ICAT Job Portal: a generic job submission system built on a scientific data catalog

Stephen M Fisher
Scientific Computing Department
Rutherford Appleton Laboratory
Didcot, OX11 0QX, UK
Email: dr.s.m.fisher@gmail.com

Kevin Phipps
Scientific Computing Department
Rutherford Appleton Laboratory
Didcot, OX11 0QX, UK
Email: kevin.phipps@stfc.ac.uk

Daniel J Rolfe
Central Laser Facility
Research Complex at Harwell
Rutherford Appleton Laboratory
Didcot, OX11 0QX, UK
Email: daniel.rolfe@stfc.ac.uk

***Abstract*—The value of metadata to the scientist is well known: with the right choice of metadata, data files can be selected very quickly without having to scan through huge volumes of data. The ICAT metadata catalog[1] (which is part of the ICAT project[2]) allows the scientist to store and query information about individual data files and sets of data files as well as storing provenance information. This paper explains how a generic job management system, exposed as a web portal, has been built on top of ICAT. This gives the scientist easy access to a high performance computing infrastructure without allowing the complexities of that infrastructure to impede progress.**

The aim was to build a job and data management portal capable of dealing with batch and interactive work that would be simple to use and that was based on tried and tested, scalable, and preferably open source technologies. For the team operating the portal, it needed to be generic and configurable enough so that they can, without too much effort, modify their software to run within the portal, add new software, and create new dataset types and parameters. Modifications to existing software should be limited to saving and loading their datasets in a slightly different way so that instead of just being saved to disk, they are registered within the system along with recording any provenance information.

CEUR-WS.org/Vol-993/paper6.pdf

I. Introduction

The ICAT Job Portal (IJP)[3] builds upon the tried and tested ICAT data catalog, an existing component written specifically to catalog datasets produced by scientific facilities. It uses ICAT as the central database component which also provides authorization via a flexible rules based system. This means that users will only be shown datasets readable by them, and any datasets produced whilst using the Job Portal will also be protected by relevant permissions.

While developing a prototype portal to meet the needs of one group it became apparent that it could be made generic and configurable enough to be used by a wide range of teams within the scientific community.

A. ICAT the metadata catalog

ICAT is a data catalog specifically aimed at scientific facilities into which data are stored based on the following hierarchy of entities: Facility, Investigation, Dataset and Datafile. The "Facility" produces the data for a group of users associated with an "Investigation". Within the investigation "Datafiles" are grouped into "Datasets".

Each entity has a small agreed set of attributes. To make the system extensible, parameter types can be defined and associated with one or more of the entity types. Actual parameters of those types can then be associated with the corresponding entities. For example a parameter type could be defined for current measured in milliamps or elapsed time measured in seconds.

Further entities Application, Job, InputDataset and OutputDataset allow the provenance of datasets to be stored within the catalog, such that it is possible to trace the derived dataset back through a chain of applications and intermediate datasets to the original raw dataset.

ICAT is implemented as a SOAP based web service using the mechanisms provided by the Java Persistence Architecture (JPA) to connect to a relational database. ICAT has rule based authorization and a powerful query language which is translated into the JPA query language (JPQL).

The data files are not stored within ICAT itself, but are stored within an ICAT Data Service (IDS)[4] as explained below.

B. IDS the ICAT Data Service

This is a component, defined by its interface which is able to store files and register their metadata in ICAT. It makes use of ICAT for authorization. If ICAT allows the file metadata to be written then the IDS will allow the file to be written. Control of who can read follows the same pattern.

II. Background

A. Use Case

This work was motivated by a request from the Lasers for Science Facility (LSF) of the UK Science and Technology Facilities Council (STFC) to help them with their data. The LSF operates the OCTOPUS imaging cluster[5], a central core of lasers coupled to a set of advanced interconnected microscopy stations that can be used to image samples from single molecules to whole cells and tissues. They had accumulated a large number of data files stored in a directory structure. They had both a range of applications to process and visualise that data[6] and an interactive program with an easy to use GUI that would scan through a selected part of the file system to collect information in memory about their data then offer lists of raw datasets and lists of processed datasets

and offer the ability to process those datasets with a fixed set of interactive jobs. The main problem with this solution was that it was not scalable; the user had to restrict himself[1] to a small part of the available data each time the GUI was launched as the program had to scan the data afresh each time it was started which took time proportional to the volume of data. In addition the user needed a machine allocated to him with a personal account on that machine to allow him to run his work. This machine was hidden from off site users by a firewall requiring his presence on site or the use of a VPN. Relieving the bottlenecks of data, job and user management would enable a significant improvement to the user experience and enable more effective exploitation of the OCTOPUS facility.

