Lecture 25 - Advanced Profiling

DSE 512

Drew Schmidt
2022-04-28

From Last Time

- Homework 4
  - Assigned
  - Due April 30
  - Extensions very unlikely
- Homework 5
  - Covers profiling
  - Smaller
  - Assigned next week?
  - Due May 10?
- No Homework 6 (Deep Learning)
- Questions?

Advanced Profiling

Advanced Profilers

- Hardware counter profiling
- MPI profilers
- GPU (CUDA) profilers

Hardware Counter Profiling

- Modern hardware (CPU, GPU) collects *hardware counters*
- Counter: number of occurrences
- Hardware counter: Number of times hardware event occurs
- Performance Application Programming Interface (PAPI) https://icl.utk.edu/papi/

Hardware Counter Examples

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

An Example

We're talking about SVD again!

PCA

- Input matrix A
  - Compute SVD of A
  - Project A onto V
- Number of floating point operations:
  - SVD requires $6mn^2 + 20n^3$ ops
  - projection requires $2mn^2$ ops
  - Source: Golub, G.H. and Van Loan, C.F., 2013. Matrix computations. JHU press.
  - Also requires mean-centering: $2mn + n$ ops
- TOTAL: $6mn^2 + 20n^3 + 2mn^2 + 2mn + n$

PCA FLOPs

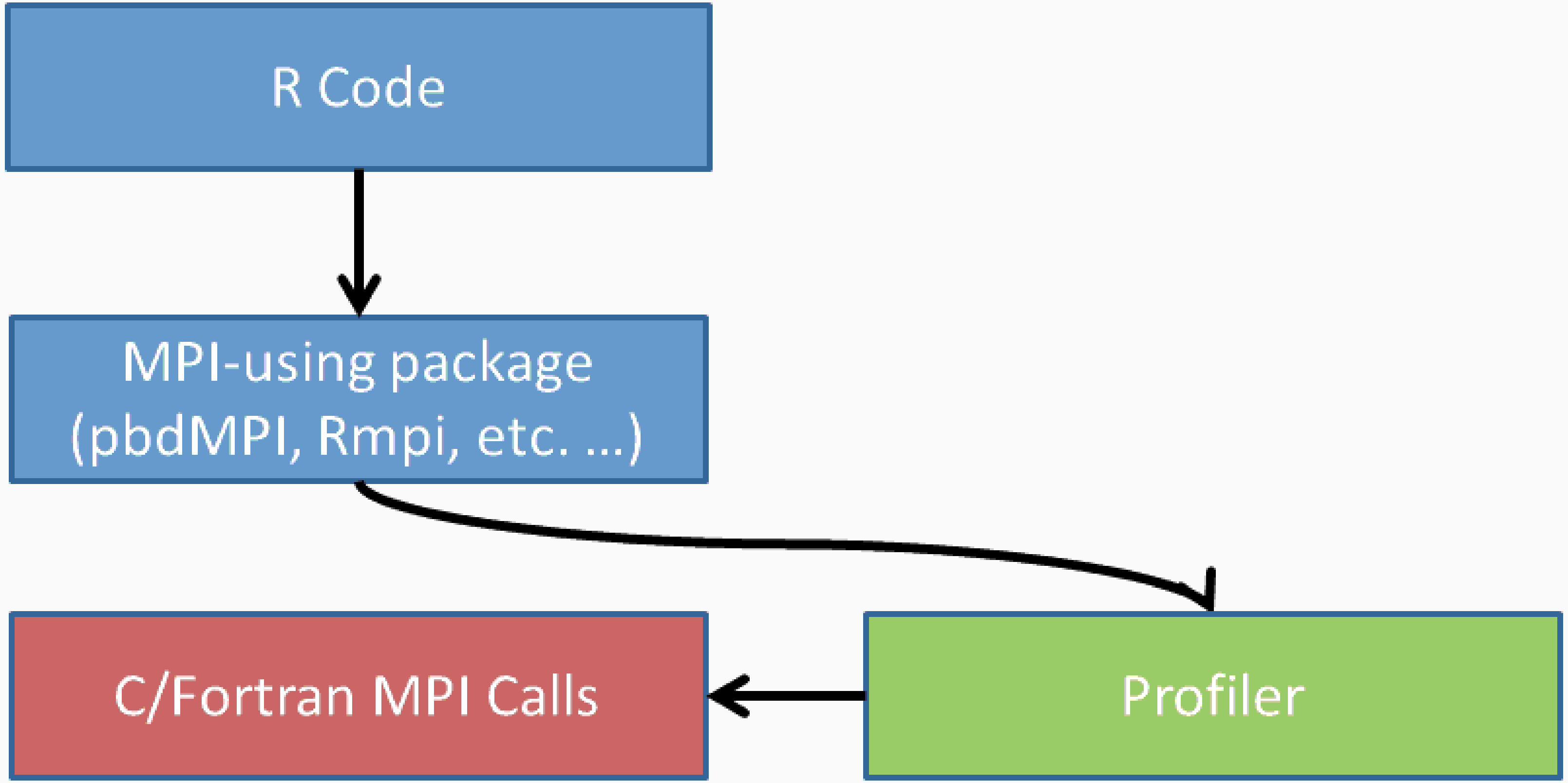
```r
m = 10000
n = 50
A = matrix(rnorm(m*n), m, n)
ret = prcomp(A)

pbdPAPI::system.flops(prcomp(A))
```

```
      m  n  measured theoretical difference pct.error   mflops
1 10000 50 212538720   203500050    9038670      4.25  2284.26
```

