An Open Source System for Life Sciences Data Management and Integration

Can Türker, Fuat Akal, and Dieter Joho

Functional Genomics Center Zurich, Winterthurerstr 190, 8057 Zurich, Switzerland
<turker|joho>@fgcz.ethz.ch, akal@acm.org

Abstract. The advances in information technology boosted life sciences research towards systems biology which aims at studying complex interactions in biological systems in an integrative way. Steadily improving high-throughput instruments, such as genome sequencers and mass spectrometers, and analysis software produce tremendous amounts of life sciences data. In this paper, we describe a life sciences data management system that we have implemented and deployed at the Functional Genomics Center Zurich. We highlight some crucial features and share some experiences we made in developing and running this system.

1 Introduction

Life sciences research aims at characterizing complex biological organisms and functions at the systems level. To achieve this ambitious goal, data produced in different research projects and groups must be linked together. As the complexity of the analytical systems in integrated functional genomics or systems biology research has reached a level where specific, isolated application-oriented data management and analysis has become apparently inefficient, a system for integrated management of experimental data and scientific annotations is needed. Such a system shall capture and integrate all generated scientific data in a way that third parties can correctly interpret, reproduce, and reuse the data.

In [18] we motivated the need for such a system in more detail and introduced *B-Fabric*, an open source life sciences data management system implemented and deployed at the Functional Genomics Center Zurich (FGCZ). Meanwhile B-Fabric is running for two years at FGCZ and is continuously extended with new features. Basically, B-Fabric allows to store and annotate all data produced at FGCZ. Any external data store can be attached to and made accessible via B-Fabric. Users do not need to care about where and how the data is placed. B-Fabric functions as data fabric capturing and providing the data transparently and in access-controlled fashion through a Web portal. Using its search and browse features, inter-experiment and inter-project analysis becomes possible. Since experimental data is captured together with instrument parameters, experiments become reproducible for third parties. Another core feature of B-Fabric is the dynamic coupling of external applications. Once an application is registered with B-Fabric, users may invoke and feed the application via B-Fabric.

We are aware of several systems with similar targets, even commercial ones such as *BioPoint* [5] and *CapitalBio* [6]. The goal of this short paper is to share some potentially useful experiences we made with implementing and running B-Fabric. Section 2 gives a quick overview of the B-Fabric architecture and

sketches roughly how data and application integration works in B-Fabric. Section 3 discusses some interesting experiences with B-Fabric and addresses some shortcomings of the current system. Section 4 provides an outlook.

2 System Architecture of B-Fabric

B-Fabric is a life sciences data management system composed of distributed, loosely-coupled components based on open source technologies. The system is dynamically extendable by allowing the coupling of external components at runtime. Figure 1 gives an overview of the B-Fabric architecture.