After development of a prototype solution it was realised that there was a need for a generic solution so that LSF could quickly and easily add new dataset types and job types without needing to go back to the developers to make coding changes. Our funders also favoured a generic solution that could be deployed for other facilities which led to formulating a set of requirements some of which are listed in the next section.

B. Requirements

Following analysis of the prototype the requirements were refined. Some of the key requirements are listed below.

1) System accessible via both GUI and command line from on and off site.
2) All the systems should have automated installation of OS and software updates.
3) Centralised user/group management.
4) A file server must be able to store raw data from microscopes, analysed data and other user data. All data must be backed up and "old" data migrated with an easy mechanism to restore it when needed.
5) All data should be managed with a single point to consult the metadata to find out what is where.
6) Ability to upload and download data.
7) The ability to submit batch jobs to a set of Linux nodes, some with CUDA GPU capability. Listing, cancelling and retrieving output from jobs must also be supported.
8) The ability to run interactive GUI based analysis/visualisation jobs able to access data.
9) Select and submit multiple datasets for processing through applications. This must cover both multiple jobs with one dataset per job, or a job which will process all selected datasets.
10) Any menus must be configurable, as must the types of datasets that can be stored, jobs that can be run and job parameters associated with a job type.

C. Possible solutions

Consideration was given to OMERO[7]; however this is more suited to viewing and performing simple analysis of images rather than the specialised analysis codes developed by LSF and it it does not meet requirement 10.

[1]Gender specific terminology should be interpreted as non-gender specific throughout this paper.

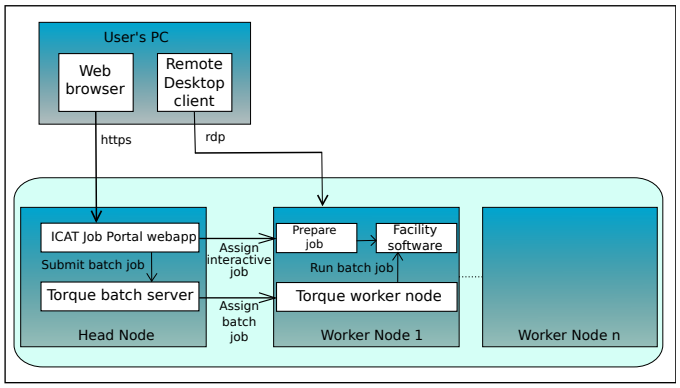

Fig. 1. Architecture overview

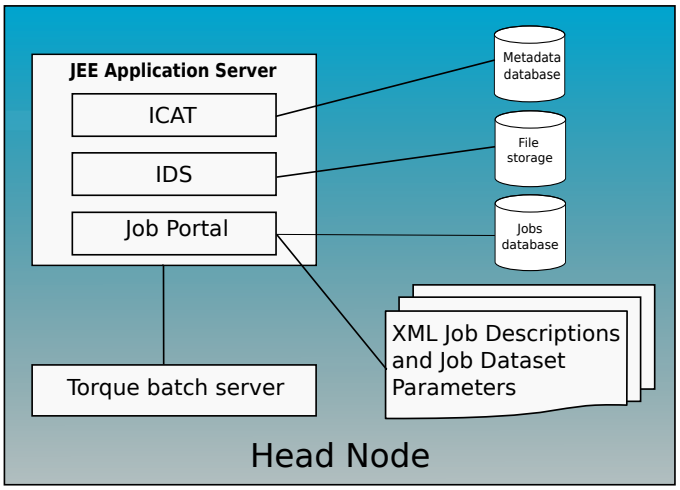

Fig. 2. The head node

IBM's Platform Application Center[8] provides a means to describe jobs in XML and submit them however though it does meet requirement 10 it fails 4 and 5.

