Connected in a small world:
Rapid Integration of Heterogeneous Biology Resources

Umut Topkara, Carol X Song
Jungha Woo, Sang P Park
Rosen Center for Advanced Computing
Purdue University

Outline

- Grid-enabled Biology Resource Collection Web Portal
- Engineering of a Software and Data Collection
- Model of a Grid Portal Evolution
  - Heterogeneous data model
  - Portal growth model
- Related Software Engineering Practices
- Agile Grid Development

Purdue Team

- Purdue University, Rosen Center for Advanced Computing
- Projects include:
  - Large contribution of compute resources to TeraGrid.
  - One of the Largest Condor Pools (>4200 nodes)
  - NanoHub: Collaborative Nanotechnology Gateway
  - Purdue Environmental Observatory Data Portal

SALSA

- Biological data and software collection
- Grid Enabled Science Gateway

- Serve *RESEARCH QUALITY* software and data
- *FAST INTEGRATION* of latest published results
- *CONNECT* these resources with TeraGrid and campus grids

Research Quality Software and Data

- Proof of Concept
- Under constant development:
  - New Scientific Insight
  - Debugging
  - More functionality
- Unstable
  - Underlying science is of primary concern
  - Lack of programming expertise
  - Pressing deadlines
- Invaluable resources
  - Novel academic products
  - Latest technology
  - Publishing quality

Fast Integration of Latest Resources

- Median publication delay 10 months*
- Life Sciences and Interdisciplinary Sciences are most selective about publication age**
- More citations to online articles
  - *336%* of offline articles *published in the same venue****
- Increasingly important
  - Additional 10M PubMed searches every year*****

Connect with Grid

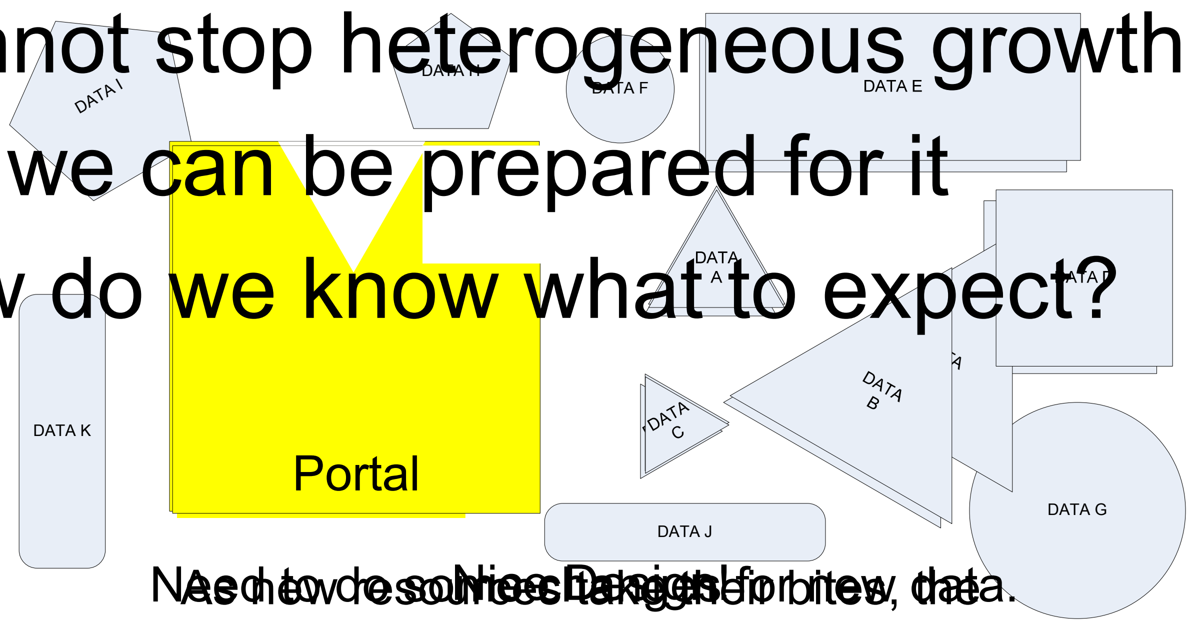
- Science Gateway:
  - Bring supercomputing to scientists
  - Utilize community resources
- Web Portal
  - Easy to access
  - Easy to use
- Collaboration friendly
  - More visibility
  - Easy to “give it a shot”

Engineering of a Biology Gateway to Grid

- Access Public resources
  - Independently developed software
  - Large amounts of data
  - Mixed quality data
  - Lack of design coordination (database schema, user interface, or API)
- Heterogeneous nature
- More of them coming constantly!

