Lecture 22 - Profiling Basics

DSE 512

Drew Schmidt
2022-04-19

From Last Time

- Homework 3
    - Graded
    - Let's talk about it
- Homework 4
    - Assigned
    - Let's talk about it

Where We've Been

Module 1: Basic Cloud and HPC

- Lecture 1 - Introduction
- Lecture 2 - Overview of HPC and the Cloud
- Lecture 3 - Introduction to Remote Computing
- Lecture 4 - Introduction to Containers
- Lecture 5 - Introduction to ISAAC
- Lecture 6 - MPI and Singularity

Where We've Been

Module 2: Performance Optimization

- Lecture 7 - Introduction to Performance Optimization
- Lecture 8 - High Level Language Optimizations
- Lecture 9 - Computational Linear Algebra Part 1
- Lecture 10 - Computational Linear Algebra Part 1
- Lecture 11 - GPGPU (The Easy Parts) Part 1
- Lecture 12 - GPGPU (The Easy Parts) Part 2
- Lecture 13 - Utilizing Compiled Code
- Lecture 14 - I/O

Where We've Been

Module 3: Parallelism

- Lecture 15 - Introduction to Parallelism
- Lecture 16 - Forks and Threads Part 1
- Lecture 17 - Forks and Threads Part 2
- Lecture 18 - MPI Part 1
- Lecture 19 - MPI Part 2
- Lecture 20 - MPI Part 3
- Lecture 21 - MapReduce

Where We're Headed

Module 4: Profiling

- Lecture 22 - Profiling Basics
- Lecture 23 - HLL Profiling
- Lecture 24 - Advanced Profiling (Hardware and MPI)

Module 5: Deep Learning

- Lecture 25 - Basic Intro
- Lecture 26 - DL for Practitioners
- Lecture 27 - Distributed Training

Profiling Basics

Profiling

- Gathering information
- Can do "profiling" on lots of different things
    - Customer profiling
    - Profiling in policing
    - ...
- We are interested in *performance profiling*

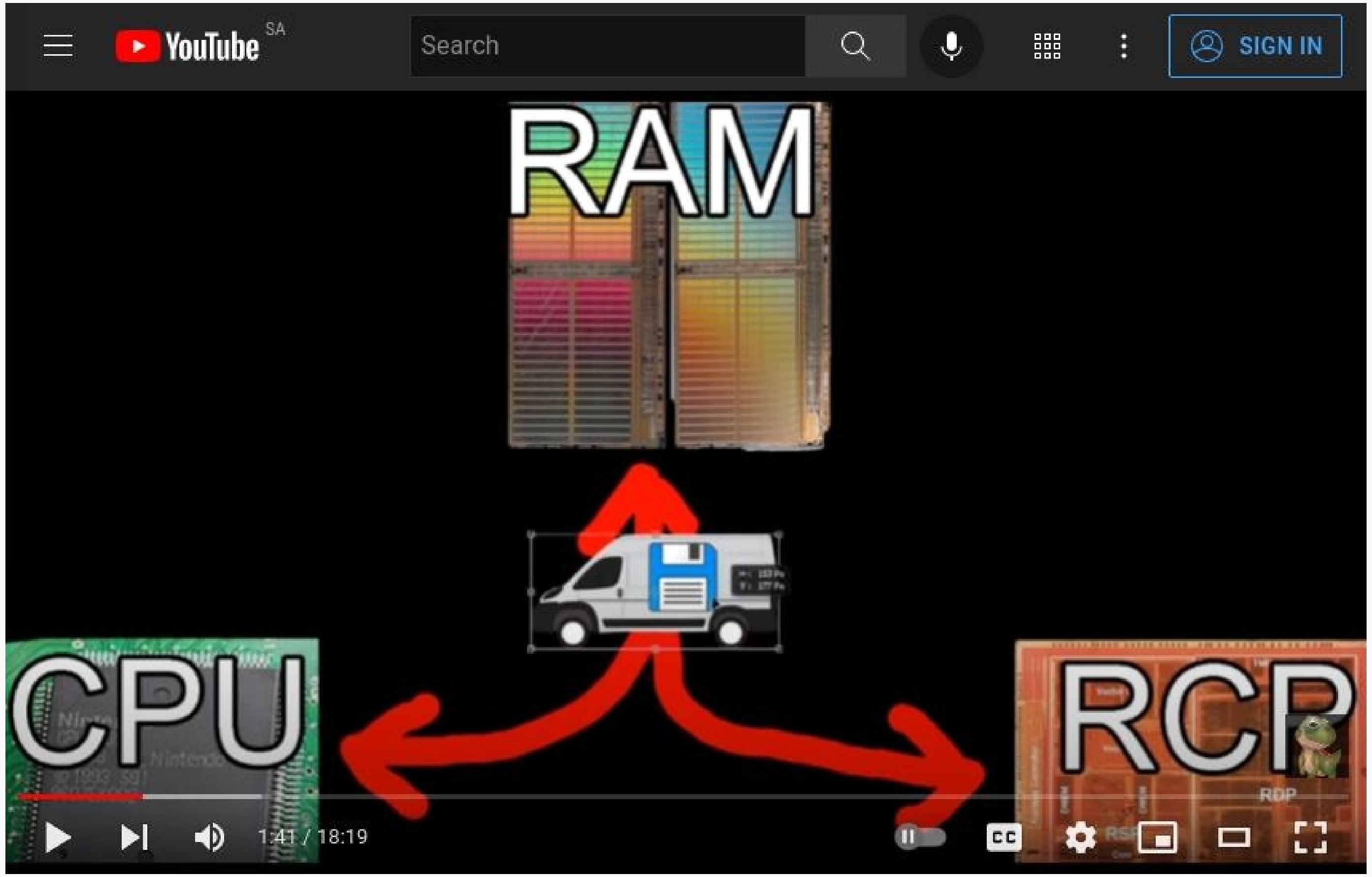
How Does Profiling Work?