The Galaxy portal[9] is quite close to meeting our requirements and also provides workflow support. Its main drawback is that it describes itself as a *genomics workbench* and as such is too focused on one discipline. This paper also contains interesting comparisons with other genomics workbenches.

As we have a good metadata catalog: ICAT, and a matched data service: the IDS, we decided to build directly on those components.

III. System Architecture

The architecture shown in Fig. 1 is based around a single head node acting as a central point for all communications and an extensible number of worker nodes which can be added to in the future in order to increase the job handling capacity of the system.

The head node which is shown in more detail in Fig. 2 hosts an application server (currently Glassfish) running the Job Portal, ICAT and IDS software and acts as the head node for a batch system (currently Torque[10]).

Worker nodes have this role within the batch system but may also be assigned temporarily to a user for interactive work.

They should be capable of running all the facility software that users require and they are able to communicate with ICAT for metadata and with the IDS for data, both of which run on the head node.

A. Batch jobs

It is essential that a batch job belonging to one user cannot access the account of any other user. To achieve this a batch job is submitted to run on an account chosen randomly from a pool. Each worker node is configured to run a very small number of concurrent jobs. The job has a *prologue* which is run before the user's job. This tries to get a lock by creating a directory to ensure that two jobs cannot run simultaneously under the same account. If it fails it will issue a return code that causes the job to be requeued. The *epilogue*, which is run after the job, frees the lock if it is run by the same job that created it. The batch pool should be sufficiently large that requeuing is rare. There is a mechanism to tidy up if things go wrong.

B. Interactive Jobs

Although most batch systems do have some kind of interactive job capability we found it convenient to provide the desired functionality outside the batch system. For these jobs, the most lightly loaded worker node is found, any running batch jobs are suspended, the node is made temporarily unavailable for new batch jobs, and the user is given exclusive use of that node to run the interactive job. To achieve this, the user is supplied with a username taken from a pool and a temporary password, allowing a remote desktop connection to be established via the RDP protocol to the worker node. This will typically be either via the Remote Desktop Connection application in Windows or using the *rdesktop* command on Linux systems. The account will have been configured such that the interactive job that the user has requested will start automatically. The user is only given a short time to connect to the worker node machine before the password is removed. Once the user has logged out the system will remove the account, along with any local files that may be left, and will release any suspended jobs that were on the machine and make the machine available to the batch system again.

C. Ganglia monitoring

All nodes within the system are configured to make use of the Ganglia Monitoring System. Currently this is being used to select the most lightly loaded machine in the cluster when an interactive job is requested. It allows a single XML stream from the Ganglia host on the head node to be parsed, giving an instant overview of the loading of each machine. Nagios monitoring is also installed but it is not an essential part of the system.

D. Job Status Information

The batch system is not well suited for holding job status information for an extended period. In addition the portal needs to hold information about jobs that are not known to the batch system. Therefore the portal maintains its own records and periodically harvests information from the batch system.

E. Command Line Interface

With the addition of a RESTful web service on the server, a Python client has been provided to allow interaction with the Job Portal via a command line interface. Both of these are very thin layers totalling only a few hundred lines of code. This provides an alternative to the GUI interface which may prove to be the preferred way for more proficient users to interact with the portal, and would be the interface of choice for anyone looking to write a script to handle their data processing.

F. Automated Configuration

The installation, configuration and upgrading of the software has been set up using the Puppet Open Source[11] framework. This means that, starting with computers with an operating system and configured to use the network, it is possible to install the head node within an hour and each worker node can be added in a few minutes. The result is a working system including the Java Development Kit, a Glassfish Application Server (running ICAT, IDS and the Job Portal software), database servers and required databases, batch system, monitoring (Ganglia and Nagios) and the scientific software provided by the team operating the portal.

IV. Creation and Use of Metadata

The use of metadata is essential to the operation of the IJP. Because it is a generic tool, the portal itself is not able to look inside domain specific datasets. It is entirely reliant on the metadata inserted into the ICAT database, and uses *only* this metadata for searching and displaying information.

