

DANSE Central Services

Michael Aivazis
Caltech

NSF Review
May 23, 2008

Goal: producing high quality software

- ✦ Correctness
- ✦ Coherence
- ✦ Robustness
 - ✦ *errors in the analysis codes*
 - ✦ *prevent, handle*
 - ✦ *environmental conditions*
 - ✦ *outages*
 - ✦ *misconfigurations*
- ✦ Usability
 - ✦ *making everybody happy...*
- ✦ Aesthetics

Mission

- ✦ Central Services provides expertise and support for
 - ✦ *hardware*
 - ✦ *project servers*
 - ✦ *computational cluster*
 - ✦ *data storage and simulation archives*
 - ✦ *software engineering tools and practices*
 - ✦ *web, wiki, bugs and feature requests*
 - ✦ *configuration and release management*
 - ✦ *software architecture and framework*
 - ✦ *components*
 - ✦ *user interface design*
 - ✦ *services*
 - ✦ *algorithms and data structures*

Hardware

- ✦ Servers
 - ✦ *web*
 - ✦ *source control*
 - ✦ *distribution*
- ✦ Cluster
 - ✦ *currently: 7 nodes (x4 Opteron cores with 2Gb of memory per core)*
 - ✦ *future expansion limited by budget*
- ✦ Data storage
 - ✦ *36 Tb of raw capacity*
 - ✦ *2.5GB per second sustained read, 1.5GB per second sustained write*
 - ✦ *data storage, simulation archiving*

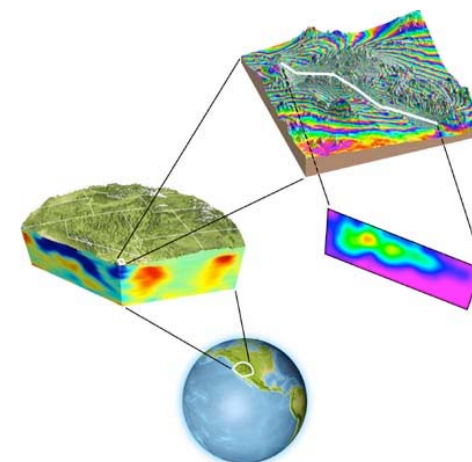
Software engineering

- ✦ Source control
- ✦ Robust and portable software build procedure
- ✦ Modern software design and implementation practices
 - ✦ *documentation of requirements, use cases, design, algorithms*
 - ✦ *agile programming: “test before you implement”*
- ✦ Quality control automation:
 - ✦ *regular (nightly?) builds of the software base*
 - ✦ *unit testing*
 - ✦ *regression testing*
- ✦ Tracking of bugs and feature requests
- ✦ Release management
- ✦ Documentation
 - ✦ *doxygen, epydoc, docbook, wiki, UML*

Pyre: the software framework

✦ Projects

- ✦ *Caltech ASC+PSAAP Center (DOE)*
- ✦ *Computational Infrastructure in Geodynamics (NSF):*
- ✦ *DANSE (NSF)*



✦ Portability:

- ✦ *languages: C, C++, F77, F90*
- ✦ *compilers: all native compilers on supported platforms, gcc, Absoft, PGI*
- ✦ *platforms: all common Unix variants, OSX, Windows*

✦ Statistics:

- ✦ *1200 classes, 75,000 lines of Python, 30,000 lines of C++*
- ✦ *Largest run: **nirvana** at LANL, 1764 processors for 24 hrs, generated 1.5 Tb*

Flexibility through the use of scripting

- ✦ Scripting enables us to
 - ✦ *Organize the large number of simulation parameters*
 - ✦ *Allow the simulation environment to discover new capabilities without the need for recompilation or relinking*
- ✦ The python interpreter
 - ✦ *The interpreter*
 - ✦ *modern object oriented language*
 - ✦ *robust, portable, mature, well supported, well documented*
 - ✦ *easily extensible*
 - ✦ *rapid application development*
 - ✦ *Support for parallel and distributed programming*
 - ✦ *a python interpreter on each compute node*
 - ✦ *MPI is fully integrated: bindings + OO layer*
 - ✦ *evolving support for both ad-hoc and grid-based solutions*
 - ✦ *No measurable impact on either performance or scalability*

Persistence

- ✦ The goal is to provide applications with seamless access to large scientific data stores that are becoming available for an increasing number of scientific domains
- ✦ DANSE applications will benefit by getting access to relevant public databases, as well as having a scalable store for archived simulation and analysis meta-data
- ✦ Design is complete; implementation has been through a 2-year beta cycle
 - ✦ *objects persist in RDBMS tables whose rows are object instances and columns are object attributes*
 - ✦ *all SQL 2003 data types are supported*
 - ✦ *PostgreSQL and MySQL backend support*
- ✦ In progress:
 - ✦ *support for more database back-ends*
 - ✦ *tighter coupling to the underlying database to eliminate the need for intermediate data representations*
 - ✦ *automatic generation of an object model based on a developer specified schema*