MPI Profilers

- fpmpi
  - Easy to install
  - Least useful
- mpiP
  - Relatively easy to install
  -
- tau
  - *Very* hard to install
  - Profiles *everything*

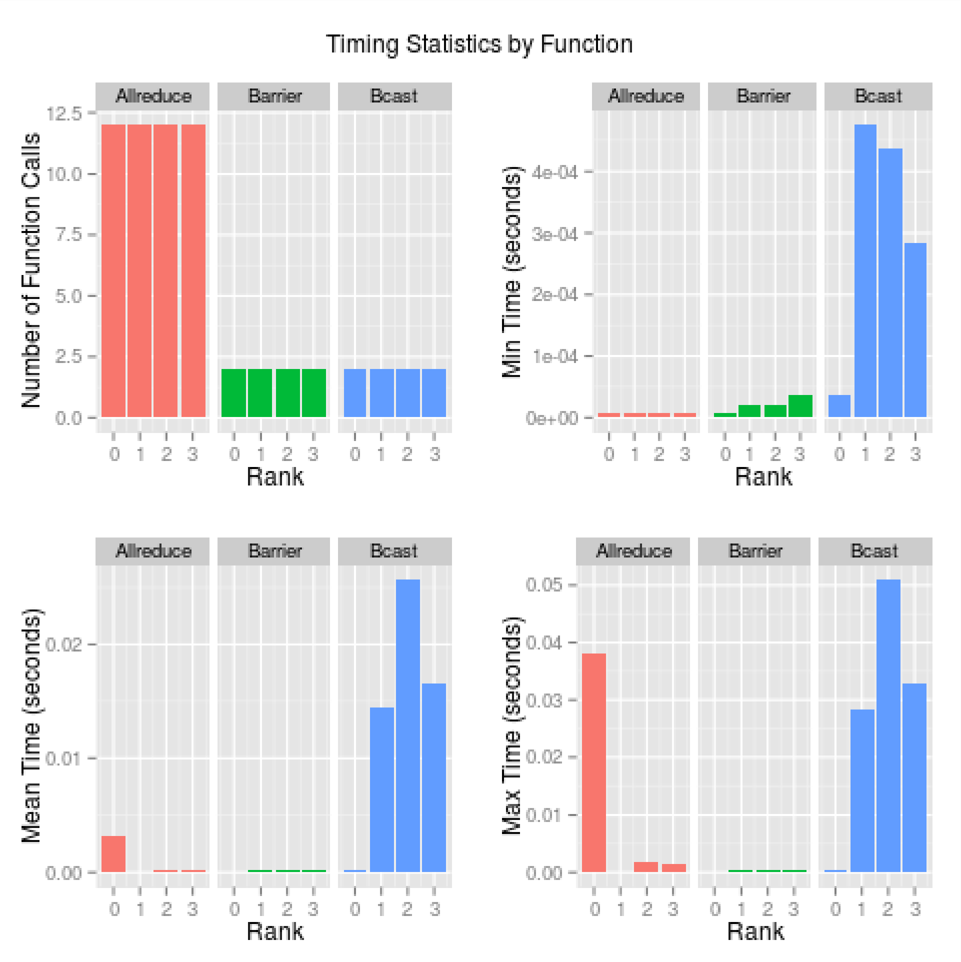
MPI Profiling



MPI Profiling Steps

- Build mpiP
- Rebuild your MPI program
  - Binary, pbdMPI, mpi4py, ...
  - Link with mpiP
- Run program
- Read profiler data