When an instrument produces data this is typically written to a local file store from which they can be ingested into the IJP system. The best people to define the metadata to associate with this raw data are the team conducting the experiment. An IJP job can be submitted each time that data need to be ingested. This job must be able to derive the metadata from the available information and upload the data files themselves to the IDS as well as creating entries in the ICAT database for the metadata.

When data are processed by an IJP job this results in new data and metadata being stored. It is the responsibility of the job to identify useful pieces of metadata to allow datasets to be subsequently selected. As it is difficult to identify all the metadata that might eventually be useful, jobs can be written to look at the data and add metadata to ICAT to hold more information about existing datasets.

The three categories of job described here: ingestion, derivation of processed data and augmentation of metadata are all just jobs for the IJP and must be installed by the facility for its users.

V. A User's View of the Portal

Users access the job portal via a web browser as shown in Fig. 3. This was developed in Java using the Google Web Toolkit[12] and communicates with a number of servlets running on the application server on the head node. Once logged in, the user is presented with a number of search options tailored to the user base of the portal, and a generic search widget listing all of the dataset parameters that are

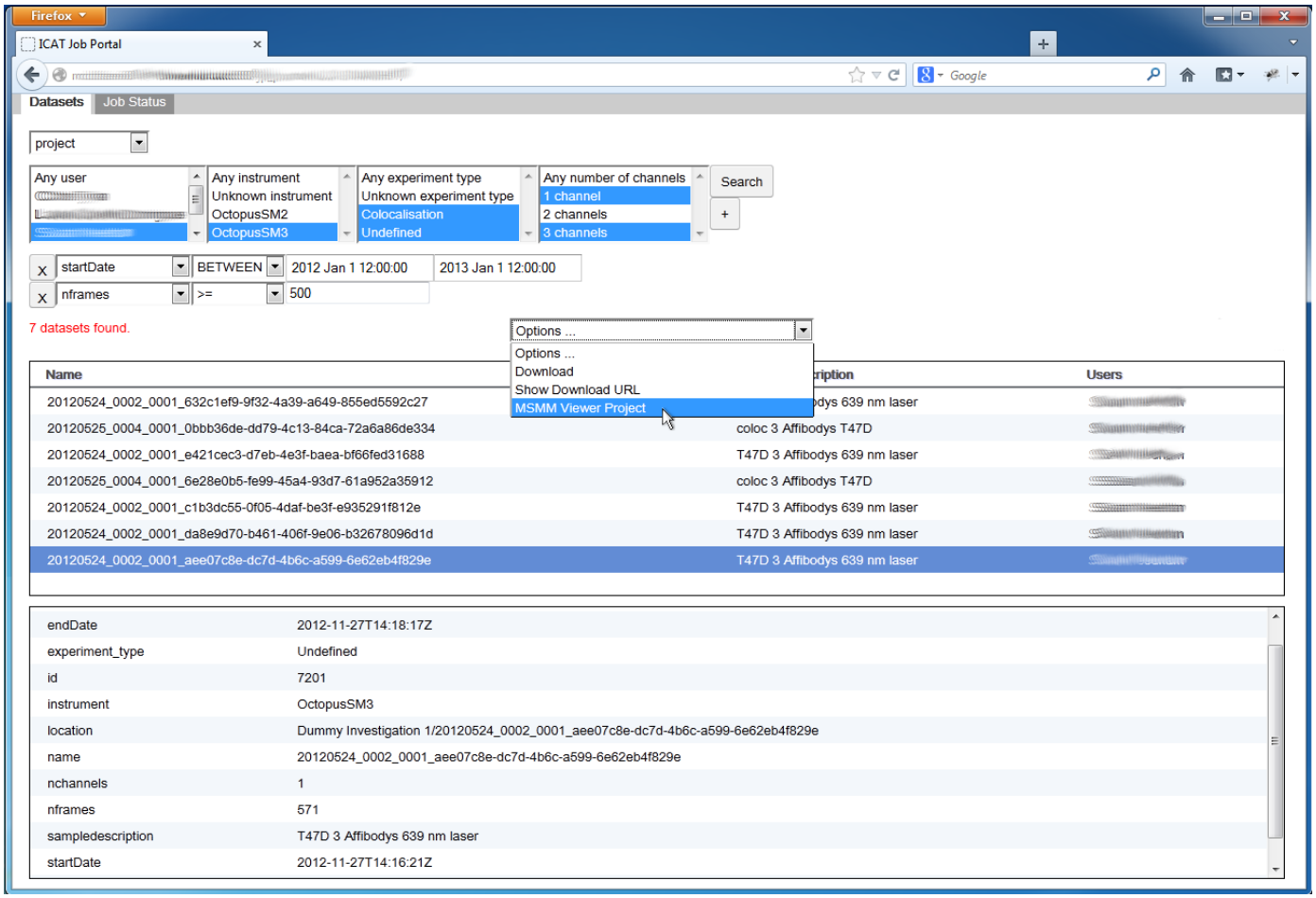

Fig. 3. Screenshot of IJP

searchable. The widget provides relevant search options for each parameter: =, !=, >, >=, <, <=, LIKE and BETWEEN depending on its type - string, numeric or date/time. The list of parameters and their types is read from the underlying ICAT database so that the portal software remains generic.

Within ICAT all datasets have to be of a type which has been pre-defined before the dataset is registered. This allows for easier searching of datasets. Once a user has selected the type of dataset in which they are interested, they can narrow down their search if they wish using the search options, then click search. A list of matching