Mouse Neuroimaging Phenotyping in the Cloud

Massimo Minervini[1], Mario Damiano[2], Valter Tucci[3], Angelo Bifone[2], Alessandro Gozzi[2], Sotirios A. Tsaftaris[1,2]
[1]IMT - Institute for Advanced Studies Lucca, Italy
e-mail: {massimo.minervini, sotirios.tsaftaris}@imtlucca.it
[2]Istituto Italiano di Tecnologia, Center for Nanotechnology Innovation@NEST, Pisa, Italy
e-mail: {mario.damiano, alessandro.gozzi, angelo.bifone}@iit.it
[3]Istituto Italiano di Tecnologia, Neuroscience and Brain Technology Department, Genova, Italy
e-mail: valter.tucci@iit.it

***Abstract*— The combined use of mice that have genetic mutations (transgenic mouse models) of human pathology and advanced neuroimaging methods (such as MRI) has the potential to radically change how we approach disease understanding, diagnosis and treatment. Morphological changes occurring in the brain of transgenic animals as a result of the interaction between environment and genotype, can be assessed using advanced image analysis methods, an effort described as "mouse brain phenotyping". However, the computational methods required for the analysis of high-resolution brain images are demanding. In this paper, we propose a computationally effective cloud-based implementation of morphometric analysis of high-resolution mouse brain datasets. We show that the proposed approach is highly scalable and suited for a variety of methods for MR-based brain phenotyping. The proposed approach is easy to deploy, and could become an alternative for laboratories that may require instant access to large high performance computing infrastructure.**

***Keywords*— High performance computing, cloud, neuroimaging, MRI, phenotyping, computer vision, image analysis.**

I. Introduction

The advent of advanced magnetic resonance imaging (MRI) methods for the study of the brain has significantly advanced our understanding of how this important organ functions in health and disease. A deeper understanding of brain physiology and function is expected to provide great benefit to the development of improved therapies for central nervous system disorders. An important tool towards this goal is the generation of transgenic animals, which have made possible to study in a controlled fashion, the effects of specific genetic mutations on brain anatomy and function. The use of transgenic mouse lines as pre-clinical analogs of human disease, together with high-resolution neuroimaging methods such as MRI have thus the potential to radically change how we approach disease understanding, diagnosis and treatment.

The application of MRI-based method to detect genetic- and environment-induced alterations in the morphology of specific organisms (in our case transgenic mouse lines), is known as MRI Phenotyping [1]. When combined with post-processing image analysis algorithms several features can be extracted that can improve our understanding of the effects of specific genetic variations on brain anatomy.

A neuroimaging pipeline (hereafter referred as pipeline) is a concatenation of (usually independent) image analysis steps that each process the input volume and output several metadata. Several processing platforms for the analysis of brain images have been proposed and have pushed the boundaries of computational neuroimaging [2]–[5]. However, most methods require remarkable computational resources. MRI phenotyping of the brain is particularly demanding since it involves the study of relative large number of subjects with high spatial resolution with different MR sequences.

Until recently, the majority of such analysis has either occured in single powerful workstations (with limited throughput) or in computing clusters to achieve higher throughput (see the reviews of [6], [7]). The scientific community reacted by establishing consortia (e.g., LONI, neuGrid, outGrid, CBRAIN) that aim at making distributed (grid) computing available for neuroscience and neuroimaging research [6], [7]. They usually provide a user-friendly graphical interface for designing and putting together any pipeline, with respect to both tools and data. However, for executing seamlessly the pipeline, access to the computer grid associated with the consortia underwriting these efforts is assumed. Some (e.g., LONI), allow the user to configure and install the framework on another grid environment or the cloud (e.g., Amazon EC2), but the user has to export the pipeline as a virtual machine and take care of the details of its execution on the cloud.