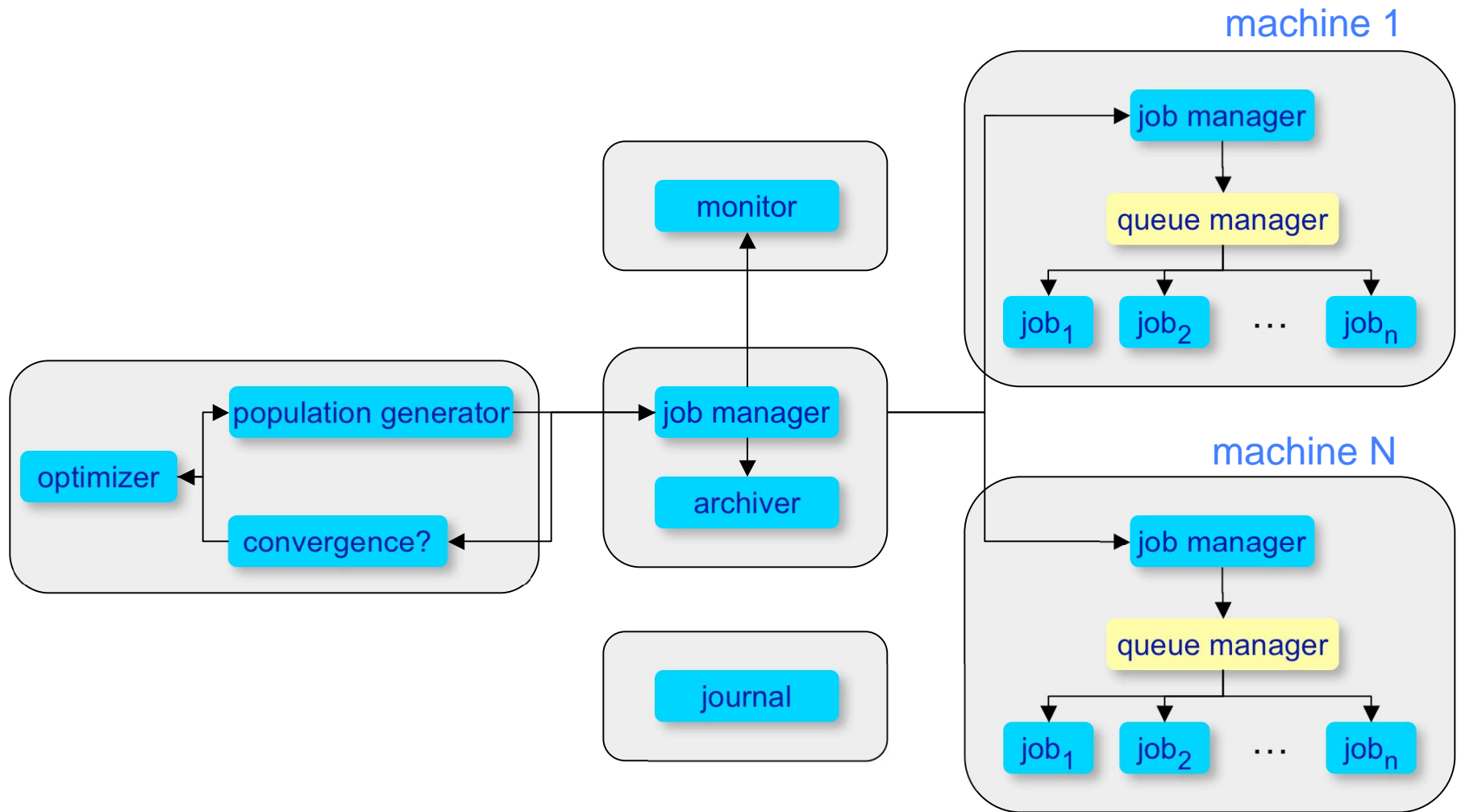
DANSE and distributed computing

- ✦ We will continue to build services and support for creating and monitoring large scale data analyses and simulations from the user's workstation
- ✦ Infrastructure
 - ✦ *encapsulation of commonly used communication protocols*
 - ✦ *protocol is negotiated at run-time*
- ✦ Superstructure
 - ✦ *creation and deployment of a distributed computation*
 - ✦ *programmatic interface*
 - ✦ *management and monitoring tools*
- ✦ Services
 - ✦ *global unique identifier generator*
 - ✦ *one-step remote authentication*
 - ✦ *visualization and monitoring*

Distributed computing support

- ⊕ We are continuing the migration of existing support for distributed processing into `gs1`, a package that completely encapsulates the middleware
- ⊕ Provide both user space and grid-enabled solution
- ⊕ User space:
 - ⊕ *ssh, scp*
 - ⊕ *pyre service factories and component management*
- ⊕ Web services
 - ⊕ *pyGridWare from Keith Jackson's group*
- ⊕ Advanced features: coming soon
 - ⊕ *dynamic discovery for optimized deployment*
 - ⊕ *reservation system for computational resources*