datasets then appears in the central panel. When one of these datasets is selected, all of its dataset parameters are displayed in the lower panel. Having selected a dataset, the central Options select box lists all of the jobs that it is possible to run on that dataset type. After selecting the desired job, a Job Options Form is displayed allowing the user to pass particular parameters to the job, if required. This form is automatically generated from an XML file defining the job within the system, as shown in Fig. 4. The options displayed can also be tailored so that only options relevant to the chosen dataset are offered.

Submitting the form results in the job being submitted to the server and the user receiving a response containing the ID assigned to the job in the batch system. The user can then use the Job Status tab to follow the progress of the job through the batch system, checking the output and error logs if required and monitoring the status until the job is complete.

As well as handling interactive and batch jobs, the portal is able to handle jobs that take either a single dataset or multiple datasets as input. Users are able to select multiple datasets and the portal uses the job definition to work out whether to submit multiple jobs each with a single dataset as input, or a single job with multiple datasets as the input. Where it is ambiguous, the user is asked to confirm what was intended.

Datasets remain registered within ICAT and available via the IDS. They are suitably protected via a rule based permission system which should have been configured to ensure that users can at least read the data they have created. These data will remain within the system. Should the user wish to download a copy of their data, this is possible via “Download” in the Options select box. There is also an option to display a URL to obtain the dataset from the IDS.

VI. An Administrator’s View of the Portal

Configuration of the portal is defined by XML files. Each team using the portal to run their software needs to have at least one person who is familiar enough with the team’s software and the datasets it uses, to be able to set up each piece of software so that it can be run as a “job” by the portal. Firstly,