What Happens in a Growing Heterogeneous Portal

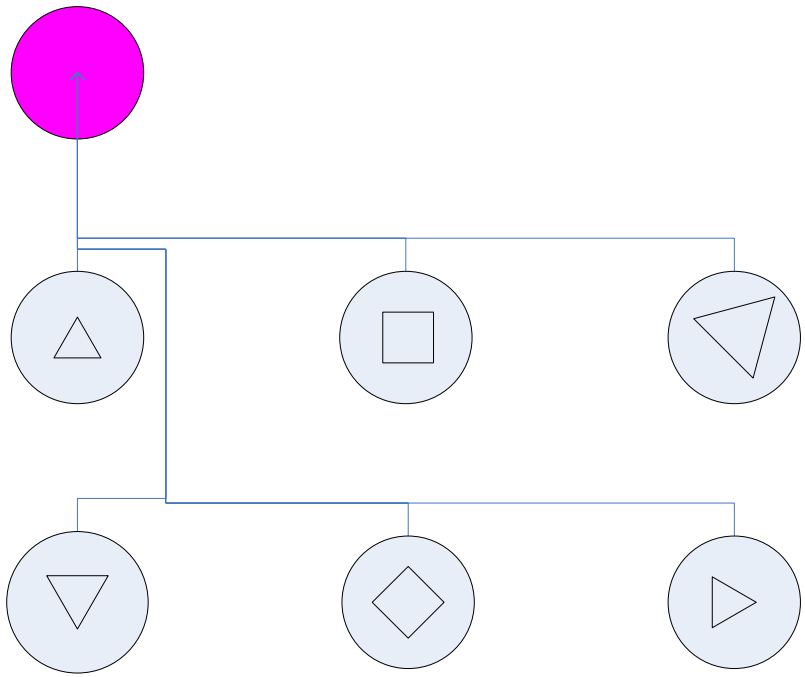
- Portal growth can hurt the design badly
- Patches eventually inhibit further additions
- Cannot stop heterogeneous growth
- But we can be prepared for it
- How do we know what to expect?



Need to do something for new data.
As new resources take their bites, the
Initial portal design is no more.
Nice Design!
Everything fits together.
But it's OK!

Model Driven Development

- Blueprint is developed first
- Computer Aided Software Engineering (CASE) tools generate end product
- Small changes can be tolerated as long as they match with the blueprint
- O(N) wrappers are enough to integrate the whole system of N components



Model Driven Development

- Canonical Models are solidified at design time
- Need to know all requirements at design time
  - In depth knowledge of the field
  - Can you predict future trends
- … otherwise, as new components are added
  - Canonical model needs to be revised
  - Becomes more and more complex
  - Eventually, inhibit growth of the collection

Design to Grow

- Do not fix “what” the system will become
- Determine “how” the system will evolve
  - Heterogeneous data model
  - Portal Growth Model
- Agile Development
  - Continuous maintenance
  - Make decisions as needed, not before

Heterogeneous Data Model

- Resources are product of uncoordinated effort of large number of researchers
- Individual design decisions by researchers that produce data and software
- Come in a multiplicity of
  - Data types
  - Formats
  - Applications
  - Implementations
- Studied by database research

