Featherlight Speculative Task Parallelism (Artifact)

Vivek Kumar

Department of Computer Science and Engineering, IIIT Delhi, India
vivekk@iiitd.ac.in

Abstract. Speculative task parallelism is a widely used technique for solving search based irregular computations such as graph algorithms. Here, tasks are created speculatively to traverse different search spaces in parallel. Only a few of these tasks succeed in finding the solution, after which the remaining tasks are canceled. For ensuring timely cancellation of tasks, existing frameworks either require programmer introduced cancellation checks inside every method in the call chain, thereby hurting the productivity, or provide limited parallel performance.
This artifact includes an implementation of *Featherlight* [1], a new programming model for speculative task parallelism that satisfies the serial elision property and doesn't require any task cancellation checks. We also include the source code of the different benchmarks presented in the paper. The interested reader can use the artifact to experiment with Featherlight implementation.

Keywords: Speculative Parallelism, Async-Finish Programming, Task Cancellation, Managed Runtimes, Work-Stealing.

1 Getting Started

Featherlight was evaluated on a dual socket 10 core Intel Xeon E5-2650 v3 processor running at 2.3 GHz and with a total of 96GB of RAM. The operating system was Ubuntu 16.04.3 LTS. All the scripts attached in this artifact are tested on the same machine and in the bash shell. We have also verified the working of Featherlight on Ubuntu 18.04.2 LTS.

1.1 Software Dependencies

1. JDK 1.6 is required for building our JVM JikesRVM. JDK 1.6 is available at Oracle website for **free** download. You can follow the steps mentioned here for installation: `https://gist.github.com/senthil245/6093389`.
2. ANT 1.9

```
wget http://apachemirror.wuchna.com//ant/binaries/apache-ant-
1.9.14-bin.tar.gz
sudo apt-get remove ant
```

```
sudo tar -xf apache-ant-1.9.14-bin.tar.gz -C /usr/local
sudo ln -s /usr/local/apache-ant-1.9.14 /usr/local/ant
sudo vim /etc/profile.d/ant.sh
#Enter following two lines in the above file:
export ANT_HOME=/usr/local/ant
export PATH=${ANT_HOME}/bin:${PATH}
#Save and quit
source /etc/profile
#Verify
ant -version
Apache Ant(TM) version 1.9.14 compiled on March 12 2019
```

3. autoconf-2.69

```
sudo apt-get install autoconf
```

4. GCC version 4.9 (or 4.8) along with multilib support.

```
sudo apt-get install gcc-4.9
sudo apt-get install gcc-4.9-multilib
sudo apt-get install g++-4.9
#Check if ``gcc -v'' shows 4.9 version
#else do the following
sudo update-alternatives --install \
        /usr/bin/gcc gcc /usr/bin/gcc-4.9 100 \
        --slave /usr/bin/g++ g++ /usr/bin/g++-4.9
sudo update-alternatives --config gcc
#choose the gcc-4.9 in above available configs
```

5. libtool 1.5

```
sudo apt-get install libtool
```

6. bison 3.0.4

```
sudo apt-get install bison
```

7. Perl

```
sudo apt-get install libperl4-corelibs-perl
```

8. Sloccount

```
sudo apt-get install sloccount
```

1.2 Software Installation

1. Directory setup:

```
wget -c https://github.com/hipec/featherlight/archive/278a060.tar.gz
tar -xvf 278a060.tar.gz
cd featherlight-278a060
#Environment setup inside each bash shell
vim ./setup-env.sh
#Change the variable ``FEATHERLIGHT'' such that it
#holds the absolute path to directory ``featherlight-278a060''
#Save and quit
source ./setup-env.sh
```

2. JikesRVM Java Virtual Machine (JVM) installation.

```
#Script will first patch on JikesRVM GitHub version
#``087d300'' and then build (Section 5 in paper)
./build-jvm.sh
#Above command might ask three questions.
#Answer with 'y' as shown below:
Is /home/vivek the right home directory for hippo?
Please type y to confirm > y
Is x86_64-linux the right type for hippo?
Please type y to confirm > y
Is /home/vivek/europar/JikesRVM the correct root directory here?
Please type y to confirm > y
```

3. Building AJWS translator and benchmarks. Note that this step won't work if JikesRVM installation failed.

```
#Script will first patch on ajws GitHub version
#``bd5535f'' and then build (Section 5 in paper)
./build-translator.sh

./build-benchmarks.sh
```

2 Step-by-Step Instructions on how to reproduce the results

Mapping of benchmark name reported in this artifact with those in the paper:

1. SLP = ShortlongPath
2. LinearSearch = ArraySearch
3. lusearch = DacapoLUSFix

2.1 Productivity Analysis (Section 5.1 in the paper)

We have presented two types of productivity analysis in the paper: a) comparing lines-of-code in all four implementations of each benchmark, and b) classroom-based study where we noted the time spent by students in coding different implementations. Here, in this artifact, we are only providing the script to validate the lines-of-code based productivity analysis.

```
cd experiment
./lines-of-code.sh
#Output used for making the
#Table 1 for Lines of Code in the paper:
Benchmark Common_Code Sequential Featherlight ManualAbort ForkJoin
UTS    545    39    39    45    58
NQueens    75    48    48    53    68
Sudoku    469    48    48    54    66
ShortlongPath    558    54    54    60    76
ArraySearch    88    44    44    46    75
TSP    158    55    55    61    84
DacapoLUSFix    143,321    222    222    242    239
```

2.2 Performance Analysis (Section 5.2 in the paper)

1. *Running the seven benchmarks.* As mentioned in Section 5 of the paper, for each benchmark, we ran 30 invocations, with 15 iterations per invocation, where each iteration performed the kernel of the benchmark. Featherlight, ManualAbort, and ForkJoin versions of each benchmark were run with 1, 4, 8, 12, 16, and 20 threads. However, reproducing this same number of invocations and threaded execution will take a lot of time, hence to save time, the scripts we have attached herewith will run 7 invocations instead of 30. Also, the number of threads that will be used with Featherlight, ManualAbort, and ForkJoin versions are 8, 12, 16, and 20. This combination took around 3 hours on the author's experimental machine.
   If you want to evaluate the artifact quickly, then use fewer thread counts and lesser number of invocations. This can be controlled by modifying the variables "INVOCATIONS" and "THREADS" in "experiment/run.sh" script. However, changes must be done accordingly in the following two scripts that calculate speedups: "experiment/speedup-over-ManualAbort.sh" and "experiment/speedup-over-Sequential.sh". All these three scripts have documentations describing the steps to do these changes.

   ```
   cd experiment
   ./run.sh
   ```

2. Speedup of Featherlight over ManualAbort by using 20 threads. This experiment is shown as Figure 3(a) in the paper.

   ```
   ./speedup-over-ManualAbort.sh
   #Output used for plotting the
   #graph Figure 3(a) in the paper:
   ====== Speedup of Featherlight over ManualAbort At Max Thread (=20) ======
   UTS    1.05
   NQueens    0.98
   Sudoku    0.89
   ```

```
ShortlongPath    1.02
ArraySearch    1.03
TSP    0.97
DacapoLUSFix    0.99
```

3. Speedup of benchmarks over their Sequential implementation. Figures 3(b)–3(h) in the paper shows the corresponding plots.

```
./speedup-over-Sequential.sh
#Output used for plotting the
#graphs Figure 3(b)--3(h) in the paper:
====== UTS Speedup over Sequential ======
THREADS    Featherlight    ForkJoin
1    1.12    0.79
4    6.80    1.39
8    9.76    2.73
12    14.08    3.90
16    16.81    4.50
20    19.68    4.31
====== NQueens Speedup over Sequential ======
THREADS    Featherlight    ForkJoin
1    0.95    0.91
4    3.24    2.42
8    5.66    4.65
12    9.12    6.52
16    11.63    8.03
20    12.90    9.21
====== Sudoku Speedup over Sequential ======
THREADS    Featherlight    ForkJoin
1    0.98    3.35
4    5.74    22.52
8    19.44    18.38
12    33.25    20.33
16    47.21    26.04
20    52.77    30.21
====== ShortlongPath Speedup over Sequential ======
THREADS    Featherlight    ForkJoin
1    0.95    0.66
4    2.78    1.70
8    4.28    3.27
12    5.47    4.48
16    6.31    5.48
20    6.85    6.19
====== ArraySearch Speedup over Sequential ======
THREADS    Featherlight    ForkJoin
1    0.95    0.89
4    3.62    3.40
```

```
8    7.43    5.80
12    9.95    8.27
16    18.04    10.77
20    19.72    12.11
====== TSP Speedup over Sequential ======
THREADS    Featherlight    ForkJoin
1    0.99    0.73
4    2.01    2.27
8    3.19    2.86
12    5.21    3.44
16    7.21    4.13
20    8.83    5.39
====== DacapoLUSFix Speedup over Sequential ======
THREADS    Featherlight    ForkJoin
1    0.97    0.83
4    3.19    2.81
8    4.86    4.46
12    5.84    5.48
16    6.23    6.15
20    6.80    5.36
```

References

1. Kumar, V.: Featherlight speculative task parallelism. In: Euro-Par 2019: Parallel Processing. Springer Berlin Heidelberg (2019), to appear