Using Pegasys for genome annotation

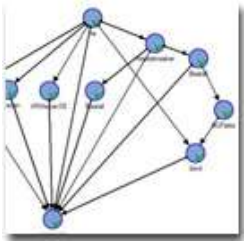
Sohrab Shah

sshah@cs.ubc.ca

Department of Computer Science, UBC

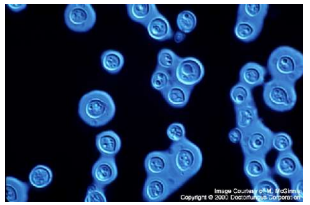
Outline

- What is Pegasys?
  - System features
  - System architecture



- Setting up Pegasys
  - Server
  - Client

- Using Pegasys
  - Pegasys for genome annotation
  - *Cryptococcus neoformans* use case



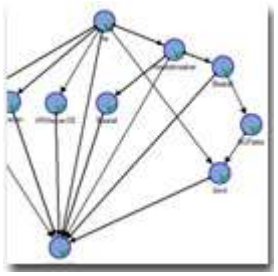
Sohrab Shah <sshah@cs.ubc.ca>

Part I: What is Pegasys?

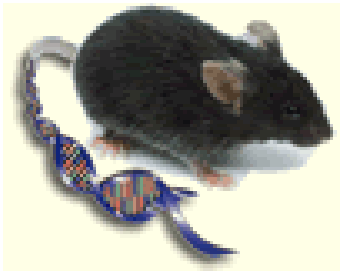
(Shah et al. BMC Bioinformatics 2004)

Sohrab Shah <sshah@cs.ubc.ca>

Motivation for the development of Pegasys

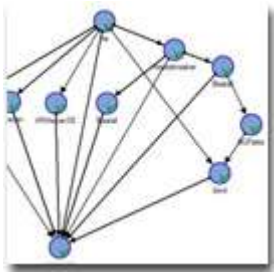


- 100's of genomes being sequenced
- Need to be annotated to gain useful scientific knowledge
  - Requires automation with human refinement
- A need for a portable, customizable, generic solution to reduce software development effort
- A need for the development of re-usable protocols for genome analysis
- A need to adapt to the changing bioinformatics landscape

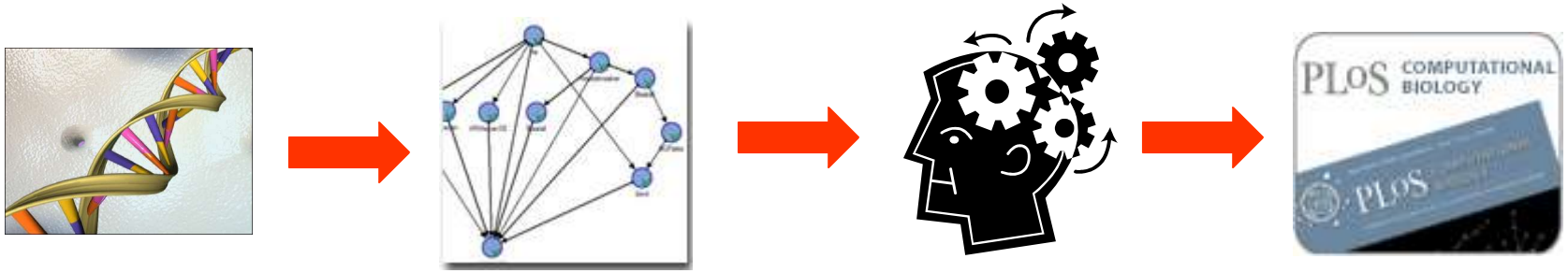


Sohrab Shah <sshah@cs.ubc.ca>

What is Pegasys?

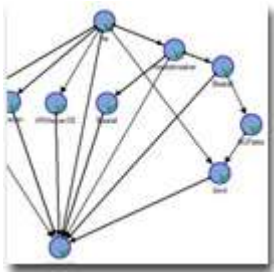


- A software platform designed to simplify and manage the process of (semi) automated genome annotation
- Pegasys:
  - provides an interface to construct sequence analysis workflows with no programming required
  - performs job scheduling and submission to a computer cluster
  - unifies heterogeneous analysis results into practical data formats
  - encodes analysis protocols
    - reusable, distributable, reproducible research

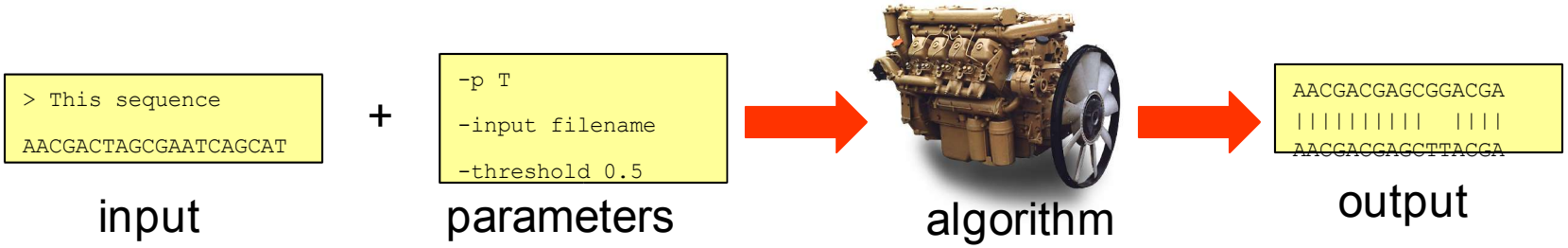


Sohrab Shah <sshah@cs.ubc.ca>

Key to Pegasys: abstraction and modularity



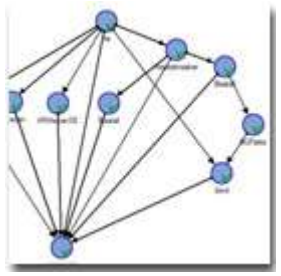
- Analysis tools for genome annotation have many things in common



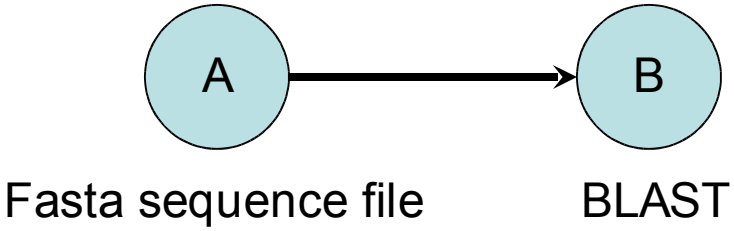
- Abstracting the concept of a sequence analysis tool allows:
  - Creating of a conceptual model for an analysis tool
  - Implementation of conceptual model into a data structure to be used in software implementation
  - Reuse of the same data structure for different tools

Sohrab Shah <sshah@cs.ubc.ca>

Workflow modeling

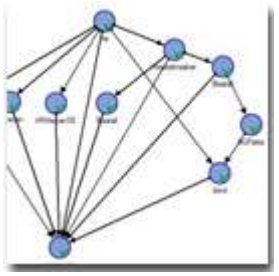


- Workflows in Pegasys modeled as a Directed Acyclic Graph (DAG)
  - Nodes = analysis programs, input/output files
  - Edges = data transfer between nodes
  - For edge A->B, output of A becomes input of B

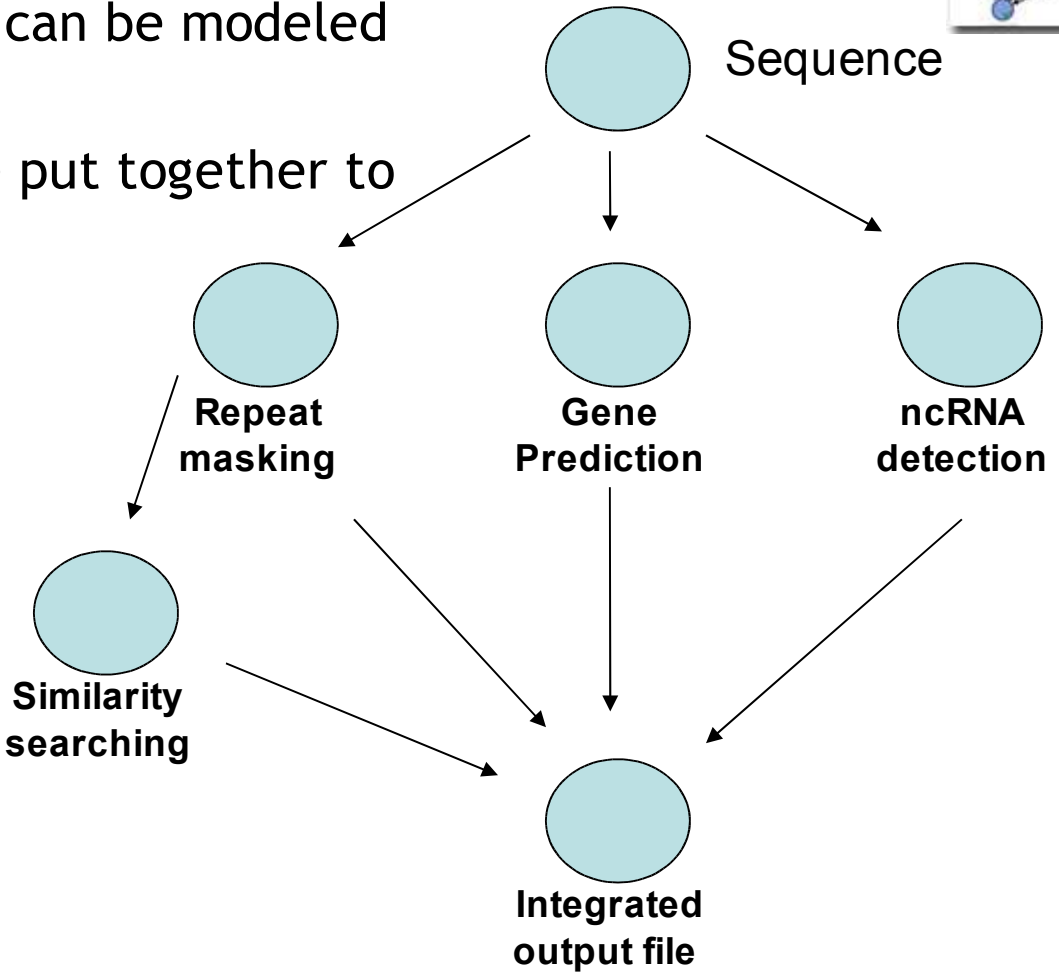


Sohrab Shah <sshah@cs.ubc.ca>

Abstraction of analysis tools

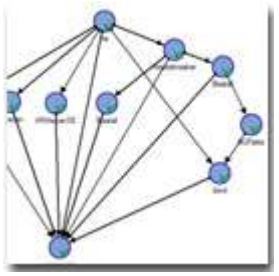


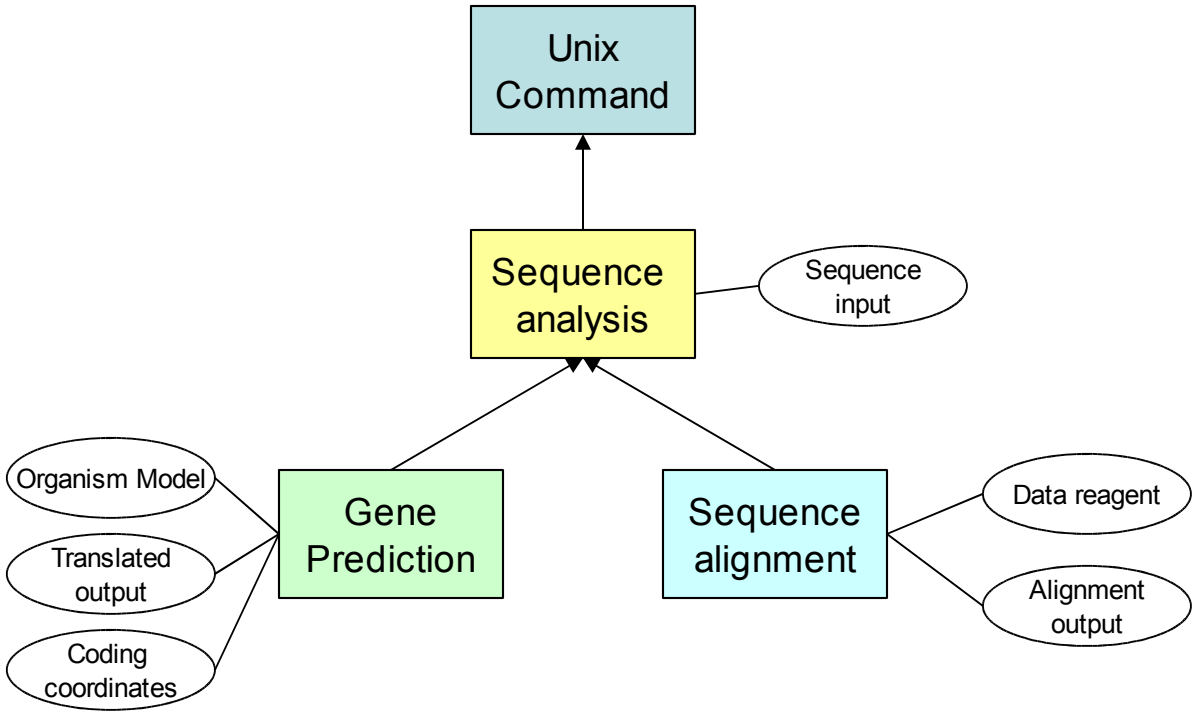
- Any analysis tool that accepts input and produces output can be modeled in this way
- Pairs of nodes can be put together to form a DAG