Heterogeneous Data Model

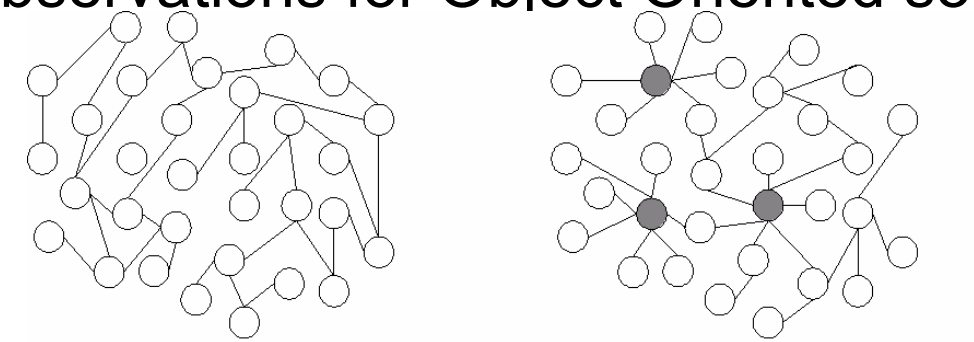
- Locally controlled updates: syntactic/ semantic changes without prior notice
- Sporadic down times: computer failures and scheduled maintenances
- Size Heterogeneity: Vary in size and quality
- Structural Heterogeneity: from relational databases to unstructured text
- Query Heterogeneity: Different limitations on the types of allowed queries
- Need to capture all of the above in a single model if Model Driven Development is to be used

Portal Growth Model

- Canonical models help us integrate N components with O(N) wrappers
- Otherwise we may need O(N^2) mediators to integrate the portal, or so?
- Scale-free networks
  - Highly connected “hubs”
  - Most components have small degree

Are Portals Scale Free

- Preferential attachment
  - Some nodes more popular (SwissProt, BLAST,etc)
- P(k): probability that a random node will connect to k other nodes
- $P(k)= k^{-\alpha}$, α : characteristic constant
- Debian packages [LaBelle]:
  - $\alpha_{in}= 0.9$ $\alpha_{out}=2.33$
  - Highly (in) referenced packages have large hit count
  - Average < 4 (out) dependencies per package
- Similar observations for Object Oriented software



(a) Random network (b) Scale-free network

Agile Portal Development

- Sustainable portal growth
  - Changing requirements for existing components
  - Addition of new components
- Agile Development
  - Modifying existing features are more costly than writing them from scratch
- Continuous process of
  - Specification : Usage scenarios
  - Design: Least specific
  - Development:
    - Improvement using *refactoring*
    - Followed by addition of new features

