Integration of a neuroimaging processing pipeline into a pan-canadian computing grid

S. Lavoie-Courchesne\(^1,2,3\), P. Rioux\(^3\), F. Chouinard-Decorte\(^1,3\), T. Sherif\(^6\), M.-E. Rousseau\(^3\), S. Das\(^3\), R. Adalat\(^3\), J. Doyon\(^1,2\), C. Craddock\(^4,8\), D. Margulies\(^5,8\), C. Chu\(^6,8\), O. Lyttelton\(^7,8\), A. C. Evans\(^3\), P. Bellec\(^1,2,8\)

\(^1\)Centre de recherche de l’institut de gériatrie de Montréal, Montréal, CA
\(^2\)Département d’informatique et de recherche opérationnelle, Université de Montréal, Montréal, CA
\(^3\)McConnell Brain Imaging Centre, Montreal Neurological Institute, McGill University, Montréal, CA
\(^4\)Virginia Tech Carilion Research Institute, Roanoke, VA, USA
\(^5\)Max Planck Institute for Human Cognitive and Brain Sciences, Leipzig, DE.
\(^6\)Section of Functional Imaging Methods, Laboratory of Brain and Cognition, NIMH, NIH, Bethesda, USA
\(^7\)Mind, Brain Imaging and Neuroethics Unit, Institute of Mental Health Research, Royal Ottawa Health Care Group, University of Ottawa, Ottawa, CA
\(^8\)The Neurobureau Research Institute, Chicago, USA

E-mail: \{bellecp@iro.umontreal.ca, lavoiecs@iro.umontreal.ca\}

Abstract. The ethos of the neuroimaging field is quickly moving towards the open sharing of resources, including both imaging databases and processing tools. As a neuroimaging database represents a large volume of datasets and as neuroimaging processing pipelines are composed of heterogeneous, computationally intensive tools, such open sharing raises specific computational challenges. This motivates the design of novel dedicated computing infrastructures. This paper describes an interface between PSOM, a code-oriented pipeline development framework, and CBRAIN, a web-oriented platform for grid computing. This interface was used to integrate a PSOM-compliant pipeline for preprocessing of structural and functional magnetic resonance imaging into CBRAIN. We further tested the capacity of our infrastructure to handle a real large-scale project. A neuroimaging database including close to 1000 subjects was preprocessed using our interface and publicly released to help the participants of the ADHD-200 international competition. This successful experiment demonstrated that our integrated grid-computing platform is a powerful solution for high-throughput pipeline analysis in the field of neuroimaging.

1. Introduction

With most modern scientific projects relying heavily on computer programs, the dissemination of research software prototypes is of central importance at the very least to enable the reproduction of published results by independent research groups [1]. In the neuroimaging field, such dissemination raises a number of domain-specific computational challenges:

(i) **Complex multi-stage processing.** Neuroimaging datasets need to go through a large number of preprocessing steps before a scientifically meaningful outcome can be inferred.
These steps can include such operations as non-linear realignment of the individual brain anatomy into a common stereotaxic space [2], the correction for motion of the participant during the data acquisition [3] or the correction of physiological noise [4], amongst many others. Even for a common type of analysis such as the multi-level general linear model [5], it is only recently that fully automated scripts, which include all analysis steps going from raw data to the final results, have been made widely available [6].

(ii) **Computational challenge.** A neuroimaging database includes a collection of multiple brain volumes for anywhere between tens to thousands of subjects, and typically representing at least several gigabytes of raw data. Some of the processing steps mentioned above can take up to several hours of computation for a single subject on a single machine, such as the extraction of the cortical surface [7]. Processing a full database can thus range from tens of hours for a small study to possibly thousands of hours for a large database or a complex analysis, e.g. [8]. Rather than a very large computational challenge faced by a single research group, hundreds of research groups in the neuroimaging community are independently facing moderately large computational tasks.

(iii) **Diversity of production environments.** The type of computational resources that are being used to process neuroimaging databases varies widely in the neuroimaging community, ranging from a personal workstation to cloud or grid computing [9] and including multi-core workstations as well as computer clusters. To reach a wide audience, a software package should ideally be able to take advantage of any of these computing environments.