MPI Profiling

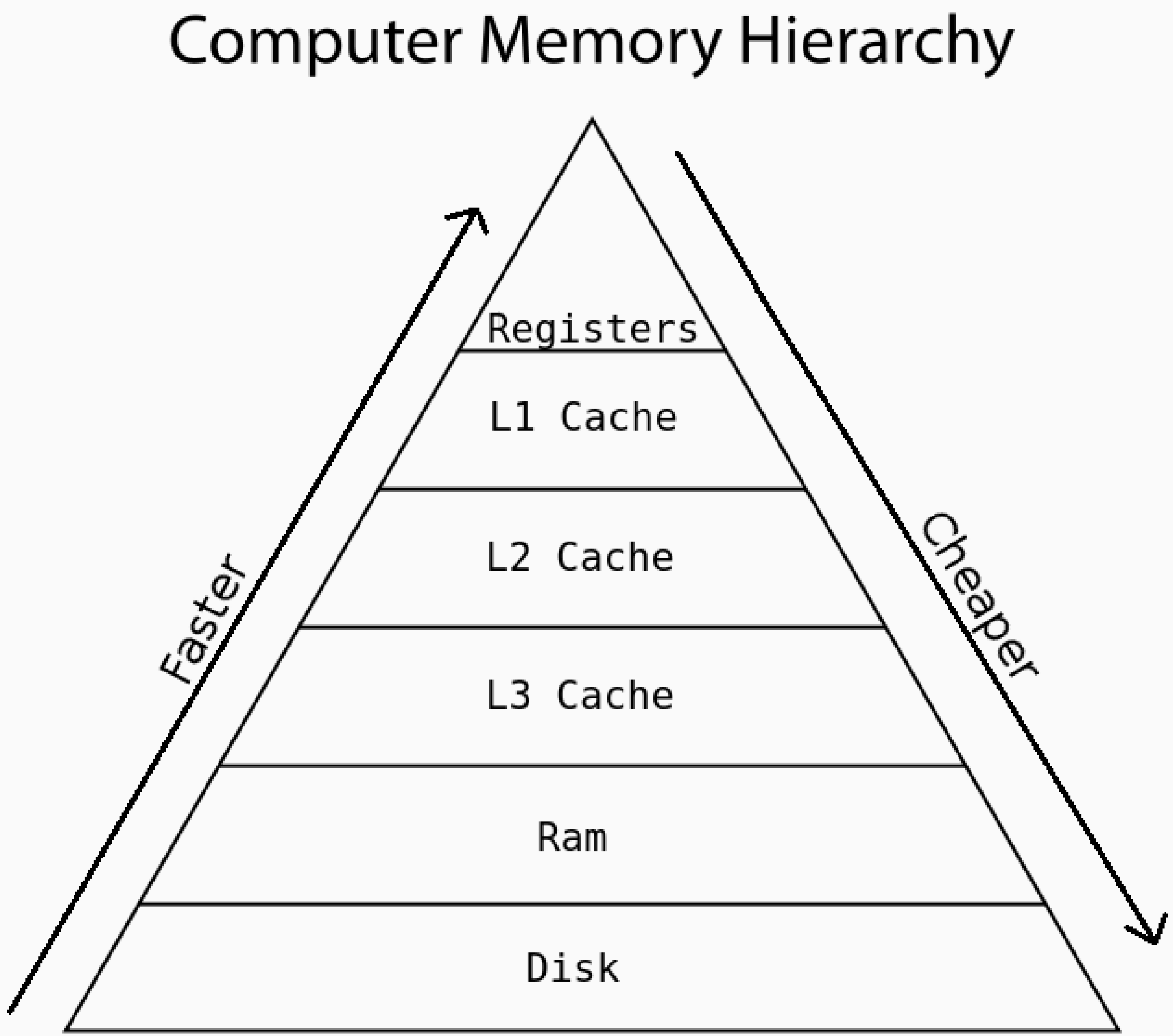


Cache

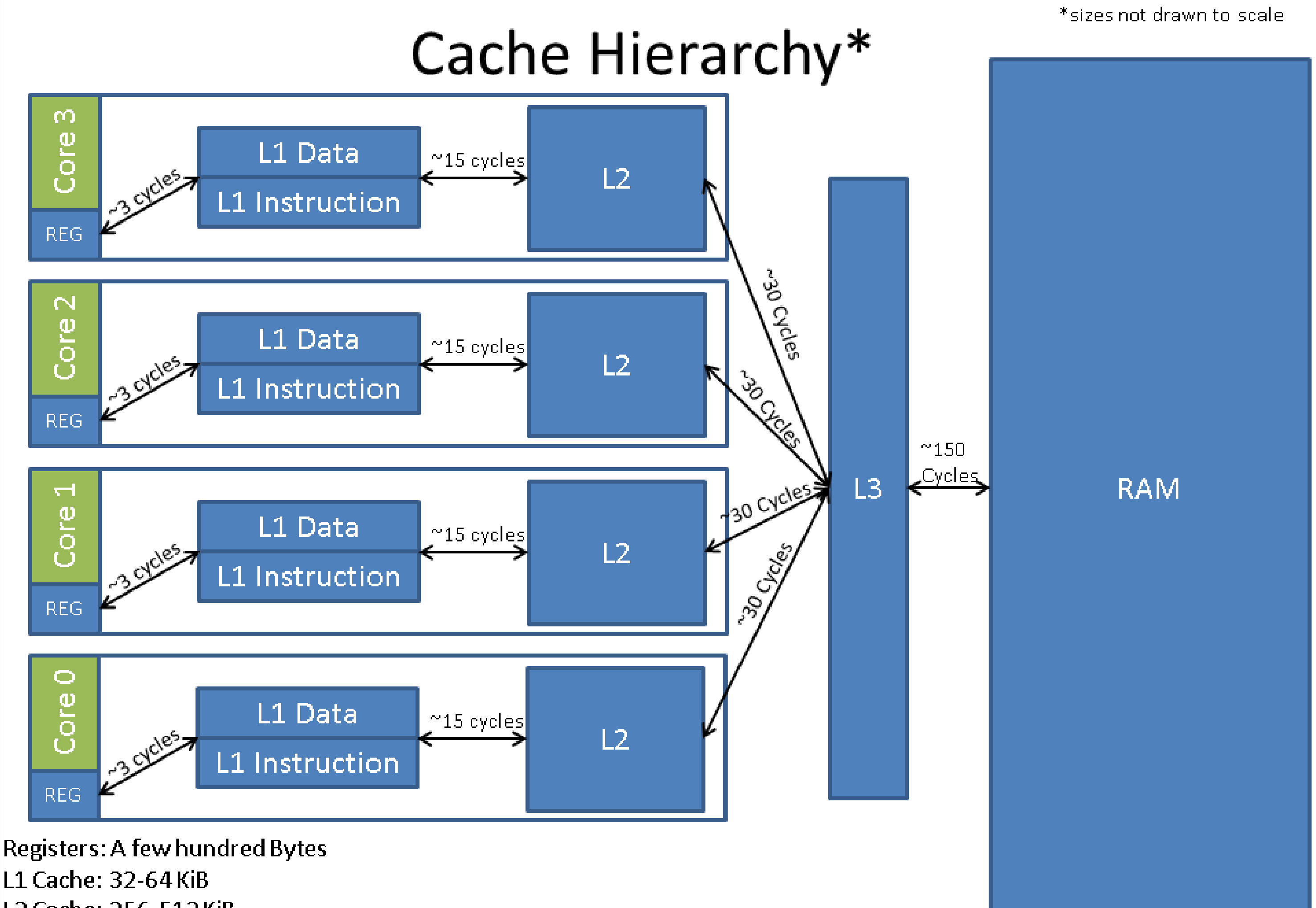
Computer Hardware

- We have to talk a bit about computer hardware
- Goal is not to become experts
- Need to be aware of some realities

Memory Hierarchy



Cache



Registers: A few hundred Bytes
L1 Cache: 32-64 KiB
L2 Cache: 256-512 KiB
L3 Cache: 4-8 MiB
Ram: 8-16 GiB

References:
http://www.eecs.berkeley.edu/~rcs/research/interactive_latency.html
http://www.7-cpu.com/cpu/SandyBridge.html

