Designing Highly Flexible and Usable Cyberinfrastructures for Convergence

Bruce Herr, Weixia Huang, Shashikant Penumarthy, Katy Börner
School of Library and Information Science, Indiana University
Wells Library, 10th Street & Jordan Avenue
Bloomington, IN 47405, USA
[bherr|huangb|sprao|katy]@indiana.edu

ABSTRACT

Innovation and progress in most areas of science requires access to advanced computational, collaborative, data acquisition, and management services available to researchers through high-performance networks. These environments have been termed cyberinfrastructures (CI) by a National Science Foundation (NSF) blue-ribbon committee lead by Daniel Atkins (Atkins, Drogemeier, Feldman, Garcia-Molina, Klein, Messerschmitt, Messian, Ostriker and Wright 2003). Today, some CIs provide access to thousands of interlinked experts, services, federated datasets, and compute resources. They have reached a complexity that is hard if not impossible to specify, implement, and manage in a top down fashion. Also, most content providers and users of these CIs are not computer scientists. They are biologists, physicists, social scientists, information scientists, and others with deep knowledge about data and algorithms and a desperate interest to save lives, secure important technical infrastructures, discover universal laws, etc. It is argued here that CIs will not only transform the way science is conducted but also will play a major role in the diffusion of expertise, datasets, algorithms, and technologies across multiple disciplines and business sectors leading to a more integrative science. This chapter presents the results of a seven-year long quest into the development of a 'dream tool' for our research in scientometrics (Börner, Chen and Boyack 2003) and more recently, network science (Börner, Sanyal and Vespignani in press). The results are two CIs: The *Cyberinfrastructure for Information Visualization* and the *Network Workbench* that enjoy a growing national and interdisciplinary user community. Both CIs use the Cyberinfrastructure Shell (CIShell) software specification which defines interfaces between datasets and algorithms/services and provides a means to bundle them into powerful tools and (Web) services. In fact, CIShell might be our major contribution to progress in convergence. Just like Wikipedia is an 'empty shell' that empowers lay persons to share text, a CIShell implementation is an 'empty shell' that empowers user communities to plug and play, share, compare and combine datasets, algorithms, and compute resources across national and disciplinary boundaries.

Keywords

Cyberinfrastructure, OSGi, Plug-in, Data Models, Analysis, Network Science, Scientometrics, Usability, Flexibility, Extensibility.

1. Introduction

Fifty years back in time, scientists did not use any computer to conduct their research. Today, science without computation is unthinkable in almost all areas of science. Scientists use commercial packages such as Microsoft Excel, SPSS (http://www.spss.com), Matlab (http://www.mathworks.com), Adobe Photoshop, etc. to analyze and model their diverse datasets and to visualize their research results. These tools come with easy to use, menu driven interfaces through which a set of standard features can be used. Data and file sharing is easy if collaborators use the very same tools and versions. However, it is not trivial to add your own or any other algorithm without major modifications. Any code has to match with the internal data format and plug-in system. It is also impossible to get a 'customized filling' of the tools with exactly those algorithms that are needed by a user or user group.

In response to this need, a growing number of grass roots efforts aim to create data and software libraries, application programming interfaces (APIs), and repositories that provide access to the best algorithms. In many cases, algorithms are made available as open source so that the concrete

implementation can be examined and improved if necessary. Sample efforts are R (Ihaka and Gentleman 1996), StOCNet (Huisman and Duijn 2003), Jung (O'Madadhain, Fisher, White and Boey 2003), and Prefuse (Heer, Card and Landay 2005). However, the mentioned packages come as APIs or require scripting of code. None of them supports the easy, wizard based integration of new algorithms and datasets by developers or provides a menu driven, easy to use interface for the application user.

Grid computing (Berman, Hey and Fox. 2003) aims to address the computational needs of "big science" problems such as protein folding, financial modeling, earthquake simulation, or climate/weather modeling. It follows a service-oriented architecture and provides hardware and software services and infrastructure for secure and uniform access to heterogeneous resources (e.g., different platforms, hardware/software architectures, and programming languages), located in different places belonging to different administrative domains over a network using open standards. It also supports the composition of applications and services, workflow design, scheduling, and execution management. There are several challenges to overcome when using grid computing. To fully take advantage of the grid, one's code must be (re)written to take advantage of running in parallel on a cluster of potentially very different operating systems and environments, which appears to be a major challenge for most non-computer scientists. Further, access to the grid is usually through command line interfaces or customized portals optimized for the needs of specific communities. Finally, while grid computing is a powerful approach to 'big science' computations, many applications can be served well without parallel computing and distributed databases. The time and effort required to make an application grid-able is considerable and only justifiable if grid resources and functionality are truly needed.

In sum, there is a gap between algorithm and application developers and application users. Many algorithm developers are searching for possible utilities to quickly disseminate their work. Many researchers and educators are in need of good algorithms but are not equipped with the mathematical sophistication and programming knowledge required to benefit from code descriptions in research papers, implemented APIs, or advanced CIs. In many cases, users are not only interested in a single algorithm but they want tools similar to MS Excel, SPSS, Matlab, or Photoshop but with the option to customize and extend them according to their needs.

