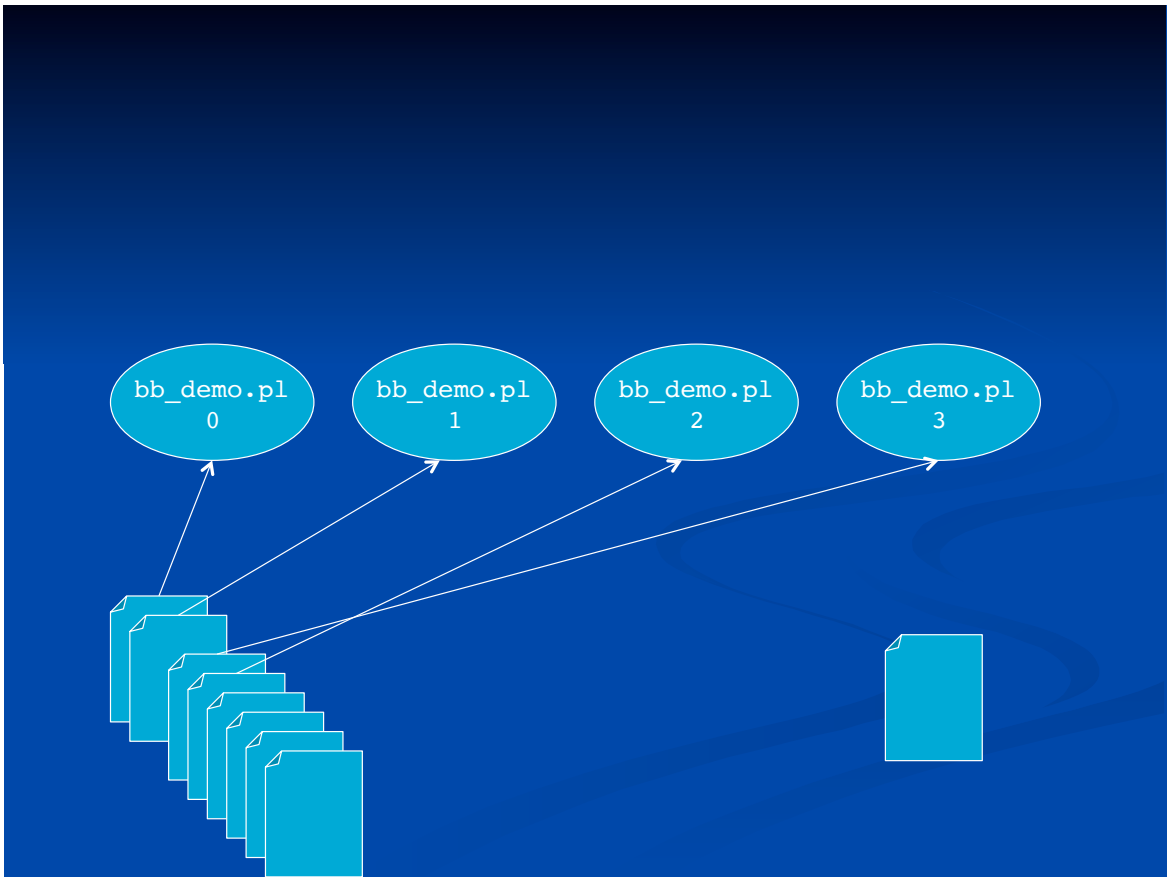
- Launch multiple instances of the identical script (each instance is called an agent)
- Each agent looks at the data and processes the next input file that needs to be processed. When it is finished, it gets another input file.
- Agents communicate with each other so they don't work on the same input files.
- When an agent has no more files to process, it dies.
- The speed up is linear in the number of agents – not going fast enough? Add more agents!
- What does this look like?

