A Web Services Based Architecture for Biomedical Applications

Sriram Krishnan
sriram@sdsc.edu
Goals

• Enabling integration across multi-scale biomedical applications
• Leveraging geographically distributed, disparate computational and data resources
Modeling and Analysis Across Scales

NBCR Tools Integrate Data, Construct Models and Perform Analysis across Scales

- Organisms
- Organ
- Tissue
- Cell
- Subcellular
- Macromolecular
- Molecule
- Atom

Modeling the Heart
- ventricles
- multicellular
- lattice
- crossbridge
- filament

Modeling Synaptic Activity
Computational Infrastructure for Multiscale Modeling

Set of Biomedical Applications

- QMView
- GAMESS
- APBS
- Autodock
- Continuity
- Gtomo2
- TxBR

Infrastructure

Computational Grid

Rich Clients

- APBSCommand
- PMV
- ADT
- Vision
- Continuity

Web Portals

- Telescience Portal

Web Services

- Workflow
- Middleware
Requirements

• Making biomedical applications *Grid-aware*
  – Remote execution on Grid resources
    • Use of Grid-based schedulers
  – Support for multiple concurrent users
  – Access via disparate user interfaces
  – Use of standards-based security mechanisms
• Integration across multi-scale applications via the use of *Workflow* tools
Towards a Services Oriented Architecture

• Applications are wrapped as services
  – Provide transparent execution on Grid resources
  – Users are free to use clients of their choice
  – Multiple standards-based security alternatives to choose from

• Services exchange strongly typed data defined using XML schemas
  – Aids in the creation of complex workflows
Talk Outline

• Motivation for a Services Oriented Architecture
• Overall end-to-end architecture
• Technical Details and Challenges
• Sample User Interfaces
• Status and Evaluation
• Conclusions
Technical Details and Challenges

- Application Services
  - Operations and Data Typing
- State Management
- Scheduling
- Security
Application Services

• APBS, GAMESS, QMView, LigPrep
  – Functionalities provided by the applications modeled as WSDL operations
  – Requests and responses for operations are strongly typed
    • Use of XML Schemas to define data structures passed around
  – Implementation details
    • Services wrap scientific codes - no (or minimal) modification required to these codes
    • Software tools used - Apache Axis, Jakarta Tomcat
Workflows and Strong Data Typing

Ligand-Protein Interaction

- Baldridge, Greenberg, Amoreira, Kondric
- GAMESS Service
  - More accurate Ligand Information via GAMESS-XML
  - Generation of Conformational Spaces
  - Assignment of parameters for APBS
- PDB2PQR Service
  - Protein preparation
- APBS Service
  - Generation of electrostatic information
- QMView Service or VMD Service
  - Visualization of electrostatic potential file
- Applications:
  - Electrostatics and docking
  - High-throughput processing of ligand-protein interaction studies
  - Use of small molecules (ligands) to turn on or off a protein function

National Biomedical Computation Resource
an NIH supported resource center

San Diego Supercomputer Center
Service Operations

- Operations can be invoked synchronously, or asynchronously
- Synchronous Operations:
  - Block until the operation is finished
  - Outputs returned as a response to initial request
  - Suitable for short jobs
- Asynchronous operations:
  - Return immediately with a jobID
  - Can query for job status and outputs using the jobID
  - Suitable for long running jobs
State Management

• Application services are stateful
  – Metadata about job inputs and outputs
  – Job status for asynchronous jobs
  – Job history

• Use of a database for storing/retrieving service state
  – Access to PostgreSQL database via JDBC

• Future Work:
  – Web Service Resource Framework (WSRF) integration
Scheduling

1. Job request using RSL via Globus CoG Kit

2. Globus Gatekeeper

3. Scheduler-specific Globus Job-manager

4. Scheduler independent job handle

Application Service

Compute Resources
Security

• GSI-based transport level (SSL) authentication
  – Use of Java CoG libraries and Tomcat to provide a secure socket connection

• Simple grid-map based authorization provided as an Axis Handler
  – Every Axis request passes through a chain of handlers before the target service is invoked
  – The grid-map Authorization Handler verifies if the client is authorized to access the service by looking up the grid-map using the Client’s Distinguished Name (DN).

