Sphinx Architecture
Architecture/API

- Sphinx as a pluggable, modular middleware
- core + set of core modules + set of additional modules + optional set of user modules
- A module *interface* that can be easily implemented
  - Module activation
  - Module de-activation
  - Module status
  - Module states (register states with the controller)
  - execute
initialize_core /* database and message interface
   initialization */
for all core modules {
   activate
   register functionality /* I am a data planner module */
   register states
}
for all additional modules {
   activate
   register functionality
   register states
   resolve conflicts (if any)
}
read config and decide the right module for
a specific work /* choose one of various exec planners etc. */
while (! finish) {
   start module
   check status of core modules
   if (down)
      restart
}
DO WORK
External API

- Extend and document external API
- SubmitJobs
- QueryJobs (by user, by site, by experiment)
- StatusJob (status of a particular job after proper authentication)
- Status (ping, no. of modules active, services available)
- Notifications (jobFin, dagFin, ...)
Clients

- Clients in python, perl and C
- All these languages have extensive libraries
- Sphinx server administration client
External Communication Module

- Web based XML-RPC with GSI
  - Clarens

- Sphinx server-client communication
  - Peer to peer communication based on Jclarens
  - XML message includes sender information
    - The information is used for response

- User - Sphinx client communication
  - Client and server communication based on java, C or python Clarens
  - User pulls necessary information from the client
Internal Communication Module

- Database tables corresponding to a module
  - Keep track of scheduling information
  - Communicate with other modules using database queries
    - select, insert, delete, update etc.
- Benefit to modulated development
  - Seamless interaction between modules
- Easy fault or error tracing
  - Failure alert back propagation
Sphinx Planning Module

- Provides interfaces to task specific planners
  - Data planning
  - Execution planning
  - Other application planning (hash join planner etc.)

- Admission controller
  - Controls planning works according to policies
  - Manages grid workload and performance

- Grid monitoring information
  - Provides system specific grid weather information
    - CPU, memory, storage, bandwidth etc.
Sphinx Prediction Module

- Provides estimated grid weather information
  - System specific information
    - Resource usage amount
      - CPU, memory, storage, bandwidth etc.
  - Interact with components in planning module
    - Assists intelligent planning
    - Smart data replication,
    - QoS satisfying scheduling, etc.

- Still a placeholder in sphinx
  - Will be implemented in the next version