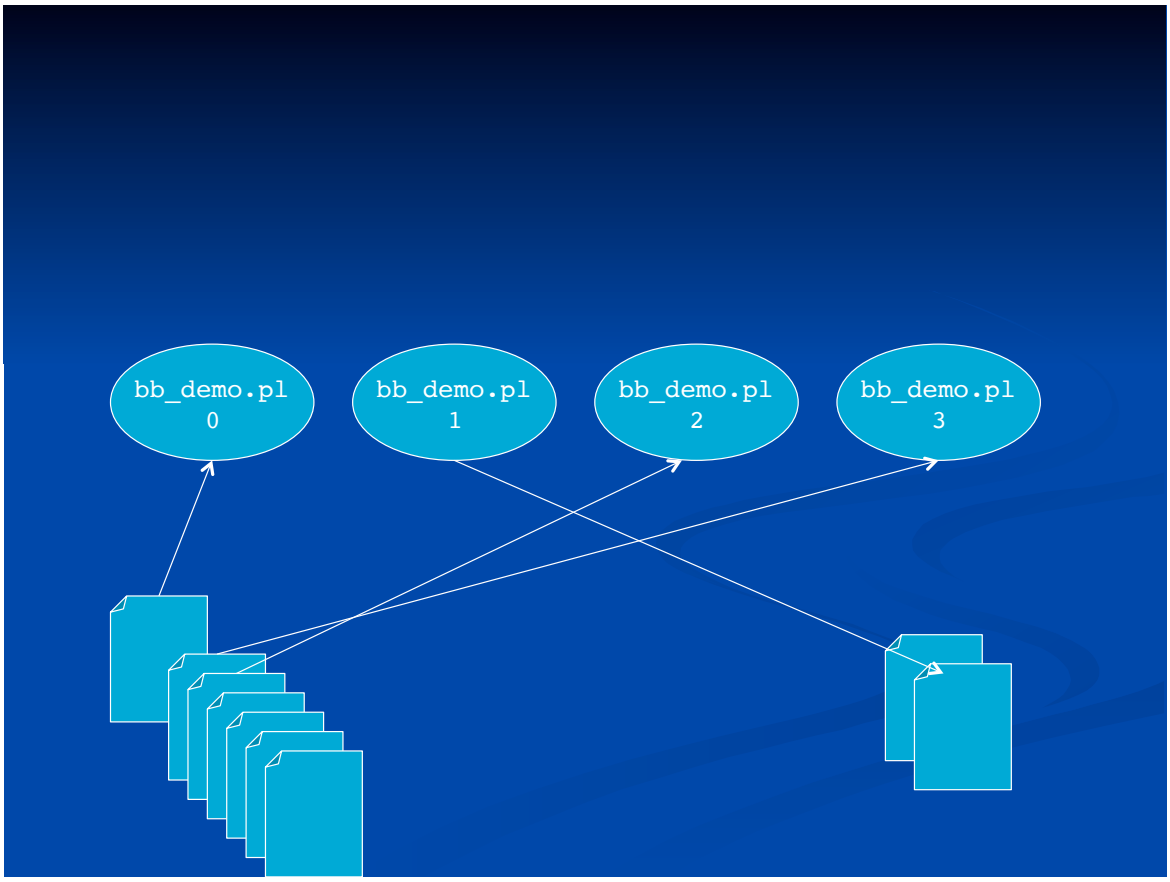
manually launch 4 collaborating agents