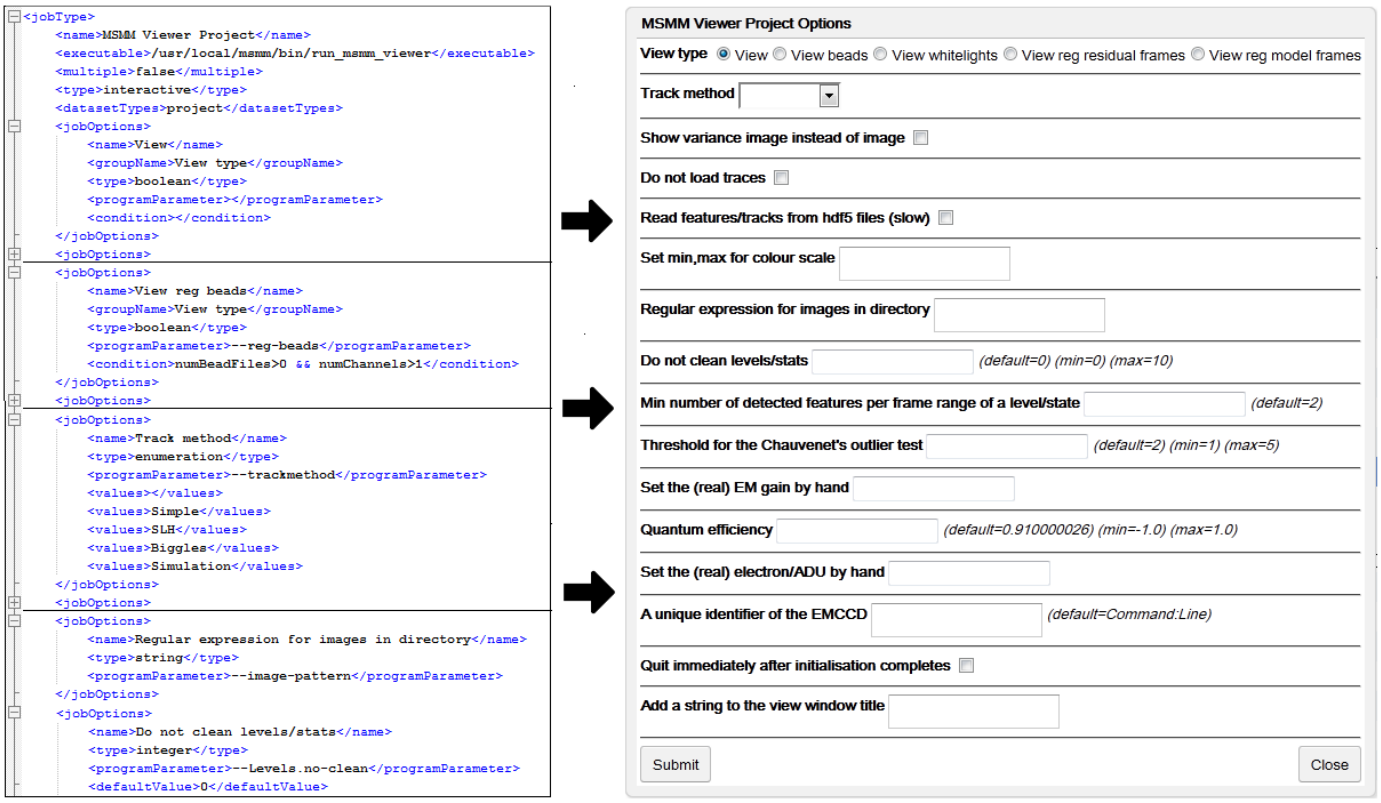

Fig. 4. Configuration of job options

there are two fairly straightforward tasks which need carrying out:

- picking out the characteristics of each dataset type which lead to different options being made available in the Job Options Form.
- creating an XML file describing each piece of software: whether it runs as an interactive or batch job, which type of datasets it needs, whether it accepts multiple input datasets, along with all of the various command line options that it accepts.

These two tasks are linked by the concept of Job Dataset Parameters. For each type of dataset, an XML file is set up allowing the administrator to define a named quantity and how it may be derived from an ICAT query. While the query can span all information the logged in user is allowed to see, a query might reasonably take into account information from the metadata associated with the dataset or any of its files and might make use of JPQL aggregate functions SUM, AVG, MIN, MAX and COUNT.

The administrator has thus defined named quantities specific to a dataset type and derivable by an ICAT query; examples include: the number of files of a particular type or the size of the largest file in a dataset. When a dataset is selected within the browser, the server runs the database queries specified within the relevant XML file, generates a map of name-value pairs and sends it back to the browser to control what appears in the Job Options Form.

Within the XML specifying each of the command line options for a job, as shown in Fig. 4, a condition can be specified in terms of the named quantities defined in the XML which if met, causes this option to appear on the Job Options Form. This takes the form of a logical expression such as `numChannels == 3 && numHdfFiles > 500`. If multiple datasets are selected, only the options that are common to all of those datasets are offered to the user.

In addition to setting up these XML descriptor files, there is a certain amount of work that needs to be done in order to make the team's existing software compatible with the job portal. This can be done either by modifying the existing applications or by providing job wrappers to perform tasks such as obtaining data from the IDS and laying it out as the program expects, storing resulting datasets back in the IDS and recording provenance information. Python libraries are being established to simplify these operations - there is a generic library and we recommend using a facility specific library that knows the facility conventions for layout of data.