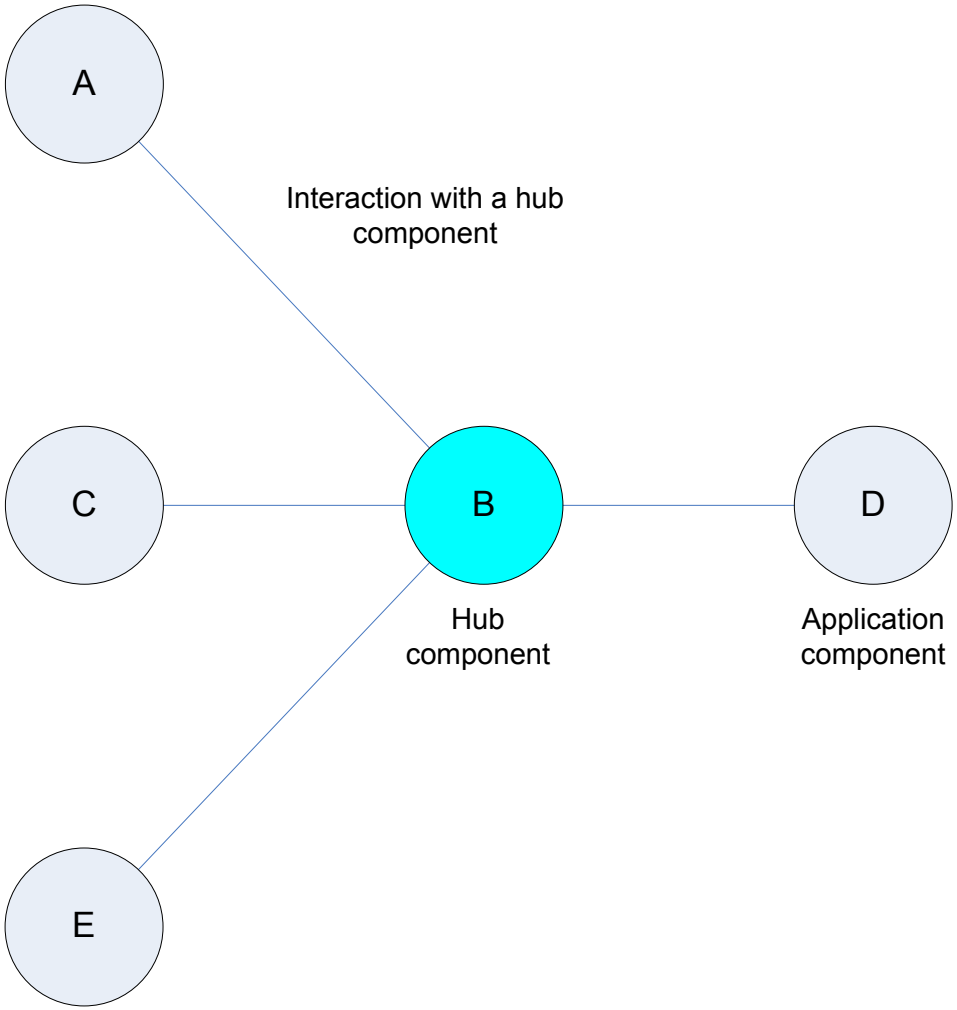
SALSA Architecture

- Aid the continuous maintenance effort
  - Capture the evolution of the portal
  - Predict future changes to the system
- Redirect development focus
  - **from** many components specific to one application
  - **to** few components that are common to all applications
- Hub components: Storage, execute, job history, FASTA databases, BLAST application

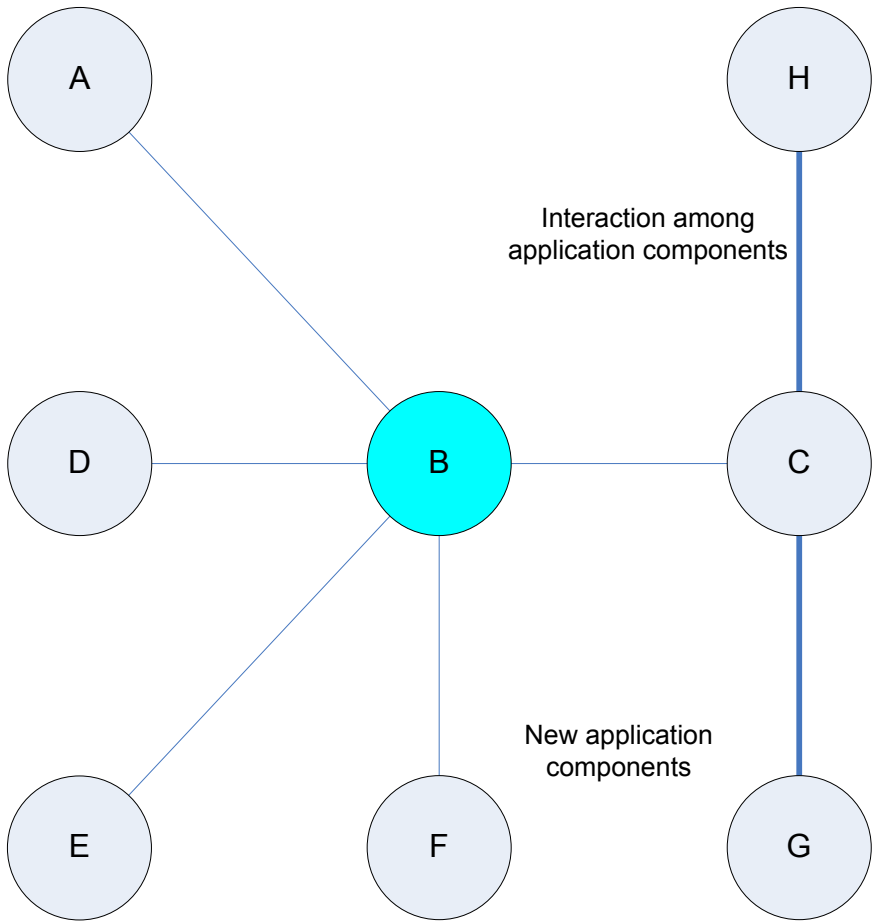
Importance of Hub Components

- Hub components are critical for the overall system
  - Stability: participate in most workflows
  - Maintainability: most applications need them to operate correctly
- Hubs emerge in the process of portal growth
  - Initially application specific components
  - Promoted to generic components through *refactoring*