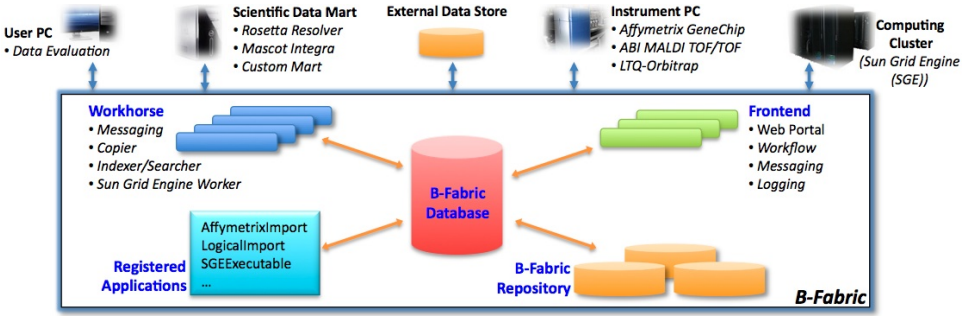

Fig. 1. B-Fabric System Architecture

The *B-Fabric Database* stores and manages all scientific annotations and base administrative data, e.g. about projects and users. Currently, *PostgresSQL* is deployed for this purpose. As B-Fabric considers databases as black-box components, any SQL database system could be used instead, for which *Apache OJB* [4] supports an object-relational mapping from Java to SQL. The *B-Fabric Repository* stores experimental (raw and processed) data. It is a *RAID-6*-based storage that provides high reliability. This repository is fully encapsulated, i.e., accessible only through B-Fabric. Users interact with B-Fabric through *Frontends* using a Web browser. A frontend acts as Web portal providing users controlled access to B-Fabric data. It is implemented with *Apache Cocoon* [2]. *Workhorses* are used to execute specific tasks, such as data copying, indexing, and searching. All relevant data is indexed by *Apache Lucene* [3], i.e., is findable via fulltext search. There could be several frontends and workhorses deployed in the system in order to improve the availability and scalability. Frontends and workhorses communicate with each other via *Apache ActiveMQ* [1], which provides persistent message queues for asynchronous communication. With the concept of *Registered Applications*, a run-time integration of arbitrary applications into B-Fabric is possible. Using the registration profiles, B-Fabric dynamically extends the frontends with appropriate buttons to invoke and feed these applications with B-Fabric data. These applications may be external ones which run autonomously beyond the control of B-Fabric. Applications can also be invoked within workflows to model scientific pipelines. B-Fabric uses *OSWorkflow* [13] as workflow engine.

There are also several external components that interact with B-Fabric. *User PCs* are standard computers running a Web browser to enable access to B-Fabric

through frontends. Typically, a scientist interacts with B-Fabric to search for and download data for analysis reasons. Various data analysis and/or data visualization tools can be deployed on these PCs to accomplish those purposes. *Scientific Data Marts* correspond to external systems that provide scientific data management, analysis, and/or visualization functionality. Currently, B-Fabric uses *Rosetta Resolver* [14] and *Mascot Integra* [11] as marts for the detailed management and analysis of transcriptomics and proteomics experiments, respectively. For some instruments, however, these data marts are not suitable. In such cases, B-Fabric implements a custom data mart to handle this data. *External Data Stores* represent life sciences data available on external systems that are accessible by B-Fabric. By configuring these data stores, their data can be made available through B-Fabric by simply maintaining links to them. As external data stores are autonomous, the corresponding external system is responsible for ensuring consistency of that data, i.e., it must guarantee that linked data is not deleted. *Instrument PCs* refer to computers that are attached to an instrument that generates and holds scientific data to be imported into B-Fabric. To support computationally and data-intensive analysis, B-Fabric can also execute applications on a *Computing Cluster/Grid*.

Figure 2 shows the integration of some external components in action by exemplifying three most frequently occurring application scenarios. The arrows depict the sequential flow of interaction between a specific scientist and B-Fabric.

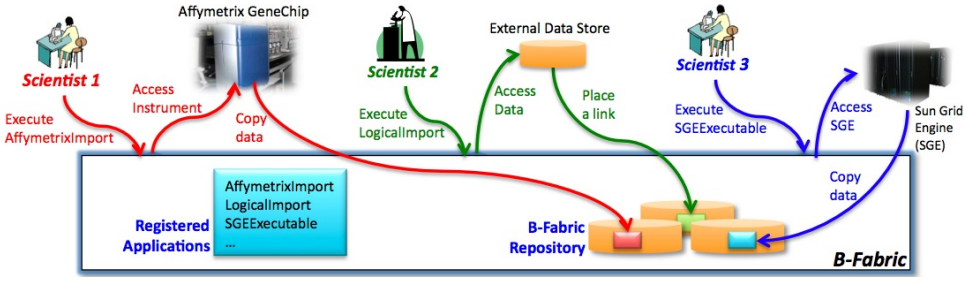

Fig. 2. Application and Data Integration

The first scenario shows how data imports are done from instruments to B-Fabric. `Scientist 1` wants to import his data generated by the *Affymetrix GeneChip* instrument. Using B-Fabric this scientist invokes a previously registered application, called *AffymetrixImport*. This application accesses the instrument and copies the data into the B-Fabric repository (*physical import*), where it will be under total control of B-Fabric. Note that when coupling an instrument to B-Fabric a corresponding import application has to be registered, too. The second scenario illustrates the case where some external data is made available via B-Fabric. To do this, `Scientist 2` executes the *LogicalImport* application. The rest is assured by this application which creates a link in B-Fabric to the data at the external data store (*logical import*). The last scenario is an example where the user has higher computing needs. `Scientist 3` asks B-Fabric to execute his job in the Grid. To meet this demand in our case, B-Fabric accesses the *Sun Grid Engine* running on a compute cluster outside of B-Fabric. The job is executed by the cluster and the results are sent back to B-Fabric.