Sohrab Shah <sshah@cs.ubc.ca>

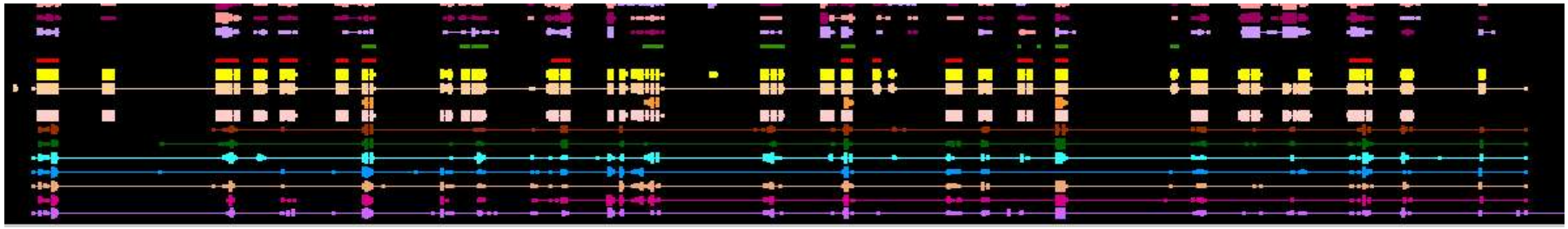
Use inheritance to be specific





Sohrab Shah <sshah@cs.ubc.ca>

Abstracting output

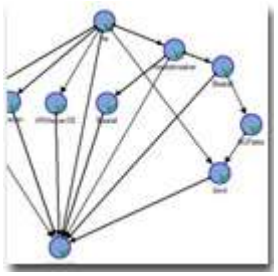


Computational feature: Start position + End position + Score

- Allows the storage and integration of heterogeneous analysis results into a common framework
- Hierarchical – can store nested results

Sohrab Shah <sshah@cs.ubc.ca>

System overview



- Wish list
  - Data integration
  - Distributed processing
  - Database backend to store all results
  - Nice interface to construct workflows
  - Workflow encoding for distributable, reproducible analysis protocols
  - Easy integration of new or changed tools
  - Highly configurable system
  - System to allow the biologist to focus on interpreting results, not on technical barriers to running analysis

Sohrab Shah <sshah@cs.ubc.ca>

Documentation

- User guide
- Administration guide
- Developer manual
- Version specific
- http://bioinformatics.ubc.ca/documentation

The Pegasys User Guide v0.6

The Pegasys development team
UBC Bioinformatics Centre
pegasys@bioinformatics.ubc.ca
http://bioinformatics.ubc.ca/pegasys

August 8, 2005

The Pegasys Administration Guide v0.6

The Pegasys development team
UBC Bioinformatics Centre
pegasys@bioinformatics.ubc.ca
http://bioinformatics.ubc.ca/pegasys

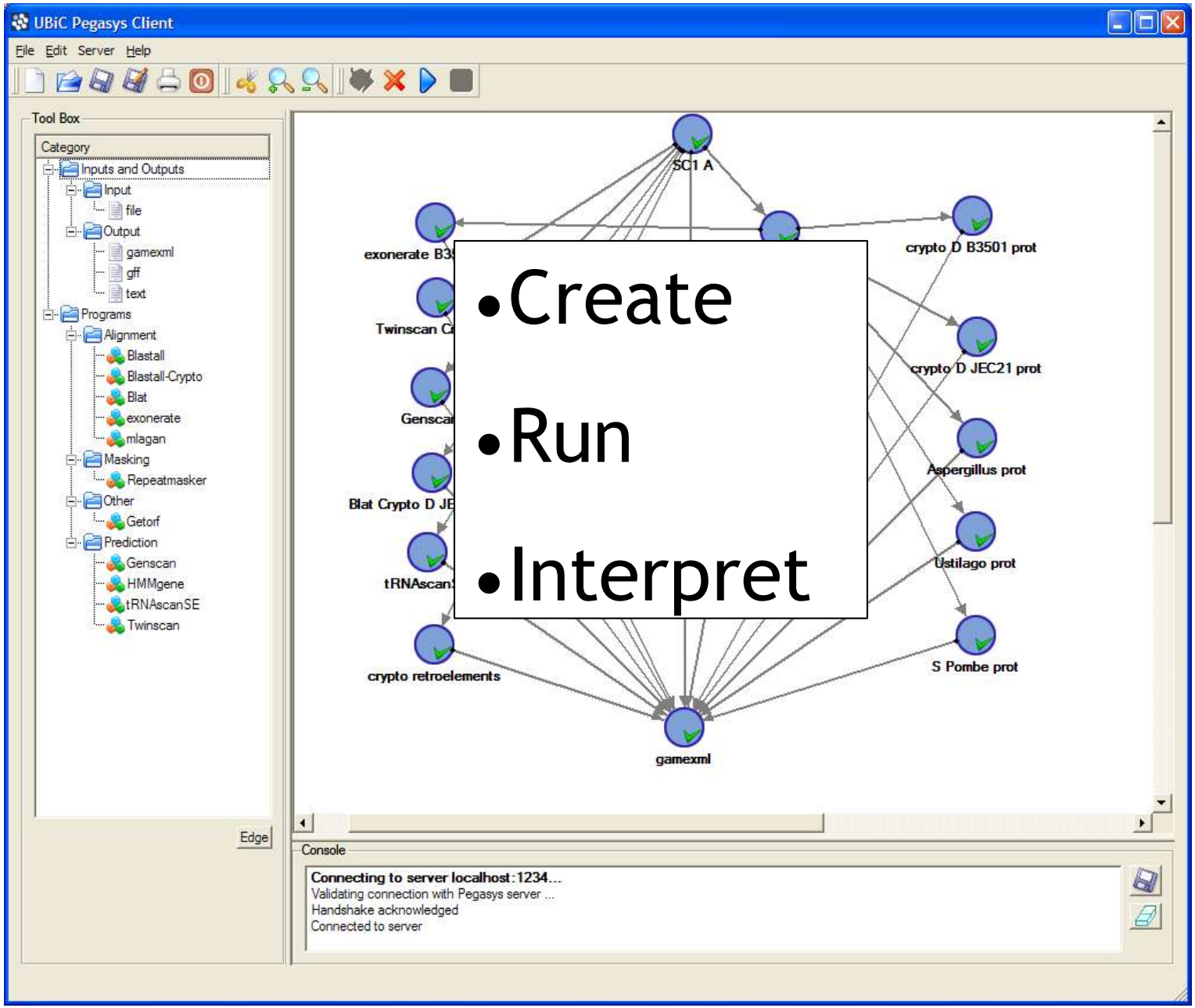
August 8, 2005

The Pegasys Developers Manual v0.6

The Pegasys development team
UBC Bioinformatics Centre
info@bioinformatics.ubc.ca
http://bioinformatics.ubc.ca/pegasys

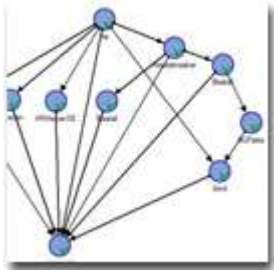
August 8, 2005

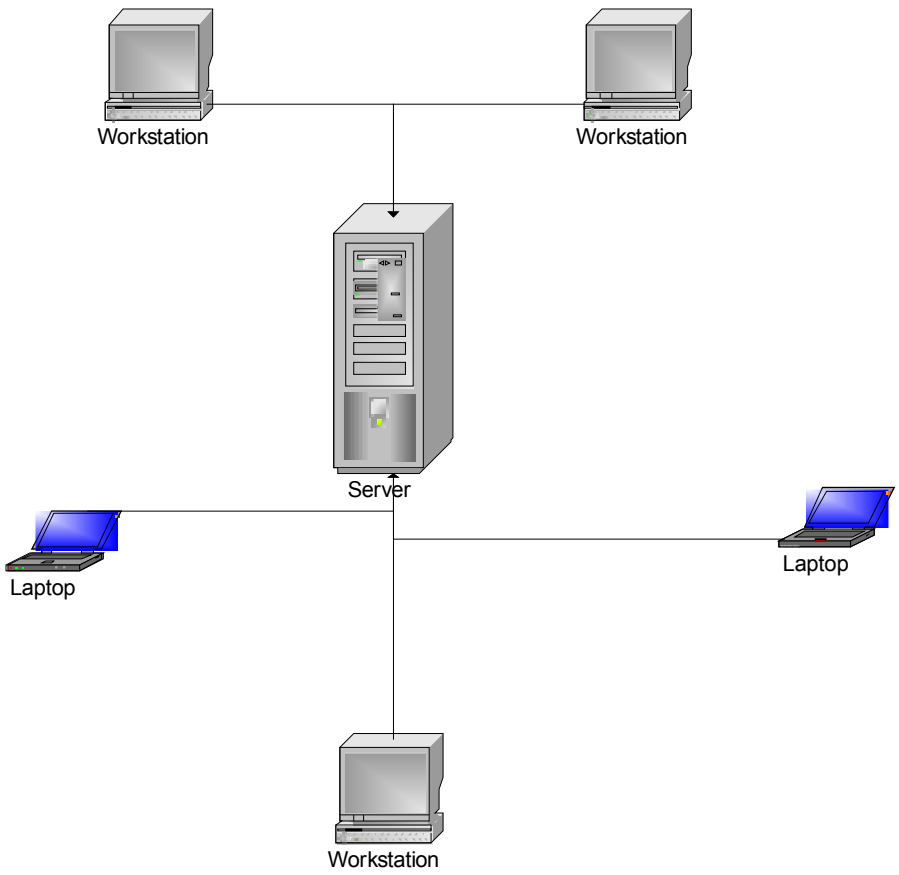
Sohrab Shah <sshah@cs.ubc.ca>



Sohrab Shah <sshah@cs.ubc.ca>

Pegasys architecture

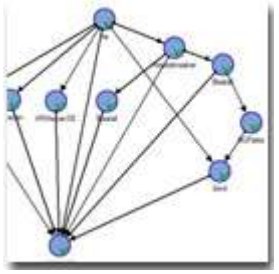


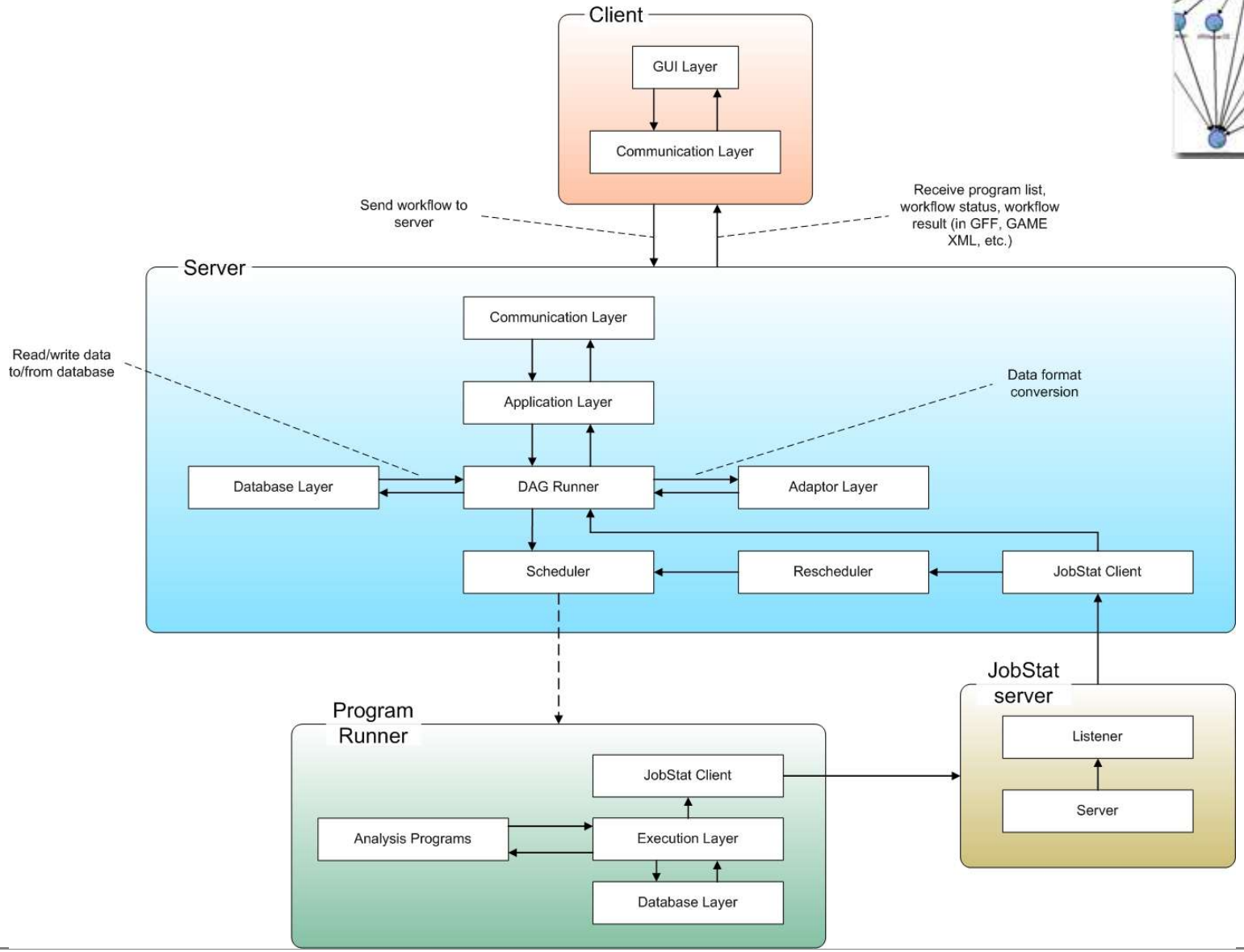


Client-Server Model

Sohrab Shah <sshah@cs.ubc.ca>

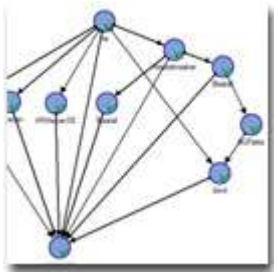
Pegasys architecture





Sohrab Shah <sshah@cs.ubc.ca>

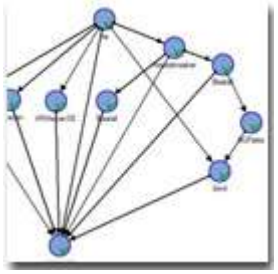
Pegasys server



- Runs as a daemon (persistent process)
- Communication layer
  - Send/receive messages from client
- Application layer
  - Handles parsing of analysis program inputs and outputs
- Scheduling layer
  - Deploy jobs in DAG on a compute cluster
- Database layer
  - Store all jobs, results
- Adaptor layer
  - Formats output

