Using JD2 for your protocols

Or: the squeaky wheel got the grease

Steven Lewis
Kuhlman lab

Outline

- What is A job distributor?
- What's in THE job distributor?
- Why should I use it?
- How do I use it?

What is a job distributor?

- IO:
  - Two for loops: over -s/-l and -nstruct
  - Reads inputs in, prints outputs out
  - Handles my_pdb_0001.pdb numbering
- Cluster architecture layer
  - Single processor
  - Multiprocessor MPI
  - BOINC
  - Oliver's BlueGene stuff

What's in it? - no time for philosophy

- JobDistributor
  - Polymorphic singleton-managed class
    - Access from any part of the code
  - Contains main running loop for Rosetta (virtual, but not yet rewritten)
  - Polymorphism allows for different cluster architectures/behaviors
  - Contains other classes plugged into slots

Input

- JobInputter
  - Determine what jobs exist
  - Determine where to get them
  - Get them when requested; cache them across nstruct
  - Subclassed for:
    - PDBs
    - Silent files
    - etc

Output

- JobOutputter
  - Prints jobs when completed
  - Determines when jobs are already complete
  - Subclassed for:
    - PDBs
    - Silent files
    - etc

JobDistributorFactory

- Reads command line (and compile-time for MPI)
- Which class to use in each hierarchy?

Job and InnerJob

- Contains ID for job and a link to the starting structure for the trajectory

Parser integration

- Parser runs under jd2
- All parser protocols automatically jd2

Why use it?

- Provides portability!!eleventyone

Why use it?

- Provides portability
- Embedded filtering: jobs can report status
  - Pass
  - Fail, repeat
  - Fail, don't repeat
  - Fail, kill my friends because they'll fail too
- Fancy extra output in PDB/scorefile like ++
- Mover refreshing between trajectories

No embedded checkpointing

Here's the push

- We really want all the release apps on jd2
  - Users ask us for this constantly!
  - Portability – example: 3.1 AbinitioMPI hack
  - Shared documentation and methods for cluster architecture
  - Unified functionality
  - We agreed to it at last minicon; some progress has been made
  - Replace release apps with jd versions for stuff like score_jd2 and docking_protocol_jd2

OK, I'm sold

- Comes free with RosettaScripts
- Your code MUST BE IN A MOVER
- Your mover requires no JD2 handles
  - But you can use them for the cool features
- Trivially short main
- Examples:
  - src/apps/pilot/JD2/jd2test.cc Only purpose is example!
  - src/apps/public/scenarios/FloppyTail.cc – complete protocol built with JD2 in mind, simple enough for one file

Moving from oldJD to JD2

- Must remove oldJD hooks!
- If you run JD2-hooked code without JD2...
  - It does NOT fail
  - It does NOT necessarily work with extra output, etc
- So you can add JD2 hooks then remove oldJD

Fancier stuff

- Oliver has multiple extra systems in place
  - Silent files only on BlueGenes?
  - I don't know how to use them so ask him
- The system is very extensible!
  - Make your own subclasses for your own purposes!