Progress Report

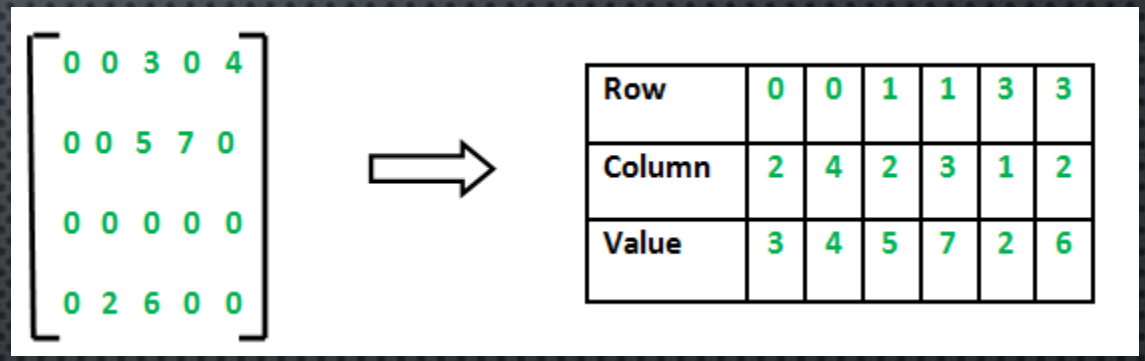
Student:何祁恩
Advisor: Prof. 賴伯承, Yuhao Fang

2021/10/13

Coding Part

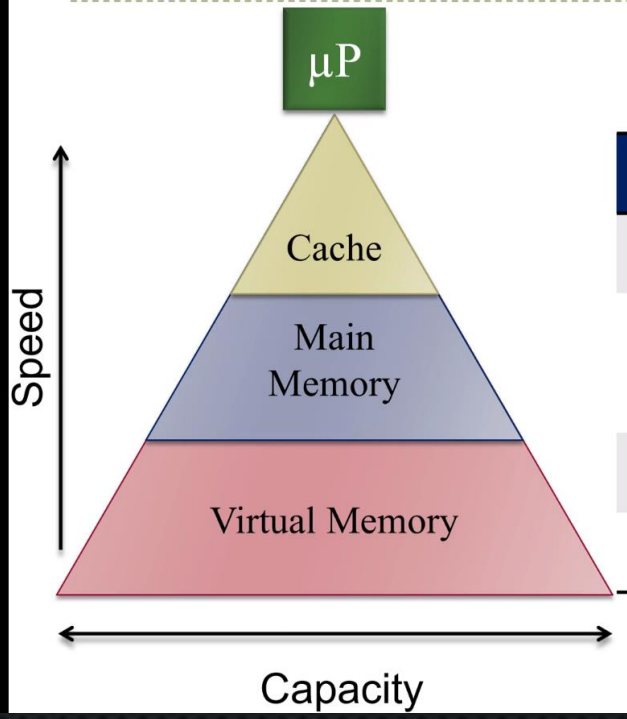
Code sparse matrix multiplication and transpose.

To decrease the memory use, I only store the non-zero term value and its row and column into the struct.



Computer Organization Part

Solution: The Memory Hierarchy



	Speed	Bit Area	Cost/GB
SRAM	~1ns	Large (≥6T)	~$10k
DRAM	~10-50ns	Small (1T & capacitor)	~$10
Disk (SSD)	~10^5ns	Very small	$1
Disk (HDD)	~10^7ns	Very small	$0.1

Numbers roughly from 2012

	Speed	Bit Area	Data lifetime	Cost/bit
SRAM	Fast (Larger arrays are slower)	Large (≥6T)	Data stable while powered	Expensive
DRAM	Slow (Multiple steps)	Small (1T & capacitor)	Must be periodically refreshed	Inexpensive
Disk	Very Slow (mechanical)	Very small	Non volatile	Very cheap

Fast cheap big capacity
trade off: memory hierarchy

Fast cache near microprocessor,
the slower the speed to access
the memory, the further distance
to the microprocessor.