On the other hand, users that may not have access to powerful computer grids or are not members of the aforementioned consortia, are left with limited computational options. They can for example outsource their analysis (a pay for service is available [8]), which can be costly and not customizable. Alternatively, they can deploy their own customized cloud solutions. There have been several systems for managing grid or cloud workflows (Soma-Workflow, Megha Workflow Management System, GridBus, Aneka), and they have been demonstrated for neuroimaging applications [9]–[11]. However, management of such systems (e.g., installation on the cloud environment, and configuration of services) may prove challenging and time consuming to unexperienced users.

The recent introduction of cloud computing and the infrastructure and platform as a service paradigms (IaaS and PaaS), offers the opportunity to create powerful pipelines that are simple to use and available to a large community of scientists, thus increasing the democratization of science.

Here, we present a new image analysis pipeline solution based on state-of-the-art image analysis methods (e.g., diffeomorphic registration, [12]) and a commercial cloud-computing platform (PiCloud [13]) for the phenotyping of small animal brain MR studies. With respect to the existing solutions, the proposed approach:

- relies on a general purpose cloud environment and not on dedicated grids and clusters;
- utilizes the PiCloud infrastructure, which offers flexible and transparent allocation of computational resources;
- is based on Python which is simpler for building complicated pipelines than other solutions (e.g., relying on bash scripts or middleware);
- is sufficiently easy to be utilized by non computer experts; and
- focuses on providing an automated solution for animal phenotyping, which is currently under-represented in the literature.

This paper is organized as follows: in Section II-A, we outline the proposed image analysis pipeline, and in Section II-B we describe the proposed cloud implementation. Section III outlines implementation details and presents our experimental findings, while finally Section IV offers conclusions and discussion.

II. Proposed cloud implementation

A. Image analysis pipeline

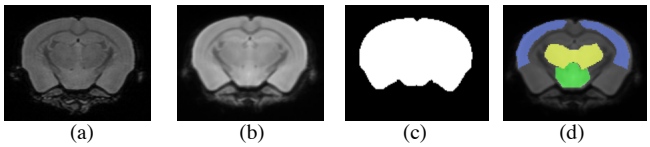
Typical image analysis pipelines in neuroimaging involve pre-processing, normalization to a template (volumetric linear and/or non-linear registration), white matter and gray matter segmentation, anatomical segmentation, and post-processing analysis of an individual's dataset [6], [7]. Each subject dataset consists of volumetric data as obtained from the scanner. In order to permit inter-group statistical comparisons of multiple subjects the volume of each subject needs to be registered to a template (or atlas). Depending on the desired outcome of the study several features can be extracted (size of anatomical structures, tensors on deformations etc). As this pipeline is typically executed independently for each subject's dataset, the process is highly suitable for parallel environments.

In typical phenotyping experiments, the experimental subjects are split in two or more groups, based on genetic differences (e.g., wild type vs. transgenic) or due to changes in environment (e.g., different diets) or a combination of the above. For simplicity we will assume here a control group (wild type) and an affected group (transgenic) under the same conditions. Our goal is to find morphometric differences in brain structure between those two groups, or more specifically differences in the volume of specific brain regions.

Our proposed pipeline relies on highly accurate, diffeomorphic, invertible non-linear registration between a subject volume (S_j^i), where $j \in [1, N]$, N is the number of subjects and a template (or atlas) $\mathbf{T}$. This allows us to propagate information from template to volume and vice versa in an accurate manner. Furthermore, it allows us to extract several voxel-level features that can be used in phenotyping [14]. Briefly described, the goal of non-linear registration is to find the optimal transformation, ϕ, within a specified transformation space which maps each $\mathbf{x}$ of volume $\mathcal{S}(\mathbf{x})$ to a location in volume $\mathcal{T}(\mathbf{z})$ such that a specified cost function relating the similarity between $\mathcal{S}$ and $\mathcal{T}$, is minimized. Recently, diffeomorphisms (ϕ) have been proposed that are differentiable, invertible, and topology preserving. With diffeomorphic registration, the following variational function is optimized