Sohrab Shah <sshah@cs.ubc.ca>

Pegasys server



- Communication layer
  - All communication done through a simple TCP/IP protocol
    - First line: integer containing how many bytes to be sent
    - Followed by a message
  - Client messages to server
    - Connection/disconnection request
    - Workflow encoded in XML
    - Halt workflow execution
  - Server messages to client
    - Connection acknowledgement
    - Node status
    - Output

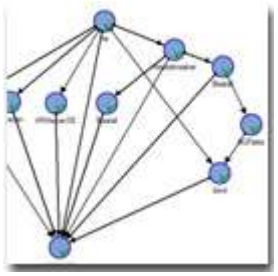
Sohrab Shah <sshah@cs.ubc.ca>

Pegasys server

- Communication layer
  - 'Simple Message Protocol'
    - Each message begins with a four byte integer that is the length of the message payload
    - message payload is read as a UTF-8 encoded string
    - messages are terminated with a four byte integer that is always 0

Sohrab Shah <sshah@cs.ubc.ca>

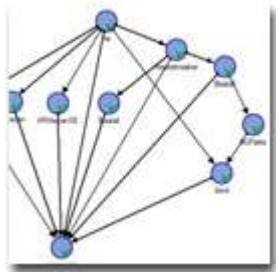
Pegasys server



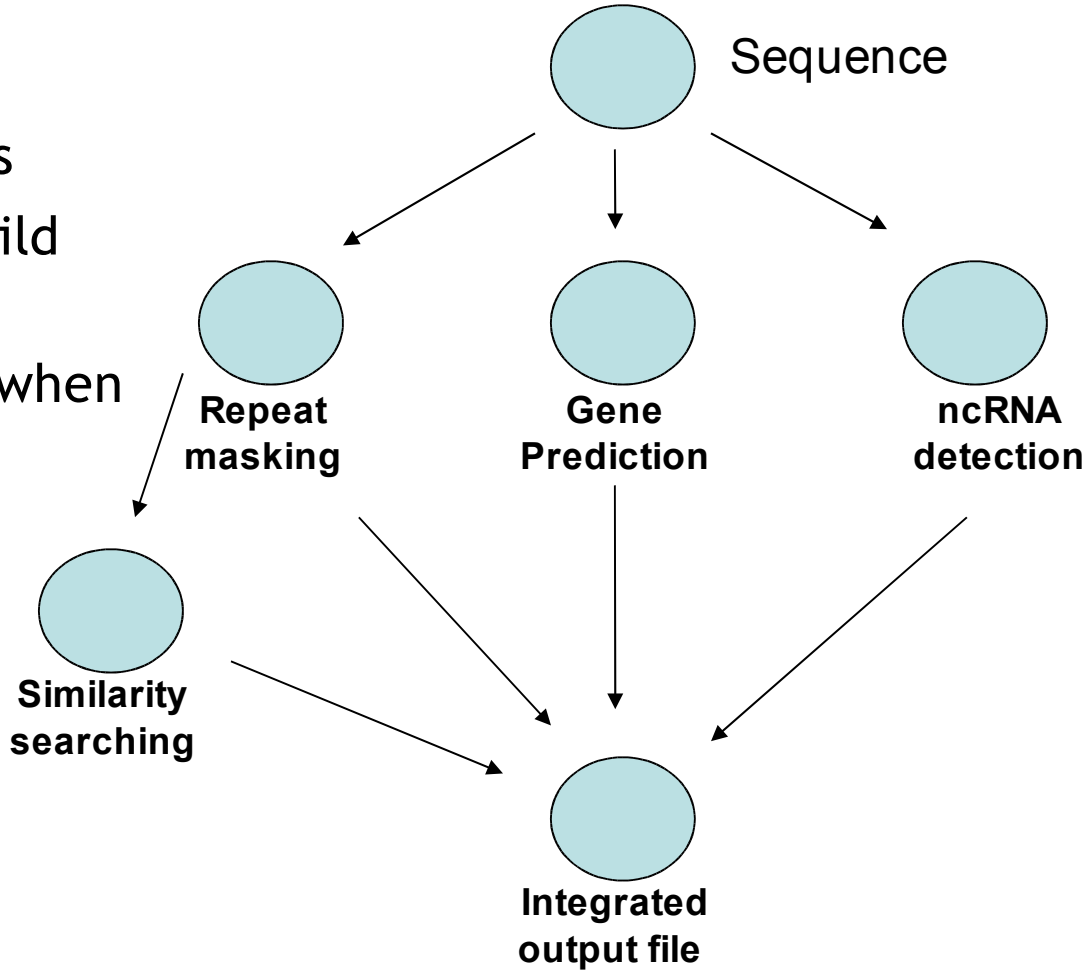
- Application layer
  - Converts workflows into DAGs
  - Creates the DAG data structure in memory and sets all appropriate variables as read from the workflow
  - Constructs appropriate command line for the particular instantiation of each program in the DAG
  - Passes the DAG to the DAGRunner
    - Uses the scheduling layer to deploy the jobs
  - 2 types of nodes:
    - Input/Output nodes
      - Input Nodes only have outgoing edges
      - Output nodes only have incoming edges
    - Program nodes
      - Have incoming and outgoing edges

Sohrab Shah <sshah@cs.ubc.ca>

Pegasys server

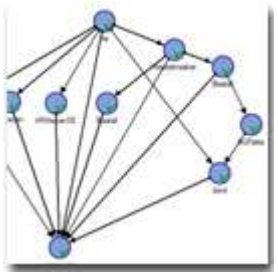


- Scheduling layer
  - Deploys jobs on compute cluster
  - Monitors status of the jobs
  - Keeps track of parent->child status
  - Notifies application layer when jobs are complete/dead



Sohrab Shah <sshah@cs.ubc.ca>

Pegasys server

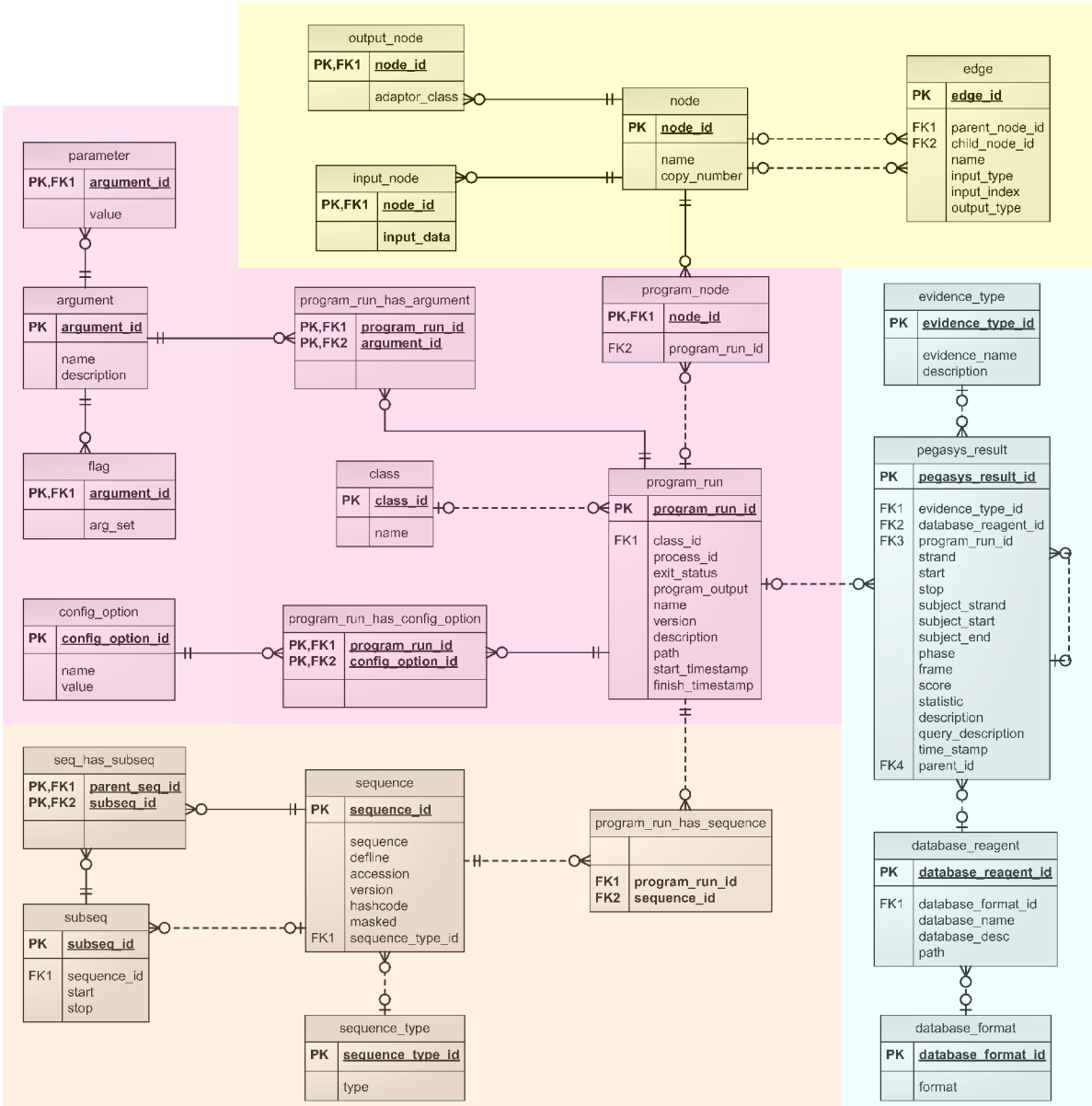


- Database layer
  - API-based layer to communicate with underlying RDBMS
  - Towards an Object-Relational model
  - Database stores workflows, jobs information, analysis results
  - Captures all data necessary to recreate data structures

Sohrab Shah <sshah@cs.ubc.ca>

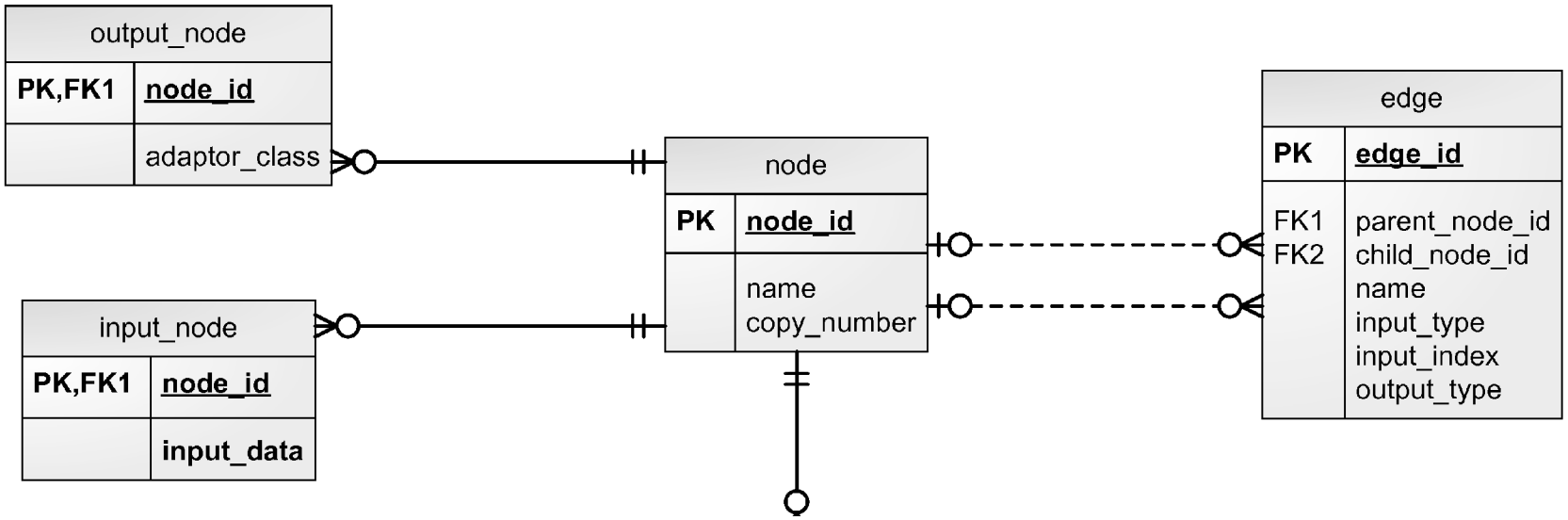
Pegasys server

- Database schema
  - Simple design:
  - DAG
  - Sequence
  - Program
  - Results