The most common approach to high-performance computing proceeds by parallelizing computations using a message-passing interface (MPI) [10]. The processing tools used in neuroimaging are generally not available in such a form, and re-implementing them with MPI is not feasible given the volume and heterogeneity of the existing code base and the fact that new algorithms are constantly being released by the community in a non-parallel format. The processing of a neuroimaging database is however inherently parallel, as multiple subjects are often treated independently. A solution to the issue of the computational load is thus to formally combine tools into a so-called pipeline, which represents the sequential application of a given series of tools to a particular dataset, and then to take advantage of the parallel nature of the pipeline flowchart to use distributed computation resources. If necessary, jobs simply interact through a file system which is shared by all the processing units. Rather than strict high-performance computing, this type of approach belongs to the emerging family of many-tasks computing [11]. A number of independent frameworks have been proposed in the neuroimaging community to design pipelines composed of several heterogeneous, non-parallel tools and then execute them in a distributed computing environment, e.g. [12]. The LONI pipeline\(^1\) [13] offers a graphical interface to design pipelines, as well as a web-based system to execute them on a dedicated supercomputer, but it is not easy to deploy that system on any given computing facility. Some libraries have recently been implemented in high-level scientific programming languages to design and execute pipelines: the pipeline system for Octave\(^2\) and Matlab (PSOM\(^2\)), Nipype\(^3\) [15] and SOMA workflow\(^4\) for Python [16]. These code-based frameworks are powerful for the implementation of a fully automated prototype pipeline, and can be deployed in a variety of computing environments, including supercomputers, but not on a grid of heterogeneous supercomputers. The CBRAIN\(^5\) initiative addresses this last issue by providing a flexible platform to run pipelines on a grid of supercomputers; however, CBRAIN

\(^1\) http://pipeline.loni.ucla.edu/
\(^2\) http://code.google.com/p/psom/
\(^3\) http://nipy.sourceforge.net/nipype/
\(^4\) http://brainvisa.info/doc/soma-workflow
\(^5\) http://cbrain.mcgill.ca/
was designed to run existing packages rather than build pipelines from scratch.

In this paper, we propose an integrated solution for the development and deployment of neuroimaging pipelines by building an interface between PSOM and CBRAIN. Since the proposed interface is generic, it is very easy to take any PSOM pipeline and make it publicly accessible via CBRAIN. We estimate that the integration of a new pipeline in CBRAIN could take as little as one day of work. The PSOM/CBRAIN integration addresses the issues (i-iii) we initially listed, as it implements fully automated multi-step analysis that can be deployed in virtually any computing environment, from a single workstation with multiple CPUs to a grid of supercomputers. To demonstrate the feasibility of our approach, we have integrated a pipeline for the preprocessing of functional magnetic resonance imaging datasets into CBRAIN. This pipeline is available as part of the “Neuroimaging Analysis Kit” (NIAK)\(^6\) [17], and was developed using the PSOM framework. We demonstrate the benefits of this integration on a large database, the ADHD-200 sample\(^7\).

2. PSOM and NIAK

PSOM is a framework to implement pipelines in a high-level programming open source GNU\(^8\) compatible language called Octave, and is also compatible with the proprietary language Matlab\(^9\). The core idea of PSOM is to build a representation of all the steps of the pipeline using a specific native Octave structure. Once this structure is defined, a generic tool called the pipeline manager can execute the pipeline in a number of computational environments, i.e. a local computer or a cluster of computers. A number of coding guidelines have also been established to help the developers keep the code concise, and ensure that the modules used in pipelines as well as the pipelines themselves can easily be combined, while keeping access to, and track of, all pipeline parameters. The description of a pipeline involves a list of jobs which are defined through five attributes:

- The **command** executed by the job. This is a (short) piece of Octave/Matlab code.
- A list of **input files** necessary to run the job.
- A list of **output files** generated by the job.
- A list of **clean-up files** that will be deleted by the job, in order to limit the amount of intermediate outputs of the pipeline.
- A list of **options** (or parameters).

These attributes are coded in a so-called Octave/Matlab structure, where each field represents a job, and each job has up to five fields coding for its attributes. An example of a simple pipeline follows:

```octave
% Job "sample" : No input, generate a random vector a
command = ['a = randn([opt.nb_samps 1]); save(files_out,''a'');'];
pipeline.sample.command = command;
pipeline.sample.files_out = 'sample.mat';
pipeline.sample.opt.nb_samps = 10;
% Job "quadratic" : Compute a.^2 and save the results
command = ['load(files_in); b = a.^2; save(files_out,''b'');'];
pipeline.quadratic.command = command;
pipeline.quadratic.files_in = pipeline.sample.files_out;
pipeline.quadratic.files_out = 'quadratic.mat';
```

\(^6\) www.nitrc.org/projects/niak
\(^7\) fcon_1000.projects.nitrc.org/indi/adhd200
\(^8\) http://www.gnu.org/copyleft/gpl.html
\(^9\) http://www.mathworks.com/
Figure 1. Dependency graph of a toy pipeline. Each job (i.e. running a specific tool on a specific dataset) is represented by a box. A plain arrow from box A to box B means that the job B uses one or multiple file(s) generated by job A. A dotted arrow from box A to box B means that the job A is using one or multiple file(s) deleted by job B.

```matlab
% Adding a job "cleanup": delete the output of "sample"
pipeline.sample.command = 'delete(files_clean);'
pipeline.sample.files_clean = pipeline.sample.files_out;
% Adding a job "cubic": Compute a.*3 and save the results
command = 'load(files_in); c = a.*3; save(files_out, ''c'');'
pipeline.cubic.command = command;
pipeline.cubic.files_in = pipeline.sample.files_out;
pipeline.cubic.files_out = 'cubic.mat';
% Adding a job "sum": Compute a.*2+c.*3 and save the results
command = 'load(files_in{1}); load(files_in{2}); d = b+c, save(files_out, ''d'');'
pipeline.sum.command = command;
pipeline.sum.files_in{1} = pipeline.quadratic.files_out;
pipeline.sum.files_in{2} = pipeline.cubic.files_out;
pipeline.sum.files_out = 'sum.mat';
```

The order in which the jobs are specified is meaningless. There are however some constraints on the order of execution for the jobs:

- If job A uses file as input, and file is an output of job B, then the execution of B must be complete before A is started.
- If job A uses file as input, and file is deleted by job B, then the execution of A must be complete before B is started.

