

# Clusterweb – a WEB-based cluster management interface

Gelato Federation, October 2003

# Benefits for typical users

- Many cluster management systems have a lot of command line options, scriptable parameters, and different tools, which are very difficult to learn for the typical user.
- ✓ WEB-based interface simplifies job submission and management. It also could be very useful for users that access cluster from non-UNIX environments, for example Windows and Mac machines.
- Typical job management includes file uploads and downloads which are usually performed using FTP protocol, which is not secure. Simple FTP client exists in all major OS distributions, on the other hand for secure file transfers additional software must be installed.
- ✓ WEB interface does not require any additional software and can be used from any HTML compliant browser. All data transfers can be made secure if server and client browser support SSL.
- ✓ Besides this, WEB-based interface to cluster can retain information concerning uploaded job files and used command scripts, which could be very convenient for all users accessing cluster. Moreover, WEB-based interface can support different user customizations and have nice graphical look.

# Implementation

- Clusterweb system is built using Java 2 frameworks and services for business logic components and Servlet API for front-end on the top of the Apache Tomcat 4.1 servlet container.
- Interface features include job management, command file management and user settings management.
- The interface can be used with different cluster management systems, which have command line interface or can be accessed through API services.
- Current version supports only English locale and language settings, but localization is straightforward.
- Future versions may include integration with external systems, i.e. Grids.

# Clusterweb features

- Clusterweb supports submission and management of parallel jobs to cluster by means of communicating with underling management system.
- Each job is represented as a number of files, compressed in a single ZIP archive. Job has a name, size, date, number of CPU and status attributes. Job may have other specific attributes, which depend only on the target management system.
- Every user has a virtual home directory, which holds all submitted jobs. User can upload new job, delete or restart already finished jobs.

# User's home page

**Main menu:**

---

[Show User Jobs](#)  
[Add New Job](#)  
[Show User Scripts](#)

---

[Logout](#)

**Job list:**

---

Name	Size	Date	CPU	Status
<a href="#">linpack</a>	235101	9/16/03	4	new

[Refresh](#)

- Home page contains main menu and a list of jobs.
- Name is an ordinary label for a job; therefore, two jobs with the same name may exist in the list.
- The size attribute shows the size of the corresponding job archive file in bytes.
- Date attribute shows the date when the corresponding job has been added to the list.
- CPU is the number of processing elements required by the job.

# Job status codes

- There are five possible values for the status code:

NEW	Job has been submitted to the list.
WAITING	Job has been started and is waiting for its execution time.
RUNNING	Job is running.
STOPPED	Job has been stopped.
FINISHED	Job finished.

# Submission of new jobs

- Submission of new jobs is performed in two steps:
  - First, user should create a single ZIP archive file, with all necessary data for job execution. This file will be further decompressed by the system into temporarily directory on a target machine. The temporarily directory will be set as a current directory for the job.
  - The second step is to create a command file for the job.

# Default command files

- Command file depends on the underlying management system; hence, creation of the command file may be troublesome for end users.
- Clusterweb can automatically create a command file for a job based on job's specific attributes.
- Command file may have several versions to support different cluster management systems. Clusterweb chooses an appropriate version automatically, each time job is executed on a target cluster machine.

# Example of submitting a new job

**Add new job**

---

Enter job name:

Enter job description:

Select cpu count:

Select file:

Select start time of the job:

After upload

Start manually

---

- To submit a new job, user should enter job name, description, required number of processing elements and job archive file. Job description is optional.
- Job can be started in two ways: either automatically after upload, or manually later. By default, starting time for a job is manual.

# Job execution on a target machine

- Job archive file should contain all data necessary for job execution and a command file named *start*.
- Before starting the job on a target machine, the system will decompress job archive file into a temporarily directory.
- After decompression, the system will try to execute (in terms of the underlying management system) command file with the name *start*.
- Job standard output and error streams should be redirected to files named *stdout* and *stderr* correspondingly. If the system fails to find these files, the contents of the corresponding output streams will not be available.

# Managing command files

- User can manage command files used for job execution.
- The scripts are organized in groups. User can add or delete script groups.
- User can delete only empty groups and scripts, which are not assigned to jobs.

# Script list page

Main menu:

---

[Show User Jobs](#)  
[Add New Job](#)  
[Show User Scripts](#)

---

[Logout](#)

Script list:

---

Sort by  Show group  [New group](#)

Name	Size	Date
<a href="#">Test script</a>	51	9/17/03
<a href="#">Linpack on mpich-qm</a>	333	9/17/03

[New script](#)

- Each script has name, size and date attributes. Two scripts with the same name may exist in the list.
- Size attribute identifies the size of the script in bytes.
- Date attribute shows the date when the corresponding script has been added to the list.

