

# Overview—Towards Portable Online Prediction of Network

## Utilization using MPI-level Monitoring

### I. Getting Started Guide

In this overview document, we focus on how to do online prediction after having the application logs of network traffic data. We used the pre-obtained logs to validate the accuracy and performance of our online prediction approach as a first step towards future more adaptive resource management.

#### 1. Dependencies and required packages

##### (1) Dependencies

- a. Python 3.6.7
- b. Pip 19.0.3 (package management system)

##### (2) Python Packages

```
python -m pip install -U pip
pip install --upgrade pip
pip install -U --user scipy scikit-learn fastdtw statsmodels
pip install -U --user matplotlib numpy tensorflow
```

- ✓ Matplotlib
- ✓ NumPy
- ✓ Tensorflow 1.12.0
- ✓ SciPy
- ✓ scikit-learn
- ✓ fastdtw
- ✓ statsmodels

### II. Step-By-Step Instructions

1. Platform used for online prediction: 2 NUMA nodes, 2 sockets, 8 cores per socket, each socket has 8-core 2.4GHz Intel Xeon CPU E5-2630 v3. 64GB
2. Datasets
  - (1) Datasets for HACC: `./hacc/hacc_0.log` (for Fig. 3a in the paper)
  - (2) Datasets for AMG: `./amg/amg_0.log` (for Fig. 3b in the paper)
3. Explanation of input dataset (only for variables that we're interested in this paper)

```
[10.63] node 0: n_msg = 233872; bytes = 463523328; sys_bytes = 118304438
```

- (1) `[10.63]`: the value in brackets indicates the time in seconds
- (2) `bytes`: network utilization obtained by MPI-level monitoring (Unit: bytes)
- (3) `sys_bytes`: network utilization obtained by system-level monitoring

4. Sequence-to-sequence online prediction Python script: `./ seq2seq_predictor.py`

5. Execution

```
python seq2seq_predictor.py <log_file> <app_name>
```

For `<log_file>`, we have one log file provided for each of the application. For `<app_name>`, specify either “hacc” or “amg” to make it run. More detailed information on how to execute can be referred to the examples as follows.

Note that **we have fixed the seed in the random number generator** in our model and this generates **reproducible outputs** almost identical to the original results we included in the paper.

6. Example – HACC (`./hacc/hacc_0.log`)

(1) Configuration

- a. Period: 60 seconds
- b. System-level Infiniband monitoring data = RAW (i.e., Infiniband readings need to be multiplied by 4)

(2) Execution

```
python seq2seq_predictor.py ./hacc/hacc_0.log hacc
```

(3) Output files

- a. `hacc-mpi.png` (`./expected_output/hacc/hacc-mpi.png`)

This figure shows how accurate the MPI-level monitoring solution compared to the system-level solution. The blue line shows the network utilization, represented in MB/s, obtained by MPI-level monitoring. The red line shows the network utilization obtained by system-level monitoring. Therefore, the more overlaps of the two lines, the higher the accuracy of MPI-level monitoring. We can reproduce Fig. 2(a) in our paper via this output file.

- b. `hacc-pred.png` (`./expected_output/hacc/hacc-pred.png`)

This figure corresponds to Fig. 3(a) in our paper. The blue line is the network utilization results obtained from MPI-level monitoring. The red and green lines indicate the predicted network utilization by our sequence-to-sequence approach and ARIMA model, respectively. The more overlaps with the blue line, which is the expected network utilization, the better the prediction accuracy of the approach.

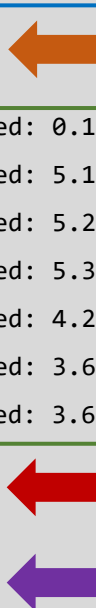
- c. `hacc-time.png` (`./expected_output/hacc/hacc-time.png`)

This figure corresponds to Fig. 4(a) in our paper. Due to the differences in hardware, the absolute results can be different from our results in the paper. However, the trend of the computational overhead should be the same. We can see decreased computational overhead of our proposed sequence-to-sequence approach compared to that of ARIMA model.

