CS 240
Computer Systems

#5: Locality of Reference and Virtual Memory

Sept. 7, 2021 · Wade Fagen-Ulmschneider

Sample Programs:

04cr.c

```
for (unsigned int c = 0; c < SIZE; c++) {
  for (unsigned int r = 0; r < SIZE; r++) {
    array[(r * SIZE) + c] = (r * SIZE) + c;
  }
}
```

04rc.c

```
for (unsigned int r = 0; r < SIZE; r++) {
  for (unsigned int c = 0; c < SIZE; c++) {
    array[(r * SIZE) + c] = (r * SIZE) + c;
  }
}
```

Running Times: **04cr.c** (Program #1):

04rc.c (Program #2):

Caching Strategies: Keeping Data Close

In working with memory in any computer system, we want to access it as quickly as possible. However, space is extremely limited in the fastest memory, so we need strategies on what data to keep close.

General Purpose Memory:

- CPU Registers: Stores one word, **only _____ general purpose registers available** on x64.
- CPU Cache: Stores a collection of 4 KB "pages" from RAM.
    - Intel Core i9-10900KF has 256 KB /CPU + 20 MB
    - Total Pages: ________ / CPU + _________
- RAM: "Dream Computer" has 128 GB of RAM
    - Total Pages: ________

Key Idea: <u>Locality of Reference</u>

System Memory: Limited, Shared, and Simple

In hardware, your system has a fixed amount of RAM:

1.

2.

3.

To help us to begin to organize this RAM, we divide the RAM up into chunks called __________.

- On most systems, a page is ______ KB.
- Linux: **`getconf PAGESIZE`**

Virtual Memory:

Modern systems provide an abstraction between the _________ and ____________________:

1. A ____________________ translates a ___________ into a physical address.

2. Virtual Memory is **NOT shared** between processes/apps:

Let's explore a sequence of allocations using a page table:

P1 Page Table:	RAM	P2 Page Table:	P3 Page Table:

Allocation Sequence:

1. Process #1 (P1): **`a = malloc(3 * 4096)`**
2. Process #3 (P3): **`b = malloc(5 * 4096)`**
3. Process #1 (P1): **`c = malloc(2 * 4096)`**
4. Process #3 (P3) exits.
5. Process #2 (P2): **`d = malloc(4 * 4096)`**
6. Process #2 (P2): **`e = malloc(5 * 4096)`**
7. Process #1 (P1): **`a = realloc(a, 5 * 4096)`**

With a virtual memory system:

- Can we meet all of the allocation requests?
- Are we limited to just RAM?

Advantages of a Virtual Memory System:

1.

05.c

```
printf("  Start of `array`: %p\n", array);
printf("    End of `array`: %p\n", &(array[(SIZE * SIZE) -1]));
```

2.

3.

Simple Simulation of Page Tables with Disk Pages

RAM:	P1 Page Table:	Disk Pages:
[0]:	[0]:	...
[1]:	[1]:	./programCode (1/5)
[2]:	[2]:	./programCode (2/5)
[3]:	[3]:	./programCode (3/5)
	[4]:	./programCode (4/5)
	[5]:	./programCode (5/5)
	[6]:	
	[7]:	
	[8]:	
	[9]:	
	[10]:	
	[11]:	hiddenImage.png
	[12]:	hiddenImage.png
	[13]:	hiddenImage.png
	[14]:	
	[15]:	...

1: Load Program

2: Run PC, pg1:
- malloc(4000)

3: Run PC, pg2:
- malloc(10000)
- Open hiddenImage.png
- Read all of image

4: Run PC, pg3:
- Access OG 4 KB
- Finish program

Q1: What is the range of possible file sizes for **`hiddenImage.png`**?

Q2: What is the range of possible file sizes for **`./programCode`**?

Q3: What is the size of the heap immediately before the program finishes?