Cache Sizes

```r
library(memuse)

Sys.cachesize()
```

```
## L1I:   64.000 KiB
## L1D:   32.000 KiB
## L2:   512.000 KiB
## L3:    16.000 MiB
```

```r
Sys.cachelinesize()
```

```
## Linesize:  64 B
```

Recall: Homework 3

Filling matrix $A = [a_{ij}]_{n\times n}$ where

$$a_{ij} = \exp\left(-\frac{i+j}{ij}\right)$$

Homework 3

R

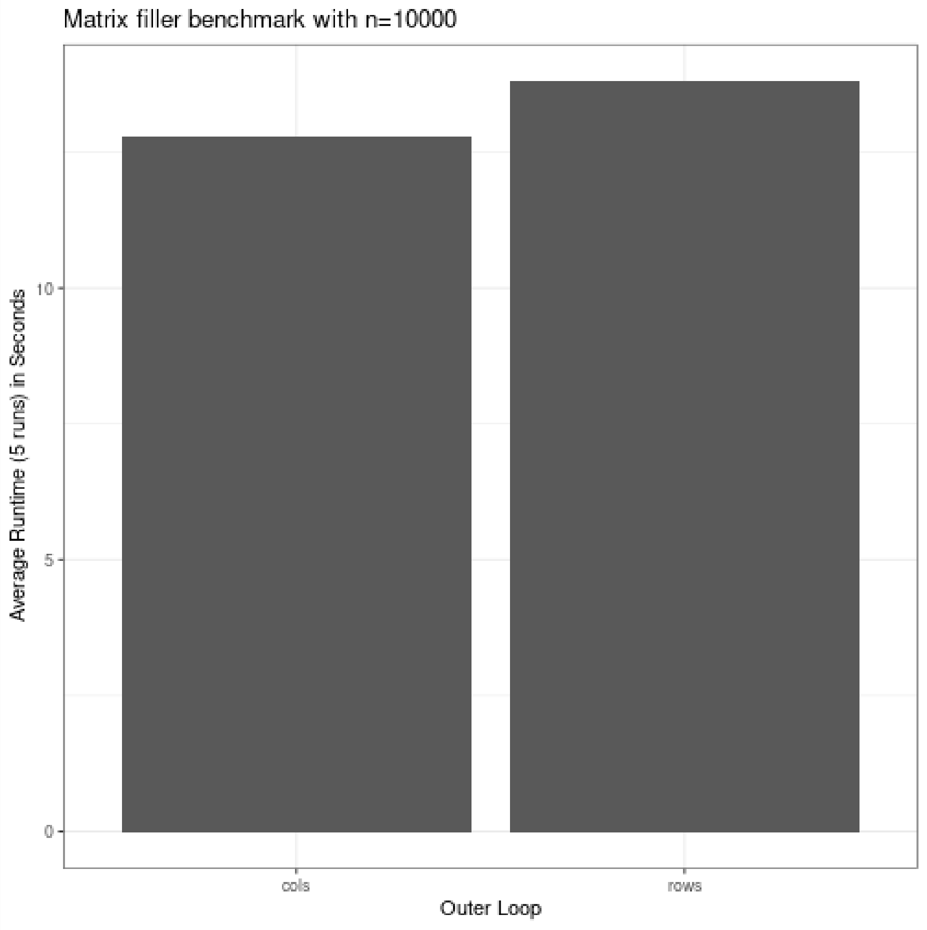
```r
rf = function(n){
  A = matrix(0, n, n)
  for (i in 1:n)
    for (j in 1:n){
      A[i, j] = exp(-(i+j)/(i*j))
  }
  A
}

cf = function(n){
  A = matrix(0, n, n)
  for (j in 1:n){
    for (i in 1:n)
      A[i, j] = exp(-(i+j)/(i*j))
  }
  A
}
```