Performance Profiling Examples

- Software profiling
  - a simple timer
  - line profiling
- API profiling
  - MPI operations
  - Tensorflow profiler
- Hardware profiling
  - Memory profiler
  - Hardware counters
  - CUDA

What Can Profiling Do?

FIXING the ENTIRE SM64 Source Code (INSANE N64 performance)



https://www.youtube.com/watch?v=t_rzYnXEQlE

Language Timers

- R
  - `system.time()`
  - `Sys.time()`
- Python's many `time` utilities
  - `time.perf_counter()`
  - ...

time

```
time Rscript -e "1+1"
```

```
## [1] 2
##
## real	0m0.287s
## user	0m0.484s
## sys	0m0.838s
```

```
time Rscript -e "x = runif(1e8)"
```

```
##
## real	0m2.203s
## user	0m2.198s
## sys	0m0.991s
```

proginfo

Project home https://github.com/wrathematics/proginfo

- Uses sampling
- Reports basic CPU and GPU info
- `MIN/MEAN/MAX (SD) / TOTAL`

```
./proginfo Rscript -e '1+1'
```

```
[1] 2

## Program Info (from 27 polls)
  CPU
  - Wall time: 0.280
  - Utilization: 3.184%
  - RAM: 19.109/19.142/19.167 (62.791) / 62.810 GiB
  GPU (CUDA=11.4 Driver=470.103.01)
  - Utilization
      + (Device 0) 6/8.96/11 (2.50) / 100%
  - RAM:
      + (Device 0) 0.579/0.579/0.579 (0.000) / 7.926
```

proginfo

Memory example

```
./proginfo Rscript -e 'x=runif\(1e9\)'
```

```
## Program Info (from 1890 polls)
  CPU
  - Wall time: 19.454
  - Utilization: 2.327%
  - RAM: 19.147/22.921/26.671 (60.604) / 62.810 GiB
  GPU (CUDA=11.4 Driver=470.103.01)
  - Utilization
      + (Device 0) 0/0.50/1 (0.50) / 100%
  - RAM:
      + (Device 0) 0.554/0.554/0.554 (0.000) / 7.926 GiB
```

GPU example

```
./proginfo Rscript -e 'suppressMessages(library(fmlr)); c = card(); x = gpumat(c, 25000,
```

```
# gpumat 25000x25000 type=f

## Program Info (from 381 polls)
  CPU
  - Wall time: 3.949
  - Utilization: 2.606%
  - RAM: 19.122/19.359/19.509 (62.692) / 62.810 GiB
  GPU (CUDA=11.4 Driver=470.103.01)
  - Utilization
      + (Device 0) 0/1.94/27 (4.76) / 100%
  - RAM:
      + (Device 0) 0.554/0.673/3.105 (0.189) / 7.926 GiB
```

