CS 382
Fall 2023

Computer Architecture and Organization

Lab 9

In CS-284 Data Structures you have learned a structure called linked list, and compared it with a traditional array. When iterating over either a linked list or an array, the time complexity for both is $\mathcal{O}(n)$. This is true theoretically, but what about in reality?

In this lab, you will complete the following two tasks, while completing a lab report provided to you.

1 Task 1: Profiling a Linked List and an Array

To see if the performance of linked lists and arrays are really comparable, we provided you a starter code. Simply go to the starter code directory, and type `make` to compile the code. Then type the following:

```
$ ./a.out -t ll,arr -s 10000
```

What this program does is to create an array and a linked list, each of which consists of 10,000 random integer numbers. Then it will iterate over the array and the list, and time the execution, respectively.

Your task here is to change the size of the lists from 10^2, 10^3, 10^4, 10^5, to 10^6. Observe how the execution time of iterating the lists changes for both linked list and array. Record this experiment result, and present a graph or chart in your lab report.

Then explain: why does the two algorithms with both $\mathcal{O}(n)$ complexity, have very different performance when n increases? You need to explain in detail from the perspective of **locality**.

2 Task 2: Locality Improved Linked List

It's a good idea to combine the advantages of both linked list and array, to improve the locality of their operations and thus the efficiency. This new structure is called **Unrolled Linked List**. Instead of storing just one data element in each node, unrolled linked list stores an array of elements in each node. When iterating over the list, in each node, we iterate over the array first, and then move on to the next node. For more information, see Wikipedia: https://www.wikiwand.com/en/Unrolled_linked_list.

We provide the starter code in both C and C++, and you are free to choose the one you're most comfortable with. In the following we will use C version as an example.

In `UnrolledLL.h`, you will see the declarations of this new struct and functions:

```c
typedef struct ull_node {
    int*        datagrp;
    u_int64_t   blksize;
    struct ull_node* next;
} uNode;

typedef struct {
    uNode*      head;
```

```
    u_int64_t   len;
} UnrolledLL;

uNode*  new_unode(uNode** prev, u_int64_t blksize);
void    init_ullist(UnrolledLL* ullist, u_int64_t size, u_int64_t blksize);
void    iterate_ullist(uNode* uiter);
void    clean_uulist(UnrolledLL* ullist);
```

Struct `uNode` is a node in a unrolled linked list. Each node has an array of size `blksize` (block size) that stores data. `u_int64_t` is an unsigned integer type of 64 bytes.

Your task is to compelete the code in `UnrolledLL.c` first, and profile the algorithm of iterating over the list:

```
$ make
$ ./a.out -t ll,arr,ull -s 10000 -b 100
```

This command will create a linked list of 10,000 nodes, an array of 10,000 random integers, and a unrolled linked list of $\frac{10000}{100} = 100$ nodes where each node contains 100 random integers. And then it will time the execution of iterating over lists for each of the structures.

Note: do not change any file provided to you other than `UnrolledLL.c` or `UnrolledLL.cpp`.

In your report, fix the block size 100, and change the size of the lists from 10^2, 10^3, 10^4, 10^5, to 10^6. Observe how the execution time of iterating the lists changes for all the structures. Record this experiment result, and present a graph or chart in your lab report.

Then explain: what is the time complexity of unrolled linked list? How does a unrolled linked list improve the efficiency of traversal in terms of locality?

3 Grading

The lab will be graded based on a total of 10 points, 3 for task 1 and 7 for task 2.

- Task 1 (3 points):
  - **-2:** the explanation is not from the perspective of locality; and/or does not consider how the two structures are stored in memory;
  - **-1:** the graph in the report does not correctly show the experiment result, or is missing.
- Task 2 (7 points):
  - **-7:** the code does not compile, or executes with run-time error;
  - **-7:** any file other than `UnrolledLL.c` or `UnrolledLL.cpp` is changed;
  - **-2:** the explanation is not from the perspective of locality; and/or does not consider how the structures are stored in memory;
  - **-2:** the time complexity of unrolled linked list is wrong;
  - **-2:** no pledge and/or name in the report (this counts for both tasks);
  - **-1:** the graph in the report does not correctly show the experiment result, or is missing;
  - **-1:** no pledge and/or name in the code.

Earlybird Extra Credit: 2% of extra credit will be given if the lab is finished by Wednesday 11:59PM EST (1 day

before the lab deadline). For specific policy, see syllabus.

Attendance: check off at the end of the lab to get attendance credit.

Deliverable

One `UnrolledLL.c` or `UnrolledLL.cpp` file, and one PDF report, no need to zip.