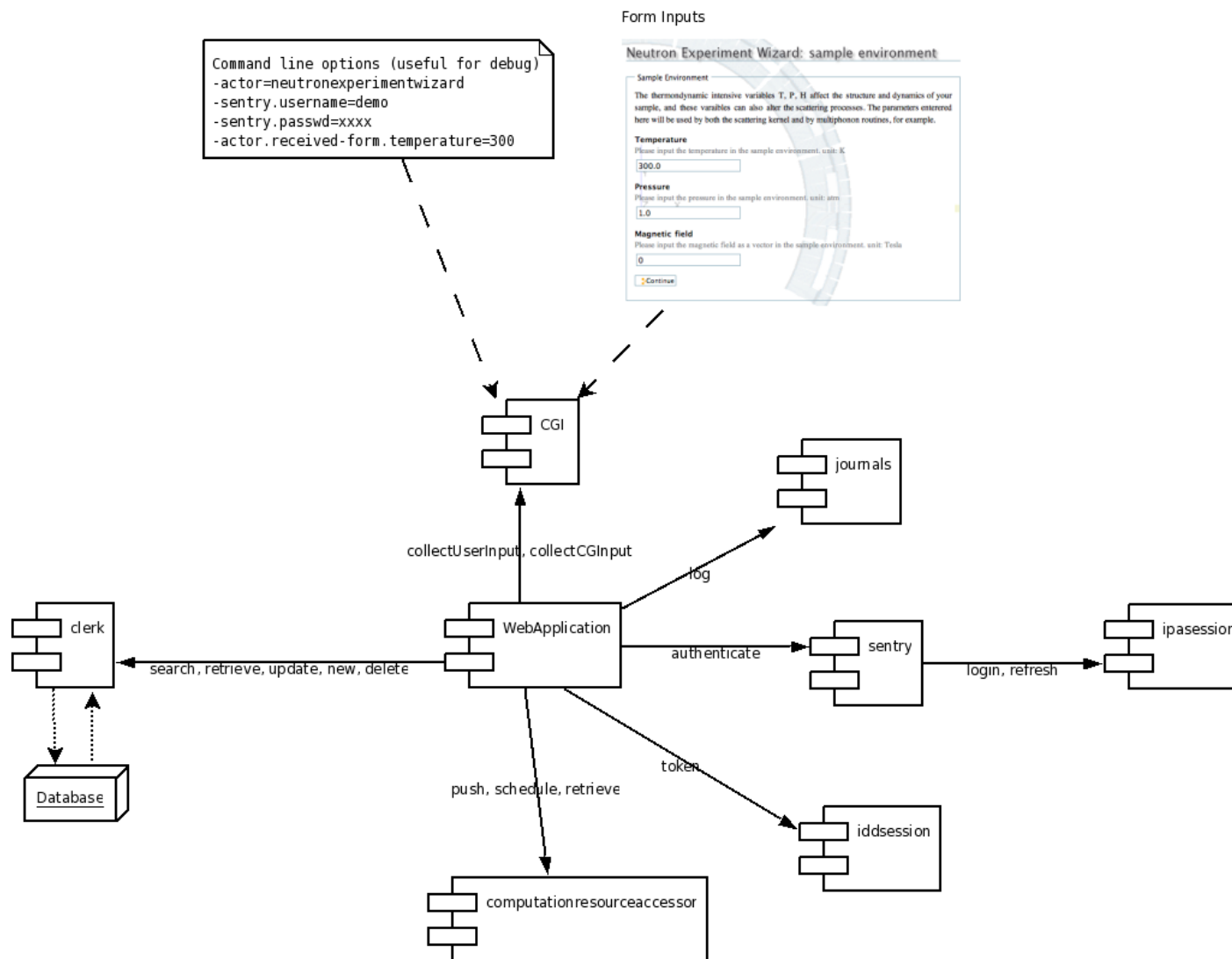
Optimization framework deployment model



User interfaces

- ✦ Identified four types:
 - ✦ *Batch scripts*
 - ✦ *Graphical user interfaces*
 - ✦ *Web-based interfaces*
 - ✦ *Services*
- ✦ Support for the development of “science portals” is the most frequently requested feature
 - ✦ *pyre applications can already be re-hosted as CGI scripts*
 - ✦ *in progress: support for near transparent re-hosting:*
 - ✦ *structured document object model for user configurable parameters, component interface invocation and result delivery*
 - ✦ *“natural” mapping between input parameter types and the widgets that collect them*
 - ✦ *complete separation of the UI specification from the underlying application behavior*
- ✦ Underway: reusing this technology to enable applications written for conventional GUI toolkits

VNF: a web hosted application



Conclusions

- ⊕ We well underway
 - ⊕ *hardware fully deployed*
 - ⊕ *support for software practices in place*
- ⊕ Adjusted plan to meet the needs of other subprojects
 - ⊕ *moved up release management and user interfaces*
 - ⊕ *postponed some implementation details of distributed computing until year three*
- ⊕ Challenges
 - ⊕ *learning curve*
 - ⊕ *maintaining project coherence*