$$\{\boldsymbol{v}_1^*, \boldsymbol{v}_2^*\} = \underset{\boldsymbol{v}_{1,2}}{\operatorname{argmin}} \Bigg\{ \int_0^{0.5} ||L\boldsymbol{v}_1(x,t)||^2 \, dt + \int_0^{0.5} ||L\boldsymbol{v}_2(x,t)||^2 \, dt + \lambda \int_\Omega \Pi_\sim \left(\mathcal{S} \circ \phi_1(\mathbf{x}, 0.5), \mathcal{T} \circ \phi_2(\mathbf{x}, 0.5)\right) d\Omega \Bigg\}$$

where, $t \in [0, 0.5]$, $\boldsymbol{v}(x,t) = \boldsymbol{v}_1(x,t)$ and $\boldsymbol{v}(x,t) = \boldsymbol{v}_2(x, 1-t)$ when $t \in [0.5, 1]$, $\Pi_\sim$ is a volume similarity metric, L is an appropriate norm, λ is a regularizer, and Ω is the domain of the diffeomorphism. Under certain assumptions, the solution above can be solved using greedy strategies that find midway (symmetric) diffeomorphic solutions [12]. To increase the probability that the greedy strategy finds the global minimum, a multi resolution strategy is adopted, where the function above is optimized first for several lower resolutions and then for the final full resolution.



(a) (b) (c) (d)

Fig. 1. Coronal slices from (a) a subject volume S_j^i, (b) from the template T^i, (c) the mask of the template T^M, and (d) the labeled anatomy T^L.

We assume that a template $\mathbf{T}$ consists of 3 volumes which are known a priori: the intensity values T^i; a binary mask T^M that has 0 for outside the brain 1 for inside, and volume anatomical labels T^L. An example of slices from a subject and template dataset is shown in Fig. 1. The goal of this study is to label the anatomy in each subject's dataset, and find the volume of specific brain regions. We achieve this via registration-based label propagation [12].

Our proposed pipeline is illustrated in Fig. 2. Observe that each box represents a step in the pipeline and several non-linear registration steps between the template intensity volume T^i and an input volume are used. The result of each registration are affine registration parameters as well as forward (subject to template) and inverse (template to subject) warp fields. The pre-processing step involves removing extra-skull material and correcting for scanner field bias (inhomogeneity correction). Then in the next step a final registration between the corrected subject dataset $''S_j^i$ is performed with

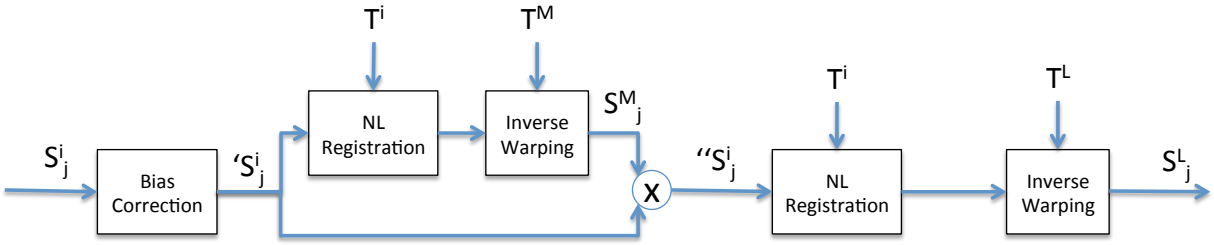


Fig. 2. The proposed pipeline.

the template (T^i). Using the inverse warp field and affine parameters obtained via the registration [12], it is possible to map (via warping) the labeled template T^L, to obtain the labelled subject volume S_j^L.

For each subject the normalized volume of a label (which corresponds to a specific region in the brain) is calculated as

$$\widehat{S_j^{L_k}} = |S_j^{L_k}|/|S_j^M|, \quad (1)$$

where $S_j^{L_k}$ is the k-th region in the labelled subject volume S_j^L, S_j^M is the subject brain mask, as found after the first registration, and $||$ denotes size. We should note that it is possible to add a label indicating the brain mask within S_j^L, thus eliminating the need to store S_j^M. Once all subjects have been processed, statistical comparisons can be performed.