Pegasys server

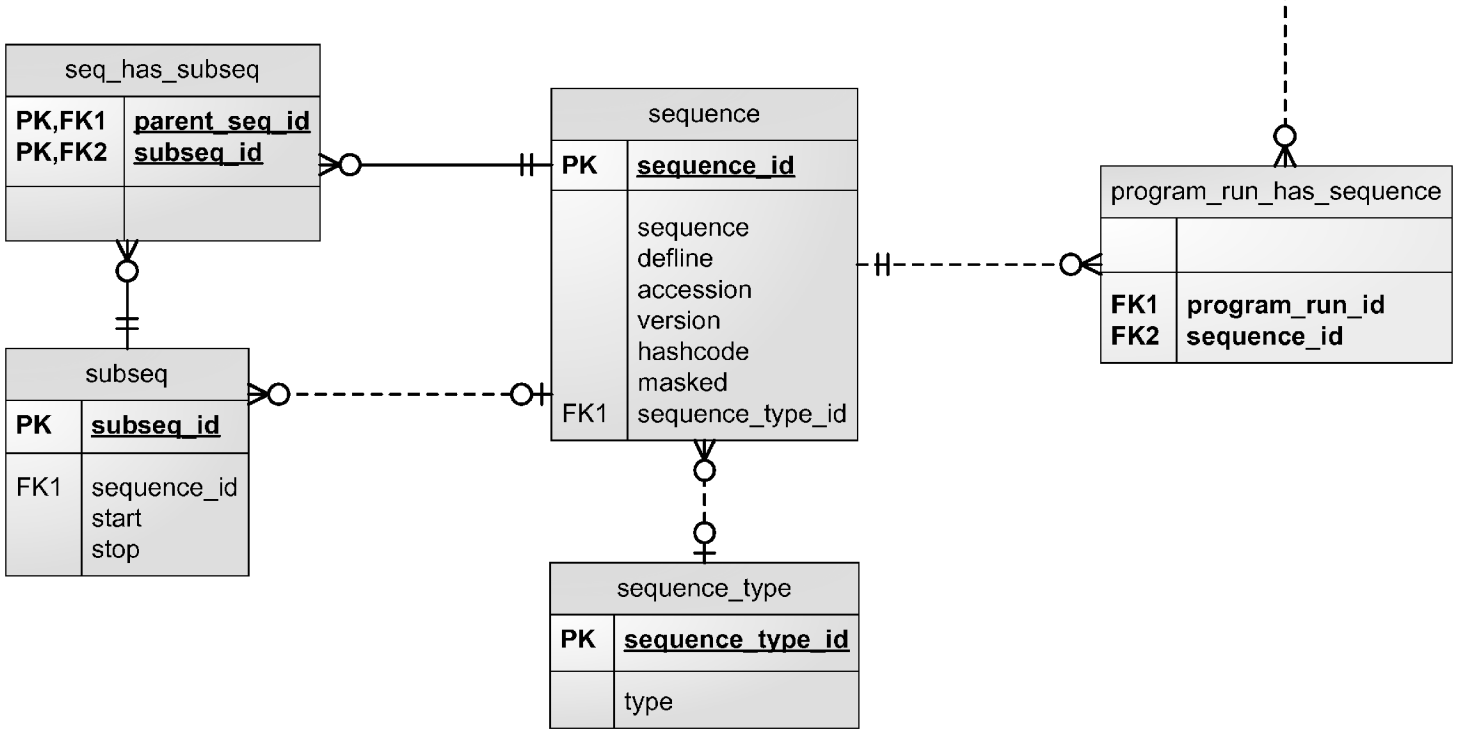
- Database schema – DAG related tables



Sohrab Shah <sshah@cs.ubc.ca>

Pegasys server

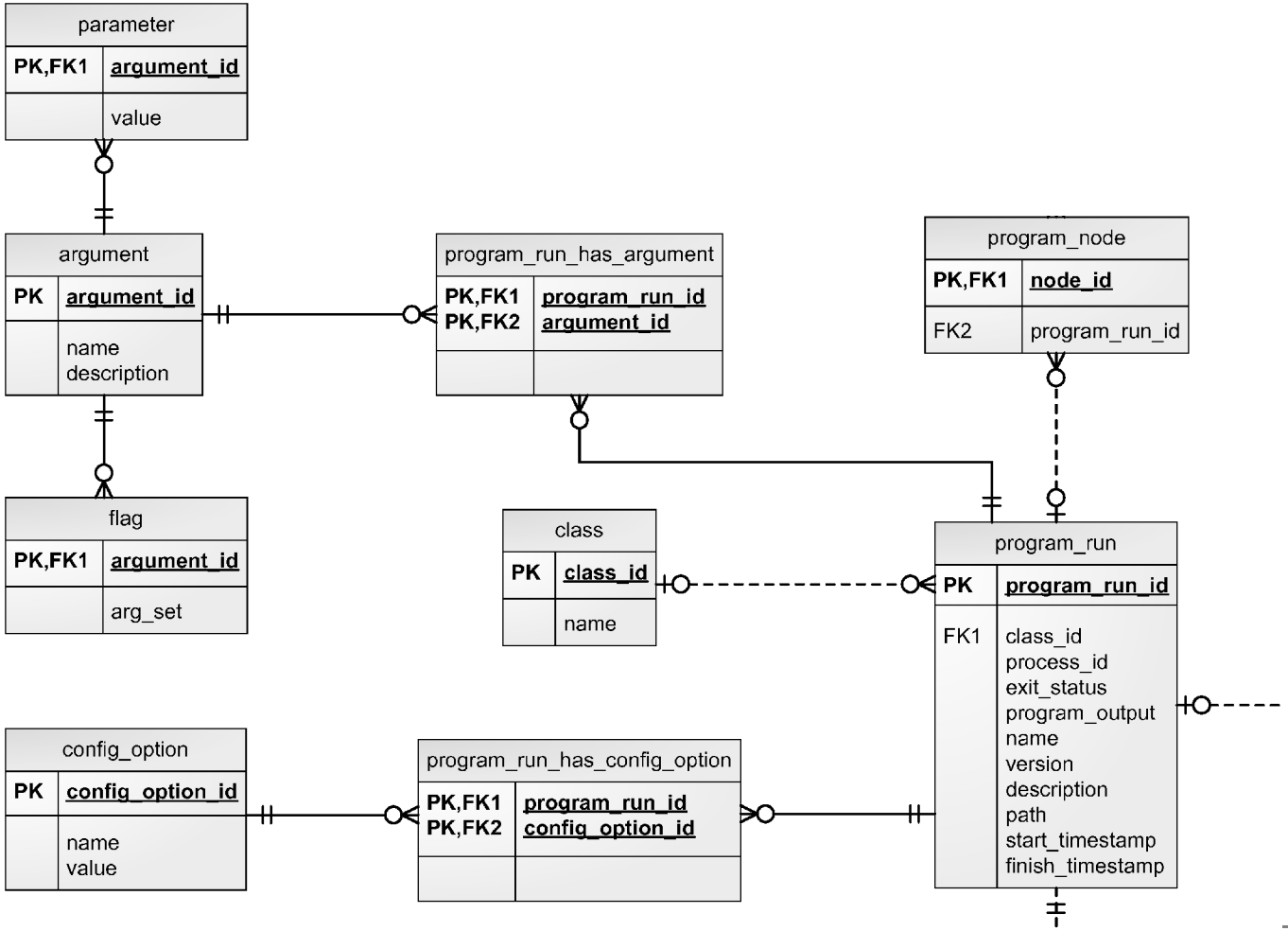
- Database schema – sequence tables



Sohrab Shah <sshah@cs.ubc.ca>

Pegasys server

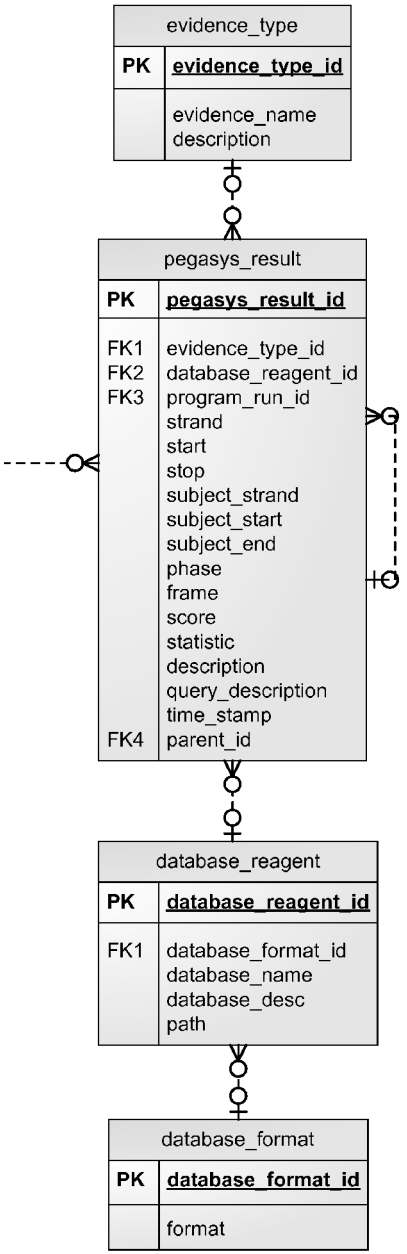
- Database schema – job related tables



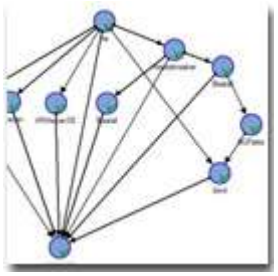
Sohrab Shah <sshah@cs.ubc.ca>

Pegasys server

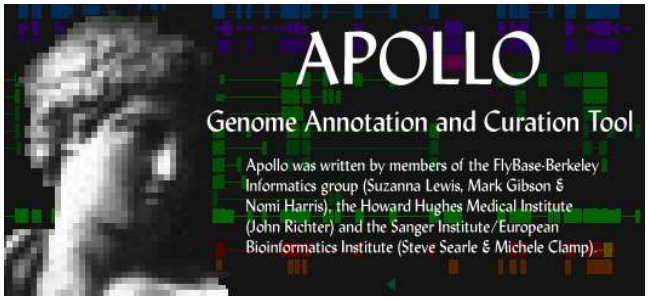
- Database schema – results tables
  - Nested results
  - All results with meta data
    - Evidence types
    - Data reagents



Pegasys server

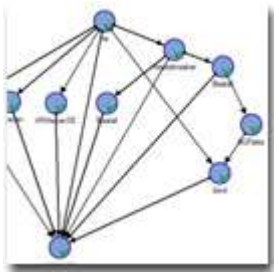


- Adaptor layer
  - Outputs results data structure to various usable text formats
  - GFF, GAME XML, raw text



Sohrab Shah <sshah@cs.ubc.ca>

Pegasys server

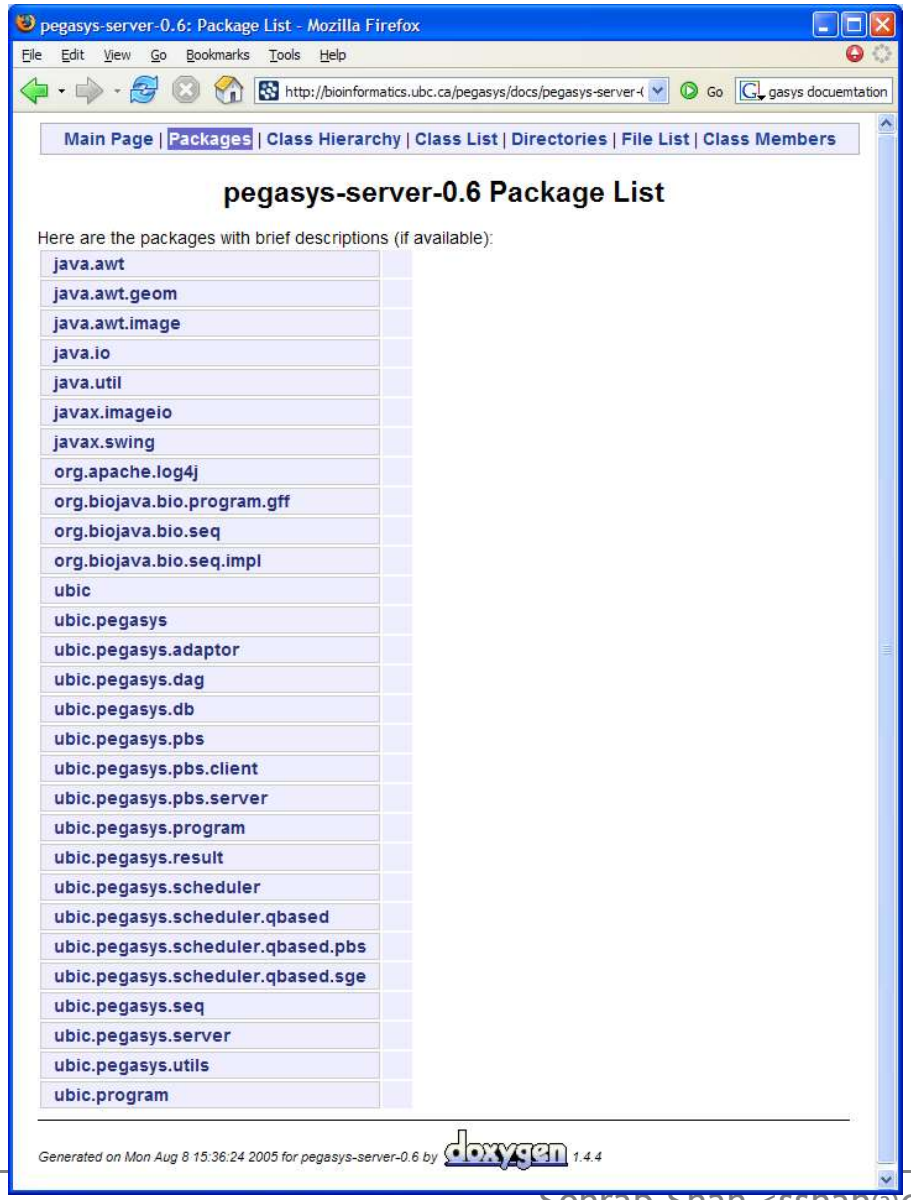


- Implementation details
  - Written in Java
  - Uses BioJava libraries
  - Runs on Postgresql RDBMS
  - Can use Sun Grid Engine OR OpenPBS scheduling systems
  - Source code is released under GPL

Sohrab Shah <sshah@cs.ubc.ca>

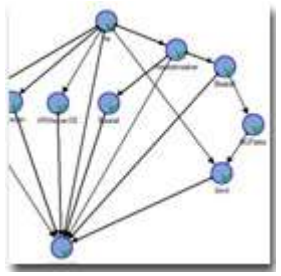
Pegasys server

- Source code documentation:

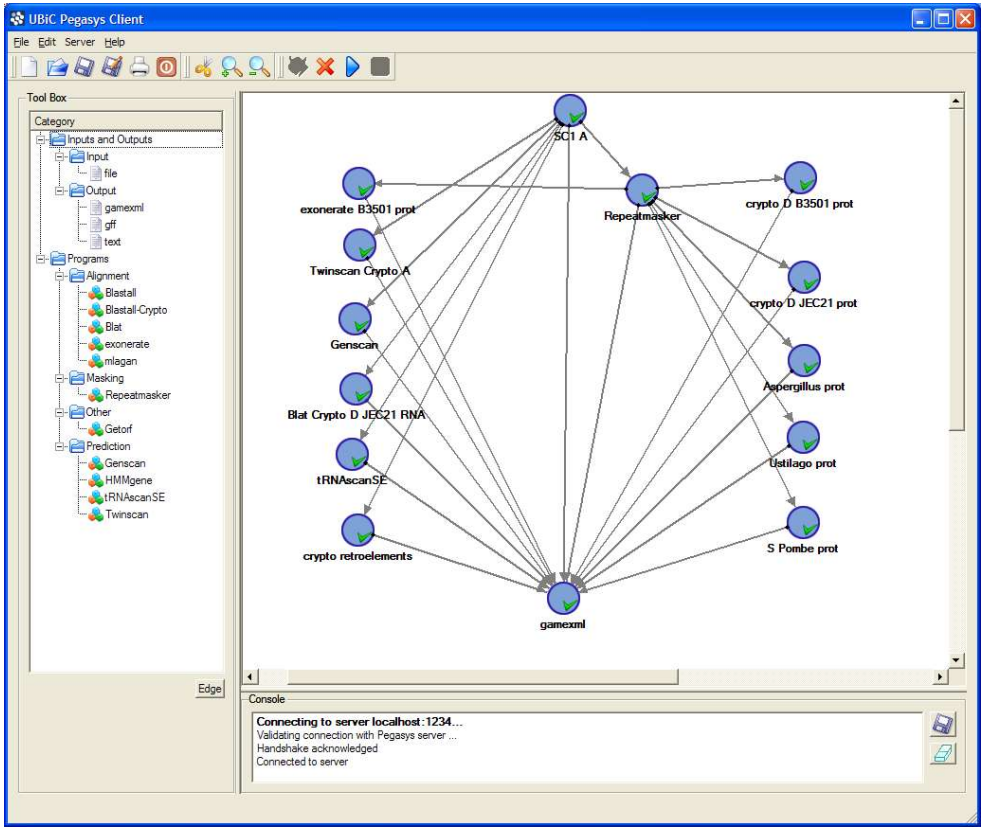


Sohrab Shah <sshah@cs.ubc.ca>

Pegasys client

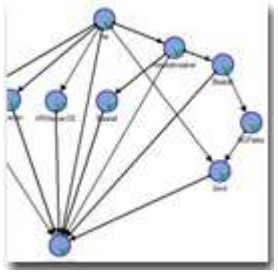


- Graphical user interface
  - Shows schematic representation of the workflow
- Create, modify and save workflows
  - Encodes workflow as an XML file that can be saved locally
- Send workflows to the server
  - Batch mode or single sequence mode
- Receive and save messages from the server
  - Status of individual nodes
  - Workflow completion
  - Output files saved to the local file system



Sohrab Shah <sshah@cs.ubc.ca>

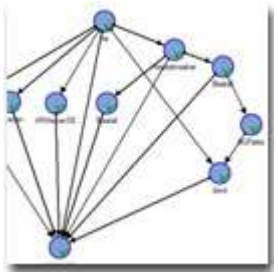
Pegasys client



- Added benefits:
  - Provides an interface to Unix command line tools
  - Gives a non computer savvy biologist the opportunity to run analyses in high-throughput

Sohrab Shah <sshah@cs.ubc.ca>

Pegasys client



- Implementation details
  - Developed using QT from Trolltech
    - Cross-platform GUI development software
    - Free for academics in Linux/Mac
    - Fee-based license required for Windows
    - Pegasys on your PDA?
  - Uses native widgets – has native look and feel
    - Fast response
  - Clean separation from server
    - All communication through a TCP-based 'Simple Message Protocol'
  - Written in C++
    - Compiles easily with QT utility

http://www.trolltech.com/

Sohrab Shah <sshah@cs.ubc.ca>

Part II: Setting up Pegasys
(adminguide.pdf)

Sohrab Shah <sshah@cs.ubc.ca>

Server installation

- Requirements
  - Java SDK 1.4.*
  - BioJava
  - Log4j
  - JArgs
  - JDom
  - PostgreSQL
  - openPBS or Sun Grid Engine
  - Ant
- Subscribe to the mailing list
  - http://bioinformatics.ubc.ca/pegasys/mailinglist.php

Sohrab Shah <sshah@cs.ubc.ca>

Server installation

- Download
  - http://bioinformatics.ubc.ca/pegasys/downloads/pegasys-server-0.6.tar.gz
  - Commands:
    - `> tar xvpzf pegasys-server-0.6.tar.gz`
    - `> cd pegasys-0.6`
    - `> ant`

Sohrab Shah <sshah@cs.ubc.ca>

Database setup and configuration

- Need PostgreSQL 7.3* database server
- Create empty database named ‘pegasysdb’
- Create user ‘pegasysdb_rw’ with permissions to create tables
- `> cd pegasys-0.6/sql`
- `> psql -U pegasysdb_rw -d pegasysdb -f pegasysdb.sql`

Sohrab Shah <sshah@cs.ubc.ca>

Database setup and configuration

- Create .dbrc file in $HOME
- Text file that contains connection parameters for databases
  - Format: 1 line per entry, space delimited
    - <database> <user> <pwd> <host>
- Example:
  - **`pegasysdb pegasysdb_rw [password] localhost`**

Sohrab Shah <sshah@cs.ubc.ca>

Server configuration

- Needs time investment!
- Mediated through 2 XML files
- Global configuration (pegasysConfig.xml)
  - Sets all system-dependent parameters
  - Server is invoked with:
    - `java ubic.pegasys.server.ServerDaemon pegasysConfig.xml`
- Program list (ProgramList.xml)
  - Specifies analysis programs supported on this server
  - Specifies supported parameters for each program

Sohrab Shah <sshah@cs.ubc.ca>

Server configuration

- Global configuration file:
  - Example given in:
  - INSTALLDIR/pegasys-0.6/server/conf/pegasysConfigExample.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pegasys_config>
<pegasys_config>

      …

</pegasys_config>
```


Sohrab Shah <sshah@cs.ubc.ca>

Global configuration

- Working directory
  - Pegasys optionally produces many (temporary) files
  - Specify where to put these files:

```
<working_dir>/tmp/</working_dir>
```


Sohrab Shah <sshah@cs.ubc.ca>

Global configuration

- Server daemon parameters:

```
<server_daemon>
   <!-- verbosity of output. can be : NONE, FATAL, WARN, INFO, DEBUG, ALL -->
   <log_level>WARN</log_level>
   <port>3488</port>
   <!-- name of PosgreSQL database to write the data -->
   <pegasysdb_name>pegasysdb_test</pegasysdb_name>
   <!-- path to XML file containing the Program list and args -->
   <proglist>/home/pegasys/progList.xml</proglist>
 </server_daemon>
```


Sohrab Shah <sshah@cs.ubc.ca>

Global configuration

- Scheduling system parameters:

```
<!-- config for the queue submission system, type: "Pbs" or "Sge" -->
<qsub type="SGE">

  …

</qsub>
```


Sohrab Shah <sshah@cs.ubc.ca>

Global configuration

- Scheduling system parameters:

```
 <!-- path to qsub application of openPBS/SGE -->

    <qsub_path>/opt/sge/bin/glinux/qsub</qsub_path>

    <!-- path to qstat application of openPBS/SGE -->
    <qstat_path>/opt/sge/bin/glinux/qstat</qstat_path>

    <qstat_server>
    <!-- hostname where Qstat server is running -->
      <host>oscar.hpc.bcgsc.bc.ca</host>
      <port>3499</port>
    </qstat_server>

    <!-- path to runner script that is submitted to PBS/SGE -->
    <!-- IMPORTANT: use sge script for SGE -->
    <script_path>/home/pegasys/pegasys-
0.5/server/shell/pegasys_sge_runner.sh</script_path>

    <!-- where to write output/err files created by SGE/PBS (default = /tmp") -->
    <tmp_file_dir>/scratch/pegasys/tmp</tmp_file_dir>

    <!-- delete the job output/err files created by SGE/PBS? (useful for debugging) -->
    <remove_tmp_files>yes</remove_tmp_files>
```


Sohrab Shah <sshah@cs.ubc.ca>

Server configuration

- ProgramList
  - Specify supported analysis tools and parameters
  - Example given in:
    - /pegasys-0.6/server/conf/progListExample.xml

Sohrab Shah <sshah@cs.ubc.ca>

Program List configuratio

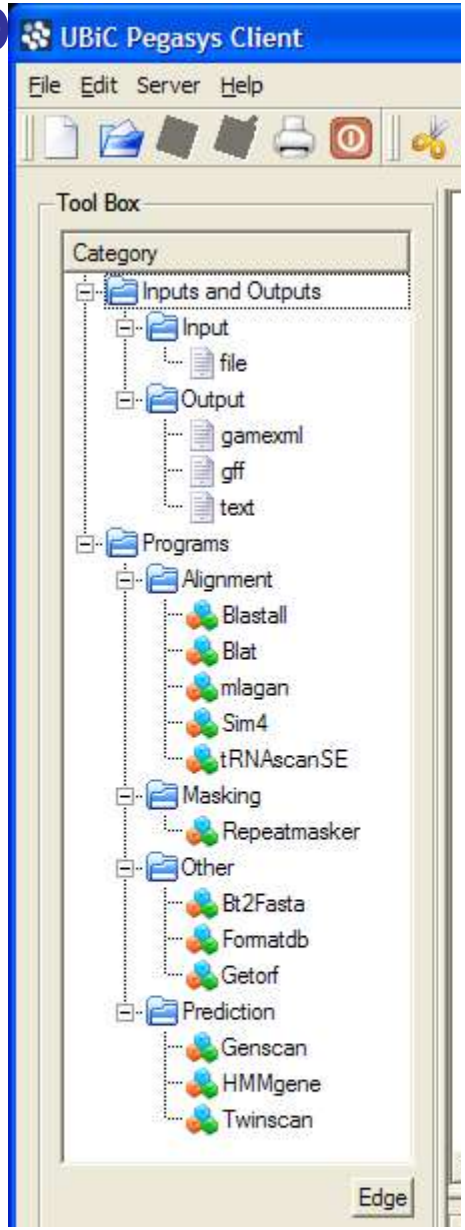
- Program List structure

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE pegasysProgXML (View Source for full doctype...)>
<pegasysProgXML>

<!-- BEGIN INPUT/OUTPUT NODES   -->
<io>
      <category name="Input">  … </category>
      <category name="Output">  … </category>
</io>
<!-- END INPUT/OUTPUT NODES   -->

<!-- BEGIN PROGRAM NODES  -->
<programs>
      <category name="Prediction"> … </category>
</programs>
<!-- END PROGRAM NODES  -->

</pegasysProgXML>
```



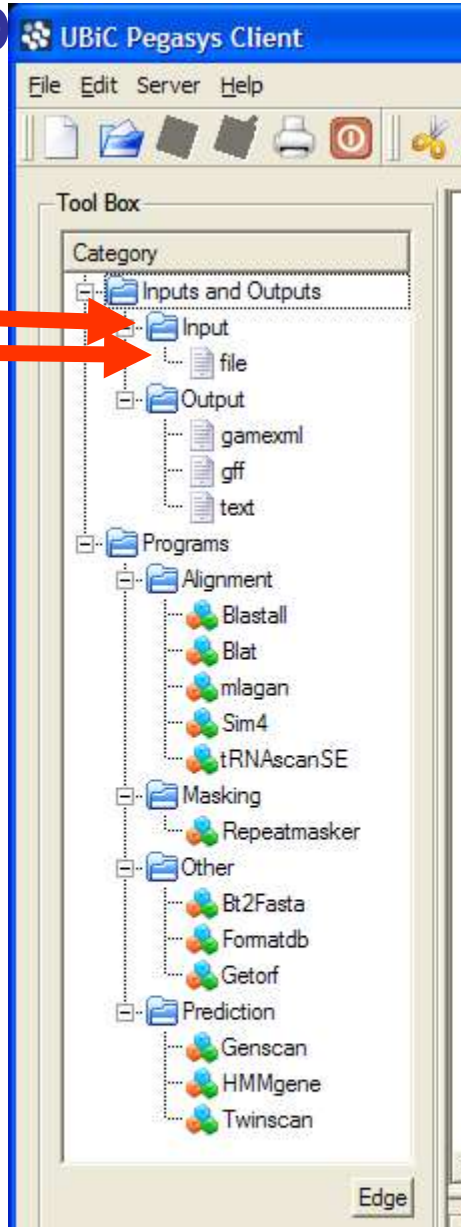
Sohrab Shah <sshah@cs.ubc.ca>

Program List configuratio

- Program List: Supported Inputs

```
<io>
    <category name="Input">
      <file>
        <outputs>
          <output name="file name" type="PLAIN_TEXT">
            <desc>The path to the file on the server</desc>
          </output>
          <output name="nucleotide FASTA sequence"
type="N_FASTA">
            <desc>A stream of nucleotides in FASTA format</desc>
          </output>
        </outputs>
      </file>
    </category>

    …
</io>
```