These constraints define a dependency graph. For a pipeline to be solvable, the dependencies must form a directed acyclic graph. The dependency graph for the toy pipeline above is presented in Figure 1.

The basic approach to integrating a PSOM pipeline into the CBRAIN platform was to generate the pipeline structure and dependencies, and then convert it into an XML structure that could be handled by the CBRAIN server. The PSOM pipeline manager is not used at all in this context. The feasibility of this approach was tested with a particular pipeline, designed to preprocess a neuroimaging database including structural magnetic resonance imaging (MRI) as well as functional magnetic resonance imaging (fMRI). This pipeline along with all the necessary tools is distributed publicly as part of the neuroimaging analysis kit (NIAK) [17]. The outline of the processing steps for one subject was as follows. The three first volumes of each run were suppressed to allow the magnetisation to reach equilibrium. Each dataset was corrected of inter-slice difference in acquisition time, rigid body motion, slow time drifts (high-pass filter with a 0.01 Hz cut-off) and physiological noise [4]. Slow time drifts and physiological noise correction were implemented in an attempt to reduce the spatially correlated noise present in the fMRI time series, which may induce spurious spatial correlations unrelated to neural activity. For each subject, the mean motion-corrected volume of all the datasets was coregistered with

10 www.nitrc.org/projects/niak
Figure 2. An example of dependency graph for the NIAK fMRI preprocessing pipeline. This example includes two subjects with two fMRI datasets each. The pipeline includes close to 100 jobs, and clean-up jobs have been removed to simplify the representation. Colors have been used to code the main subpipelines in the analysis: correction for slice timing and motion, coregistration, temporal and spatial filtering, spatial resampling and correction of physiological noise (CORSICA).

A T1 individual scan using minctracc [18], which was itself non-linearly transformed to the Montreal Neurological Institute (MNI) non-linear template [2] using the CIVET pipeline [7]. The functional volumes were resampled in the MNI space at a 3 mm isotropic resolution and spatially smoothed with a 6 mm isotropic Gaussian kernel. The spatial smoothing was implemented in an attempt to minimize the residual variability in anatomy and functional organization of individual brain in stereotaxic space. These operations were implemented through generic medical image processing modules, the so-called MINC tools\textsuperscript{11}. Some of the operations (motion correction, correction of physiological noise, CIVET) are themselves pipelines, involving a number of steps. A flowchart for the pipeline is represented in Figure 2.

3. CBRAIN
CBRAIN’s primary goal is to render the computational aspects of neurological analyses as simple as possible. Researchers interact with the platform through an intuitive web interface. Complex

\textsuperscript{11} http://en.wikibooks.org/wiki/MINC
Figure 3. CBRAIN Platform Architecture Layers. The CBRAIN platform is a fully distributed grid of data and compute resources implemented using the Ruby on Rails Framework and common UNIX tools and libraries. It consists of 3 main layers: 1- The user layer (top), which represents the Rich Web Client. An authenticated user has full control of his/her data, tasks, tools or resources through a standard web browser. 2- The service layer (middle) contains all web services, presentation layers, models, logic, authentication/authorization, reporting and platform metadata. This layer does not contain scientific data, nor does it perform heavy computation. 3- The resource layer (bottom) represents all registered data storage (file systems or databases) and compute resources. A low-footprint Execution Controller (EC) coordinates and reports all activities on each HPC resource. Note that the Data Access API on the EC accesses data directly from the data resource without creating a bottleneck in the service layer, similar to every other grid architecture.
command-line tools are wrapped in simple web forms. On the back end, interactions with remote data and High Performance Computing centers (HPCs) are handled as transparently as possible. CBRAIN connects to HPCs and data around the world, giving users access to processing power previously unavailable to most research groups, while abstracting away all the complexities of handling the notoriously heterogeneous environments of distributed resources. CBRAIN’s fundamental architecture is modular and generic, meaning that its applicability is by no means restricted to neuroimaging.

CBRAIN is currently connected to 9 HPCs across Canada and Europe, for a total of around 80 000 cores. Given the heterogeneous nature of HPC setups and policies around the country and the world in general, it was necessary to create an API for CBRAIN to interact with various cluster management programs simply and effectively. Although some active projects, such as the DRMAA working group [19], have been carefully examining scheduler integration issues, none of them handled the sheer variety of schedulers, queuing setups and policies they encountered in a satisfactory way. Issues vary greatly, from unsupported scheduler versions, core scheduling versus full node scheduling, multi-level queuing, queuing limits to completely home-brewed schedulers (i.e. using UNIX scripts or Perl). To avoid imposing some queuing requirements on every compute center, the CBRAIN design aims at keeping any remote footprint as lightweight as possible. CBRAIN’s Simple Cluster Interface in Ruby (SCIR) library does just that and can currently interact with HPCs using various flavors and setups of SGE, Torque, PBS, Moab, Maui as well as a variety of custom setups.