VII. Current Status

Having developed a prototype to prove the concept and help the users to define the features that they need from the IJP, we are currently completing the work and plan to have a first deployment for production use in a few months time.

VIII. Future Developments

We anticipate that requirements will be clarified further once we receive feedback from users of the deployed pro-

duction system. Based on existing feedback we are already planning the following enhancements.

A. Visualisation of Provenance

Provenance information is stored within ICAT when a new dataset is stored but there is currently no way to visualise this information within the portal. A new panel will be added to represent the provenance information in a graphical format. This will allow the user to select the dataset they are interested in and expand it to see the input and output datasets and files associated with it. Those datasets and files can, in turn, be selected and expanded to follow the chain of provenance.

A further development would be the addition of a provenance based search facility which would allow searches such as all datasets derived from a given dataset or all datasets produced directly or indirectly by a specific version of an application.

B. Workflow Support

It would be a particularly useful feature to have the Job Portal integrated with a Workflow Management System. This would make it possible to set off a chain of data processing jobs with the output of the first job becoming the input to later jobs, and so on. As the job relating to each stage of the process completes, the next job in the workflow is automatically submitted on behalf of the user.

One workflow management system which would be of particular interest would be Taverna[13]. It is open source, domain independent and written in Java, and therefore should integrate well with the server side of the portal software which is also written in Java. Taverna has already been used behind a portal[14] by a number of projects which demonstrates its suitability to being used in this way.

C. Software as Data

Initially our preferred solution for deploying facility software was to use the native packaging system of the operating system - typically RPMs or DEBs. While convenient for the IJP developers this may not meet the needs of some of our users who would like more freedom to run the software of their own choosing without arranging to have it officially installed and who want to have multiple versions of the software available. Following a suggestion[15], we are considering the implications of storing a job as a dataset known to ICAT. A job wrapper would then first download the application software before setting up the data and running the downloaded application. This would probably require some kind of caching mechanism and would require a means of specifying software dependencies to ensure that the correct packages are available for the desired software. This solution, which will require some operations to be run as root to install dependencies, needs careful evaluation.

D. Administration console

The system is currently rather opaque to the administrator and requires the use of the native batch system commands to find out what is going on. We plan to provide a browser based web application allowing administrators to monitor and control the system and the jobs it is running. This will support common tasks such as monitoring job distribution and loading on the worker nodes, pausing and terminating jobs, taking worker nodes offline and bringing them back online, user and group administration, modification of authorization rules and removal of unwanted datasets.

E. Alternative remote desktop mechanism

A possible alternative to using either Remote Desktop Connection in Windows or rdesktop on a Linux system is to have the remote desktop session also run within the browser. Currently, the RDP server port needs to be accessible on each of the worker nodes which is not a problem within the local site network. The system is, however, intended to be used remotely from other institutions, which may contravene security policies. Having the possibility of running the Remote Desktop session via https within a browser may be the solution.

One solution of interest to solve this problem is Guacamole[16], an HTML5 client-less remote desktop. It supports remote desktop protocols such as VNC and RDP, and is able to deliver a remote desktop within a web browser without the need for any browser plugins or client software installation.

F. Alternative batch system

We only support Torque as a batch system. We plan to include Maui as a scheduler because the inbuilt Torque scheduler (pbs_sched) is very basic. Maui would enable scheduling policies to be defined to allow more control of which job is selected to be run when a slot becomes free.

We also plan to make the choice of batch system configurable. The batch system might even act as a front-end to a grid or cloud solution. We already have a request to support IBM Platform LSF[17].

G. Portability

The Puppet configuration is only available for Ubuntu[18] and has only been tested on version 12.04 (64 bit). This is a concern for existing infrastructures which are not able to easily accommodate these decisions. We plan to make the system easy to install on other platforms and to support alternative subcomponents where practical.

We have a request to support Red Hat Enterprise Linux[19] version 6 (64 bit) and will probably include CentOS[20] version 6.4 (64 bit) at the same time.