Python

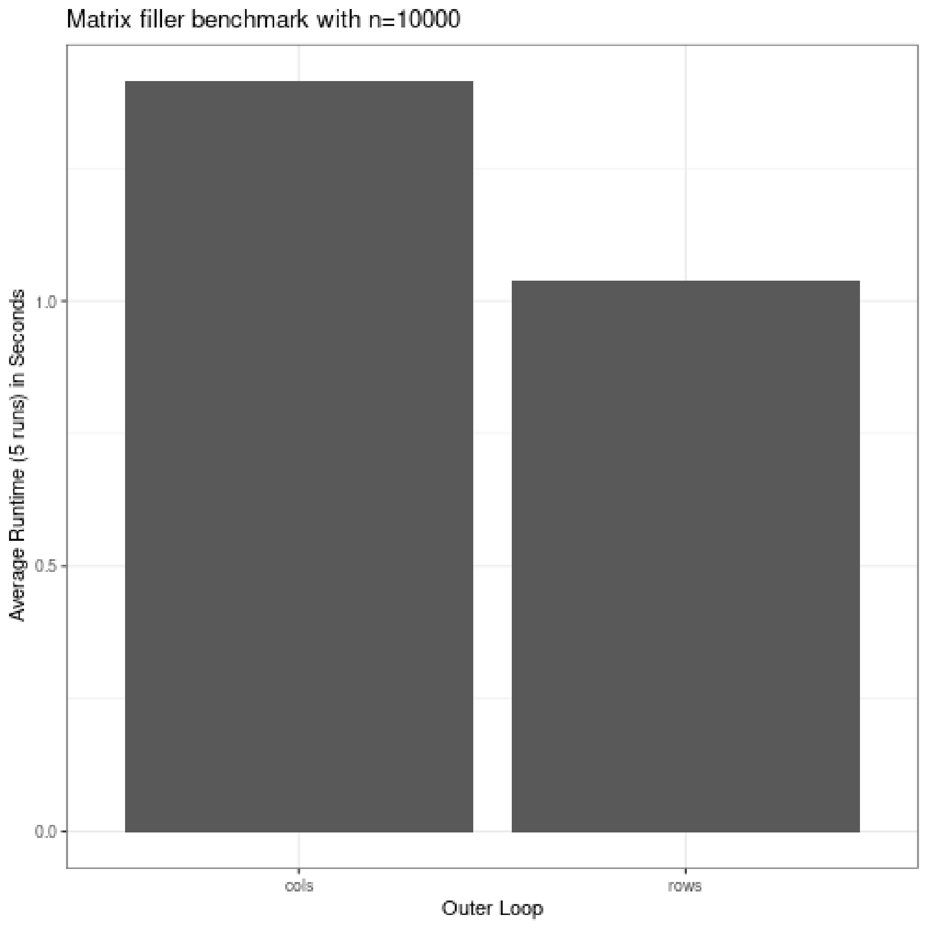
```python
@jit(nopython = True)
def rf(n):
  A = np.zeros((n, n))
  for i in range(0, n):
    for j in range(0, n):
      A[i, j] = math.exp(-((i+1)+(j+1))/(
  return A

@jit(nopython = True)
def cf(n):
  A = np.zeros((n, n))
  for j in range(0, n):
    for i in range(0, n):
      A[i, j] = math.exp(-((i+1)+(j+1))/(
  return A
```

Homework 3

R



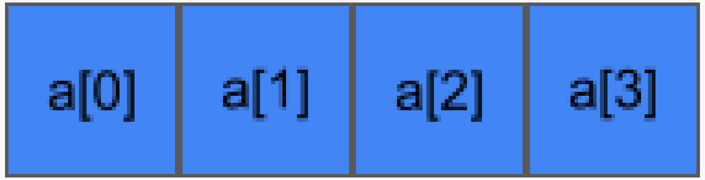
Python (with Numba)



Why Would It Matter?