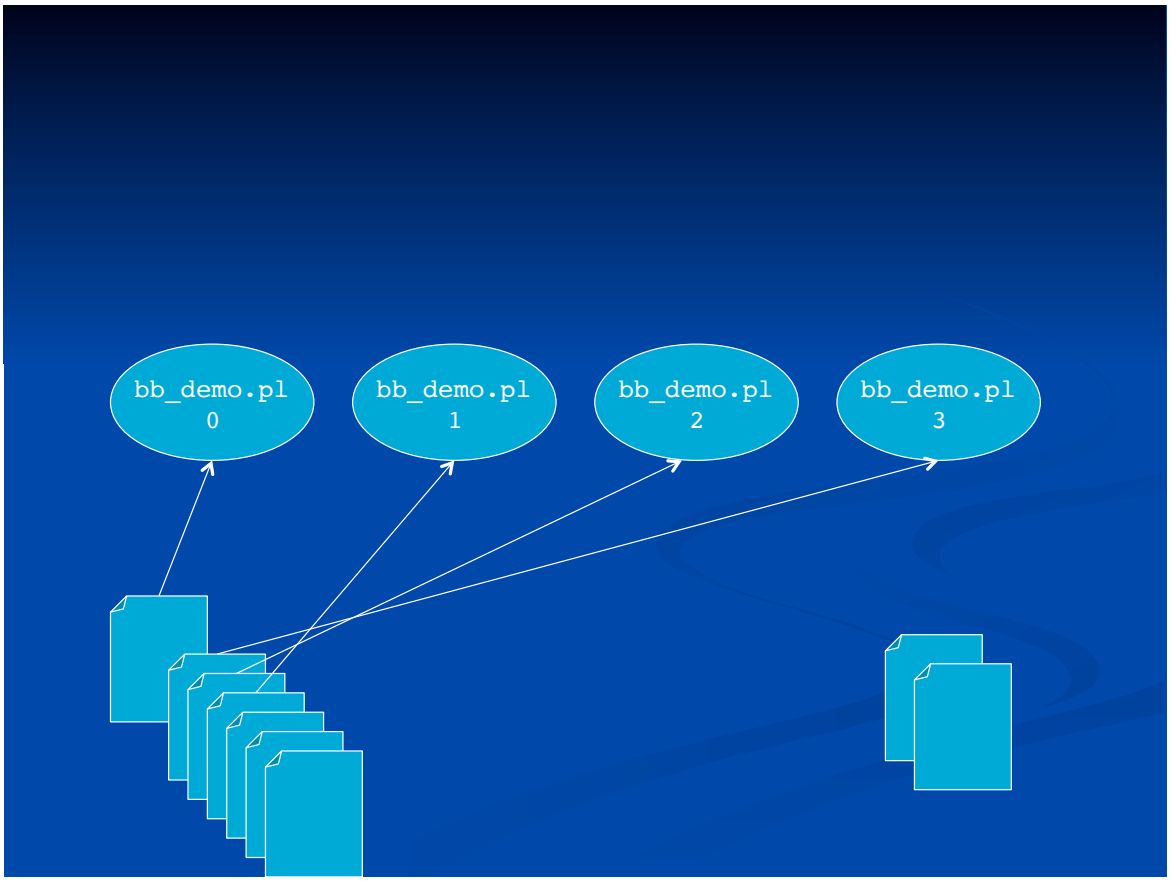
```
./bb_demo.pl
./bb_demo.pl
./bb_demo.pl
./bb_demo.pl
```