This pipeline is general and several additional steps (e.g., white-matter and gray-matter segmentation) can be introduced.

B. Cloud implementation

The PiCloud [13] platform provides to the user a Python library and middleware that enable the use of cloud computational power from Amazon Web Services (AWS) via the Python interpreter. Although other Python interfaces exist to use the infrastructural services offered by Amazon Web Services [15], [16], PiCloud's, relieves the user of dealing with virtual servers, connections, resource scaling, etc.

As of March 2012, among the types of service offered by PiCloud, the following were available:

- m1 core type, offering 3.25 compute units and 8 GB of memory at $0.30/hour;
- c2 core type, offering 2.5 compute units and 800 MB of memory at $0.13/hour;
- c1 core type, offering 1 compute unit and 300 MB of memory at $0.05/hour;
- on demand or real-time operation;

where compute units are defined by Amazon Web Services, as 1 compute unit to provide the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

Depending on resource availability, jobs launched on demand on PiCloud can be queued, thus incurring a variable waiting time. To guarantee an immediate execution, without any variable waiting time, the required computational resources can be allocated a-priori with the real-time (RT) cores feature provided by PiCloud. As long as the core usage meets a certain minimum, the real-time allocation is free of charge. Data storage and download incur a cost, whereas environment setup and data uploading are free of charge.

Via the PiCloud system and secure shell (SSH), we utilized the environment feature to setup a Ubuntu Linux virtual machine with the customized libraries and applications required for our operation. In particular, we optimized (by enabling multi-threading) and compiled the ANTs toolkit, which offers registration (`ants`), warping (`WarpImageMultiTransform`), bias correction (`N3BiasFieldCorrection`), and other standalone applications. NeuroDebian [17] repositories have been added too, so that in principle it is possible to easily install any neuroimaging application described in [6]. We tested this environment by executing simple computational experiments in the shell.

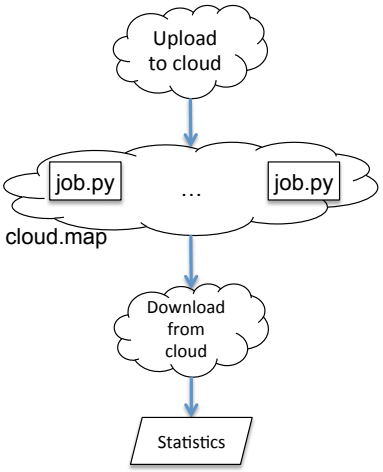
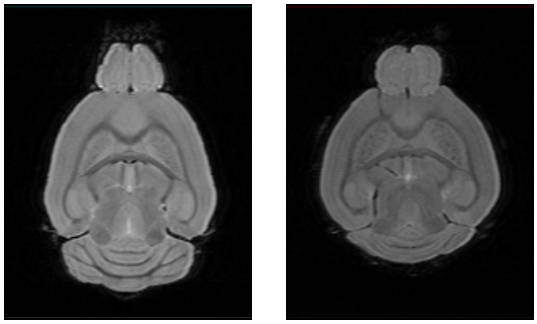


Fig. 3. Schematic of the workflow.

To execute our pipeline on the cloud we wrote customized Python code (see Appendix) to execute the workflow illustrated in Fig. 3. Massive parallelization occurs by splitting each dataset to be processed to independent jobs submitted via the PiCloud system. More specifically,

(a) We use the PiCloud file storage interface to *upload to cloud* the input data, i.e., subjects and template to the cloud from the local workstation.
(b) We wrote a Python function `job.py` that executes the pipeline for each input data set: it retrieves the input



(a) (b)

Fig. 4. Example axial slices from (a) control and (b) transgenic individuals.

data (template and subject) from the cloud storage, runs the applications composing the pipeline as subprocesses and finally saves the output files on the cloud storage. Temporary storage is deleted on job completion.

(c) A single Python instruction of the PiCloud library (`cloud.map`) executes (maps) the pipeline (`job.py`) for each subject, specifying the type of computational resources (core type, on demand or real time) and custom environment to use. This function essentially implements the core of the data parallelism that we employ here.