The CIShell specification aims to serve the needs of all three user groups: algorithm developers, application designers, and application users. Building on the Open Services Gateway Initiative (OSGi) specification, it supports the easy, wizard-driven plug and play of existing and new algorithms and datasets that are implemented as services. Data, algorithms and additional CIShell services such as graphical user interfaces, schedulers, logging services, etc. can be combined and deployed as a stand alone tool, (Web) service, or peer-to-peer service, among others. The core CIShell implementation can be 'filled' with high performance services or the datasets and services needed in a classroom setting. Hence, CIShell creates a bridge between algorithm developers, application developers, and their users as shown in Figure 1.

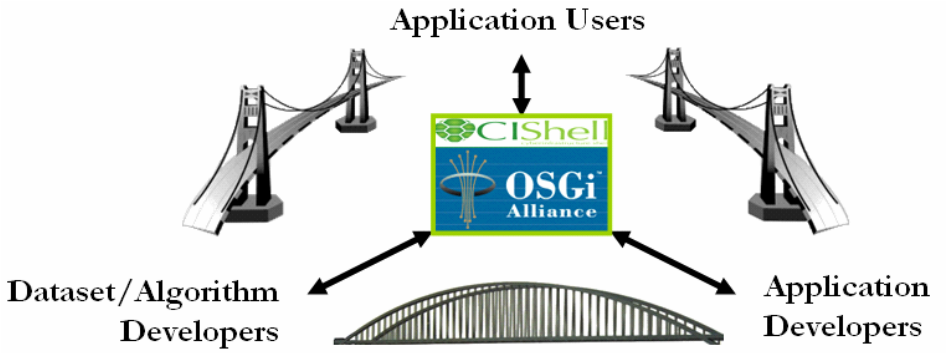

Figure 1. CIShell as a bridge among users, dataset and algorithm developers, and application developers.

The remainder of this chapter is organized as follows: Section 2 details the needs of the user groups CIShell aims to serve. Section 3 describes the inner workings of CIShell on an abstract level. Section 4

explains how CIShell is used by the three different user groups. Section 5 and 6 introduce diverse reference implementations. The chapter concludes with a discussion and an outlook of future work.

2. What Users Want

2.1 What Algorithm and Application Developers Want

Today, it is not only computer scientists which develop and create novel datasets and algorithms but also biologists, physicist, social scientists, etc. In many cases, the datasets and algorithms are used almost exclusively by the person, lab, or project that created them. Some are distributed via private web pages. Consequently, many algorithms are implemented and re-implemented uncountable times – a true waste of life time and resources. Given the effort it takes to implement a set of algorithms, comparisons of novel with existing algorithms are rare. Some algorithms are made available in compiled form or as source code. These efforts are truly appreciated by the community and lead to higher citation counts of related papers. However, there is no way to get an overview of all existing datasets and algorithms. To make things worse, datasets come in diverse formats and algorithms are implemented in very different languages.

The diffusion of high quality datasets and novel algorithms would greatly benefit from a means to easily integrate and utilize existing datasets and algorithms.

There is also a need for the design of tools that provide access to exactly those datasets and algorithms that are needed by a researcher, group, or community. Commercially available tools often provide menu driven interfaces, remote services, or scripting engines. They have workflow support, scheduling support, etc. Users will expect this from custom designed tools.

2.2 What Application Users Want

Analyzing and making sense of datasets frequently involves multiple steps, such as sampling, cleaning, analyzing, and sometimes visualizing for means of communication. For means of illustration, Figure 2 shows the diverse datasets (given in italics and underlined) and processing steps involved in a scientometric study (Shiffrin and Börner 2004) aiming at the identification of the topic coverage of a document dataset. The analysis starts with a list of documents – one document per line. This list is parsed and a term document matrix is generated in which each cell entry states how often a certain term occurred in a certain document. The resulting term document matrix is used to identify the top 50 most frequent terms. Next, the co-occurrence similarity of those top 50 terms is calculated. The more often two terms occur together in the same document (in the same line of the original list of all documents) the higher their similarity. The similarity matrix is then converted into a list of nodes and edges that can be read by the Pajek network layout tool (Batagelj and Mrvar 1998). Unfortunately, the generated layout labeled with (1) is not very readable. Given this, almost all terms co-occur with each other in at least one of the many documents. Layout optimization is needed. In a first attempt, we might apply a threshold to eliminate links below a certain similarity. However, this strategy disintegrates the network into one larger subgraph – labeled (2) – and many unconnected nodes. In a second attempt, Pathfinder Network Scaling (Schvaneveldt 1990) is applied to ensure that all 50 nodes stay connected yet a more readable layout is achieved, see visualization labeled with (3).

Most analyses in scientometrics and other fields of science require many more processing steps. Commonly, the output of one processing step is not compatible with the input of the next step. Tools and algorithms applied in one and the same study might be implemented in different programming languages and might only run on certain operating systems making data transfer across and the switch between platforms necessary. Many analyses are highly iterative. It is only after the entire sequence of processing steps is completed that data errors or layout optimization needs become visible. Some algorithms have a high computational complexity and have to be scheduled.