(4) Output printed on the terminal (for prediction accuracy)

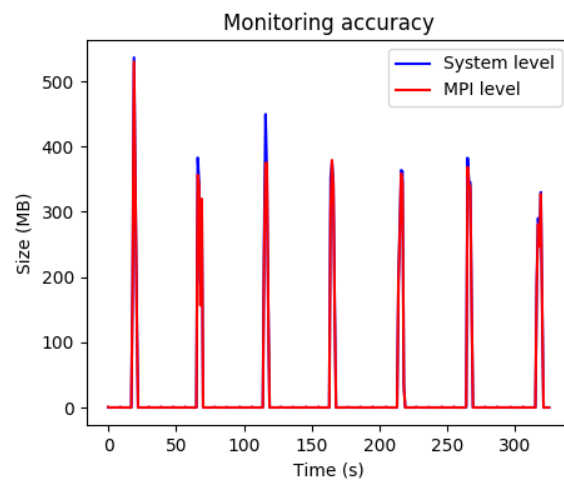
The printed output information is shown in the following figure. The blue box shows the information printed from our OnlineS2S model and the green box shows the information printed from the statistical ARIMA model. To compare the computational overhead of the two approaches, the training time for each step for both approaches are shown. To correspond the results of the left table of Table 2 in our paper, you can see the places where red and orange arrows indicate. The results which the purple arrow is pointing is the corresponding result shown in Table 1 of the paper.

```
Using TensorFlow backend.
Parsing hacc_0.log
Total data size = 506, init_data_size = 120
TensorFlow's version : 1.0 (or more)
2019-06-02 10:43:55.826827: W tensorflow/python/util/util.cc:297] Sets are not currently
considered sequences, but this may change in the future, so consider avoiding using them.
First train, iter = 300, size = 1, loss = 688.176758, training time = 26.435460
Training, iter = 130, loss = 4178435.000000, training time = 12.717893
Training, iter = 62, loss = 5762571.000000, training time = 6.995627
Training, iter = 10, loss = 8325992.000000, training time = 1.353708
Training, iter = 12, loss = 7215837.500000, training time = 1.433205
Training, iter = 8, loss = 7773440.000000, training time = 0.979137
Training, iter = 10, loss = 14874484.000000, training time = 1.321046
DNN MSE: 6194.232950
DNN FastDTW: 2736.97
Timestep 120, time elapsed: 0.10
Timestep 180, time elapsed: 5.10
Timestep 240, time elapsed: 5.28
Timestep 300, time elapsed: 5.31
Timestep 360, time elapsed: 4.25
Timestep 420, time elapsed: 3.69
Timestep 480, time elapsed: 3.60
ARIMA MSE: 14433.043824
ARIMA FastDTW: 4344.42
MPI-Mon MSE: 36.359880
MPI-Mon FastDTW: 376.75
```