(d) To *download from cloud* we use the PiCloud file storage interface to retrieve the output data (e.g., S_j^L).

On the local workstation, once all data have been collected, normalized volume sizes are estimated using Eq. 1 for statistical analysis.

The web interface and the APIs available by PiCloud, provide several statistics (run times), metadata, and output logs on each job which is collected for analysis and comparisons.

III. Experimental results and discussion

A. Imaging data and hypothesis

Briefly, animal brains were scanned ex-vivo with a 7T Bruker Scanner, using a T2-weighted RARE sequence with the following imaging parameters: TR=500ms, TE=33ms, RARE factor=8, echo spacing=11ms, at an isotropic resolution of $94\mu m$. Overall, 20 adult male mice were studied: N=10 transgenic individuals carrying a mutation thought to mimic genetic variation related to a rare human disease, and N=10 wild-type (Control) C57BL/6J littermates. For the analysis, due to differences in skull size and shape, two group-based templates (one for control and one for transgenic) were developed using standard protocols. These templates were also manually labeled following standard methods.

Our aim was to identify and quantitate the presence of brain morphological abnormalities in transgenic individuals. Representative horizontal brain slices from control and transgenic individuals are shown in Fig. 4.

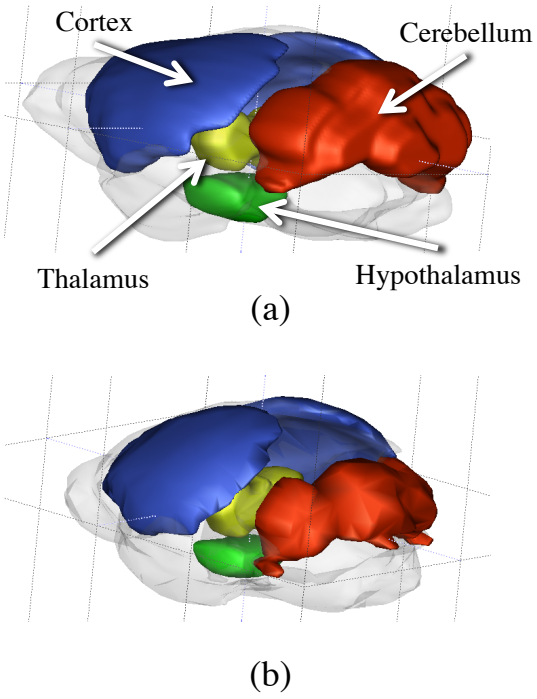


Fig. 5. 3D renderings of average anatomy for the control (a) and transgenic (b) groups, respectively.

B. Testing configuration

We executed the image analysis pipeline of Fig. 2, using the proposed PiCloud implementation and in one of our workstations. The local workstation was equipped with a 4 core 2.7 GHz Intel i5-M560 and 4 GB RAM. Two settings were tested, one using three out of four cores (3-core) and another using only one core (1-core). The workstation had similar operating system characteristics (64-bit Linux) and the same code-version of the ANTs toolkit. We wrote a bash script that would execute the pipeline of Fig. 2, using 1 or 3 cores.

C. Brain morphometric analysis

In total 20 pipeline executions were performed; 10 control individuals using the control template; 10 transgenic individuals using the transgenic group template. These executions were repeated locally and in the PiCloud environment. Figure 5 shows 3D renderings of average volumes for each of the group templates. Four different anatomical structures are shown with different colors, while the brain volume is reported as semi-transparent blue gray. Figure 6 shows average (and standard deviation as error bars) normalized volume sizes $\widehat{S_j^{L_k}}$ for the four areas of interest ($k = 1, \ldots, 4$). By performing our analysis we found that there are several anatomical differences between control and transgenic individuals (Mann-Whitney-Wilcoxon non parametric tests).