Computer Organization Part

DRAM

1. Know what exactly how to access the memory read and write the data
2. Cell + Binary decoder + Multiplexer + Demultiplexer to read and write the data

Computer Organization Part

Spatial Locality:

If used data recently, likely to use data close to it soon.

Temporal Locality:

After compile the code to assembly code, if instruction used recently, likely to use it soon.

Locality

Two method to construct the 2d-array(matrix):

```
//first method
int matrix1[ROW][COL];
```

```
//second method
int **matrix2 = malloc(sizeof(int *)*ROW);
for (int i = 0; i < ROW;i++){
    matrix2[i] = malloc(sizeof(int) * COL);
}
```

The address of the element :

```
matrix1 address:
181370368  181370372  181370376  181370380  181370384
181370388  181370392  181370396  181370400  181370404
181370408  181370412  181370416  181370420  181370424
181370428  181370432  181370436  181370440  181370444
181370448  181370452  181370456  181370460  181370464

matrix2 address:
3288442592  3288442596  3288442600  3288442604  3288442608
3288442624  3288442628  3288442632  3288442636  3288442640
3288442656  3288442660  3288442664  3288442668  3288442672
3288442688  3288442692  3288442696  3288442700  3288442704
3288442720  3288442724  3288442728  3288442732  3288442736
```

The address in both method in same row are continuous, and non-continuous in same column.

And there is something different between the red circle side.

Locality

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

#define ROWS 100000
#define COLS 10000

int matrix[ROWS][COLS];

int main(){

    int i, j;

    for (i = 0; i < ROWS;i++){
        for (j = 0; j < COLS;j++){
            matrix[i][j] = rand();
        }
    }

    //get the sum of all elements in matrix
    int64_t sum = 0;
    for (i = 0; i < ROWS;i++){
        for (j = 0; j < COLS;j++){
            sum+= matrix[i][j];
        }
    }
    printf("sum = %ld\n", sum);
    return 0;
}
```

byrow.c

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

#define ROWS 100000
#define COLS 10000

int matrix[ROWS][COLS];

int main(){

    int i, j;

    for (i = 0; i < ROWS;i++){
        for (j = 0; j < COLS;j++){
            matrix[i][j] = rand();
        }
    }

    //get the sum of all elements in matrix
    int64_t sum = 0;
    for (j = 0; j < COLS;j++){
        for (i = 0; i < ROWS;i++){
            sum+= matrix[i][j];
        }
    }
    printf("sum = %ld\n", sum);
    return 0;
}
```

bycol.c

```
CC = gcc
CFLAGS = -g -Wall
OBJ = byrow bycol

all: $(OBJ)

%: %.c
    $(CC) $(CFLAGS) $< -o $@

#evaluate the time of the two kinds of program
time: $(OBJ)
    time ./byrow
    time ./bycol

clean:
    rm $(OBJ)
```

Makefile setup

Measure the executed time

The only difference is the method to get the sum of the matrix elements. (red circle)

Locality

```
[coherent@taipei ~]$ make
gcc -g -Wall byrow.c -o byrow
gcc -g -Wall bycol.c -o bycol
[coherent@taipei ~]$ make time
time ./byrow
sum = 10737653331795883 77
10.49user 0.91system 0:11.40elapsed 99%CPU (0avgtext+0avgdata 3906728maxresident)k
0inputs+0outputs (0major+846417minor)pagefaults 0swaps
time ./bycol
sum = 10737653331795883 77
22.08user 0.92system 0:23.00elapsed 99%CPU (0avgtext+0avgdata 3906732maxresident)k
0inputs+0outputs (0major+846417minor)pagefaults 0swaps
```

2X executed time due to spatial locality.

```
matrix1 address:
181370368  181370372  181370376  181370380  181370384
181370388  181370392  181370396  181370400  181370404
181370408  181370412  181370416  181370420  181370424
181370428  181370432  181370436  181370440  181370444
181370448  181370452  181370456  181370460  181370464

matrix2 address:
3288442592  3288442596  3288442600  3288442604  3288442608
3288442624  3288442628  3288442632  3288442636  3288442640
3288442656  3288442660  3288442664  3288442668  3288442672
3288442688  3288442692  3288442696  3288442700  3288442704
3288442720  3288442724  3288442728  3288442732  3288442736
```