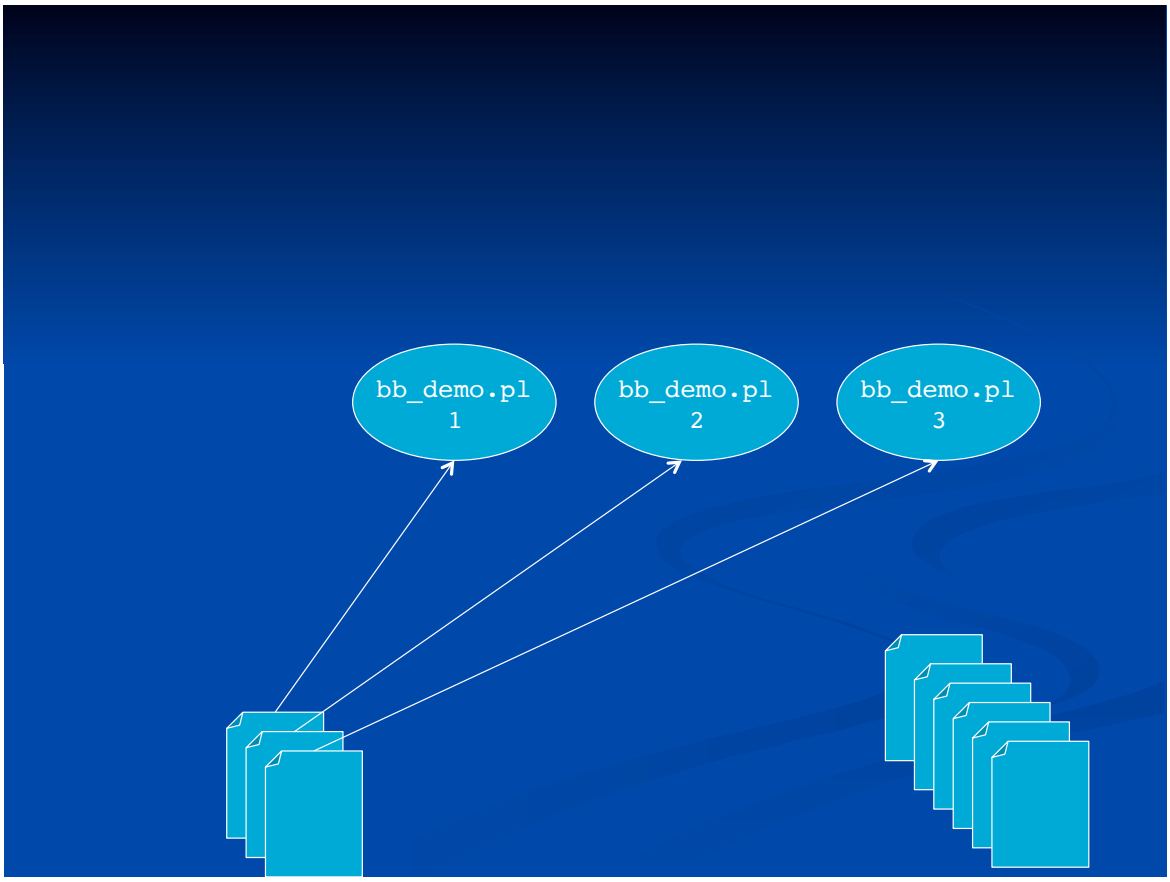


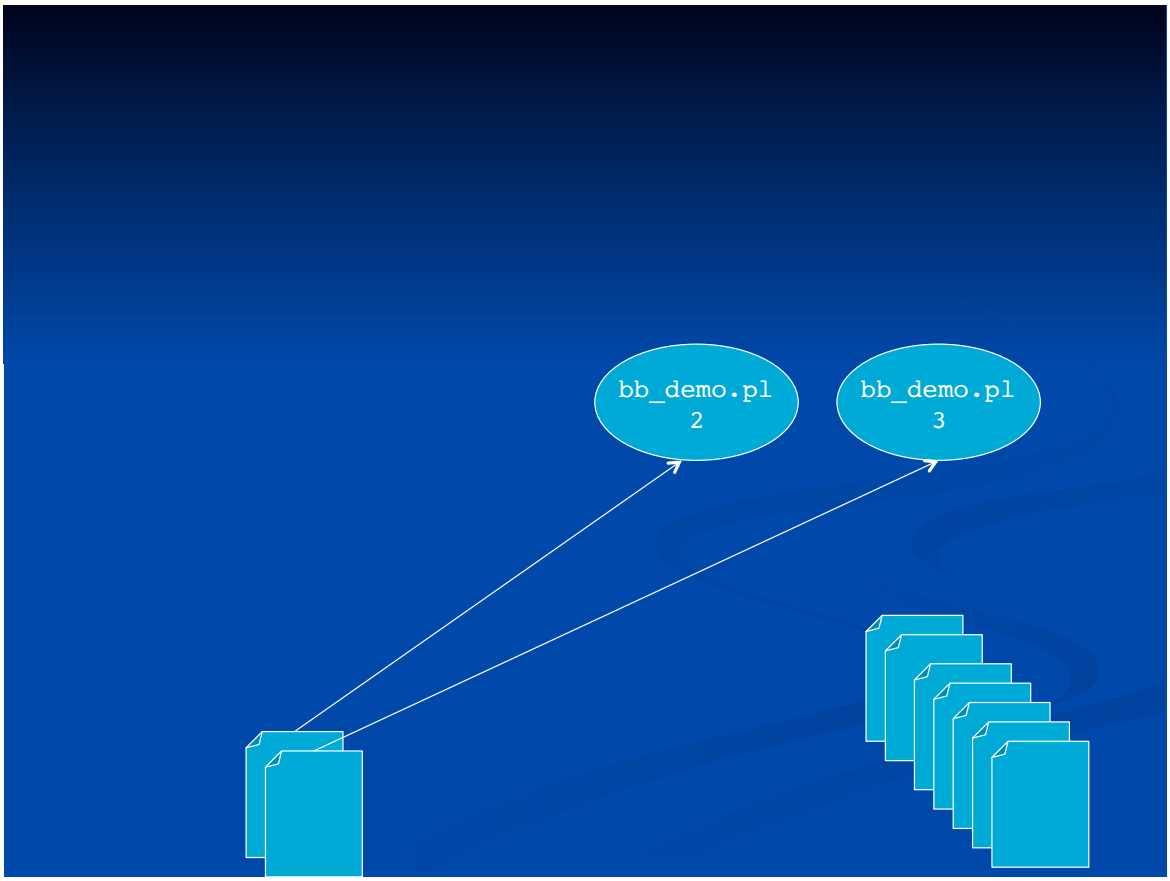

bb_demo.pl
0
bb_demo.pl
1
bb_demo.pl
2
bb_demo.pl
3

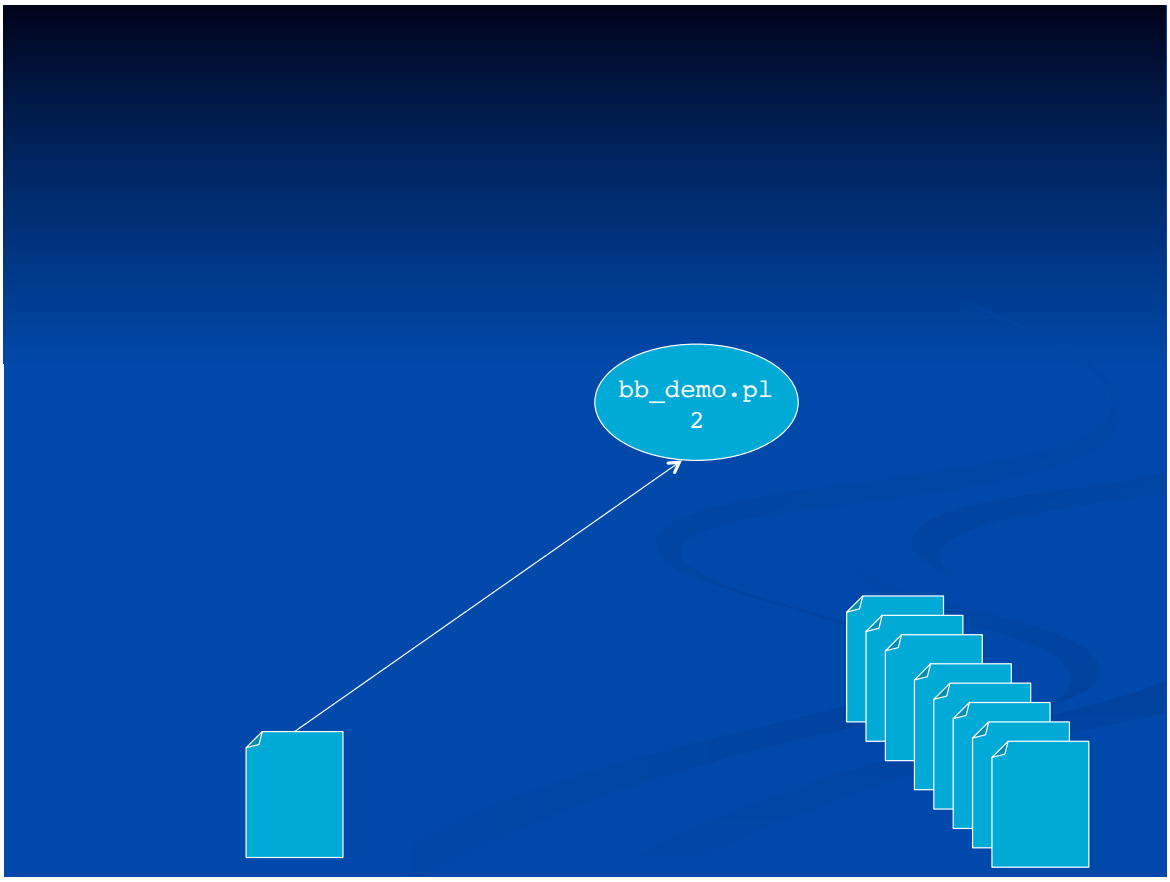

bb_demo.pl
0
bb_demo.pl
1
bb_demo.pl
2
bb_demo.pl
3


bb_demo.pl
0
bb_demo.pl
1
bb_demo.pl
2
bb_demo.pl
3

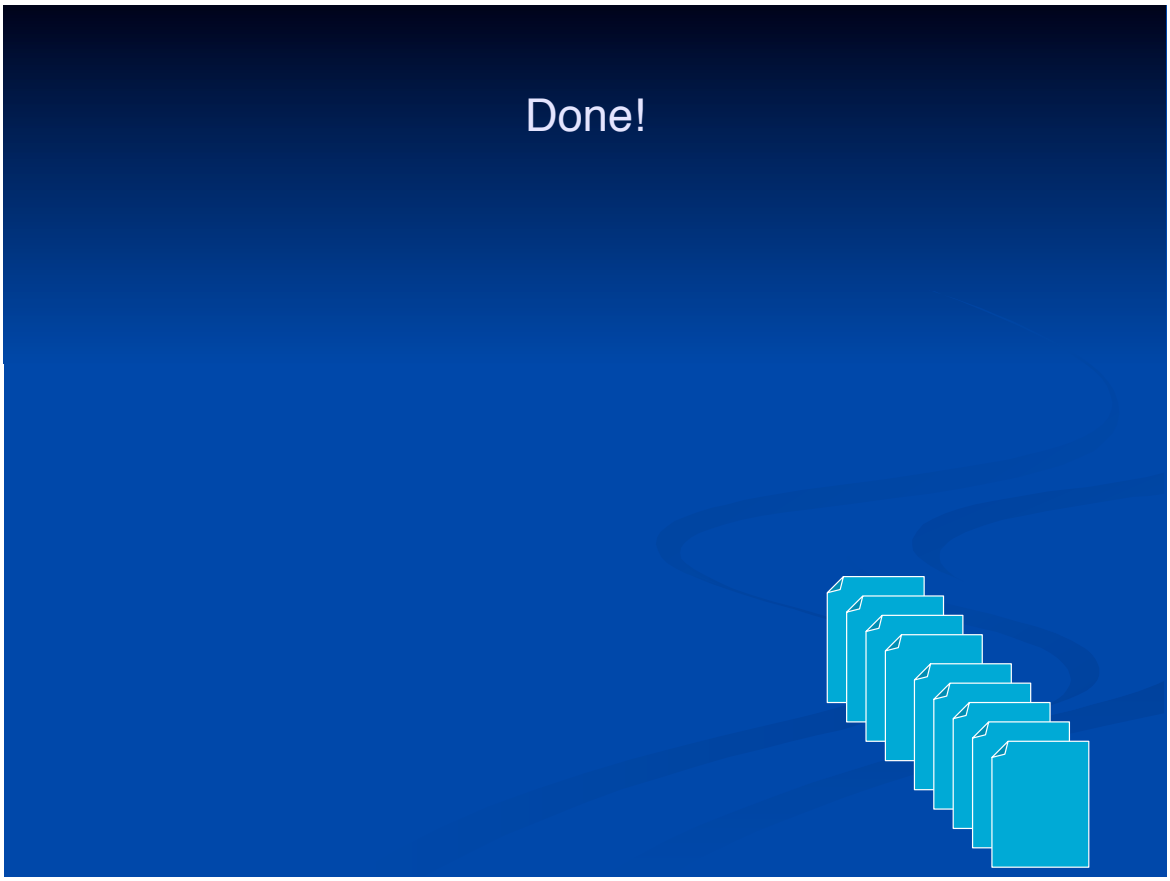

bb_demo.pl
0
bb_demo.pl
1
bb_demo.pl
2
bb_demo.pl
3


bb_demo.pl
1
bb_demo.pl
2
bb_demo.pl
3


bb_demo.pl
2
bb_demo.pl
3



Done!



A simplified blackboard for concurrency control

- Use the file system to maintain status files in a special directory so that agents can keep track of tasks and other agents.
- When an agent starts working on a task, it writes a status file in the blackboard directory.
- Only one agent is allowed into the blackboard directory at a time in order to prevent a "race condition" which would result in two agents working on the identical task.
- Status files have standard names and contain very limited data (process id, time stamp, task status)
- SimpleBlackboard.pm is a perl module that supports this strategy.