# Creation of new scripts

- During job upload user can create a new script or select some script from the list and edit it if necessary.

**New script**

---

**Create in group**

**Enter script name:**

**Enter script description:**

The script for running linpack test suite on myrinet

**Enter script body:**

```
#!/bin/bash

#PBS -o stdout
#PBS -e stderr

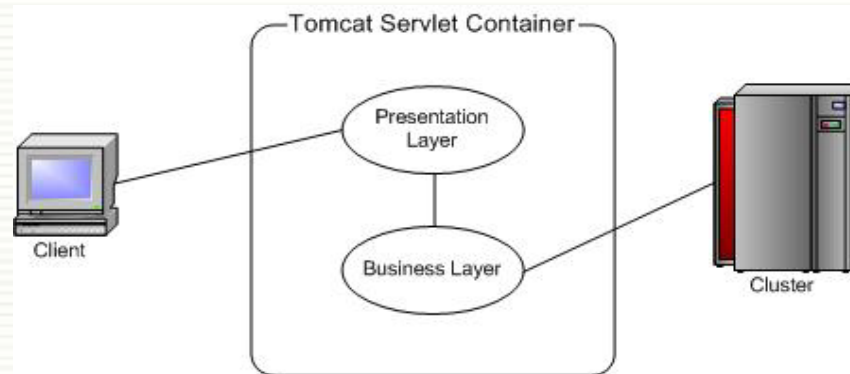
# Export all environment variables to the job
#PBS -V

# start execution
cd $PBS_O_WORKDIR
# make

# run program
cat $PBS_NODEFILE | awk '{print $1":2"}' > mf

ncpus=$(wc -l mf | awk '{print $1}')
/export/mpich-gm-1.2.5-10/bin/mpirun.ch_gm -np $ncpus -machinefile mf xhpl
```

# Clusterweb architecture



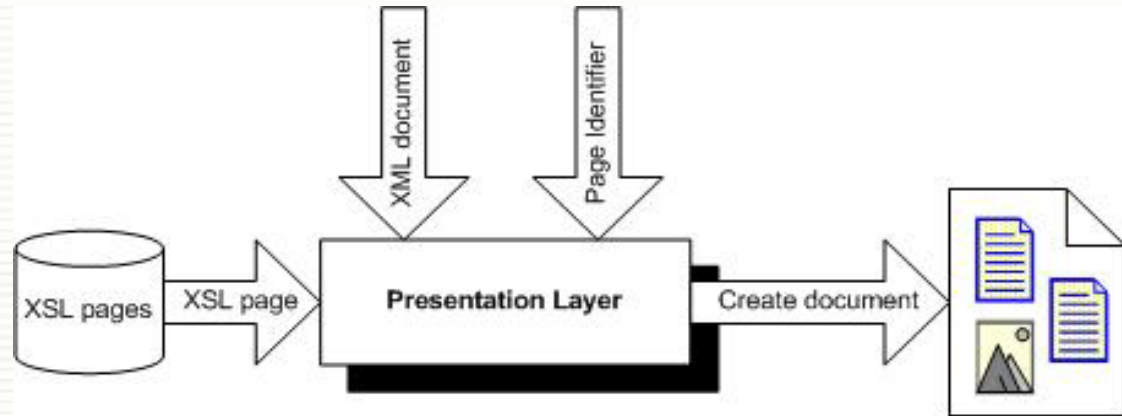
- Clusterweb architecture is a variation of the Model-View-Controller (MVC) design pattern.
- The system clearly separates the layout (presentation) information from the content, so that it is possible to modify the layout and/or translate it to another language (localize) not affecting the business processing.

# Presentation layer

- The purpose of the Presentation Layer is to provide a mechanism to create presentation documents (HTML) with the following goals kept in mind:
  - The system must be able to handle a number of locale-specific versions of each page. The system must choose the most appropriate page version transparently to the underlying processing layers.
  - The stylesheets must be completely separated from the application logic, so that it were possible to modify the layout, perform localization and otherwise edit stylesheets without any impact on the code. Such separations of concerns implies that people without Java programming skills can customize layout as far as it doesn't affect screen navigation, essential data passed and other functional aspects.
  - Although the current version does not deal with presentation media other than HTML, it is possible that the forthcoming versions will. Hence, it should be possible to add another presentation media with no impact on core business logic and minimal impact on other layers.