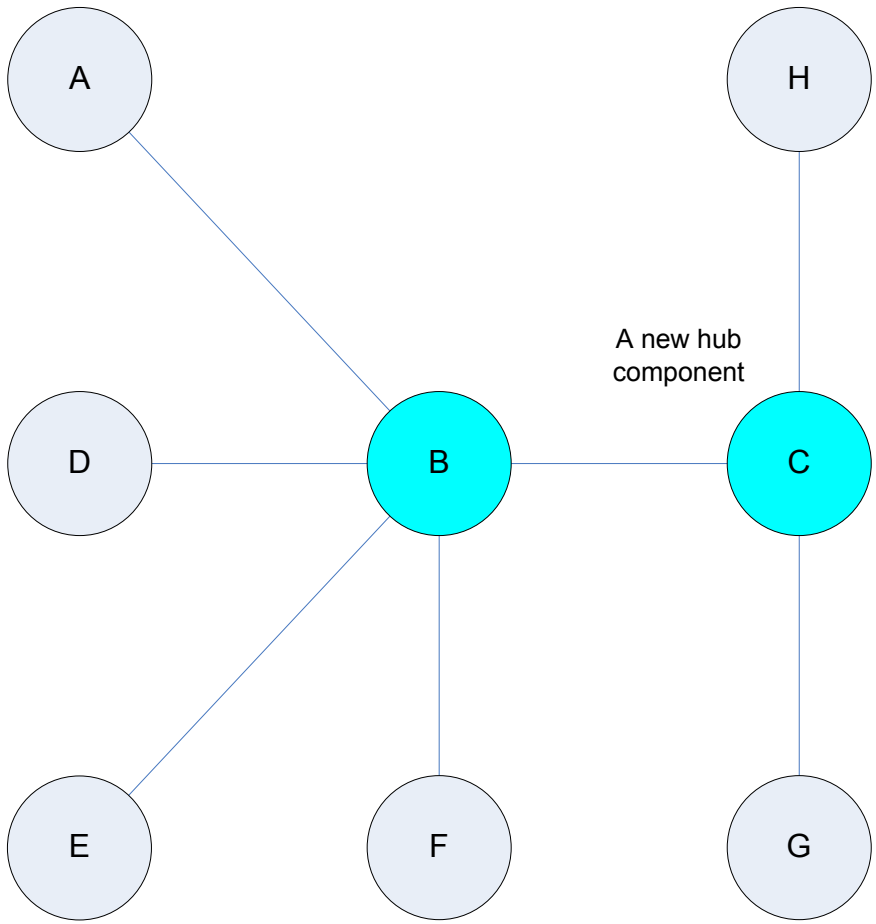
Maintenance of Portal 1



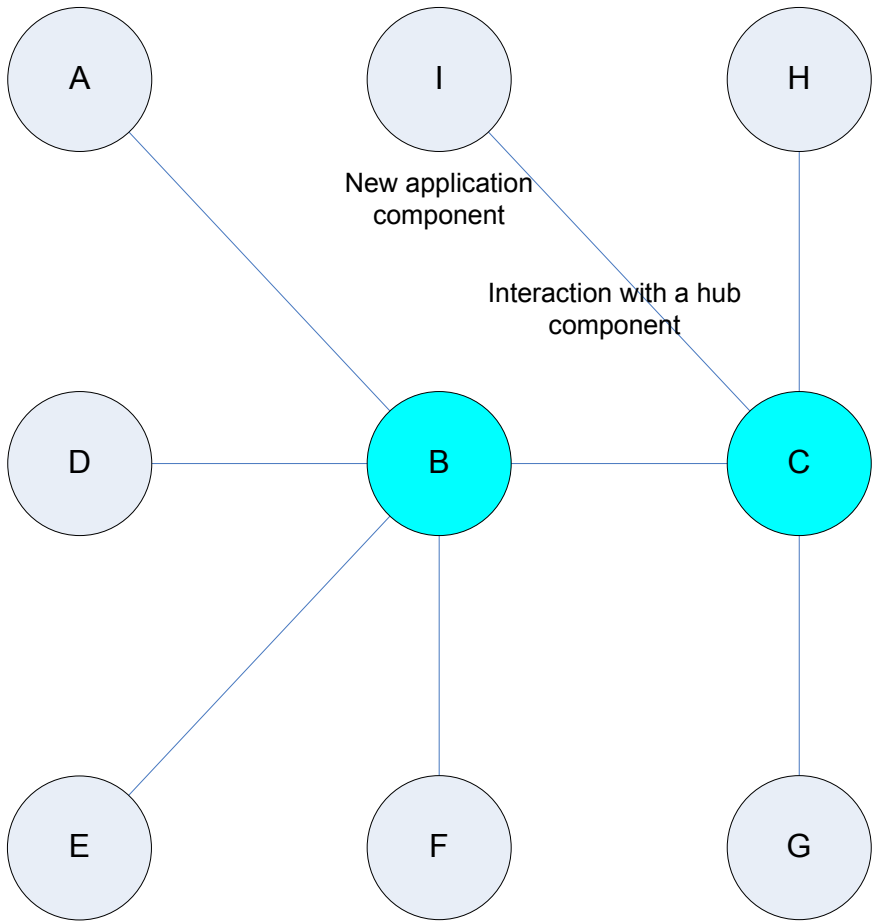
Maintenance of Portal 2



Maintenance of Portal 3



Maintenance of Portal 4



Hub Development

- A popular component becomes a hub
- Refactoring a component:
  - Code and design improvement
  - Documentation improvement
  - Performance improvement
  - Integration of similar components (BLAST → pattern matching)
- Since hubs will gain more dependencies
  - This will ease overall maintenance load