D. Computational results

Average time for pipeline execution as well as total cost and total processing time are shown in Table 1. The reduction of

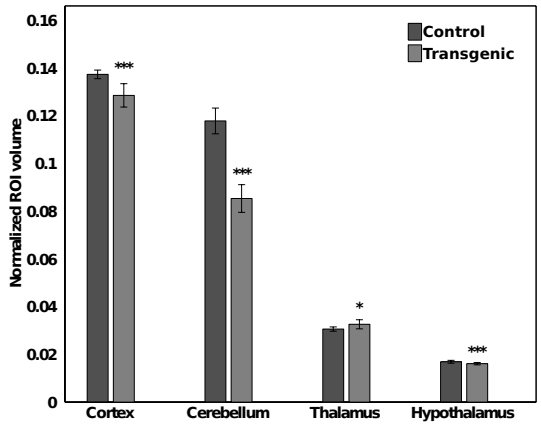


Fig. 6. Volume sizes between Control and Transgenic groups. *,*** indicate significant difference at the 0.05, 0.005 levels, respectively.

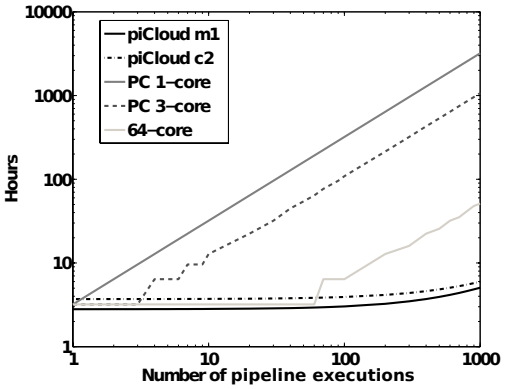


Fig. 7. Execution times (hours) for various configurations as a function of number of pipeline executions. (Shown in log-log scale.)

85% in execution time, when compared to the 3-core system, came at minimal monetary cost. Although the computational power of the c2 core type is only 80% the power of m1, its per-hour unit cost is less than 50% of the m1; it was able to finish all 30 executions with half the cost. Wait time for upload/download from our local storage was minimal. The execution time includes time spent uploading results from the PiCloud units to the cloud storage. Furthermore, wait time for on demand cores was minimal although it exhibited high variability particularly for the m1 service. This is due to PiCloud internal job handling, which dynamically leases units from Amazon according to a bundling of all jobs in the PiCloud workflow. Due to current pricing policies at PiCloud, real-time cores (denoted with RT) had no additional cost; they exhibited a fixed waiting time but guaranteed that all jobs would start at the same time. Registration takes longer when the deformation is large, which explains the variability in processing time observed in all systems.

Table 1. Average times and cost for cloud and workstation.

Type of node	Upload, Download (secs)	Processing (hrs)	Waiting (mins)	Cost ($)	Total time (hrs)
PiCloud					
M1	3.9(1.8) 4.1(1.6)	2.74(0.18)	10(20)	19.5	3.3
C2	3.9(1.8) 4.1(1.6)	3.68(0.3)	2(0.1)	10.7	3.76
M1 (RT)	3.9(1.8) 4.1(1.6)	2.74(0.18)	–	19.5	2.9
C2 (RT)	3.9(1.8) 4.1(1.6)	3.68(0.3)	–	10.7	3.72
Workstation					
3-cores	–	3.2(0.5)	0	0	22
1-core	–	3.2(0.3)	0	0	61

On average the maximum memory needed for each pipeline execution was 1 GB. The c1 core type was not able to execute the pipeline and results based on this core service are not reported. It should be noted, that although the c2 core type had less than 1 GB of memory it was able to execute the pipelines. PiCloud environments use a local HD swap, which increases the virtual memory size of the environment. However, the c1 core type failed, indicating that swap size varies according to core type. Although, frequent HD swapping may impact computational performance, in our case, this was limited since the multi-resolution nature of the registration algorithm uses a lot of memory only at the final full resolution step.