Sohrab Shah <sshah@cs.ubc.ca>

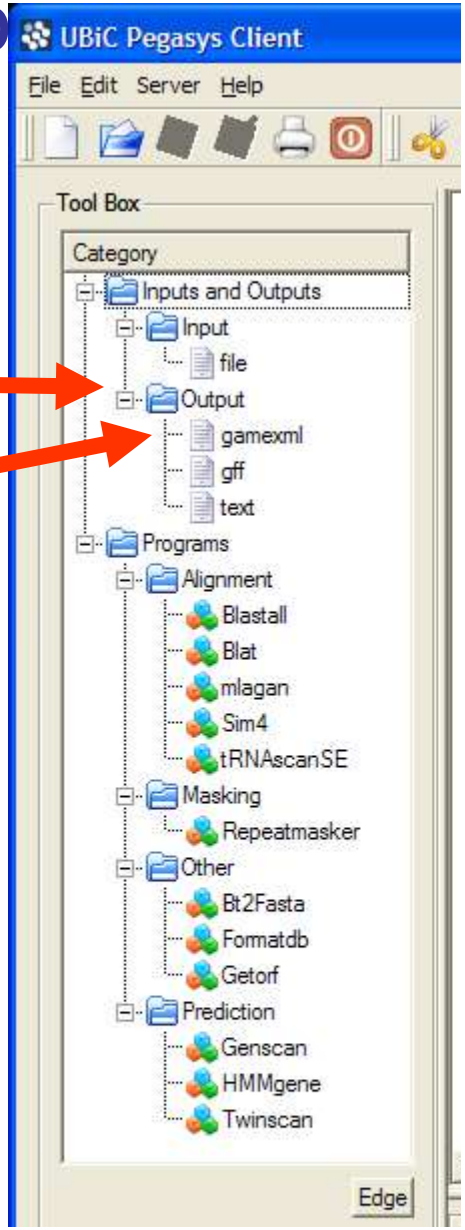
Program List configuratio

- Program List: Supported Outputs

```
<io>

…

<category name="Output">
      <text>
        <inputs>
          <input name="text" type="PLAIN_TEXT" cardinality="1">
            <desc>Plain text from any source</desc>
          </input>
        </inputs>
      </text>
      <gamexml>
        <inputs>
          <input name="nucleotide FASTA sequence" type="N_FASTA"
                                cardinality="1">
            <desc>A stream of nucleotides in FASTA format</desc>
          </input>
          <input name="Analysis results" type="ANALYSIS_RESULT"
                                cardinality="*">
            <desc>Results generated from an analysis program to be
                                included in the GameXML file</desc>
          </input>
        </inputs>
      </gamexml>
</category>
</io>
```



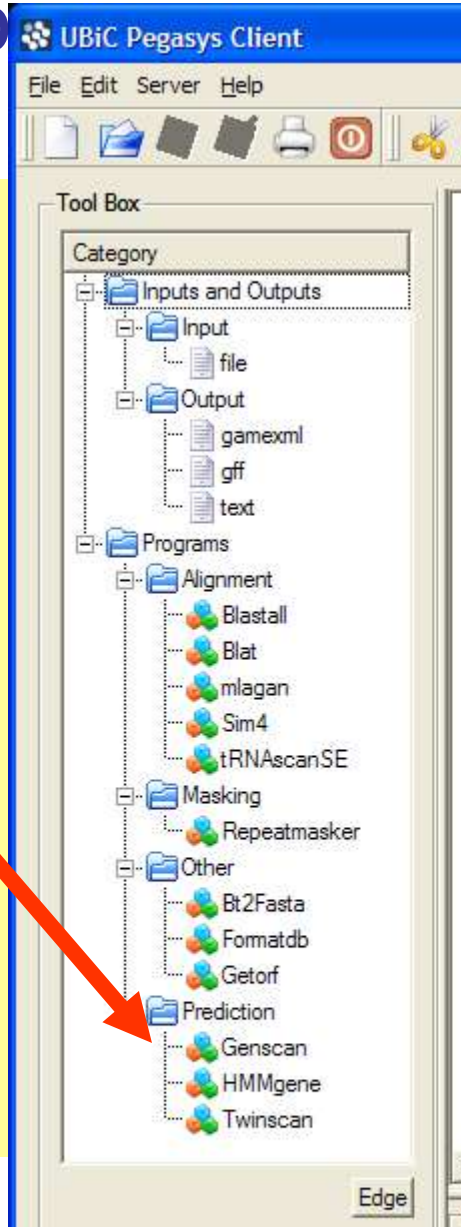
Sohrab Shah <sshah@cs.ubc.ca>

Program List configuratio

- Program List: Supported Programs

```
<!-- BEGIN PROGRAM NODES-->
  <programs>
    <category name="Prediction">
    <!-- BEGIN GENSCAN -->
      <program name="Genscan" classname="Genscan">
        <path>/usr/local/bin/genscan/GENSCAN/genscan</path>
        <config>
          <option
name="libdir">/usr/local/lib/genscan/GENSCAN/</option>
        </config>
        <parameters>
          <parameter name="parfname" required="true" enabled="true">
              <desc>Organism specific parameter file</desc>
              <values>
                 <value default="true">HumanIso.smat</value>
                 <value>Maize.smat</value>
                 <value>Arabidopsis.smat</value>
              </values>
           </parameter>
        </parameters>

    …
      </program>
    <!-- END GENSCAN -->
    …
  </programs>
```

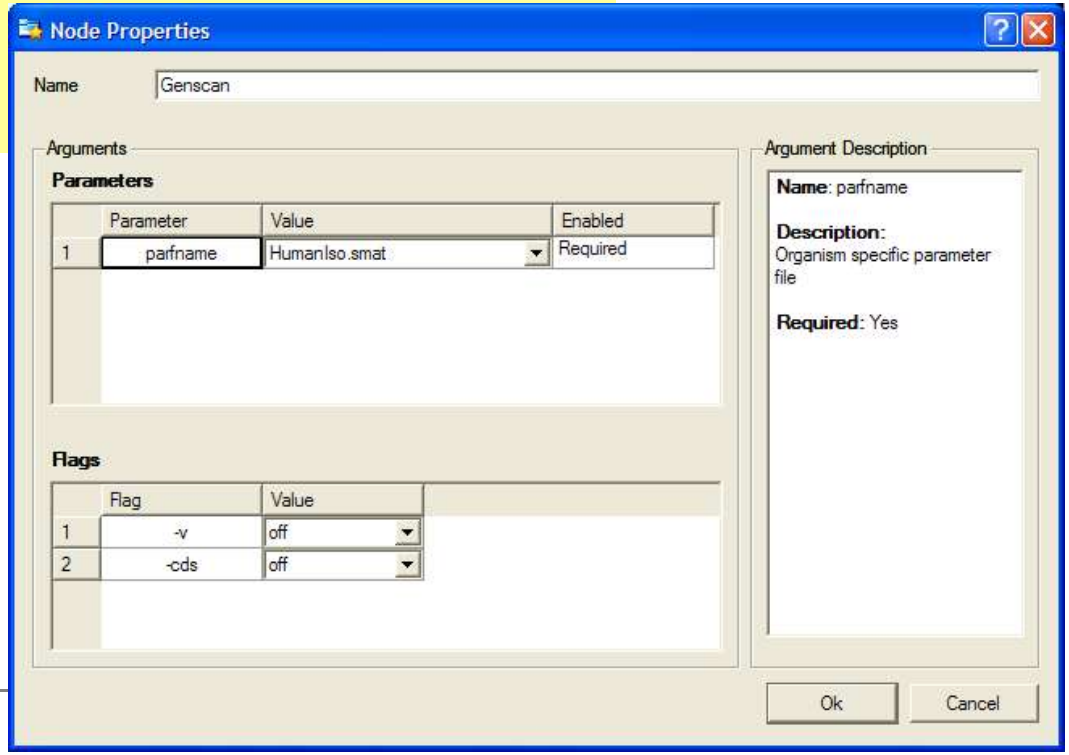


Sohrab Shah <sshah@cs.ubc.ca>

Program List configuration

- Supported programs – parameters/flags

```
<program name="Genscan" classname="Genscan">
…
          <flags>
           <flag name="-v" val="false">
             <desc>verbose output</desc>
           </flag>
           <flag name="-cds" val="false">
             <desc>print predicted coding sequences</desc>
           </flag>
         </flags>
…
</program>
```



Program List configuration

- Supported programs – Input/Outputs

```
<program name="Genscan" classname="Genscan">
…
<inputs>
    <input name="nucleotide FASTA sequence" type="N_FASTA" cardinality="1">
       <desc>A stream of nucleotides in FASTA format on which gene structures will be
                  calculated</desc>
    </input>
</inputs>
<outputs>
    <output name="predicted peptides" type="AA_FASTA">
       <desc>A list of predicted peptides in amino acids FASTA format</desc>
    </output>
    <output  name="predicted transcripts" type="N_FASTA">
       <desc>A list of predicted transcripts in nucleotide acids FASTA format</desc>
    </output>
    <output name="Genscan output" type="PLAIN_TEXT">
       <desc>The Genscan output as plain text</desc>
    </output>
    <output name="Analysis results" type="ANALYSIS_RESULT">
       <desc>Results generated from this analysis program</desc>
    </output>
</outputs>
</program>
```


Sohrab Shah <sshah@cs.ubc.ca>

Program List configuration

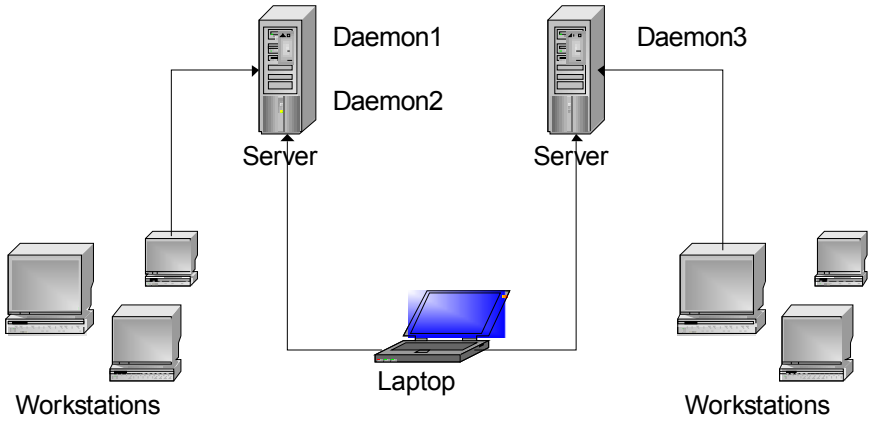
- Supported analysis programs

RepeatMasker	3.0.8
WU-BLAST	2.0MP-WashU
exonerate	0.9.0
NCBI BLAST	2.2.10
Smith-Waterman	EMBOSS 2.10.0
Getorf	EMBOSS 2.10.0
genscan	1.0
HMMgene	1.1
InterProScan	4.0
Mlagan	1.1
Sim4	
TrnaScan-SE	1.23
Twinscan	2.0 beta
GeneSplicer	05/26/04
BLAT	27.1

Sohrab Shah <sshah@cs.ubc.ca>

Server configuration

- Can deploy different servers on the same machine
  - Different configurations
  - Invoke specialised servers for different groups of users
    - Different analysis tools, different databases
    - Only present tools of interest to selected groups
    - Limit compute intensive applications to selected groups
    - Create configuration files for machine-supported tools
  - Only one system required for multiple different projects
  - Facilitates project management for users



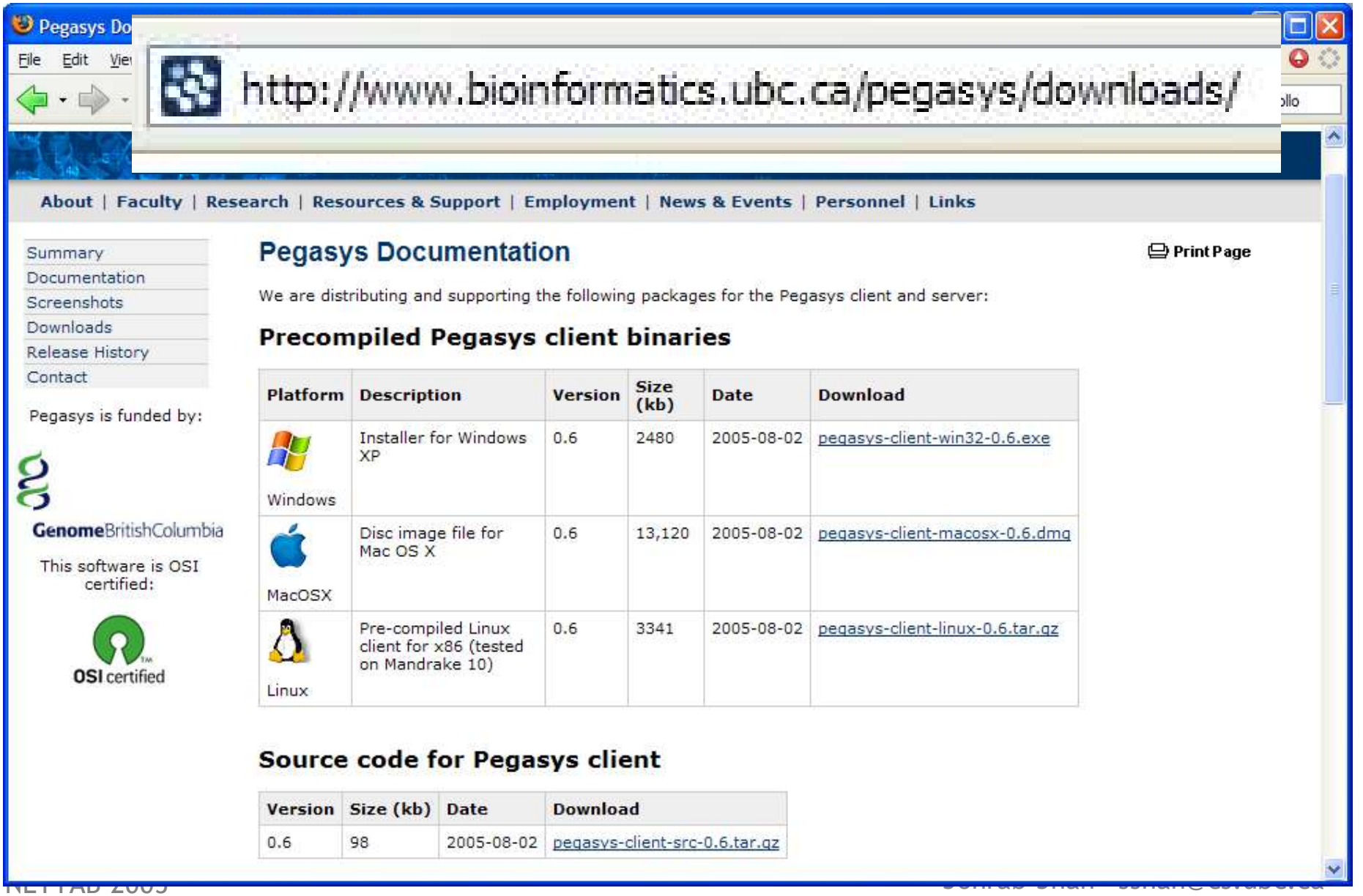
Sohrab Shah <sshah@cs.ubc.ca>

Server configuration

- Caution!
  - Maintaining tools and databases requires some investment
  - Mailing lists
  - Automated scripts
  - Ultimate: have a large compute server where resources are maintained by permanent personnel