Data in CBRAIN is handled through a flexible data provider API, allowing for the integration of any network-enabled storage or database into the platform. All registered data sources in CBRAIN are transparently presented as if they were from a single system. A project may contain data from multiple sites shared under the “virtual organization” model, thereby restricting access on an individual or group basis. By default, all data or generated results can only be accessed by the user whom registered them in the system. The user can then share these data with other registered members of CBRAIN. Collaborative projects usually share common data and, if necessary, roles can be established so that not all project members have read and write access.

4. Interface between PSOM and CBRAIN
At a user level, the execution of the pipeline starts with the selection of a file collection, see Figure 8 (Annex). The file collection has to follow a rigorous organization, making it possible to infer the identification labels as well as the neuroimaging datasets available for each subject. In this case, the canonical file organization followed the one adopted by the ADHD-200 consortium (see next Section 5 and Figure 4 for an outline). Adapting the pipeline to other types of file organization involves minimal coding, but it is one of the few development steps that is not automated at this point. The user then specifies the parameters of the pipeline in a dedicated web form coded in Ruby on Rails, see Figure 9 (Annex). Once the parameters of the pipeline have been specified, the interactions between CBRAIN and PSOM are as follows (see Figure 5):

1. CBRAIN saves the pipeline parameters in an XML file, along with a list of all the files that are going to be used in the pipeline execution.
2. A generic tool available as part of PSOM (psom_read_xml) is used to read the XML parameters file, parse it and convert it into an Octave/Matlab structure.
3. A dedicated tool (niak_pipeline_fmri_preprocess) is used to generate an fMRI preprocessing pipeline in NIAK based on the Octave/Matlab structure. At this stage, no dataset is actually processed but all steps of the analysis are described in an Octave/Matlab structure, as described in Section 2.
4. A generic tool available in PSOM (psom_write_pipeline2xml) is used to convert the pipeline into a series of XML files containing the job name, the input and output file names,
Figure 4. File organization of the ADHD-200 database. The files are organized with one folder per site contributing neuroimaging datasets (e.g. Kennedy Krieger Institute\textsuperscript{13}). The site-specific folder contains a file with comma-separated values coding for the phenotypic information for all participants, as well as one folder per subject named with the unique subject ID. The folder of each subject contains a number of subfolders with arbitrary name, each one corresponding to an imaging session (visit of the participant to the scanning site). For each imaging session, there can be one or multiple folders anat\_k, where k is an positive integer, each containing a structural MRI volume of the subject, called mprage\_noface.nii.gz. There are also one or multiple folders rest\_k, where k is an positive integer, each containing a resting-state fMRI acquisition of the subject, called rest.nii.gz.

the dependencies between jobs, as well as a comprehensive description of the attributes of the job in Octave/Matlab format. This last description can be used to execute single jobs using a dedicated command in PSOM.

(5) The execution manager in CBRAIN finally reads the series of XML file, groups some jobs to increase efficiency and executes each defined job (using psom\_run\_job) on the specified execution server while respecting the graph of dependencies. The pipeline execution terminates when all jobs have completed or an error has occured. The user monitors the execution of the pipeline in real time on a dedicated web page, see Figure 10 (Annex).

Even though the feasibility of the integration of PSOM in CBRAIN was demonstrated using the NIAK fMRI preprocessing pipeline, the interface described above can be applied to any PSOM pipeline. The creation of the interface for a new pipeline can be achieved by creating a new parameter page on CBRAIN and replacing the call to the pipeline generation used in step (3). Since the steps (1), (2), (4) and (5) are completely generic, the generation of a new interface can be done rapidly. The time of development for such new interface is estimated to
Figure 5. Workflow of the interface between CBRAIN and the NIAK fMRI preprocessing pipeline. The simple application program interface is based on generic tools implemented in PSOM. The main pipeline-specific stage is number 3, which only involves a few line of pipeline-specific code.

a few hours on a test execution server. The main work for integrating new tools in CBRAIN using the PSOM framework is to install all the necessary libraries in all the execution servers available to CBRAIN, which already adds up to 9 supercomputers and may require the limited involvement of a system administrator at each site.