# Presentation layer details

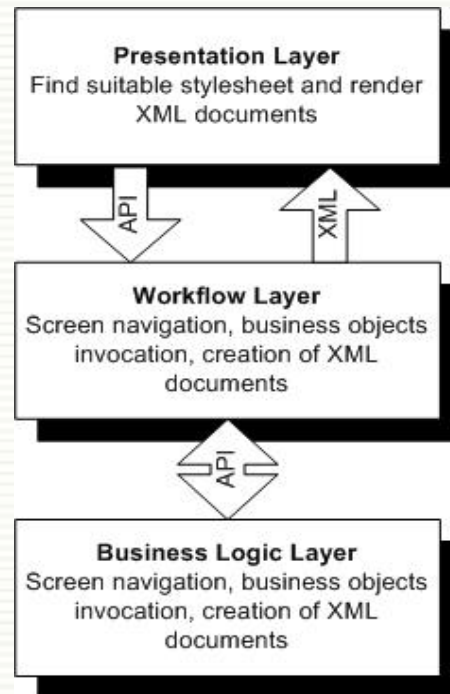
- In order to satisfy the goals above, XML/XSL-based approach was chosen as an underlying technology for the Presentation Layer.
- Presentation layer is responsible for rendering of the XML documents created by the underlying layer using the most appropriate chosen stylesheet.
- Both Presentation and Business layers coexist within Workflow layer.



# Workflow layer

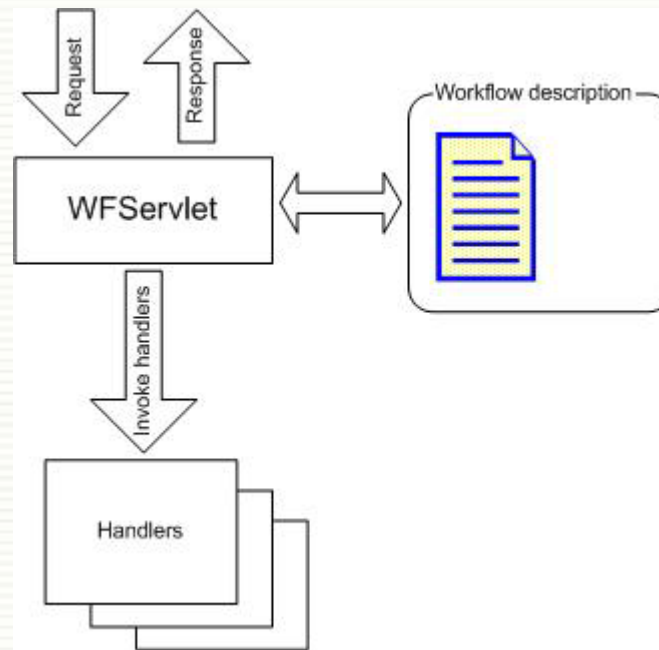
- Workflow Layer is designed to keep Business Objects in the Business Layer free from state management and interaction with the Presentation Layer.
- Its responsibility is to convert user actions into calls to stateless transactional Business Objects, gather data and prepare XML documents to be rendered by the Presentation Layer.
- Unlike the Presentation, Workflow Layer includes both framework parts and pluggable pieces of code, which implement functionality specific to particular pages and page groups.
- Workflow Layer also works as a “defense perimeter”: this is where the request parameters are parsed, validated, and permissions are checked. The calls from Workflow Layer to Business Objects must ensure that according to a particular Business Object contract, the call will not violate the security constraints.

# Clusterweb layers



# Workflow layer details

- Workflow layer corresponds to a common Three-Tier Architecture.
- WFServlet analyzes incoming request, invokes appropriate request and response handlers based on some conditions.
- Handlers modify XML document (DOM object), which is used for constructing response.