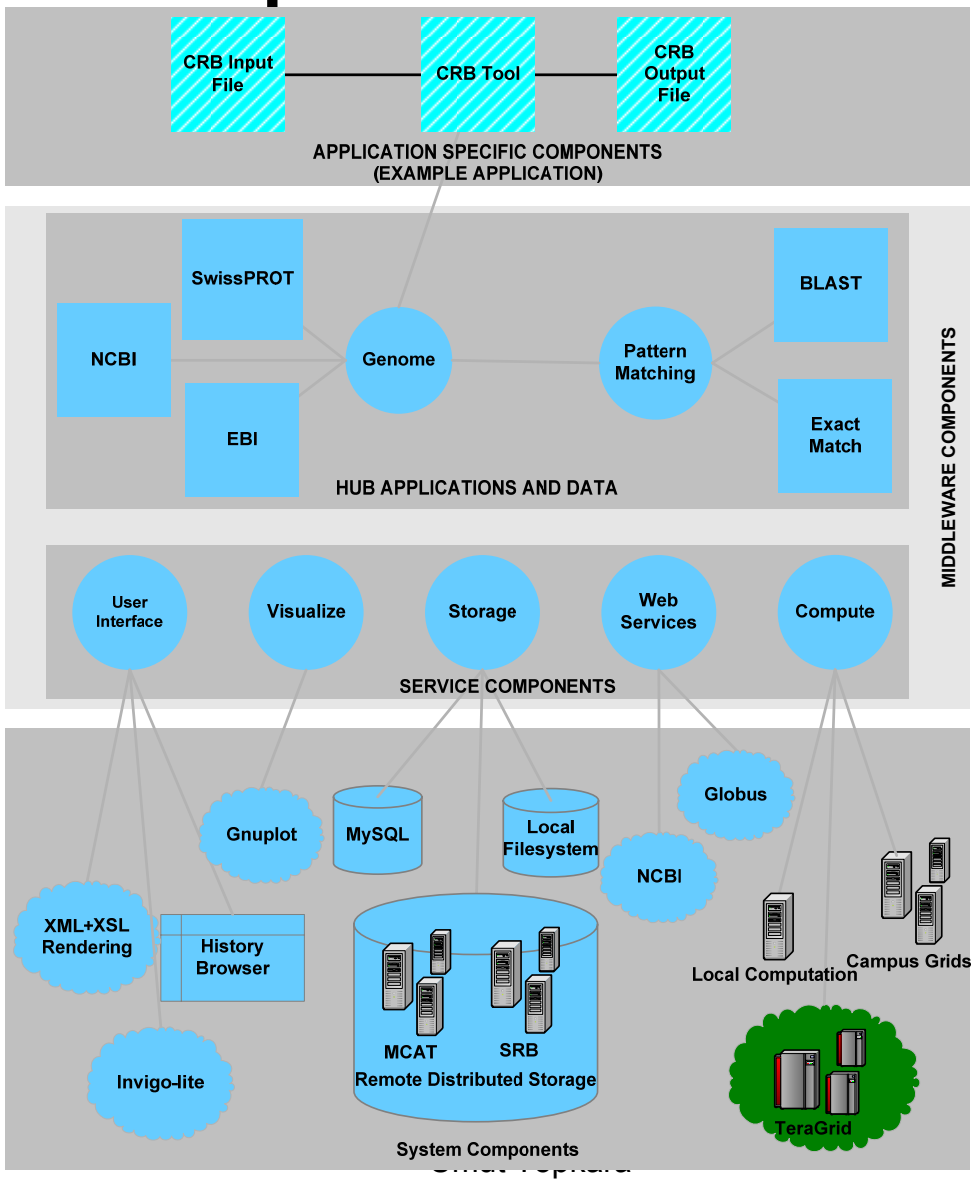
Hub Models

- Design of hubs evolve over time
- Description of hubs are specific to their usage in the portal
  - Every application can be defined independently from the rest of the portal
  - Useful when the portal grows to be very large
- Can improve error-tolerance of the overall system
  - Stability of the system depends on stability of hubs
  - Alternative implementations allow redundancy (local storage, remote storage)

Implementation

- Web based collection
- Aim to grow:
  - Large collection
  - Up to date research data and software
- Small number of service middleware components:
  - User interface : XML, XSL, JavaScript
  - Storage : SRB, MySQL
  - Compute : Condor
  - Web services : GridSphere
  - Plot Visualization : GnuPlot
- Application based hubs:
  - Pattern matching: BLAST, exact match
  - Sequence data: User FASTA, SwissProt
- Redundancy
  - Data caching
  - Alternative implementations (flat file, mySQL, SRB)
- Virtual Machines: VMware appliances
  - Easier Development
  - Packaging

Implementation



Multi-Purpose XML

- Human readable documentation
  - Annotations by developers and users
- Machine readable book-keeping
  - Input/output types
  - Input values
  - User credentials
  - CPU requirements
  - Remote data locations