5. The ADHD-200 global competition

Overview of the competition. The ADHD-200\textsuperscript{14} sample is a data-sharing initiative dedicated to accelerating the scientific community’s understanding of the neural basis of attention deficit hyperactivity disorder (ADHD) in human development. The project was coordinated by Michael Milham at the Child Mind Institute, NY, and involved 8 sites which contributed neuroimaging datasets as well as various phenotypic informations and clinical evaluation. One of the immediate goals of the ADHD-200 release was to organize a global competition for the image analysis pipeline that provided the best predictor (biomarker) of ADHD and ADHD subtype. The first (training) database comprised 776 children, with and without ADHD, and was released in April 2011. This database featured all phenotypic information, including ADHD diagnosis, and was provided to researchers in order to design an effective method. The competition will last until August 30th 2011, at which time competing researchers will submit an neuroimaging-based prediction of ADHD diagnosis and subtype for the unlabeled subjects. This is an unusually large volume of datasets, as a typical sample for an fMRI study was about 20 subjects before the advent of open data-sharing efforts such as the 1000-functional connectomes project [20]. Moreover, the training and test databases have to be preprocessed under very stringent deadlines.

The NIAK preprocessing. We decided to use the ADHD-200 global competition to demonstrate that our pipeline analysis tools can be utilized in a real large-scale project. Specifically, we preprocessed, quality-controlled and publicly released the training database as well as the test database. Table 1 shows a summary of the computational aspects of the preprocessing of the training ADHD-200 database. The initial size of the database was about 70 Gb, and over

\textsuperscript{14}http://fcon_1000.projects.nitrc.org/indi/adhd200/
Table 1. Computational metrics for the preprocessing of the ADHD-200 training sample. The columns respectively code for: scanning site; the number n of participants; the size iSize of the datasets before preprocessing (in Gb); the size fSize of the datasets after preprocessing (in Gb); the ratio fSize/iSize, called %size; the number of jobs composing the NIAK pipeline; the total time eTime spent running jobs; the effective computation time pTime using parallelization; the ratio eTime/pTime, called accel.

| Site                                             | n  | iSize | fSize | %size   | jobs  | eTime   | pTime   | accel   |
|--------------------------------------------------|----|-------|-------|---------|-------|---------|---------|---------|
| Peking University                                | 195| 9.1   | 27    | 296.70% | 5951  | 155:32:07 | 60:21:30 | 257.69% |
| Kennedy Krieger Institute                        | 84 | 4.9   | 15    | 306.12% | 2551  | 66:36:17  | 22:00:22 | 302.66% |
| neuroIMAGE                                        | 49 | 6.8   | 20    | 294.12% | 1477  | 47:35:30 | 16:22:14 | 290.72% |
| New-York University Child Study Center            | 223| 42    | 125   | 297.62% | 9574  | 239:11:14 | 106:52:49 | 223.79% |
| Oregon Health and Science University              | 80 | 3.5   | 10    | 285.71% | 6487  | 56:46:19  | 32:47:32 | 173.13% |
| University of Pittsburgh                         | 90 | 3.4   | 10    | 294.12% | 2738  | 64:58:40  | 23:31:16 | 276.25% |
| Washington University in St-Louis                | 31 | 3.1   | 9     | 290.32% | 3311  | 44:08:53  | 23:49:23 | 185.32% |
| **Total**                                        | 782| 72.8  | 216   | 296.70% | 32089 | 674:49:00 | 285:45:06 | 236.15% |

200 Gb of final results were generated. This does not include the intermediate results that were deleted as soon as possible in the pipeline. The NIAK fMRI preprocessing pipelines combined across all the sites comprised over 30000 jobs, which took a total of about 675 hours to execute. A factor of 2 speedup was achieved using parallel computing on a machine with 20 computing cores available for the pipeline, which represents a low degree of parallelization efficiency. Computational efficiency was not a primary objective of this proof of concept. We believe that the low parallelization efficiency is related to the mechanisms of job submission and can be dramatically improved in the future. Possible remedies to that issue are presented in the discussion section. As an indication of a minimal level of achievable efficiency, the parallelization efficiency observed with the PSOM pipeline manager in a similar computing environment and the same pipeline is about 67%, i.e. the computation time was divided by 67% times the number of cores [17]. When the tools will be deployed in a large supercomputer, we expect that the whole analysis will run in less than a day.

Quality control. A common concern expressed in the neuroimaging community regarding the pipeline processing of large volumes of data is the lack of control over the quality of individual steps, which is perceived as greater than with tools that are less automatic. This question is of central importance, because it undermines the scientific credibility of pipeline analysis, despite obvious advantages in terms of computational power and reproducibility. The results of the preprocessing were carefully yet rapidly quality controled using summary indices derived at the group level. These indices notably included the spatial (Pearson’s) correlation coefficient between each individual brain volume and the average of the population, after non-linear coregistration to stereotaxic space was performed, which was done for both structural and functional MRI. For example, at the KKI site, there were 4 subjects for which the coregistration between the structural and the functional volumes clearly failed, see Figure 6. This was further confirmed by the visual examination of individual datasets. A battery of summary brain maps were also generated, such as the average and standard deviation (std) of all brain volumes after non-linear coregistration, both for anatomical and functional scans, as well as the average of brain masks extracted in both modalities to identify variations in the field of view across subjects. For the KKI site, the std images clearly revealed substandard coregistration, with significant signal found outside of the brain in the stereotaxic space, see Figure 7. Those abnormalities were created by the four subjects where the coregistration between the structural and functional volumes catastrophically failed. Some group-level summary metrics were also generated to summarize...
the maximal transition in motion parameters, translation and rotation, for each subject (results not shown). This allowed us to tag subjects with excessive, moderate and negligible levels of motion during the acquisition of the fMRI datasets. A detailed report of the results of our quality control procedure was released publicly on the internet\textsuperscript{15}. Note that the results of the subjects where coregistration problems were identified could all be fixed by adjusting the parameters of the pipeline.