Basic Memory Profiling

- Can be quite sophisticated
    - valgrind
    - gdb
- Or also very simple

```r
library(memuse)
Sys.procmem()
```

```
## Size:  95.207 MiB
## Peak:  95.207 MiB
```

```r
x = runif(1e8)
mu(x)
```

```
## 762.939 MiB
```

```r
rm(x);invisible(gc())
Sys.procmem()
```

```
## Size:   95.477 MiB
## Peak:  858.188 MiB
```

Line Profiling

```r
m = 10000
n = 250
x = matrix(rnorm(

Rprof()
pca = prcomp(x)
Rprof(NULL)

summaryRprof()
```

```
$by.self
                self.time self.pct total.time total.pct
"La.svd"             0.68    69.39       0.72     73.47
"%*%"                0.12    12.24       0.12     12.24
"aperm.default"      0.04     4.08       0.04      4.08
"array"              0.04     4.08       0.04      4.08
"matrix"             0.04     4.08       0.04      4.08
"sweep"              0.02     2.04       0.10     10.20
### output truncated by presenter

$by.total
                 total.time total.pct self.time self.pct
"prcomp"               0.98    100.00      0.00     0.00
"prcomp.default"       0.98    100.00      0.00     0.00
"svd"                  0.76     77.55      0.00     0.00
"La.svd"               0.72     73.47      0.68    69.39
### output truncated by presenter

$sample.interval
[1] 0.02
```

Line Profiling

```r
m = 10000
n = 250
x = matrix(rnorm(

Rprof(interval=.9
pca = prcomp(x)
Rprof(NULL)

summaryRprof()
```

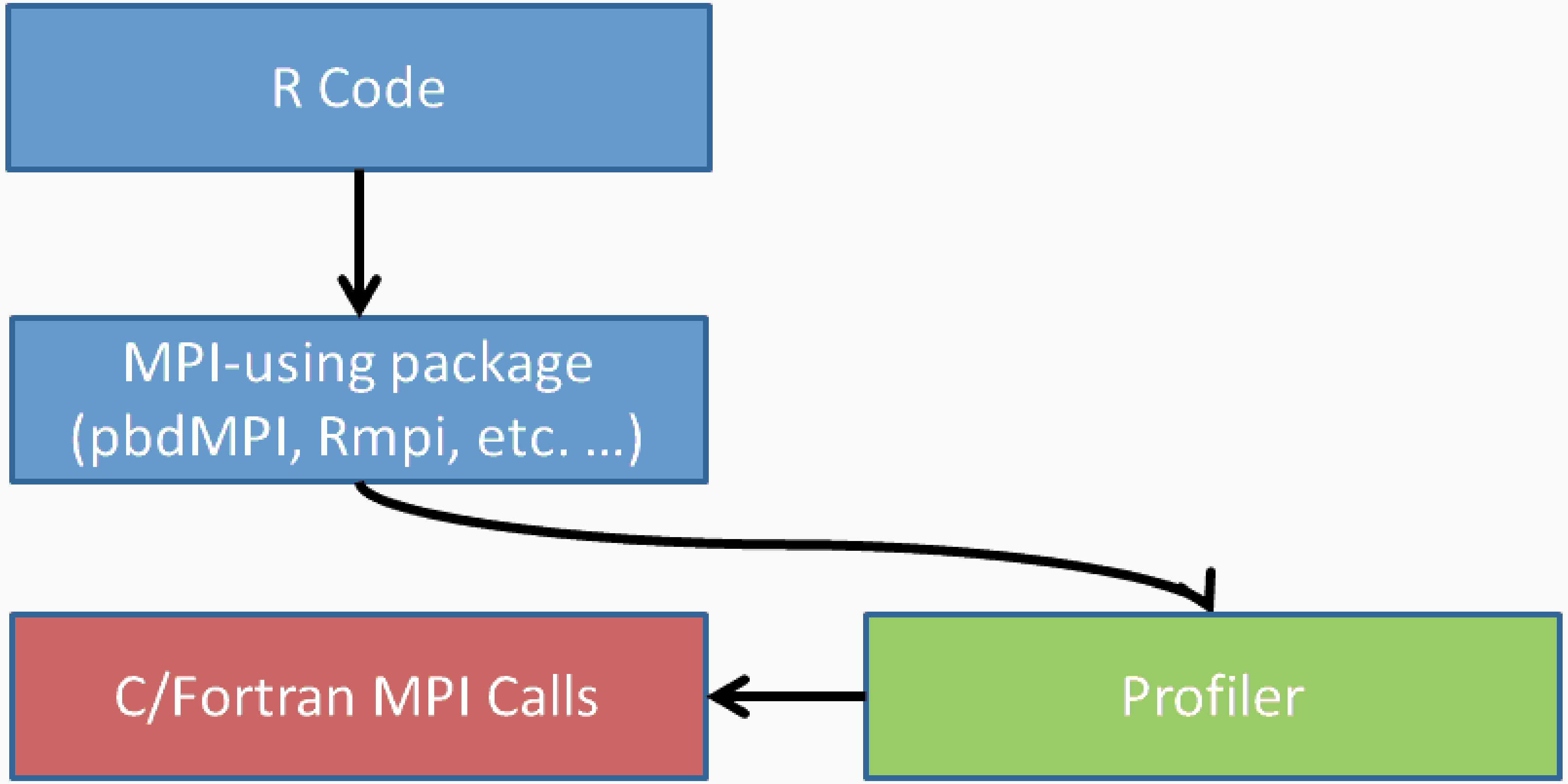
```
$by.self
[1] self.time   self.pct    total.time total.pct
<0 rows> (or 0-length row.names)

$by.total
[1] total.time total.pct  self.time   self.pct
<0 rows> (or 0-length row.names)

$sample.interval
[1] 0.99

$sampling.time
[1] 0
```