3 Experiences with B-Fabric

Data modeling and vocabulary. Many data integration projects failed simply due to the ambitious goal to provide an integrated data scheme that can capture all potentially created data in the application domain at a detailed level of granularity. Already at the beginning of the B-Fabric project, it became clear that in a very dynamic research environment as the FGCZ it is practically impossible to agree on a scheme. Sticking one-to-one with standards such as MIAME [12] or GO [7] was however not a solution. First, there are many researchers arguing that they are doing research on issues that have not been defined yet. Second, depending on the concrete area of the scientists they use different notions and granularities to describe their scientific data. Third, these standards are so huge in size that for most scientist they are impractical for daily use. Therefore, it is vital to have a simple but exact vocabulary. It is a balancing act to use terms that are as precise as needed, but not precise enough that users cannot find them easily within an overwhelming set of terms. B-Fabric hence concentrates on a very small data scheme describing only the metadata of central entities such as samples and extracts. After many discussions with our researchers at FGCZ we came to the conclusion that only a small set of commonly agreeable attributes should be fixed for such entities and that the vocabulary defined by the different standards should be initially restricted to that part which is potentially needed for the research technologies supported at FGCZ. To be open to new research directions, B-Fabric allows to dynamically extend the vocabularies at run-time. As soon as a new entry is provided by a user, a reviewing process is started. Depending on the project membership of that user, the coach of that project, in our case an FGCZ employee, is triggered to release or reject the new entry.

Application integration and loose coupling Most facilities like the FGCZ have applications and workflows for data processing that are now running stable for years. Moving such functionality into a central system can become a complicated task for several reasons. First, the original functionality is provided by proprietary tools whose internal processing is not known. Second, the scientists and developers that created and installed the application often are no longer available. Third, replacing existing code eats up resources without providing new benefits for the users. These were the reasons why we started implementing connectors to different external applications. However, writing such a connector for every application that had to be connected to B-Fabric was quite laborious. Needs and requests for new applications popped up faster than the implementation could progress. This was the reason why we introduced the *Application Registration* concept in B-Fabric. First, a connector is written for a certain type of external application, e.g. for running *R* scripts [15] on an *Rserve* system [16]. Then, a minimal interface is defined to describe how the external application gets its input. Finally, the scientist writes the external application in any language. The advantage of this approach is that an upgrade or even a replacement of components of an external application is possible without touching the B-Fabric system. Defining the interface towards the external application was quite tricky. Allowing too much flexibility may result in errors since the user usually is not aware of the restrictions of the corresponding external application. Additionally, the heterogenous data that is stored in B-Fabric provides a lot of traps.

External responsibility. Although B-Fabric knows the scientific annotations best, it may not be able to interpret and process the experimental data correctly if the instruments produce data in proprietary formats. This problem is solved by moving the responsibility of the data processing to the external application. B-Fabric provides the application input via a minimal interface. The external application is then able to fetch the information to process the raw data correctly. B-Fabric then just has to wait for the result. This approach requires that B-Fabric trusts the external application to provide the result in a correct way. The external applications need to take additional effort to process the data. Our experiences show that examples and documentation on the development of connectors are vital to support stable and generally accepted applications. However, if a certain level of commitment is achieved, additional functionality can be provided quickly and without changing the B-Fabric interface.

Transparency. To consistently link together experimental data with scientific annotations, the data must be provided in a reliable environment. B-Fabric provides these services on its own (B-Fabric Repository, B-Fabric Database). However, as most facilities do already have such an infrastructure on their own and duplicating data can get quite expensive, B-Fabric allows to place links to resources located on external data stores. In this way, existing data can virtually be integrated into B-Fabric without additional costs in terms of infrastructure. It is even better than just referencing to external data since B-Fabric provides a transparent access to the data, i.e., a user can use the same functionality whether the data is located in the B-Fabric repository or somewhere else. This is achieved by handling files and other data resources through URIs. Each resource knows where its data is located (host and path) and also how to get it (protocol).