Contents of the blackboard and status files

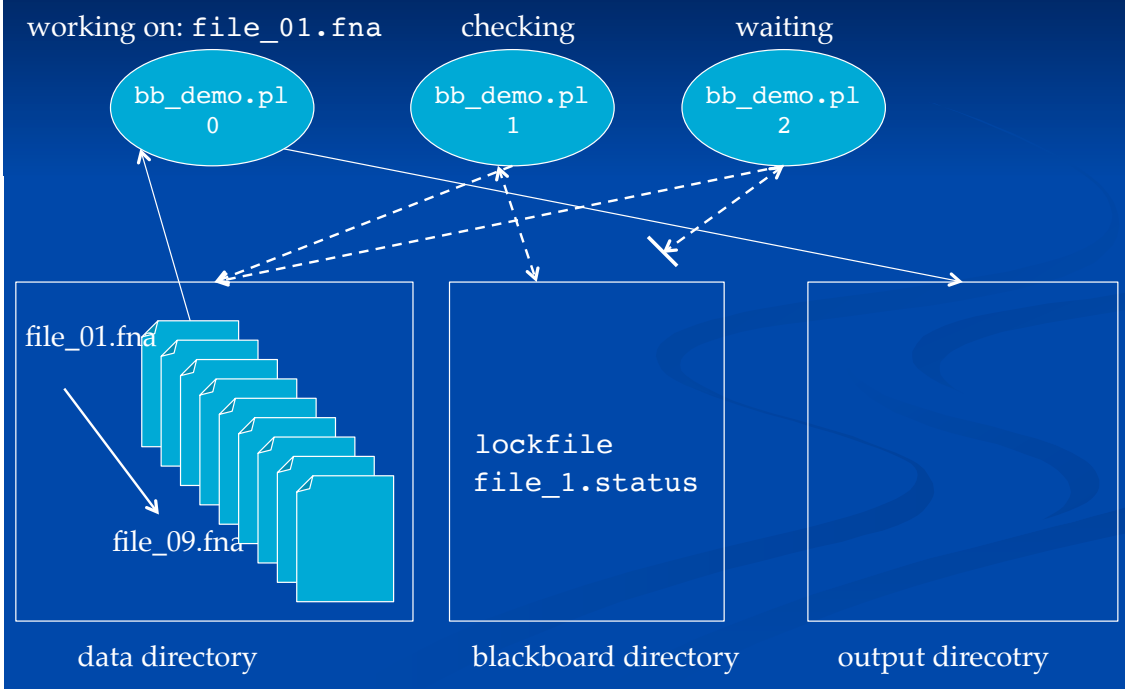
```
enigma2> ls -l
-rw-r--r--+ 1 jrwhite markhamlab 43 Oct  6 00:08 138_26.shorah.status
-rw-r--r--+ 1 jrwhite markhamlab 44 Oct  5 22:41 138_27.shorah.status
-rw-r--r--+ 1 jrwhite markhamlab 43 Oct  6 01:20 139_17.shorah.status
-rw-r--r--+ 1 jrwhite markhamlab 44 Oct  5 23:47 139_18.shorah.status
-rw-r--r--+ 1 jrwhite markhamlab 44 Oct  5 23:37 139_19.shorah.status
-rw-r--r--+ 1 jrwhite markhamlab 44 Oct  6 01:42 139_20.shorah.status
-rw-r--r--+ 1 jrwhite markhamlab 44 Oct  5 23:51 139_21.shorah.status
enigma2> cat 139_21.shorah.status
compute-0-23.local:18495:139_21:shorah:done
```

host : process_id : file_prefix : task : status

what the perl script does

1. scan input directory and make a list of files to process
2. attempt to write the lockfile in the blackboard directory
   1. wait and repeat until successful
3. for each file in the input list, check for a corresponding status file, e.g. to check if the preprocess task has to be performed on `138_26.fna` check for
   `138_26.preprocess.status`
4. if no corresponding status file exists create it, with a status of "started"
5. Delete the lockfile and start performing the task
6. When completed, attempt to write the lockfile again and wait until success.
7. Update the status in the status file to 'done'.
8. repeat steps 1. – 7. until no more files to process

To access the blackboard directory, a process must write a lockfile. If can't write the lockfile, then it waits and retries later



demo: agent.pl

- 1. Add the sample code directory to your PATH
  export PATH=/users/sph140636/shared/code_examples/24_blackboard/$PATH
- 2. In your home directory make a directory named "demo"
  mkdir ~/demo
- 3. cd into demo and make a directory named 'blackboard'
  cd demo
  mkdir blackboard
- 4. launch two agents with qsub
  qsub launch_agent.sh
  qsub launch_agent.sh
- 5. monitor the progress
  while cat *.status; do sleep 2 && clear; done