• Future Work:
  – Message Level Security
  – SAML-based authorization techniques
Certificate Management

GAMA: Grid Account Management Architecture
User Interfaces

• Web services are language and platform independent
  – Can be accessed via a multitude of clients
• Java
  – Gridsphere-based Web portals
  – Workflow tools: Kepler, Informnet
• Python
  – Python Molecular Viewer (PMV)
  – Workflow tools: Vision
• Other
  – Gemstone: Mozilla-based Web services front-end
PMV APBS Client: Michel Sanner, et al

- **Calculation Parameters**
  - Calculation type: Electrostatic potential
  - Poisson-Boltzmann equation type: Linearized
  - Boundary conditions: Single Debye-Huckel
  - Charge discretization: Trilinear hat-function
  - Surface-based coefficients: No smoothing

- **Molecule Data**
  - Select Molecule: None specified

- **File I/O**
  - Energy output: None
  - Project folder: /Users/sriramkrishnan/Projects/pmw/apbs-project
  - Output files...

- **Invoke remote Web service**

- **San Diego Supercomputer Center**
  - National Biomedical Computation Resource: an NIH supported resource center
Initial Evaluation

- SOAP/HTTP not the most ideal technology to transfer large inputs and outputs
  - XML representation of molecule data (in PQR format) approximately an order of magnitude larger
  - Larger transfer times
- Axis de-serialization very expensive for large inputs
  - Large memory footprint
  - Very time consuming
Status and Software Availability

• Application services: http://nbcr.net/services
  – Alpha version of APBS service available for download and testing
  – GAMESS, QMView, LigPrep services available soon
• Gemstone: http://grid-devel.sdsc.edu/gemstone
• GAMA: http://grid-devel.sdsc.edu/gama
  – Version 1.0 available for download
• Informnet: http://grid-devel.sdsc.edu/informnet
• PMV: http://www.scripps.edu/~sanner/python
Summary

• An end-to-end infrastructure for Grid-enabling biomedical applications that provides:
  – Remote execution on Grid resources
    • Access to schedulers
  – State management
  – Concurrent access via disparate interfaces
  – Standards-based security
• Ability to use workflow tools for coupling multi-scale biomedical applications
(Incomplete) Acknowledgements

- Karan Bhatia
- Phil Papadopoulos
- Brent Stearn
- Steve Mock
- Kurt Mueller
- Sandeep Chandra
- Nadya Williams
- Peter Arzberger
- Wilfred Li
- Kim Baldridge
- Jerry Greenberg
- Robert Konecny
- Michel Sanner
- Wibke Sudholt
- APBS Team
Appendix
Sample Service: APBS

- Operations provided:
  - calculateBindingEnergy
  - calculateSolvationEnergy
  - calculateElectrostaticPotential

- Operations accept and return strongly typed parameters in XML format
  - Described by an XML Schema
  - Data binding provided by stub generators in various languages
    - WSDL2Java provided by Apache Axis
    - WSDL2PY provided by Python ZSI
**APBS Input Types**

- **InputType**
  - Grid Parameters
    - No. of Grid points: \{x, y, z\}
    - Fine & Coarse Grids
    - Length: \{x, y, z\}
    - Center: \{x, y, z\}
  - Calculation Parameters
    - Equation Type
    - Boundary Conditions
    - Charge Discretization
    - Surface Coefficients
    - Molecules[ ] — Atoms[]
    - Energy Output Type
  - Physical Parameters
    - Solvent Temperature
    - System Temperature
    - Protein Dielectric
    - Solvent Dielectric
    - Ions[ ]
    - Charge
    - Concentration
    - Radius

- Field Name
  - Atom Name
  - Atom Number
  - Residue Name
  - Residue Number
  - Coordinates
  - Atomic Charge
  - Atomic Radius
  - Symmetry Unique
SOAP Performance: Alternatives

- Parsing techniques
  - Streaming
  - Pull-based
- Binary XML
  - More compact representation of data
  - More efficient data transport and parsing
  - Smaller memory footprint
- Data Format Description Language (DFDL)
  - Definition of structure of binary and character files
  - Files transferred in their native formats
    - Smaller sizes, hence faster transfer