Sohrab Shah <sshah@cs.ubc.ca>

Client installation

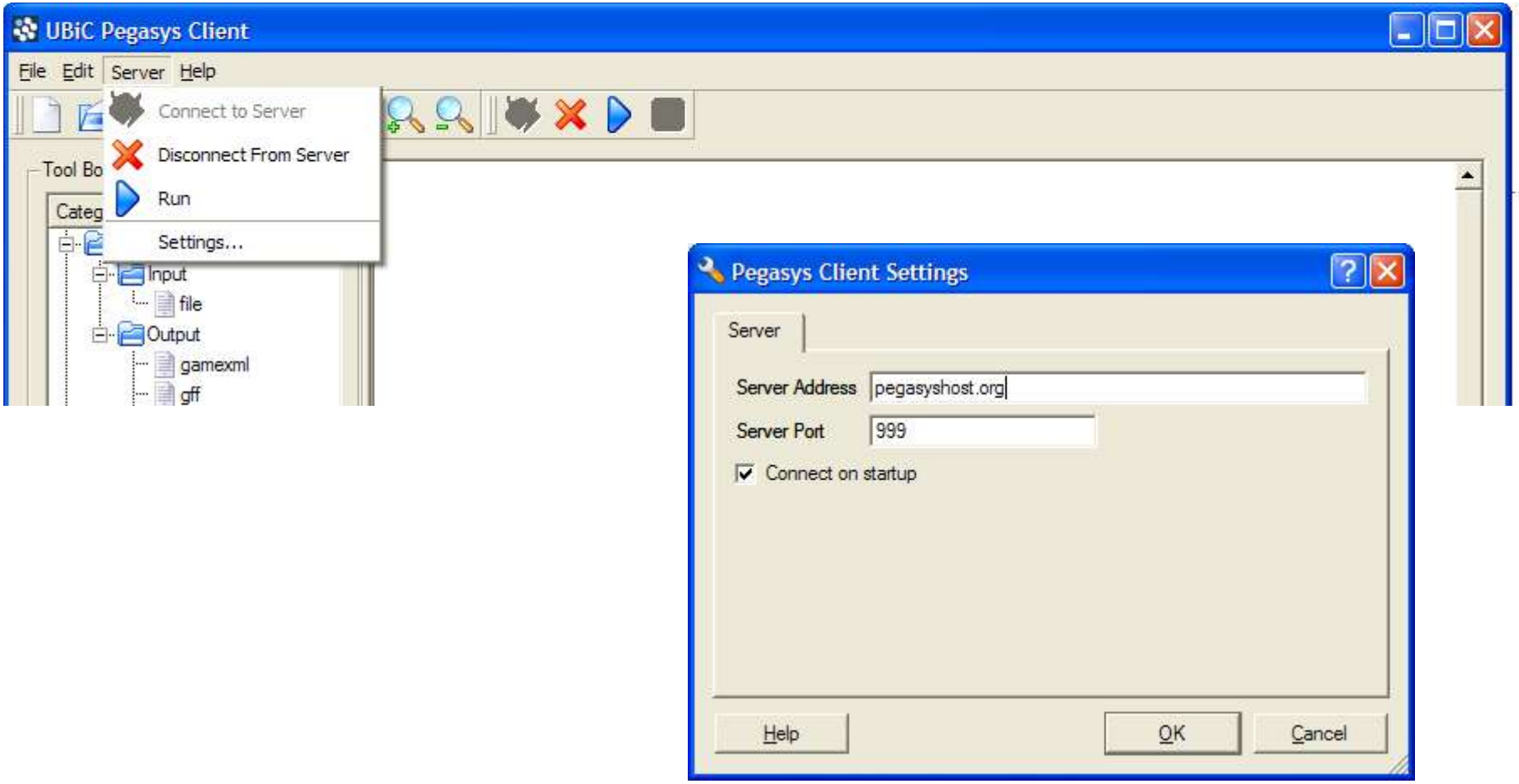


Client installation

- From source under Linux
  - Download:
  - http://www.bioinformatics.ubc.ca/pegasys/downloads/client/pegasys-client-src-0.6.tar.gz
  - **> tar xvpzf pegasys-client-src-0.6.tar.gz**
  - **> cd pegasys-client-src-0.6**
  - **> qmake**
  - **> make**
  - **> ./pegasysGUI**
- Dependencies
  - QT 3.0 or higher from Trolltech
    - Comes with Mandrake (Mandriva) and RH fedora core distributions

Sohrab Shah <sshah@cs.ubc.ca>

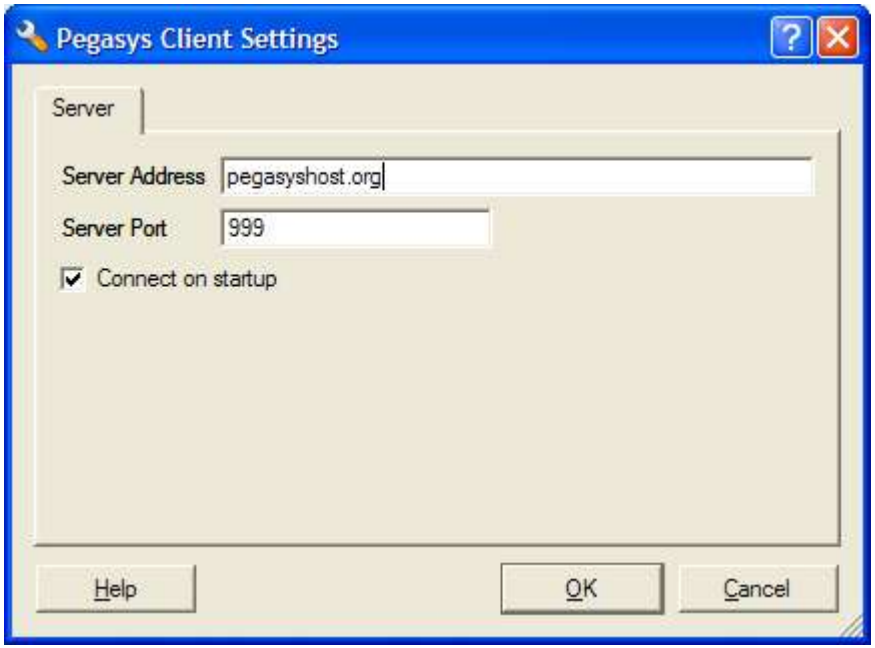
Client configuration



Sohrab Shah <sshah@cs.ubc.ca>

Client configuration

- Insecure way:
  - Open a port outside your firewall
- Secure way
  - Create an ssh tunnel
  - For Unix – use ssh –L
  - For Windows
    - Use ‘Putty’
    - Create a tunnel
    - I will demo this



Sohrab Shah <sshah@cs.ubc.ca>

Part III: Using Pegasys

(userguide.pdf)

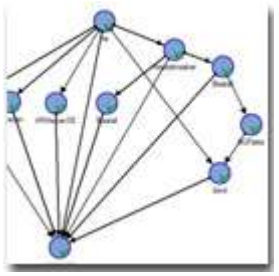
Sohrab Shah <sshah@cs.ubc.ca>

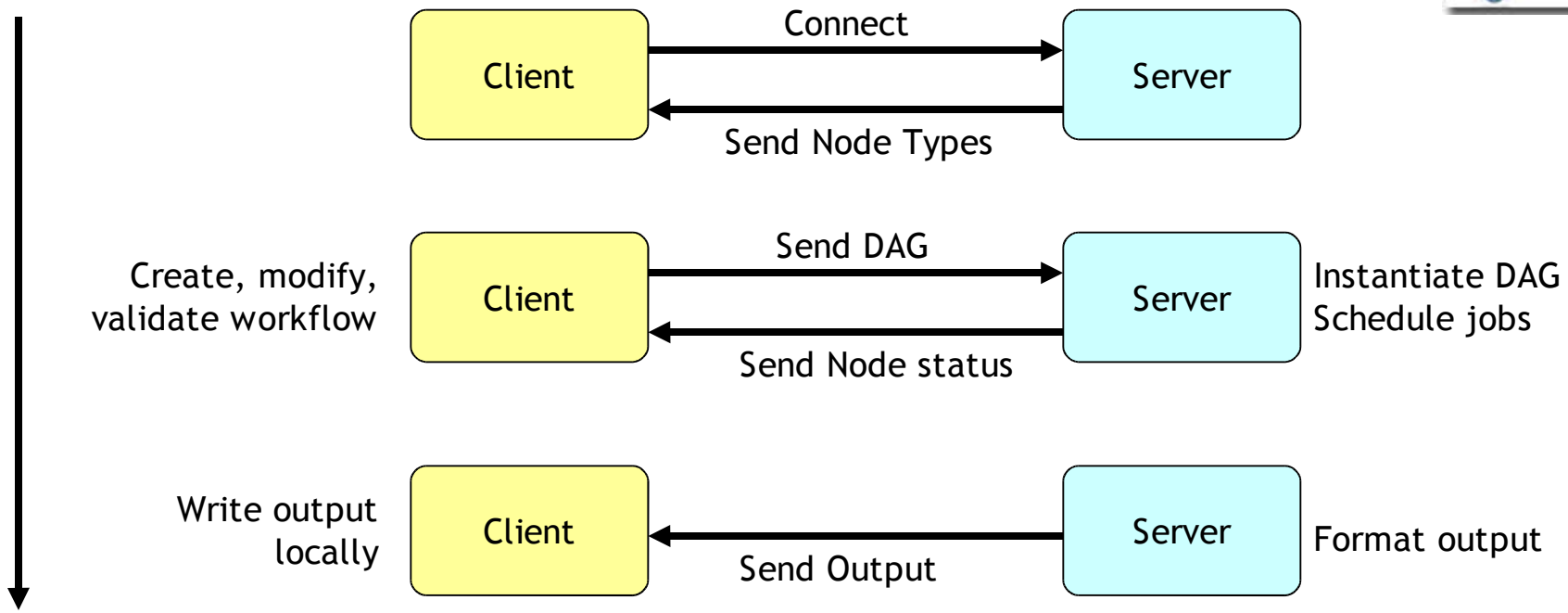
Using Pegasys

- Invoking the server
  - **`java ubic.pegasys.server.ServerDaemon pegasysConfig.xml`**
  - Server runs as a daemon (persistent process)
  - listens for TCP/IP requests on a dedicated port