Scientific significance. The ultimate proof of the scientific value of a tool is that it is used by the community and significantly contributes to scientific discoveries. All of the preprocessed datasets have been publicly released as part of a community initiative organized by the Neurobureau\textsuperscript{16}, which includes a variety of tools and analysis besides the results of the NIAK fMRI preprocessing pipeline. In less than 6 months, this public sharing has generated over 5500 downloads on the NITRC website\textsuperscript{17}. Note that this number includes all downloads of individual preprocessing packages, yet we estimate based on our analytics software that at least 134 different researchers have accessed these resources, 30 of which have accessed the NIAK release. The winners of the ADHD-200 competition\textsuperscript{18}, a group from Johns Hopkins University, was made up of statisticians who used the Neurobureau release. Processing rapidly a large volume of neuroimaging datasets is a real challenge, especially for researchers who may lack some of the domain-specific knowledge. The NIAK pipeline as integrated in the CBRAIN platform has thus proven to be a powerful solution for large-scale processing tasks. The NIAK pipeline will be made accessible soon and without restrictions to the whole CBRAIN users community.

6. Discussion
In this paper, we presented a general strategy to implement complex pipelines composed of heterogeneous, non-parallel tools, and deploy them in a grid-computing platform. The core of the work consisted of writing an interface between two systems: PSOM, a code-oriented pipeline development framework, and CBRAIN, a web-oriented platform for grid computing. We demonstrated the feasibility of this approach by integrating a pipeline for preprocessing structural and functional magnetic resonance imaging into CBRAIN. We also showed the capacity of this approach to handle real large-scale databases by processing close to 1000 subjects

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{group_level_summary_metrics.png}
\caption{Group-level summary metrics of coregistration procedures.}
\end{figure}

\begin{itemize}
\item http://www.nitrc.org/plugins/mwiki/index.php/neurobureau:NIAKPipeline
\item http://www.theneurobureau.org
\item http://www.nitrc.org/projects/neurobureau
\item http://fcon_1000.projects.nitrc.org/indi/adhd200/results.html
\end{itemize}
Figure 7. Group-level summary maps of coregistration procedures. The first three rows are derived from anatomical MRI (mean across subjects, standard deviation across subjects, mean of the individual brain masks). The last three rows are derived from the individual average of all functional MRI (mean across subjects, standard deviation across subjects, mean of the individual brain masks).

released as part of the ADHD-200 competition. Despite the limited computational resources available to us in our test environment, we were able to process this massive dataset in a timely fashion, using semi-automated metrics to perform a quality control of the results of the pipeline. We publicly released the preprocessed datasets which have been used by competing teams around the world. We believe that this experiment makes a strong case for the feasibility and usefulness of grid-computing in neuroimaging.

The major limitation we have experienced thus far using the proposed approached is computational resources. We notably deployed the pipeline on a test execution server featuring only 20 computing cores, because we did not have time to deploy the software packages in a supercomputing environment for this pilot project due to the tight release schedule of the ADHD-200 competition. There is however a more fundamental issue: the fMRI preprocessing pipeline has been designed for PSOM which can submit at least three jobs per second to a queuing system, and much more for multi-core processing on a single machine. The pipeline was thus implemented at a very high level of granularity. There are notably hundreds of cleaning
jobs which only delete files on the system and usually take a few hundreds of milliseconds to run. This becomes a problem with a grid-level job submission system like CBRAIN, which has an overhead of several seconds to submit jobs. In addition, due to the nature of queuing in public resources, pipelines featuring large number of jobs are more exposed to long queue wait times. Another potential issue is that many HPC centers allocate a full multi-CPU node for a single task, so using a single core for a task would represent a considerable waste of resources. The CBRAIN framework addresses these issue by meta-scheduling: several PSOM jobs are regrouped under the control of serializers and parallelizers, and these meta-jobs are actually submitted to the cluster. Serializers combine sets of PSOM jobs that are directly dependent on each other in the graph of dependencies. Parallelizers are used to run several serialized and/or independent PSOM jobs in parallel. The end result is that any single cluster’s job will typically be responsible for running about 20 PSOM jobs from different parts of the pipeline, and all of this in the proper order such that dependencies are respected. Achieving an optimal grouping of tasks however requires precise estimates for the execution time of the jobs composing the pipeline, and we are currently working on a generic approach to optimize that process.

