High Performance Computing

Homework #3

Due: Wednesday Sept 22 2021 Before 11:59 PM (Midnight)

Email-based help Cutoff: 5:00 PM on Tue Sept 21 2021

Name: Maciej Wozniak

Description

Instead of this text – Provide a brief description of the objective of the experiment and the design of the micro-benchmarks. What important aspect of programming languages are the benchmarks designed to compare? Why is this aspect important in general for software development?

Experimental Platform

The experiments documented in this report were conducted on the following platform (**Note**: commands to obtain the information are provided in the template version of this report):

Component	***Details***
CPU Model	Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
CPU/Core Speed	2.40GHz
Operating system used	Linux
Name and version of C++ compiler	g++ (GCC) 8.4.0
Name and version of Java compiler	openjdk version "1.8.0_302"
Name and version of Python environment	Python 3.6.8

Runtime Observations

Record the runtime statistics collated from your experiments conducted using the specified command-line arguments below. Job information should be placed in the appendix at the end of this report.

Command-line Argument	C++ (-O3)		C++ (-O3 and PGO)		Java		Python	
	Elapsed time (sec)	Peak memory (KB)	Elapsed time (sec)	Peak memory (KB)	Elapsed time (sec)	Peak memory (KB)	Elapsed time (sec)	Peak memory (KB)
10	0:00.09	2284	0:00.09	2320	0:00.25	36876	0:11.81	14652
10	0:00.05	2288	0:00.06	2320	0:00.18	34452	0:11.41	14656
10	0:00.05	2292	0:00.06	2320	0:00.18	34344	0:11.58	14656
10 (Avg)	**0:00.06**	2288	**0:00.07**	2320	**00:00.2**	35224	**00:11.6**	14654.67
13	0:03.15	3088	0:03.91	4112	0:04.95	55356	16:13.79	64864
13	0:03.15	3084	0:03.90	4116	0:04.97	37400	15:37.85	65068
13	0:03.19	3088	0:03.91	4112	0:04.93	37460	15:31.26	64864
13 (Avg)	**00:03.2**	3086.667	**00:03.9**	4113.333	0:04.95	43405.33	**15:47.6**	64932
15	0:52.99	6160	1:15.80	10256	1:20.25	41420	3:58:18	240172
15	0:52.86	6160	1:15.97	10256	1:20.53	42932	4:03:06	240172
15	0:53.00	6160	1:15.88	10256	1:20.98	58200	4:02:31	240176
15 (Avg)	**00:53.0**	6160	**01:15.9**	10256	**01:20.6**	47517.33	**4:01:13**	240173.3
16	3:48.73	10015	7:12.03	18452	5:24.19	53840	Optional	Optional
16	3:56.39	10252	6:30.66	18448	5:26.73	62692	Optional	Optional
16	4:01.21	10256	6:17.90	18444	5:25.22	69484	Optional	Optional
16 (Avg)	**03:55.4**	10174.33	**06:40.2**	18448	**05:25.4**	62005.33	Optional	Optional

Inferences & Discussions

Now, using the data from tables and the discussion points from the project grading rubric, document your inferences in the space below. Ensure you address all of the 12 aspects mentioned in the grading rubric. The discussion will be critically graded and only the comprehensive / complete discussions will be assigned full score.

The benchmark code we are using in this experiment is Ackermann algorithm. It is recursive method which is call *itself* multiple times. Therefore, instead of just writing for loop for calling it the method millions of times, we can run is once and get similar results (in the case of performance measurement).

We can observe that PGO is not faster than -O3 as we suspected before. It should be faster due to additional optimization being done on the code. PGO is optimizing runtime compiling and helps with program profiling.

The runtime in all the cases increases exponentially, when memory footprint has more linear pattern. We can see that the highest memory footprint and running time occurred in case of python due to JIT compilation.

Three programs were relatively similar and very easy to implement.
Inside each benchmark, methods were called by the command line:

- Cpp : **./ackerman**
- Cpp + pgo: **/ackerman_pgo**
- Java: **java -Xss20m Ackermann**
- Python **python3 ./ackermann.py**

Results summary.
We can observe that Cpp outperform any other language. It is mainly caused that Java and especially Python are based on C code. It causes additional compilation step where python has to be transform into C and then compile. We can see that due to these aspect python cannot really be used in heavily computational applications with small memory and computational power. From the other hand C++ can be used in the task such as gaming, thanks to the fast compilation and execution time. From the other hand, Python is easily understood and comes with many handy functions and libraries, which allows users to easily manipulate and analyze data. Therefore, Python is often used in these types of jobs/tasks, where the computational power and memory is not crucial. Finally Java is somewhere in the middle with performance and being user friendly. Thanks to that it is popular among many mobile and web app developers all around the world.

Appendix

Copy-paste the SLRUM job script that you used to collect the runtime statistics in the space below:

```
#!/bin/bash

#SBATCH --account=PMIU0184
#SBATCH --job-name=hw3_cpp_java
#SBATCH --time=24:00:00
#SBATCH --mem=4GB
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1

# To keep the benchmarks more upto date, we load the more recent
# versions of the tools available on the cluster
module load gcc-compatibility/10.3.0 java/11.0.8 python/3.7-2019.10

# Function to run given command 3 times 10 13
function run3() {
    cmd="$*"
    for arg in 10 13 15 16; do
        echo "-----------------------------------------"
        echo "Running 3 reps of ${cmd} ${arg}"
        for rep in `seq 1 3`; do
           /usr/bin/time python3 ./ackermann.py ${arg}
        done
    done
}

# This script assumes the benchmarks have been precompiled.

echo "================================================"
run3 ./ackermann

echo "================================================"
run3 ./ackermann_pgo

echo "================================================"
run3 java -Xss20m ackermann
```

separate job for python so it runs faster

```
#!/bin/bash

#SBATCH --account=PMIU0184
#SBATCH --job-name=hw3_cpp_java
#SBATCH --time=24:00:00
#SBATCH --mem=4GB
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1

# To keep the benchmarks more upto date, we load the more recent
# versions of the tools available on the cluster
module load gcc-compatibility/10.3.0 java/11.0.8 python/3.7-2019.10

# Function to run given command 3 times 10 13
function run3() {
    cmd="$*"
    for arg in 10 13 15 16; do
        echo "-------------------------------------------"
        echo "Running 3 reps of ${cmd} ${arg}"
        for rep in `seq 1 3`; do
           /usr/bin/time python3 ./ackermann.py ${arg}
        done
    done
}
run3
```