Developer Guide

This section covers how to continue development of Hivemind.

## Installation

Please reference the User Guide for the Anaconda installation steps that must be performed before following the rest of this guide.

The installation process (note that the setup.py step is different here):

1. git clone ssh://git@altssh.bitbucket.org:443/amirbahmani/elasticmedflow
2. cd elasticmedflow/Team29
3. python setup.py develop

To allow running of unit tests (further details below):

1. pip install coverage

To build the Sphinx HTML documentation:

1. pip install sphinx
2. cd elasticmedflow/Team29/docs
3. make html
4. Open elasticmedflow/Team29/docs/build/index.html

Below is the directory structure of Hivemind. The core components are the mpi package, pipeline package, and util module. main.py, while not being part of Hivemind, is an example of how Hivemind master and workers can be created in an MPI environment.

examples/main.py

hivemind

├── \_\_init\_\_.py

├── mpi

│ ├── \_\_init\_\_.py

│ ├── master.py

│ ├── worker.py

├── pipeline

│ ├── \_\_init\_\_.py

│ ├── pipeline.py

│ ├── rank.py

│ ├── task.py

├── util.py

**MPI**

Hivemind uses the Master/Worker paradigm to distribute work across the cluster.

* master.py : the master MPI process
  + Creates a PipelineFramework from the Tasks and their requirements
  + Ranks the framework if ranking function is given
  + Creates the ConcretePipelines from the framework and patient data
  + Manages a PriorityQueue of Tasks
  + Manages a Queue of Workers (contains all Workers at start of program)
  + Determines what Tasks can be run based on the ConcretePipelines and checkpointing and puts them onto the PriorityQueue
  + Loops through the following steps until no more work is left and all Workers have been told to exit
    - Sends out available Tasks to available Workers (based on the two queues)
    - Close out Workers that are no longer needed due to the max concurrency of the ConcretePipelines
    - Receive Task completion messages from a Worker
    - Marks the given Task as done (creates checkpoint file, updates max concurrency)
    - Adds the newly available Tasks onto the PriorityQueue
    - Puts the given Worker back onto the Queue
* worker.py : the worker MPI process
  + Uses a tag field to determine when to exit. This could be extended to have the worker have different behavior as necessary.
  + Loops through the following steps until told to exit
    - Receive a Task from the Master
    - Run the given Task
    - Send Task completion message to Master

**Pipeline**

Hivemind uses NetworkX (Python package) for creating and manipulating pipelines.

* pipeline.py : creates a specific pipeline for each patient in the CSV file.
* task.py : represents a task inside a pipeline
* rank.py : uses ranking algorithms for ranking and prioritizing tasks in the pipeline. We implemented two ranking methods:
  + rank\_by\_successors - gives tasks with more immediate successors a higher priority
  + rank\_by\_total\_successors - gives tasks with more total successors (children, grandchildren, etc) a higher priority
  + Other methods could be added, such as rank\_by\_topological\_sort or rank\_by\_parallel\_topological\_sort

**Util**

This module contains all of the utility functions used throughout the code. For example, the code for creating directories on the file system, reading JSON / CSV files and generating a random checkpoint directory name is contained here. Changes to the JSON specification would have to be reflected here as well as task.py in the pipeline package.

**Test suite**

Before beginning development on Hivemind, both the unit tests and blackbox test suite should be run to verify that it is correctly set up in your environment.

To run the unit tests, first install the coverage package (see above) in your Python virtual environment, then cd into the tests directory, and run

./unittests.sh

To run the blackbox tests, cd into the tests/blackbox, and run

./blackboxtests.sh

## Words of Caution

1. Never change the UID of Task after it has been created. This will change its hash, which will break how the pipeline works.
2. Make small changes and then run both testing suites. Hivemind is less than 350 lines of code and small changes can have large effects.