Data Import. To annotate scientific data, the system must first be aware of the corresponding files. In B-Fabric, this is supported by importing or linking data from the instrument PCs. Pre-configured data providers then take care of the original location of the scientific data on the instrument PC. These data providers support transparently many protocols like *SSH* [10], *SMB* [8], *JDBC* [9] etc. and therefore provide simple and unified data handling. This allows a user to import his data without having any further knowledge about the infrastructure. Currently the data providers are Spring [17] configured components. This was an easy and straight-forward way to implement them, but this approach has the drawback that it is not possible to add or modify such providers at run-time. By changing data providers into hot-pluggable components, B-Fabric could allow to integrate new instruments without restarting the system.

Data Export. Regardless of the systems functionality, there are always users with needs that go further than the system can support. The best solution to address this issue is to provide an easy access to the data. Scientists should be able to export data and do their calculations on their own. B-Fabric provides several methods for data export. For instance, search results can be exported. Another example is the download of data resources, i.e., files. Due to the huge size of the interconnected data, B-Fabric compresses the data and provides a download link for the user. The download of predefined reports is another example. For specific needs and applications, B-Fabric allows reports that collect and prepare data for later use. If it turns out that scientists use a certain type of export very

often to perform a specific task, this task becomes a candidate for a B-Fabric internal functionality.

Data Access Control. At FGCZ data access is controlled at project level, i.e., project state and membership define access to all the data within a project. This is a good approach in environments where we have small projects with clear and exact aims. If there are huge projects that run for many years, this approach has some disadvantages. Especially if there are scientists from different facilities working within the same project, but should not be able to access each others results. As a simple effective solution is to divide such projects into several smaller projects according to their goals and members.

4 Outlook

B-Fabric was designed according to the specific needs of the Functional Genomics Center Zurich (FGCZ), for instance, w.r.t. data access policies and data structuring. To make B-Fabric also very attractive to other research groups/centers, future releases will generalize it in different directions. First, ad-hoc coupling of external data stores will be supported. To avoid inconsistencies, the autonomous data stores have to follow basic data storage rules. Second, a more flexible permission management will be implemented. The owner of the data shall be allowed to grant and revoke access rights on different granules of his data also if the corresponding project is not published yet. Third, the data networks shall be graphically visualized in such a way that the link and correlation between the different life sciences data become reproducible.

References

1. Apache ActiveMQ. http://activemq.apache.org/
2. Apache Cocoon. http://cocoon.apache.org/
3. Apache Lucene. http://lucene.apache.org/
4. Apache Object Relational Bridge - OJB. http://db.apache.org/ojb/
5. BioPoint's Life Sciences Information Management System. http://www.turningpointsystems.com/biopoint.html
6. CapitalBio Life Science Data Management System. http://www.capitalbio.com/life_sciences/Products/Software_InformationSystem
7. Gene Ontology. http://www.geneontology.org/
8. JCIFS. http://jcifs.samba.org/
9. Java Database Connectivity - JDBC. http://java.sun.com/products/jdbc/
10. Java Secure Channel - JSch. http://www.jcraft.com/jsch/
11. Mascot Integra - Data Management for Proteomics. http://www.matrixscience.com/integra.html
12. Minimum Information About a Microarray Experiment - MIAME. http://www.mged.org/Workgroups/MIAME/miame.html
13. OpenSymphony - OSWorkflow. http://www.opensymphony.com/osworkflow/
14. Rosetta Resolver System. http://www.rosettabio.com/products/resolver/
15. R-Project. http://www.r-project.org/
16. Rserve. http://www.rforge.net/Rserve/
17. Spring Framework. http://www.springsource.org/
18. Türker, C., Stolte, E., Joho, D., Schlapbach, R.: B-Fabric: A Data and Application Integration Framework for Life Sciences Research. DILS 2007: 37-47