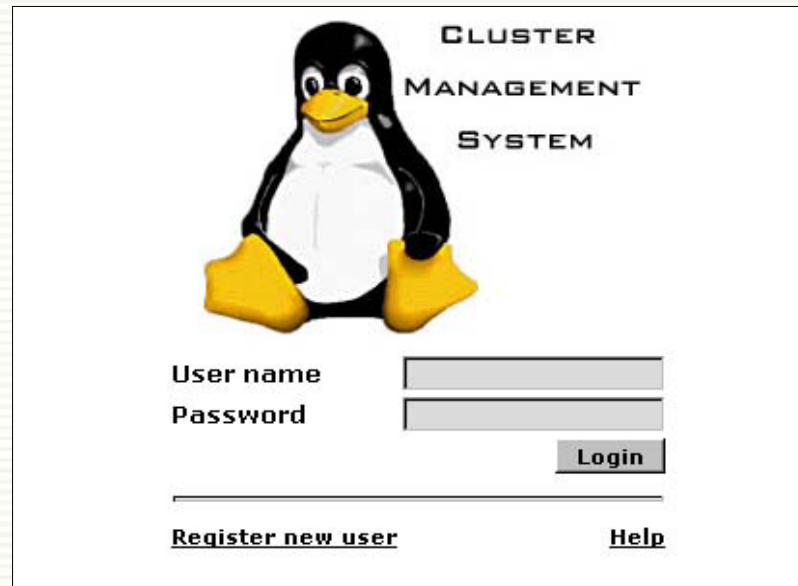


# Workflow benefits

- WEB application consists of handlers and XSL pages.
- Handlers represent application logic, while XSL pages hold presentation information.
- Localization requires a set of new XSL pages and a set of localized messages (Java Resource Bundles).
- Each handler is independent from others. All interaction between handlers is performed through conditions and HTTP session.
- Error processing is straightforward - developer should provide corresponding XSL page.
- All request-response mapping is stored in a single XML file called workflow description.

# Access to the system

- To start working with Clusterweb user should supply a login name and a password. Clusterweb has a stand-alone access list, which is checked for existence of the provided user name and password.



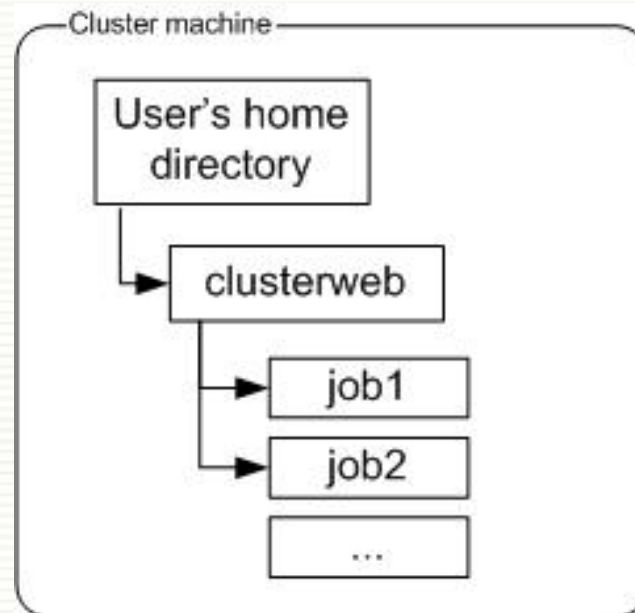
The image shows a login interface for the Cluster Management System. At the top, the text "CLUSTER MANAGEMENT SYSTEM" is displayed in a bold, sans-serif font. To the left of this text is a cartoon penguin character with a black body, white belly, and yellow feet. Below the penguin and text, there are two input fields: "User name" and "Password". The "User name" field is a simple rectangular box, while the "Password" field is a rectangular box with a horizontal line through it, indicating it is a password field. To the right of the "Password" field is a "Login" button. Below the input fields, there are two links: "Register new user" on the left and "Help" on the right.

# Interaction with cluster management system

- After successful authentication user can start working with Clusterweb.
- When the user submits a new job, the system opens Secure Shell (*SSH*) connection to the desired cluster server and executes corresponding command line utilities.
- All necessary files are transferred with *SCP* utility.
- This approach requires that the user under which Tomcat server is run should have public key *SSH* access to the cluster machine with the desired user id. Hence, Tomcat *SSH* public identity key should be copied to the corresponding user's *authorized\_keys* file on the cluster machine.

# Working directory

- When Clusterweb does a secure shell login into the user's system account it creates working directory named *clusterweb*.
- Each uploaded job is given a subdirectory inside this directory.
- When the job finishes, all related files are removed from the directory.



# Related work

- Monitoring
  - Ganglia Toolkit
  - ClusterWorX 2.0
  - Cluster Management System from Streamline Computing
- Job submission and management
  - PBSWeb

# Current work

- Testing
- Administrator interface (only Clusterweb)
- User settings management
- Interface improvements

# Future work

- Packaging and deployment tools
- Integration with external systems (Grid)
- Administrator interface (communication with cluster management system)