- XML as Component Descriptors
  - Component specification
  - Couple with user interface through XSL and Javascript

XML lifetime

<opt>	<opt>	<opt>
<input>	<input>	<input>
<integer>	<integer>	<integer>
<attrib>	<attrib>	<attrib>
<id>ncat</id>	<id>ncat</id>	<id>ncat</id>
<default>71</default>	<default>71</default>	<default>71</default>
<max>1000</max>	<max>1000</max>	<max>1000</max>
<min>0</min>	<min>0</min>	<min>0</min>
	<reason>	
	Value greater than maximum limit (1000).	
	</reason>	
	<uservalue>2000</uservalue>	<uservalue>1000</uservalue>
	<validity>0</validity>	<validity>1</validity>
</attrib>	</attrib>	</attrib>
<description>	<description>	<description>
Number of words in catalogue file.	Number of words in catalogue file.	Number of words in catalogue file.
</description>	</description>	</description>
<label>Number of words in catalogue</label>	<label>Number of words in catalogue</label>	<label>Number of words in catalogue</label>
</integer>	</integer>	</integer>
.....	.....	.....
<jobinfo>	<jobinfo>	<jobinfo>
	<allowed>0</allowed>	<allowed>1</allowed>
	<createdate>01/01/2006-12:00</createdate>	<createdate>01/01/2006-12:00</createdate>
<des>TestJob</des>	<des>TestJob</des>	<des>TestJob</des>

XML Rendering

CRB Toolkit

Number of words in catalogue: 71

Sequence length: 35937

Catalogue file: Browse...