The computational complexity of non-linearly registering mouse brains is significant due to the high spatial resolution of the datasets and the subtle differences that we aim to uncover. The registration problem involved has upwards of 28M degrees of freedom and for very high resolution data requires up to 8hrs for processing time. Identifying differences in ex-vivo fixed skulls can be additionally complex, since the brain may be deformed (as a result of fixing). Thus the registration process must first recover these large deformations prior to recovering local structural variability.

Our test case used a few subjects; however, there are a number of neuroimaging pipelines that require several non-linear registrations (which composed the majority of our pipeline processing time) even for a few subjects. It has been suggested recently, that is best to use multiple atlases when performing a registration-based label propagation such as the one used here [18]. All subjects are registered using multiple templates and the final anatomical labels for each subject are obtained by fusing the results of each registration based propagation. As the number of subjects and atlases increases, performing such experiments in conventional computational environments (local workstations, or small clusters) becomes computationally demanding.

Figure 7 shows a hypothetical scenario where 1 to 1000 pipeline executions involving two non-linear registrations are considered, performed either on a local workstation (1 or 3 cores), a hypothetical small cluster (64 cores same per core characteristics as the local workstation), or the proposed cloud implementation. Average numbers from Table 1 for each case, were used to calculate these curves. Working on local environments becomes soon prohibitive, while the cloud approach is

clearly outperforming the 64-core cluster. Naturally the only limiting factor of the cloud is the time spent uploading and downloading data, thus pipelines should be designed such to minimize the amount of data transferred in/out of the cloud. However, it is possible to deploy a version of PiCloud on a local cluster for a fee. It is expected that, when core-types that are multi-core and GPU capable become available (according to PiCloud this will occur within 2012), and when our software has been optimized for such environments (e.g., using the FastFlow library [19]), the potential gains to be even more significant.

IV. Conclusions

In this paper we presented a cloud-based neuroimaging pipeline solution, for the application of morphometric phenotyping of the mouse brain. We are currently developing more complex pipelines (e.g., multi-atlas label propagation [18]) and integrating this platform with the nipype framework [20]. The possibility to perform hundreds of parallel executions with a fraction of the cost of owning a workstation opens the possibility to many laboratories around the world to develop state-of-the-art image analysis pipelines, without having to rely on (shared) distributed computing facilities.

V. Appendix

The following sample code illustrates the main parts of the proposed implementation, discussed in Section II-B, and demonstrates the ease of use of PiCloud in deploying an imaging pipeline on the cloud and Python as a 'glue' language.

Two Python modules `cloud, subprocess` need to be imported in any Python script as preamble to allow the use of the PiCloud infrastructure and call external binaries.

The pipeline, part (b) in Section II-B ('job.py' in Fig. 3) is wrapped in a function that retrieves the input data, runs the required tools (e.g., ANTs) as external processes and then saves the output files on the cloud storage.

```
def job(n,params):
  cloud.files.get('template file')
  subject = 's' + str(n) + '.nii.gz'
  cloud.files.get(subject)
  # parse parameters and input file
  build_ants_string = '... params ... '
  args = 'ANTS' + build_ants_string
  popen = subprocess.Popen(args, shell=True, \
              stdout=subprocess.PIPE, \
              stderr=subprocess.STDOUT)
  popen.communicate()
  cloud.files.put('...') #save output files
```

The PiCloud file storage interface is used to both upload the input data (part (a))

```
cloud.files.put('subject_file')
```

and collect the output data (part (d)).

```
cloud.files.get('output_data1') ...
cloud.files.get('output_data2')
```

The same pipeline is executed on multiple subjects in parallel on the cloud with a single instruction, that also specifies the desired core type (e.g., 'c2') and the custom environment (e.g., 'test') to use,

```
subjects = range(1,1000)
cloud.map(job, subjects, params, \
          _env='test', _type='c2')
```

which is part (c) of our implementation.

Acknowledgment

The authors would like to thank Alberto Galbusera for assistance with preparing the subjects for scanning, and PiCloud Inc, particularly Ken Elkabany, for providing financial and technical assistance.