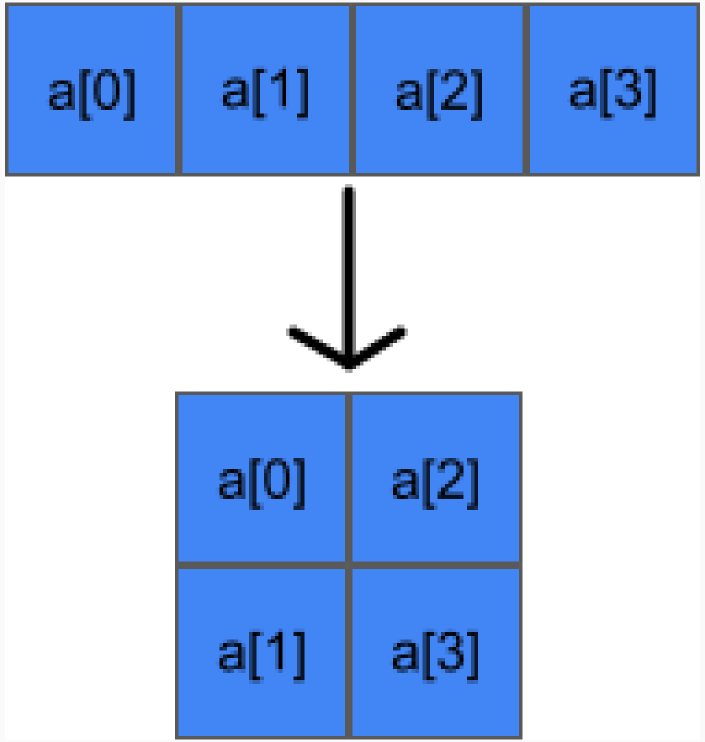
Matrices

- Computers don't have a concept of "matrix"
- Computers *do* have a concept of "array"
- Array: A contiguous block of memory



Matrices

- Matrices are (almost always) just arrays
  - Can be arrays of arrays
  - This is usually not a good idea
  - I don't want to get into it
- Ordering (column-major / row-major)
- Col: `A[i, j] = A[i + m*j]`
- Row: `A[i, j] = A[i*n + j]`



Looping "Wrong"

- CPUs will fetch data to cache(s) that it expects you'll need
- A cache "miss" means data is not in cache
  - have to go get it from RAM
  - RAM → L3 → L2 → L1
- A "miss" is not inherently bad
  - If everything fit in cache, GREAT!
  - Probably it won't
  - In that case, *there will be cache misses*
- What you *don't* want is **unnecessary** misses

Another Example

We're simulating π again!

Implementations

Rcpp

```
double mcpi_rcpp(const int n){
  int i, r = 0;
  double u, v;
  Rcpp::RNGScope scope;

  for (i=0; i<n; i++){
    u = R::runif(0, 1);
    v = R::runif(0, 1);
    if (u*u + v*v <= 1)
      r++;
  }

  return (double) 4.0*r/n;
}
```

C

```
SEXP mcpi_c(SEXP n_){
  SEXP ret;
  int i, r = 0;
  const int n = INTEGER(n_)[0];
  double u, v;

  GetRNGstate();
  for (i=0; i<n; i++){
    u = unif_rand();
    v = unif_rand();
    if (u*u + v*v <= 1)
      r++;
  }
  PutRNGstate();

  PROTECT(ret = allocVector(REALSXP, 1));
  REAL(ret)[0] = (double) 4.0*r/n;
  UNPROTECT(1);
```

Benchmark

```r
library(Rcpp)

sourceCpp(code=Rcpp_code)
sourceCpp(code=C_code)

n = 1e6
rbenchmark::benchmark(mcpi_c(n), mcpi_rcpp(n))
```

```
##             test replications elapsed relative user.self sys.self user.child
## 1      mcpi_c(n)          100   1.609    1.000     1.598    0.004          0
## 2 mcpi_rcpp(n)          100   3.099    1.926     3.084    0.001          0
```

Data Cache

```
pbdPAPI::system.cache(mcpi_c(n))
```

```
## Level 1 Cache Misses: 1358
## Level 2 Cache Misses: 720
## Level 3 Cache Misses: 400
```

```
pbdPAPI::system.cache(mcpi_rcpp(n))
```

```
## Level 1 Cache Misses: 1316
## Level 2 Cache Misses: 747
## Level 3 Cache Misses: 248
```

Instructions

Total instructions executed

```r
pbdPAPI::system.event(mcpi_c(n), events="PAPI_TOT_INS")
```

```
## Instructions Completed: 131035426
```

```r
pbdPAPI::system.event(mcpi_rcpp(n), events="PAPI_TOT_INS")
```

```
## Instructions Completed: 227028352
```

Instruction Cache

Instruction cache misses

```
pbdPAPI::system.event(mcpi_c(n), events="PAPI_L1_ICM")
```

```
## Level 1 Instruction Cache Misses: 604
```

```
pbdPAPI::system.event(mcpi_rcpp(n), events="PAPI_L1_ICM")
```

```
## Level 1 Instruction Cache Misses: 873
```

Wrapup

Wrapup

- Profiling is information gathering
  - runtimes
  - memory consumption
  - cache misses
  - MPI comms
- The more you understand your computer's memory hierarchy, the faster your programs will be.
- That's it for profiling!
- Deep learning next

Questions?