Note: these instructions are to reproduce the first half of the poster's results, i.e. data referring to PLB, not DPLB.

A ARTIFACT DESCRIPTION

A.1 Abstract

We provide code and scripts to reproduce the same experiments which were used in evaluation of our proposed method - Progressive load balancing. The artifact are meant to provide evidence for the contributions of the paper by producing similar results to the ones published. We also provide the original data presented in the paper.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Progressive load balancing.
- **Run-time environment:** Linux. Require C compiler with MPI and OpenMP support.
- **Hardware:** Shared memory node with 1 or 2 sockets and Intel CPUs, preferably a recent Xeon with a total of 12 or more cores in the node.
- **Execution:** Sole user of node, process and thread pinning, preferably a batch submission system like PBS.
- **Output:** Figures and table from paper.
- **Experiment workflow:** Customise Makefile and batch submission script. Use provided bash and python scripts to generate experiments used in the paper and submit them to back end node. Use python script to generate figures.
- **Experiment customization:** Edit experiment configuration file and submission script.
- **Publicly available?:** Yes.

A.2.2 How delivered. Download code from public git repository at `https://bitbucket.org/Justs/ia3_2017.git`

A.2.3 Hardware dependencies. Shared memory node with 1 or 2 sockets and Intel CPUs, preferably a recent Xeon with a total of 12 or more cores in the node.

A.2.4 Software dependencies.

- Linux
- C compilation environment with support for OpenMP and MPI
- python 2
- python 3 with modules:
  - matplotlib
  - seaborn
  - numpy
  - pandas

A.3 Installation

Edit Makefile to use your preferred C compiler and link with MPI and OpenMP. Run `make` to compile.

A.4 Experiment workflow

There are detailed instructions and examples in `README` files in the root and experiments folders. The workflow is as follows:

1. Use the provided submission scripts and `README` in the root folder to produce your own submission script, with thread/process pinning appropriate for your environment.
2. Navigate to the folder `experiments`.
3. Use the `README` in the folder `experiments` to augment your submission script for experiment output redirection.
4. Use the script `generate.py` to generate experiments used in the paper.
5. Launch experiments using the script `runBatch.sh`.
6. Run `bash generate_plots.sh` to summarise experiments as figures.

A.5 Evaluation and expected result

The final outputs are 3 pdf figures and 1 table, corresponding to the ones in the Evaluation section of the associated paper. The results likely will not match the ones presented in our paper exactly, but we expect the trends to be the same. In particular:

- **Landscapes:**
  - The update spread increases significantly (moves up on y axis) for async methods without load balancing when noise is added.
  - The iteration rate decreases significantly (moves left on x axis) for sync and ssync method when noise is added.
  - The load balanced async methods do not move significantly when noise is added. Additionally, they are in bottom right corner of the landscape plot (i.e. spread is low and iteration rate is high (in most cases). (paper contributions 1 and 2)
- **Boxplot:** Some of the load balanced version are fastest when noise is added. (paper contribution 3)
- **Table:** Load balanced versions experience least slowdown when noise is added. (paper contribution 3)

A.6 Experiment customization

New experiments can be created by following the pattern of the files in the `experiments` folder. All algorithm settings are set in the application's input file `config.txt`. The meaning of the variables is explained in the `README` in the root folder.

A.7 Notes

The original data is available as a separate download; see instructions at the end of `experiments/README`.