Sohrab Shah <sshah@cs.ubc.ca>

Summary of data flow sequence

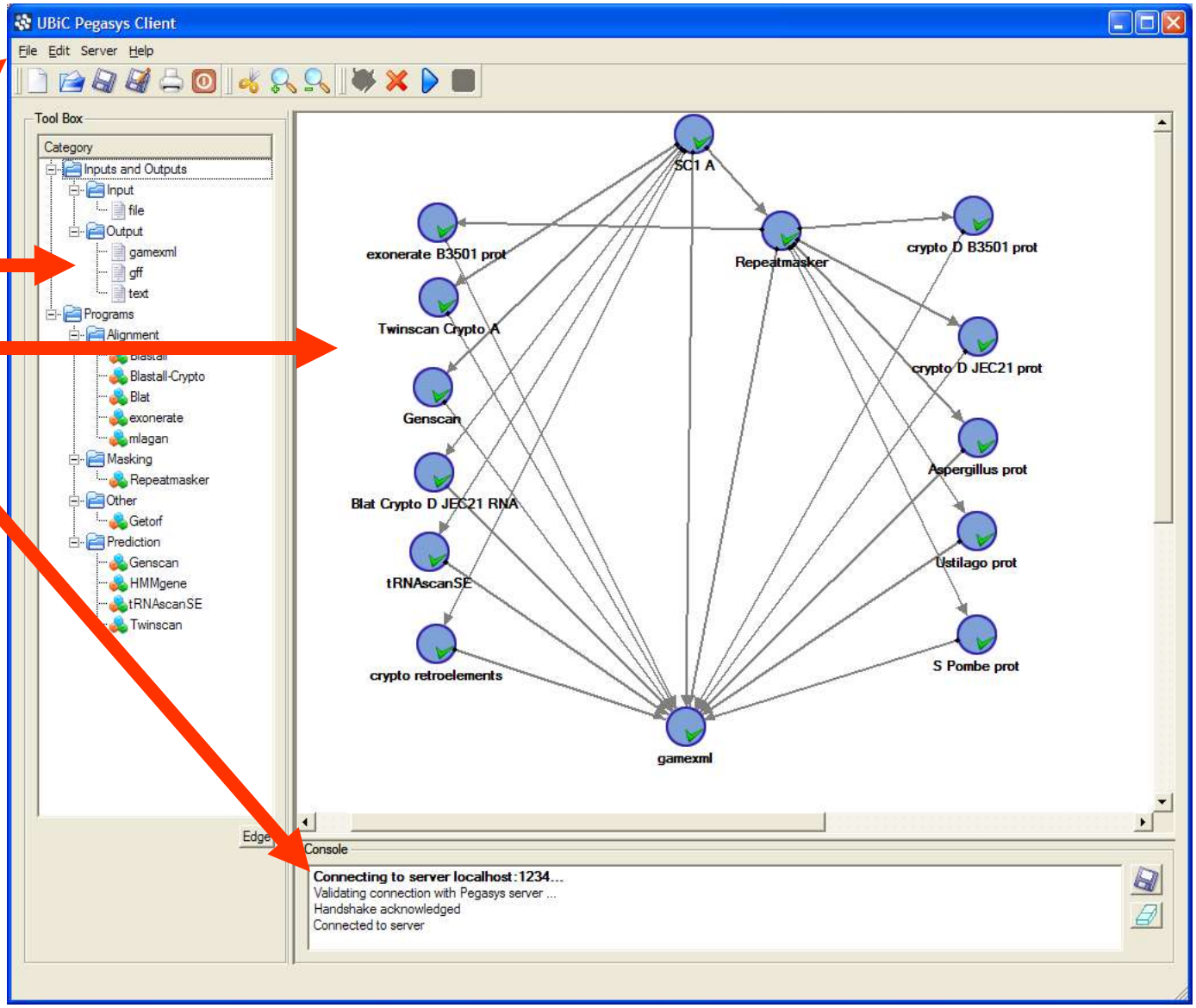




Sohrab Shah <sshah@cs.ubc.ca>

Client components

- Menu
- Toolbox
- Canvas
- Console

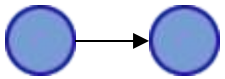


Sohrab Shah <sshah@cs.ubc.ca>

Using the client

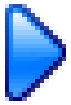
- Step 1
  - Connect to server

- Step 2
  - Create workflow



- Step 3
  - Save workflow

- Step 4
  - Send workflow



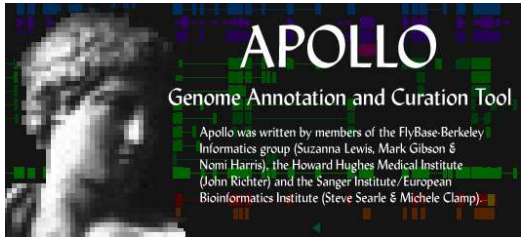
- Step 5
  - Evaluate results

Sohrab Shah <sshah@cs.ubc.ca>

Run-time details

- Supported input formats
  - FASTA nucleotide/protein
- Supported output formats
  - Raw text from any analysis program
  - Generic Feature Format (GFF) 2.0
    - Used in many visualisation tools, annotation servers
  - GAME XML
    - Rich XML-based modeling language for biological features and computational features
    - Developed for the Apollo genome editor

Lewis SE, Searle SM, Harris N, Gibson M, Lyer V, Richter J, Wiel C, Bayraktaroglir L, Birney E, Crosby MA, Kaminker JS, Matthews BB, Prochnik SE, Smithy CD, Tupy JL, Rubin GM, Misra S, Mungall CJ, Clamp ME.
Apollo: a sequence annotation editor. *Genome Biol*. 2002;**3**(12)



Sohrab Shah <sshah@cs.ubc.ca>

Demo 1: Simple Blast

- Creating a file node
- Creating a program node
- Connecting file nodes with program nodes
- Setting parameters in a program node
- Creating an output node
- Demo different formats

Sohrab Shah <sshah@cs.ubc.ca>

Demo 2: Repeatmasker->Blast

- Linking Program nodes
- Useful protocol under many different scenarios
  - Multiple alignment -> phylogenetic tree
  - Mask for repeats -> sequence alignment
  - *Ab-initio* gene prediction -> sequence alignment
  - Many others…

Sohrab Shah <sshah@cs.ubc.ca>

Demo 3: multiple different Blasts

- Blast against different databases
- Save results separately
- Integrate results using GAME XML output node
- Open results in Apollo

Sohrab Shah <sshah@cs.ubc.ca>

Demo 4: Batch input

- Show previous example with multiple FASTA input

Sohrab Shah <sshah@cs.ubc.ca>

Importance of Studying Cryptococcus

- *Cryptococcus neoformans var Gattii*
- Opportunistic fungal pathogen of people with compromised immune systems
  - Causes life threatening infections in ~10% of AIDS patients
- Can also infect immunocompetent people
  - Emergence of Cryptococcosis on Vancouver Island
    - 98 reported human cases and 200+ animal cases since 1999

Sohrab Shah <sshah@cs.ubc.ca>

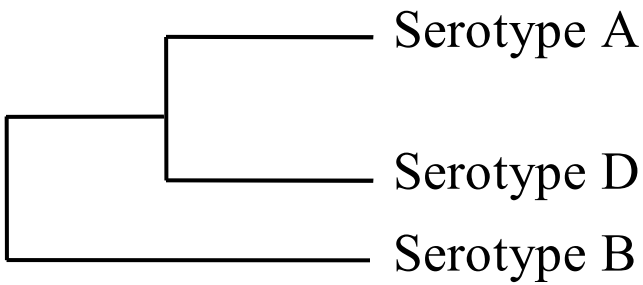
Serotypes A, B, D

- Serotype A
  - Strain 184A – Oklahoma
  - Strain H99 – Broad Institute
- Serotype B
  - Strain R265 – Broad Institute
  - Strain WM276 – UBC
- Serotype D
  - Strain JEC21 – TIGR
  - Strain B3501-A - Oklahoma/Stanford

Sohrab Shah <sshah@cs.ubc.ca>

How are the serotypes related?

- Serotype A and D are more closely related
  - Approximately 20 million years apart
- Serotype B is more distant
  - Approximately 40 million years apart



Sohrab Shah <sshah@cs.ubc.ca>

Cryptococcus neoformans - The Genome

- Approximately 19Mb
- 14 chromosomes
- Estimated 6,000 genes – to be annotated!
  - Average of 5.7 exons per gene (based on TIGR annotations)

Sohrab Shah <sshah@cs.ubc.ca>

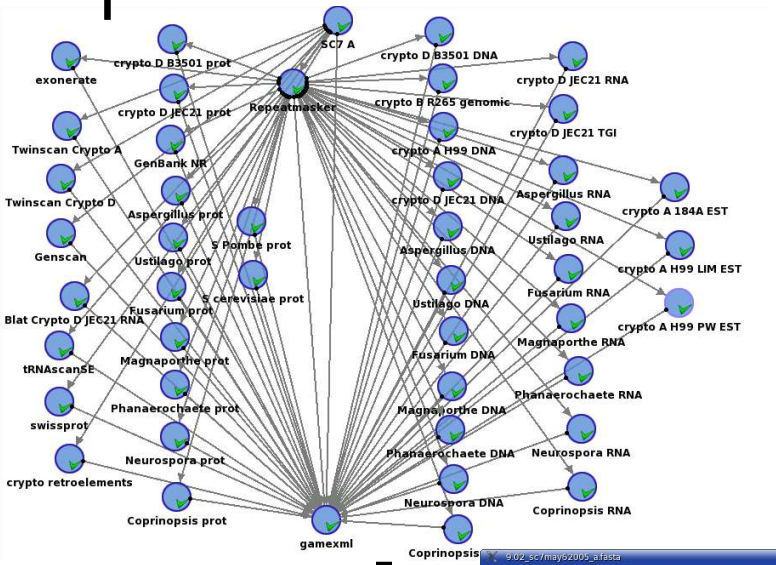
Demo – full cryptococcus

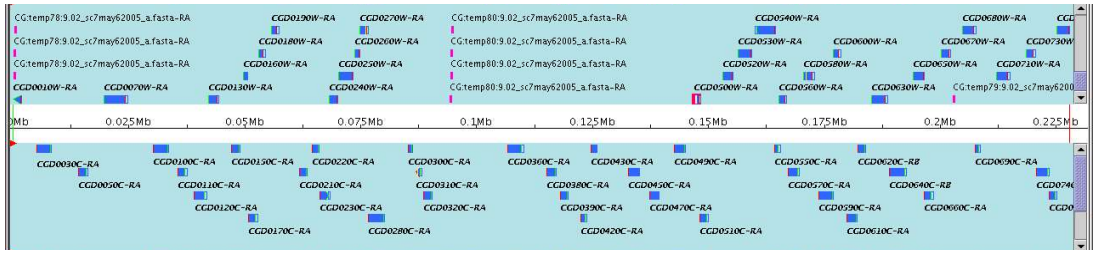
- Pegasys at its most complex

Sohrab Shah <sshah@cs.ubc.ca>

Annotation procedure

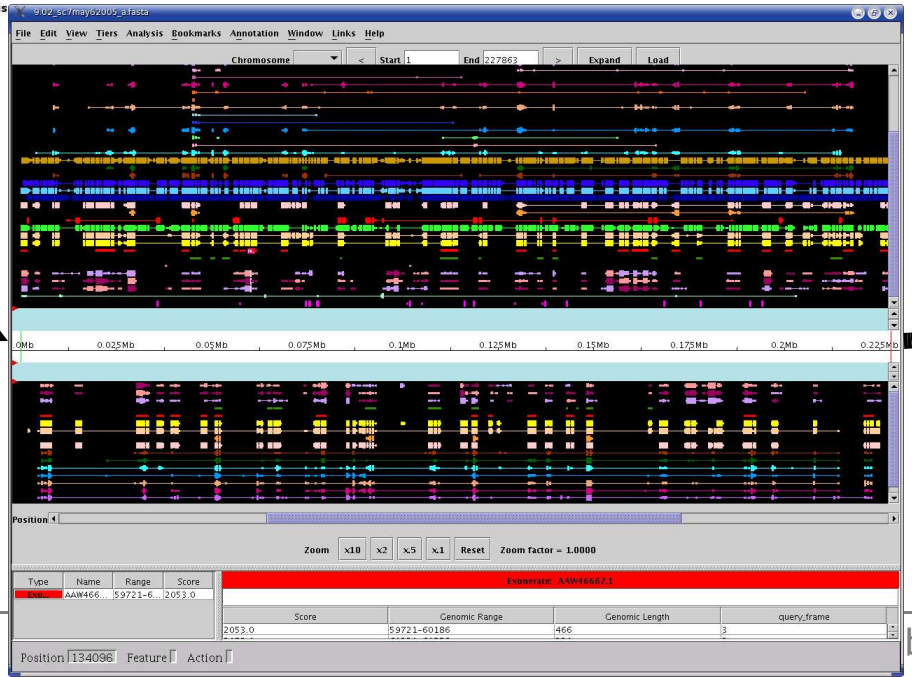
1





3

2



b Shah <sshah@cs.ubc.ca>

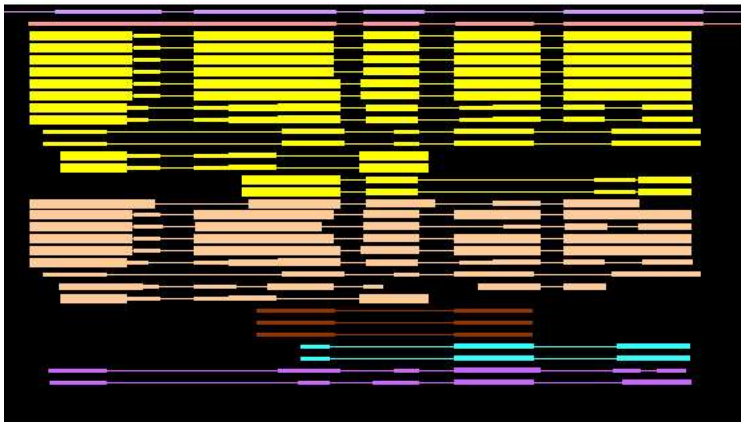
Pegasys - Limitations

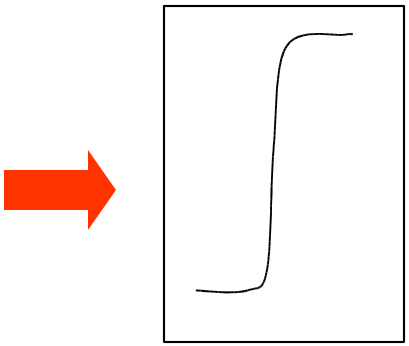
- What about web services?
  - Chinook at Michael Smith Genome Sciences Center
- Significant overhead for server setup
  - But client side users – no overhead
- Genome annotation-specific
  - Gene expression analysis?
  - Proteomic analysis?
- Client is too lightweight
  - No sessions
  - Username/password authentication?
  - Cannot talk to database directly
  - Partial re-run of workflow
  - Possible due to architecture, but not currently implemented
- Others?

Sohrab Shah <sshah@cs.ubc.ca>

Pegasys future work

- Scientific
  - Create algorithms to automate the annotation





Machine learning
Computational statistics
Probabilistic inference

Partially automated annotation

Sohrab Shah <sshah@cs.ubc.ca>

Pegasys future work

- Scientific
  - Evaluation of protocols
  - What is the best set of tools to use for annotation?
  - What parameter settings work best for my genome?

Sohrab Shah <sshah@cs.ubc.ca>

Pegasys future work

- Software
  - Administration GUI tool
  - ‘heavier client’ for sessions/partial workflow re-run
  - Possibility of other tools besides genome annotation?
  - Streamlining server memory usage

Sohrab Shah <sshah@cs.ubc.ca>

Where do we go from here?

- Technologically
  - Incorporate web services
  - Reduce burden of system administration
  - Use sessions
- Within genome annotation
  - Streamline start->finish processing of a genome
  - Intelligent ways to integrate heterogeneous analysis results
    - Multi-parametric regression/classification techniques to automatically process results
  - Evaluate protocols
    - Which workflows produce the best results?
- Beyond genome annotation
  - Gene expression
  - Gene regulation
  - Comparative genomic hybridization

Sohrab Shah <sshah@cs.ubc.ca>

Acknowledgements

- **UBC Bioinformatics Centre**
  - **Francis Ouellette**
  - **Graham McVicker (University of Washington)
  - Joanne Fox
  - Jessica Sawkins
  - **David He (University of Toronto)
  - **Mack Yuen
  - Miroslav Hatas
  - Jonathan Falkowski
- **Michael Smith Laboratories**
  - **Jim Kronstad**
  - Cletus D'Souza
- **BC Genome Sciences Centre**
  - Greg Taylor
  - Bernard Li

Sohrab Shah <sshah@cs.ubc.ca>