Another future improvement of the framework is to modularize the pipeline parameters, and give some memory to the CBRAIN pipeline manager. Currently, the same parameters apply to all subjects in a database. To reprocess a subset of a database with different parameters, the only solution is to copy this subset and then treat it as an independent database. In addition, if a parameter of the processing changes, CBRAIN is not able to identify which jobs of the pipeline actually changed and which did not. As a consequence, the whole pipeline has to be restarted if any change is made to the pipeline parameters. This is quite computationally inefficient, because the datasets that do not pass quality control are usually a small subset of subjects and the issues can usually be fixed easily by tweaking parameters and restarting a very limited number of stages. PSOM features these capabilities, and we will extend them to the CBRAIN pipeline manager in the near future.

The interface between PSOM and CBRAIN represents a very powerful approach for integrating neuroimaging pipelines in a web-based grid computing environment that can easily be shared with a large community of people with very limited information technology skills. We see this as a fantastic opportunity to connect a new community with the world of high-performance computing. We also believe that such progress would dramatically help a large number of neuroscientists perform the research they want at a higher pace or at a scale they would not currently envision. One priority is thus to port existing popular tools such as AFNI, FSL and SPM as pipelines conforming to the PSOM framework, such that they can be easily deployed into CBRAIN. Of important note is that a parallel effort to PSOM/NIAK, the so-called Nipype/Nipy projects [15], is being developed in Python by a very active community and already includes a number of pipelines coded under a format very similar to the one used in PSOM. We are hoping that creating an interface between Nipype/Nipy and CBRAIN is going to be as straightforward as interfacing PSOM/NIAK in CBRAIN. This will dramatically accelerate the integration of new tools into that platform.

While the early days of neuroimaging have been dominated by small studies collected by independent researchers who kept their datasets private, there are now substantial efforts toward the systematic sharing of large databases. This will open new research avenues for large-scale data processing and data mining in the field [21]. The pilot project reported here demonstrate that CBRAIN has the potential to stand as a key computing infrastructure to face this upcoming challenge.

**Acknowledgements**

The authors are grateful to the members of the ADHD-200 consortium for publicly releasing the eponym data sample. SLC and PB were partly supported by a pilot project grant of the Quebec
bioimaging network (QBIN), “fonds de recherche en santé du Québec” (FRSQ). The CBRAIN project is funded by Canada’s advanced research and innovation network (CANARIE).

Bibliography

[1] J. P. Mesirov, “Accessible reproducible research.” Science (New York, N.Y.), vol. 327, no. 5964, pp. 415–416, Jan. 2010. [Online]. Available: http://dx.doi.org/10.1126/science.1179653

[2] V. Fonov, A. C. Evans, K. Botteron, C. R. Ahlmi, R. C. McKinstry, and D. L. Collins, “Unbiased average age-appropriate atlases for pediatric studies,” NeuroImage, vol. 54, no. 1, pp. 313–327, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.neuroimage.2010.07.033

[3] D. R. Pickens, Y. Li, V. L. Morgan, and B. M. Dawant, “Development of computer-generated phantoms for fMRI software evaluation.” Magnetic Resonance Imaging, vol. 23, no. 5, pp. 653–663, Jun. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.mri.2005.04.007

[4] V. Perlberg, P. Bellec, J.-L. Anton, M. Pelegrini-Issac, J. Doyon, and H. Benali, “CORSICA: correction of structured noise in fMRI by automatic identification of ICA components,” Magnetic Resonance Imaging, vol. 25, no. 1, pp. 35–46, Jan. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.mri.2006.09.042

[5] K. Worsley, “A General Statistical Analysis for fMRI Data,” NeuroImage, vol. 15, no. 1, pp. 1–15, Jan. 2002. [Online]. Available: http://dx.doi.org/10.1016/0926-6492(00)00133-7

[6] Y. Chao-Gan and Z. Yu-Feng, “DPARSF: A MATLAB Toolbox for "pipeline" data analysis of resting-state fMRI.” Frontiers in systems neuroscience, vol. 4, 2010. [Online]. Available: http://dx.doi.org/10.3389/fnysm.2010.00013

[7] A. P. Zijdenbos, R. Forghani, and A. C. Evans, “Automatic "pipeline" analysis of 3-D MRI data for clinical trials: application to multiple sclerosis,” IEEE Transactions on Medical Imaging, vol. 21, no. 10, pp. 1280–1291, Oct. 2002. [Online]. Available: http://dx.doi.org/10.1109/02.866738

[8] P. Bellec, P. Rosa-Neto, O. C. Lyttelton, H. Benali, and A. C. Evans, “Multi-level bootstrap analysis of stable clusters in resting-state fMRI.” NeuroImage, vol. 51, no. 3, pp. 1126–1139, Jul. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.neuroimage.2010.02.082

[9] M. Hanke and Y. O. Halchenko, “Neuroscience runs on GNU/Linux,” Frontiers in Neuroinformatics, vol. 5, 2011.

