The Job Distributor

That got made up last night

while most of the designers were drinking

Steven Lewis

What is a job distributor?

- This is the code responsible for:
  - Managing input and output associated with each cycle (essentially the nstruct flag)
  - In some cases, distributing work across processors
- Basically nested `for` loops over your input files and nstruct

Current model

- Three methods
- not really a job distributor

```
//for now we are not bothering with the job distributor
Size const nstruct(option[nstruct].value());
for( core::Size i(1); i <= nstruct; ++i){
	mover.apply( pose );
	std::string filename(
"result_"+ObjexxFCL::right_string_of(i,4,'0')+".pdb" );
	combined.dump_scored_pdb( filename, *score_fxn); //take
that, job distributor!
	combined = copy; //resetting for proper sampling
  }
```

Current model

- Three methods
- Simple but rigid methods from `standard_mains.cc`
- `main_plain_pdb_mover(argc, argv, mover, scorefunction)`

Current model

- Three methods
- Hardcore write your own job distributor using Ian's tools
  - Docking seems to do this

Five problems

1) Inside your protocol, you don't know what iteration (nstruct) you are on for output
   - Centroid poses in centroid/fullatom protocols
2) Cannot suppress output for bad trajectories
   - Docking filters failed
   - Loops left unclosed
3) Hard to get arbitrary output (loop phi/psi) included with PDBs or elsewhere

Five problems

4) What sorts of output a protocol should have should be runtime-determinable
   - PDB, silent, compressed concatenated PDB
5) What architecture your protocol runs on should not require editing code (compile time flags unavoidable)
   - MPI, condor, pseudoparallel

Changes to system from the view of a protocol

- Job distributor is a singleton class
  - Polymorphic under the hood
- Your protocol can use this hook to request that the job distributor give it ways to output things
- The job distributor decides when and how to output these things
- Protocol can also tell job distributor that this run was a waste and should not be saved

Polymorphic

- Three classes in the works
  - VanillaJobDistributor (base class)
    - On your laptop, or pseudoparallel
  - MPIJobDistributor
    - True parallel
  - BOINCJobDistributor

Changes to Mover base class

- Mover base class will implement a virtual function failed()
  - This tells the job distributor that this run should not be output
  - Different protocols can reimplement as necessary to determine if the nstruct counter should be implemented, etc.

Fingers of Blame

Andrew LF
John K
Monica B
Jeff G
Doug R
Ian D
Oliver L
TJ B
Not myself

Changes to movers

- Movers need either:
  - A reinitialize function to ensure that one mover passed to the job distributor can be “restarted” if necessary
  - A no-argument default constructor which will set up everything intelligently (probably from the options system)
- (It wasn’t decided which)

Andrew says

- Can’t OP a stream (because C++ class)
- Make an object which derives from ReferenceCount for OPs...and holds only stream objects
- This is a hack which will work