References

[1] J. P. Lerch, J. G. Sled, and R. M. Henkelman, "MRI phenotyping of genetically altered mice." *Methods in molecular biology (Clifton, N.J.)*, vol. 711, pp. 349–361, 2011.

[2] ANTs. http://www.picsl.upenn.edu/ANTS.

[3] FreeSurfer, http://surfer.nmr.mgh.harvard.edu/.

[4] LONI, http://www.loni.ucla.edu/Software/.

[5] MNI, http://www.bic.mni.mcgill.ca/ServicesSoftware/HomePage.

[6] G. B. Frisoni, A. Redolfi, D. Manset, M.-E. Rousseau, A. Toga, and A. C. Evans, "Virtual imaging laboratories for marker discovery in neurodegenerative diseases," *Nature Reviews Neurology*, vol. 7, no. 8, pp. 429–438, Jul. 2011.

[7] I. Dinov, K. Lozev, P. Petrosyan, Z. Liu, P. Eggert, J. Pierce, A. Zamanyan, S. Chakrapani, J. Van Horn, D. S. Parker, R. Magsipoc, K. Leung, B. Gutman, R. Woods, and A. Toga, "Neuroimaging Study Designs, Computational Analyses and Data Provenance Using the LONI Pipeline," *PLoS ONE*, vol. 5, no. 9, pp. e13 070+, Sep. 2010.

[8] Cambridge Neuroimaging, http://www.cambridgeneuroimaging.com.

[9] C. Vecchiola, S. Pandey, and R. Buyya, "High-Performance Cloud Computing: A View of Scientific Applications," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*. IEEE, Dec. 2009, pp. 4–16. [Online]. Available: http://dx.doi.org/10.1109/I-SPAN.2009.150

[10] S. Pandey, W. Voorsluys, M. Rahman, R. Buyya, J. E. Dobson, and K. Chiu, "A grid workflow environment for brain imaging analysis on distributed systems," *Concurr. Comput. : Pract. Exper.*, vol. 21, no. 16, pp. 2118–2139, Nov. 2009.

[11] G. Antoniu, L. Bougé, B. Thirion, and J.-B. Poline. (2010, Sep.) AzureBrain: Large-scale Joint Genetic and Neuroimaging Data Analysis on Azure Clouds. [Online]. Available: http://www.irisa.fr/kerdata/lib/exe/fetch.php?media=pdf:inria-microsoft.pdf

[12] B. B. Avants, C. L. Epstein, M. Grossman, and J. C. Gee, "Symmetric diffeomorphic image registration with cross-correlation: evaluating automated labeling of elderly and neurodegenerative brain." *Medical image analysis*, vol. 12, no. 1, pp. 26–41, Feb. 2008.

[13] PiCloud, http://www.picloud.com/.

[14] J. Ashburner and K. J. Friston, "Voxel-based morphometry–the methods." *NeuroImage*, vol. 11, no. 6 Pt 1, pp. 805–821, Jun. 2000.

[15] BOTO, https://github.com/boto/boto.

[16] Apache Libcloud, http://libcloud.apache.org/.

[17] NeuroDebian, http://neuro.debian.net/.

[18] R. A. Heckemann, J. V. Hajnal, P. Aljabar, D. Rueckert, and A. Hammers, "Automatic anatomical brain MRI segmentation combining label propagation and decision fusion," *NeuroImage*, vol. 33, no. 1, pp. 115–126, Oct. 2006.

[19] M. Aldinucci, M. Danelutto, P. Kilpatrick, M. Meneghin, and M. Torquati, "Accelerating code on multi-cores with fastflow." in *Euro-Par (2)*, ser. Lecture Notes in Computer Science, E. Jeannot, R. Namyst, and J. Roman, Eds., vol. 6853. Springer, 2011, pp. 170–181.

[20] K. Gorgolewski, C. D. Burns, C. Madison, D. Clark, Y. O. Halchenko, M. L. Waskom, and S. S. Ghosh, "Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python." *Frontiers in neuroinformatics*, vol. 5, 2011.