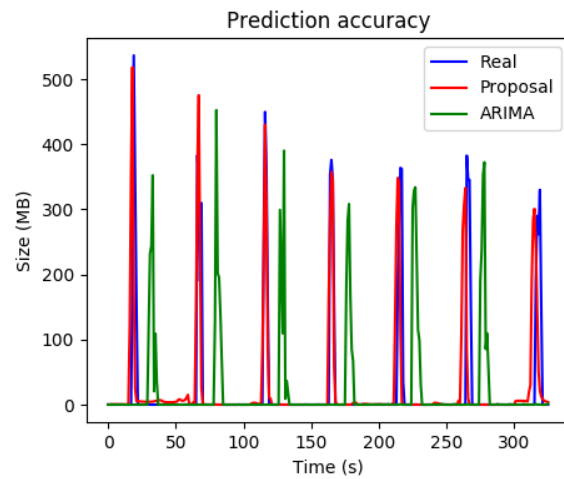


(5) Expected output figures

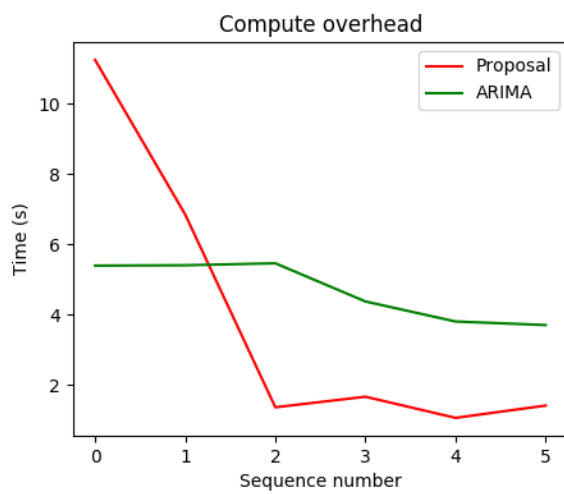
a. expected\_output/hacc/hacc-mpi.png



b. expected\_output/hacc/hacc-pred.png



c. expected\_output/hacc/hacc-time.png



## 7. Example – AMG (./amg/amg\_0.log)

### (1) Configuration

- a. Period: 20 seconds
- b. system-level Infiniband monitoring data is pre-processed (no need to multiply by 4)
- c. Note that we cut the data of an initialization phase that was different from the rest because we assume periodic behavior.

### (2) Execution

```
python seq2seq_predictor.py ./amg/amg_0.log amg
```

### (3) Output printed on the terminal (for prediction accuracy)

The outputs of AMG are similar to the outputs of HACC. Therefore, amg-mpi.png, amg-pred.png and amg-time.png correspond to 2(b), 3(b), and 4(b) in our paper, respectively.

Using TensorFlow backend.

Parsing ./amg/amg\_0.log

AMG: Processed Infiniband data, can be used directly

Total data size = 135, init\_data\_size = 40

TensorFlow's version : 1.0 (or more)

2019-06-02 21:45:50.246266: W tensorflow/python/util/util.cc:297] Sets are not currently considered sequences, but this may change in the future, so consider avoiding using them.

First train, iter = 100, size = 1, loss = 0.110518, training time = 2.724811

Training, iter = 46, loss = 1.329680, training time = 0.993951

Training, iter = 29, loss = 1.826567, training time = 0.641402

Training, iter = 10, loss = 3.216334, training time = 0.243796

Training, iter = 13, loss = 2.164866, training time = 0.313761

DNN MSE: 0.007966

DNN FastDTW: 4.77

Timestep 40, time elapsed: 0.10

Timestep 60, time elapsed: 2.40

Timestep 80, time elapsed: 2.21

Timestep 100, time elapsed: 2.22

Timestep 120, time elapsed: 2.58

ARIMA MSE: 0.016858

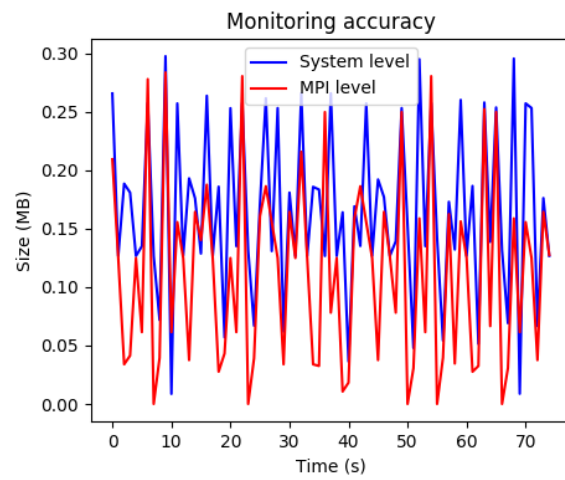
ARIMA FastDTW: 7.14

MPI-Mon MSE: 0.007403

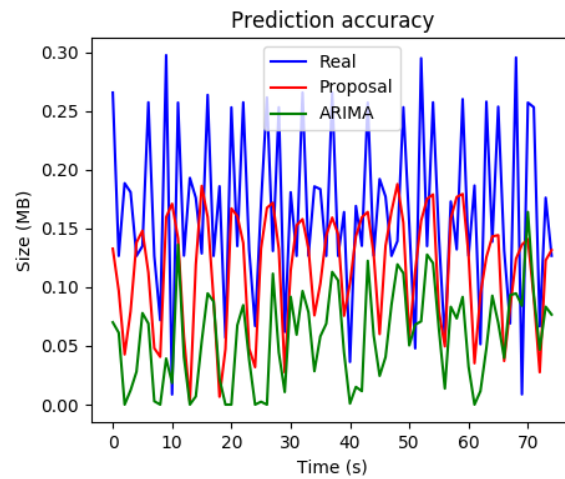
MPI-Mon FastDTW: 4.57

#### (4) Expected output figures

##### a. expected\_output/amg/amg-mpi.png



##### b. expected\_output/amg/amg-pred.png



##### c. expected\_output/amg/amg-time.png