Data file: Browse...

Additional parameters: string input

Submit Job

CRB Toolkit

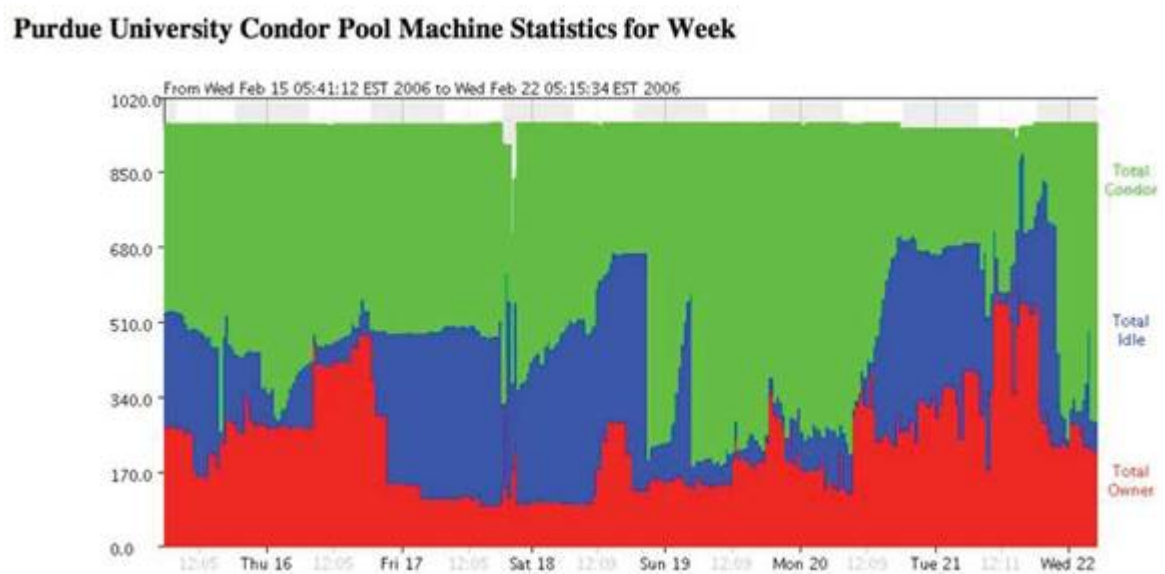
Job Description	TestJob
Date/Time	01/01/2006-12:00
Status	WAITING
User Input Validity	**There was an error**

Tried Input Data			
Number of words in catalogue	2000	Invalid	Value greater than maximum limit (1000).
Sequence length	4000a	Invalid	Value not integer.
Catalogue file	mycat.txt	Invalid	File is not uploaded.
Data file	mydat.txt	Invalid	File is not uploaded.
Additional parameters	NA	Invalid	Unknown parameters

The output files	
Posterior Probability of a Regulatory Region	pycurve.gif
Regulatory region prediction output	pyout.txt
Word predictability output	pyout.txt
Log output	diagout.txt

Compute Component

- Utilize Purdue Campus Condor Pool
- > 4200 nodes
- Utilize TeraGrid using Condor-G interface
- Example application: CRB
  - Bayesian model for locating regulatory regions
  - Process segments of sequence simultaneously



Related Work

- Software engineering literature on
  - Agile development
  - Commercial off the shelf (COTS) based development
- True cost of using COTS is *maintenance*

Conclusion

- Emphasis on rapid integration
- Focus on highly connected components
- Expect to continuously update *design*
- Ability to capture trends looking at connectivity of nodes
- Redundancy at hub components increase reliability and error tolerance

Thank you

- Anonymous Referees
- National Science Foundation (TeraGrid Resource Partners grant OCI-0503992)