IX. Conclusion

We have successfully built a job portal for ICAT users on top of the basic metadata catalog and the IDS. The initial prototype was very valuable as it allowed us to get something out quickly to ensure that we were on the right track and to understand what needed generalising.

Though the generalisation was not a trivial task; the result is a tool that we believe is now very easy to configure for many scientific disciplines.

The IJP allows rapidly changing, mature and wrapped "legacy" software to be made available, side by side, with

a uniform and modern style of interface to a scientific community.

We already have a number of groups from the existing ICAT community interested in the project and we anticipate a good uptake of the software.

ACKNOWLEDGMENT

The authors would like to thank Dave Clarke from STFC's Lasers for Science Facility for supporting this work and attracting funding.

We would like to acknowledge the assistance and funding from STFC's Harwell Imaging Partnership, which has supported this development from inception (http://www.stfc.ac.uk/hip).

The diagrams in this paper were produced by Noris Nyamekye.

Finally we thank our colleagues Brian Mathews, Alistair Mills and Erica Yang who all provided helpful comments on drafts of this paper.

REFERENCES

[1] The ICAT Metadata Catalog website. [Online]. Available: http://code.google.com/p/icatproject/

[2] The ICAT project website. [Online]. Available: http://www.icatproject.org/

[3] The ICAT Job Portal website. [Online]. Available: http://code.google.com/p/icat-job-portal/

[4] The ICAT Data Service website. [Online]. Available: http://code.google.com/p/icat-data-service/

[5] D. T. Clarke, S. W. Botchway, B. C. Coles, S. R. Needham, S. K. Roberts, D. J. Rolfe, C. J. Tynan, A. D. Ward, S. E. D. Webb, R. Yadav, L. Zanetti-Domingues, and M. L. Martin-Fernandez, "Optics clustered to output unique solutions: A multi-laser facility for combined single molecule and ensemble microscopy," *Review of Scientific Instruments*, vol. 82, no. 9, p. 093705, 2011. [Online]. Available: http://link.aip.org/link/?RSI/82/093705/1

[6] D. Rolfe, C. McLachlan, M. Hirsch, S. Needham, C. Tynan, S. Webb, M. Martin-Fernandez, and M. Hobson, "Automated multidimensional single molecule fluorescence microscopy feature detection and tracking," *European Biophysics Journal*, vol. 40, no. 10, pp. 1167–1186, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00249-011-0747-7

[7] The OMERO website. [Online]. Available: http://www.openmicroscopy.org/site/products/omero

[8] The IBM Platform Application Center. [Online]. Available: http://www.ibm.com/support/entry/portal/documentation_expanded_list/software/platform_computing/platform_application_center

[9] J. Goecks, A. Nekrutenko, J. Taylor, and The Galaxy Team, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome Biology*, vol. 11, no. 8, p. R86, 2010. [Online]. Available: http://genomebiology.com/2010/11/8/R86

[10] Adaptive Computing's Torque website. [Online]. Available: http://www.adaptivecomputing.com/products/open-source/torque/

[11] The Puppet Open Source website. [Online]. Available: https://puppetlabs.com/puppet/puppet-open-source/

[12] The Google Web Toolkit website. [Online]. Available: https://developers.google.com/web-toolkit/

[13] P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble, "Taverna, reloaded," in *SSDBM 2010*, M. Gertz, T. Hey, and B. Ludaescher, Eds., Heidelberg, Germany, June 2010. [Online]. Available: http://www.taverna.org.uk/pages/wp-content/uploads/2010/04/T2Architecture.pdf

[14] Taverna: Behind a portal. [Online]. Available: http://prototype.taverna.org.uk/introduction/taverna-in-use/portal/

[15] Rich Wareham, Cambridge, private communication, 2012.

[16] The Guacamole website. [Online]. Available: http://guac-dev.org/

[17] IBM Platform LSF. [Online]. Available: http://www.ibm.com/systems/technicalcomputing/platformcomputing/products/lsf/index.html

[18] The Ubuntu website. [Online]. Available: http://www.ubuntu.com/

[19] Red hat enterprise linux. [Online]. Available: http://www.redhat.com/products/enterprise-linux/

[20] CentOS: The Community ENTerprise Operating System. [Online]. Available: http://www.centos.org/