Westfälische
Wilhelms-Universität
Münster

Patrick Schiffler, Jan-Gerd Tenberge
University of Münster, Münster, Germany
schiffler@uni-muenster.de

NPS: Neuroimage Management and Processing System Built on Open Source Software

Introduction

We present the Neuroimage Processing System (NPS). NPS can reliably perform arbitrary imaging pipelines in a reproducible fashion and is built upon well-tested, freely available open source software. This is our first publication of the system and presents a high-level overview of its purpose and capabilities. Along with this abstract we publish the NPS documentation to foster interoperability with other systems.

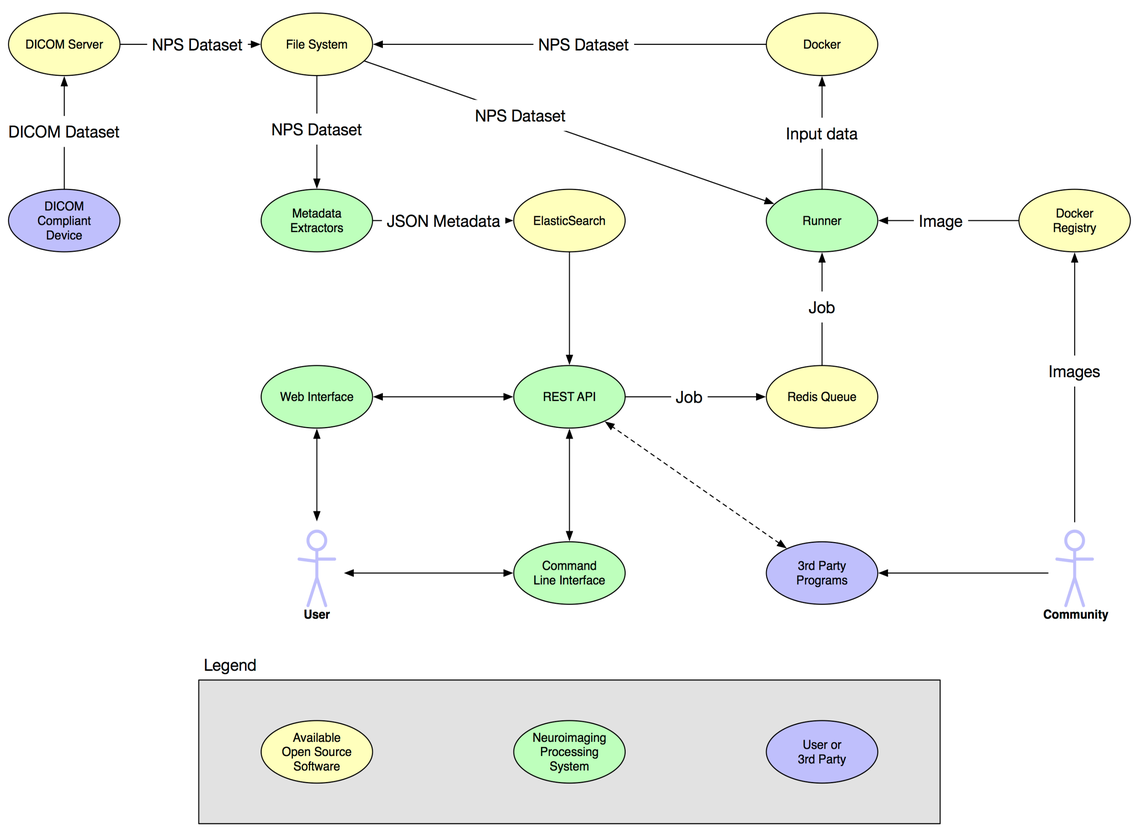
Methods

Data enters the system through the DICOM Receiver, each received DICOM Series Instance gets converted to a NPS dataset. A dataset in NPS is any set of immutable files and folders. NPS datasets are periodically indexed by an extendable set of Python scripts that extract metadata from the files and writes it to an ElasticSearch instance, which also stores the relationship of different datasets to each other.
Since datasets are immutable every operation in the system consists of multiple read-only input datasets and a job description and creates a single new output dataset. Job descriptions specify a command to run, a Docker [1] image to run the command in, and the mount-points of the input datasets in the container. All Docker images in the system are versioned, making sure that identical job definitions on identical input always create the same output. Combined with our approach of defining longer pipelines this enables us to cache individual steps of the pipeline. Jobs can be automatically crated from a pipeline definition (see abstract 1979). A RESTful API allows third party software components to work with NPS metadata and the job queue without direct interaction with Redis or ElasticSearch.
The job definition itself is a simple JSON document that is pushed into a Redis queue to be scheduled for execution. Jobs are fetched from the queue by NPS Runners that mount the dataset file system. The Runner script, which is written in Python, fetches a single job from the Redis queue, parses the definition, downloads the Docker image, creates a container, and runs the command. A temporary dataset is created and mounted in the container at /output. Once the container terminates, this dataset is converted to a regular NPS dataset and fed back into the system. The output dataset identifier is determined only by its job definition and therefore known before the actual processing of the job. Our system uses this information to skip jobs whose output is already known from earlier executions and handle dependencies between multiple stages of a pipeline.
Any log output produced while the job is running is sent to ElasticSearch and accessed by a Kibana instance providing real-time access and search functionality for all present and past logs.
As Docker images may sometimes not be publicly shared for licensing or privacy reasons, the system also contains a private Docker Registry which can be used to store images on premise. An optional GitLab instance can hold Git repositories with sources and data for each image, which will be automatically built, versioned and made available on each change to the source.
A schematic overview of the system and its components is shown in the figure . It shows the interaction of the single subsystems as well as the reciprocal action between the free, open-source, and self-developed components.

Results

The described system offers an easy way to develop and execute reusable and shareable neuroimage computation pipelines. It allows the automatization, parallelization and distributed execution of the developed processing tasks and pipelines. The technical documentation of the system can be read under https://github.com/neuro/nps-docs.

Our current implementation of the system manages ~150,000 datasets with peak rates of more than 10,000 jobs processed per day on a cluster of 45 machines.



Conclusions

The Neuroimage Processing System is a big step towards the goal of reproducibility [2], data management and distributed parallel execution in neuroimage processing. We invite feedback and Pull Requests on our GitHub page.

References

[1] Docker, Inc. (2013) Docker, http://www.docker.com/
[2] Peng, R. D. (2011). Reproducible research in computational science. Science, 334(6060), 1226-1227