Profiling Overhead

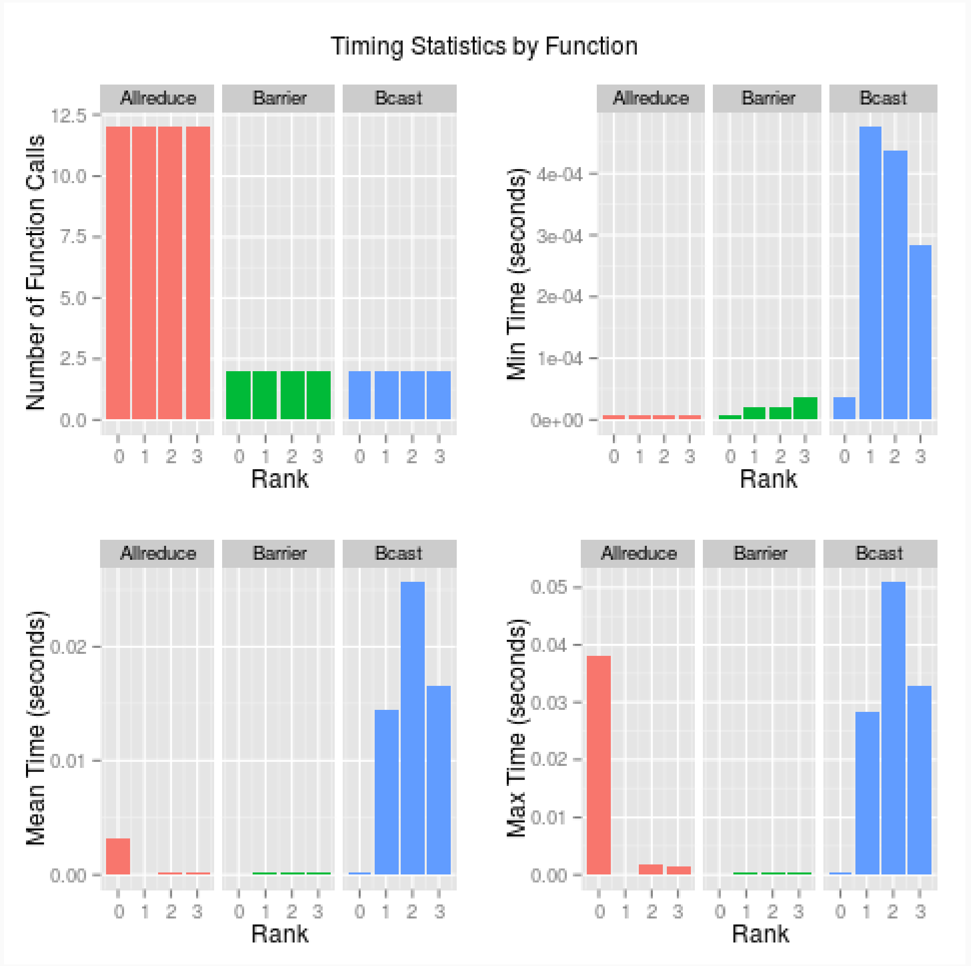
Hardware counters

- Cache misses
  - Data cache
  - Instruction cache
- Flops
- Others

MPI Profiling



MPI Profiling



Questions?