[10] M. Snir, J. Dongarra, J. S. Kowalik, S. Huss-Lederman, S. W. Otto, and D. W. Walker, MPI: The Complete Reference (Vol. 1) - 2nd Edition, vol. 1 - The MPI Core, second edition ed. The MIT Press, Sep. 1998. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262692155

[11] I. Raicu, I. T. Foster, and Y. Zhao, “Many-task computing for grids and supercomputers,” Many-Task Computing on Grids and Supercomputers (MTAGS08), Proceedings of the IEEE Workshop on, pp. 1–11, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1109/MTAGS.2008.4777912

[12] J. D. D. Van Horn and A. W. Toga, “Neuroimaging workflow design and data-mining: a frontiers in neuroinformatics special issue.” Frontiers in neuroinformatics, vol. 3, 2009. [Online]. Available: http://dx.doi.org/10.3389/neuro.11.031.2009

[13] I. D. Dinov, J. D. Van Horn, K. M. Lozev, R. Magsipoc, P. Petrosyan, Z. Liu, A. Mackenzie-Graham, P. Eggert, D. S. Parker, and A. W. Toga, “Efficient, distributed and interactive neuroimaging data analysis using the LONI pipeline.” Frontiers in neuroinformatics, vol. 3, 2009. [Online]. Available: http://dx.doi.org/10.3389/neuro.11.022.2009

[14] J. W. Eaton, D. Bateman, and S. Hauberg, GNU Octave Manual Version 3. Network Theory Ltd., Oct. 2008. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/095461206X

[15] K. Gorgolewski, C. D. Burns, C. Madison, D. Clark, Y. O., Halchenko, M. L. Waskom and S. S. Ghosh, D. S. “Nipype: A flexible, lightweight and extensible neuroimaging data processing framework in python.” Frontiers in neuroinformatics, vol. 5, 2011. [Online]. Available: http://dx.doi.org/10.3389/fninf.2011.00013

[16] W. F. Punch and R. Enbody, The Practice of Computing using Python, 1st ed. Addison Wesley, Mar. 2010. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0136110673

[17] P. Bellec, F. M. Carbonell, V. Perlberg, C. Lepage, O. Lyttelton, V. Fonov, A. Janke, J. Tohka, and A. C. Evans, “A neuroimaging analysis kit for Matlab and Octave,” in Proceedings of the 17th International Conference on Functional Mapping of the Human Brain, 2011.

[18] D. L. Collins, P. Neelin, T. M. Peters, and A. C. Evans, “Automatic 3D intersubject registration of MR volumetric data in standardized Talairach space.” Journal of computer assisted tomography, vol. 18, no. 2, pp. 192–205, 1994. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/8126267

[19] P. Troger, H. Rajic, A. Haas, and P. Domagalski, “Standardization of an API for distributed resource
management systems,” *Cluster Computing and the Grid, IEEE International Symposium on*, vol. 0, pp. 619–626, 2007. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2007.109

[20] B. B. Biswal, M. Mennes, X.-N. Zuo, S. Gohel, C. Kelly, S. M. Smith, C. F. Beckmann, J. S. Adelstein, R. L. Buckner, S. Colcombe, A.-M. Dogonowski, M. Ernst, D. Fair, M. Hampson, M. J. Hoptman, J. S. Hyde, V. J. Kiviniemi, R. Kötter, S.-J. Li, C.-P. Lin, M. J. Lowe, C. Mackay, D. J. Madden, K. H. Madsen, D. S. Margulies, H. S. Mayberg, K. McMahon, C. S. Monk, S. H. Mostofsky, B. J. Nagel, J. J. Pekar, S. J. Peltier, S. E. Petersen, V. Riedl, S. A. R. B. Rombouts, B. Rypma, B. L. Schlaggar, S. Schmidt, R. D. Siedler, G. J. Siegle, C. Sorg, G.-J. Teng, J. Veijola, A. Villringer, M. Walter, L. Wang, X.-C. Weng, S. Whitfield-Gabrieli, P. Williamson, C. Windischberger, Y.-F. Zang, H.-Y. Zhang, F. X. Castellanos, and M. P. Milham, “Toward discovery science of human brain function,” *Proceedings of the National Academy of Sciences*, vol. 107, no. 10, pp. 4734–4739, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1073/pnas.0911855107

[21] D. S. Marcus, J. Harwell, T. Olsen, M. Hodge, M. F. Glasser, F. Prior, M. Jenkinson, T. Laumann, S. W. Curtiss, and D. C. Van Essen, “Informatics and data mining tools and strategies for the human connectome project.” *Frontiers in neurinformatics*, vol. 5, 2011. [Online]. Available: http://dx.doi.org/10.3389/fninf.2011.00004
Annex

Figure 8. Caption of the CBRAIN interface: file collections. The CBRAIN users can upload and download file collections through the web, and transfer these between data repositories. A secured sftp mechanism is running in the background, but this process is transparent for registered users.
Figure 9. Caption of the CBRAIN interface: running NIAK. A simple web form is available to set the parameters of the NIAK fMRI preprocessing pipeline.
Figure 10. Caption of the CBRAIN interface: execution servers. The status of jobs running an any execution server can be monitored in real-time on a dedicated web page.