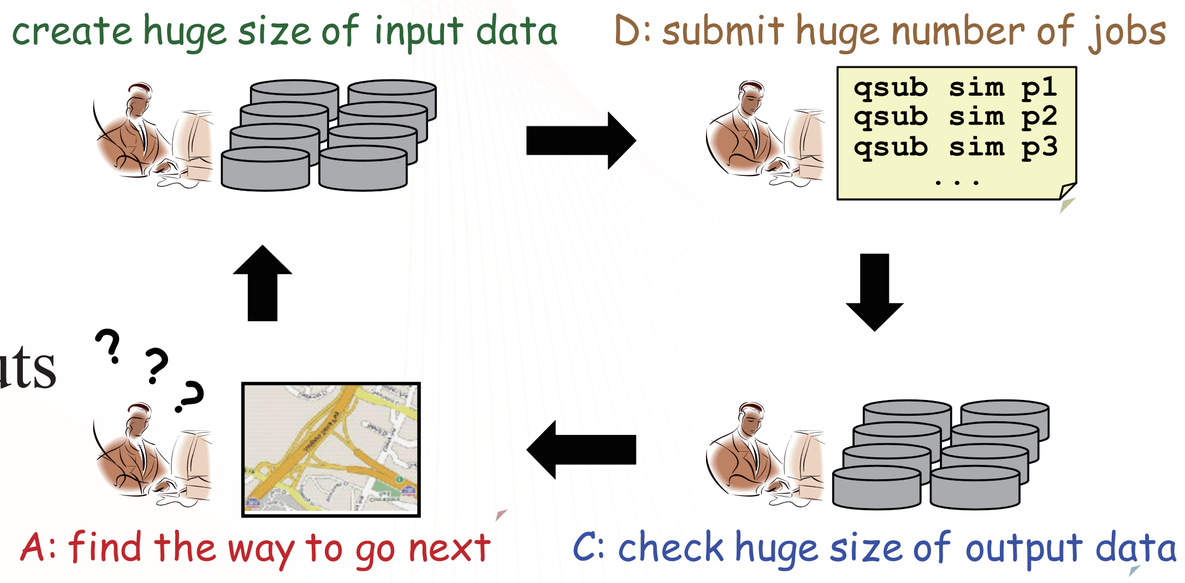
Open Supercomputer

Xcrypt
Highly Productive Parallel Script Language

What's the Problem?

R&D of computational science often has PDCA cycles with *many* runs of a *single* program.

- Automation is not so easy due to
  - preparing/analyzing inputs/outputs
  - generating job scripts
  - managing plenty of asynchronously running jobs



Design Concept

- Language for *job-level* (coarse grained) *parallel* programming
- *Easy-to-write* for simply using predefined magical spells
- Rich *flexibility* and *capability* to enable wizards to add various spells (e.g., smart search algorithms) as *modules*

Features

- *Perl-based*, *object oriented* script language
- *Declarative* job definition, *imperative execution*
  - easy to manage asynchronously running jobs (`submit()`/`sync()`)
- Job script generation
  - *supporting a wide variety of schedulers* by writing configuration files
- Libraries for *generation/extraction* of input/output files
  - *easier than legacy Unix tools* (`sed`/`awk` etc.) to use
- High extensibility
  - Perl wizards can write *class libraries* to implement various features as add-ons

```
use base qw(graph_search core);     # load modules
%mySimulation = (
 exec=> "geom_optimize.exe",
 arg1=> "input.dat",
 arg2=> "output.dat",
 initial_states=> "molecule_conformations.dat",
 before=> sub {       # invoked before submitting each job
  choose a structure from state pool and generate "input.dat"
 }
 after=> sub {              # invoked after each job finished
  evaluate "output.dat" and add new structures into state pool
 }
 end_condition => &isStationary,     # Perl function
);
prepare_submit_sync (%mySimulation);
```

Open Supercomputer All in One