CS622A

Advanced Computer Architecture

Assignment 2

Memory Reuse and Sharing Profile Analysis

GROUP 16

Aditya Rohan
160053

Aniket Pandey
160113

Instructor:
Dr. Mainak Chaudhury

September 24, 2019

1 Introduction

In this assignment, we use PIN tool to instrument a set of parallel programs and collect thread-wise memory access trace and break it down to x86 machine accesses. Then with the resulting trace, we analyze the sharing profile and memory reuse for the given parallel programs.

2 Analysis Results

PART 1: Collection of machine-access traces

The results were varying across individual runs. Hence, we have collected 5 results for a particular program and picked *addrtrace.out* corresponding to the middle value (highlighted).

Programs	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
prog1.c	**128988038**	128988149	128987956	128988046	128987901
prog2.c	2528955	2513452	2521172	**2524574**	2532314
prog3.c	**9508261**	9510696	9501049	9497081	9521463
prog4.c	1061544	1061507	1061492	1061525	**1061515**

Table 1: Machine accesses count across 5 runs

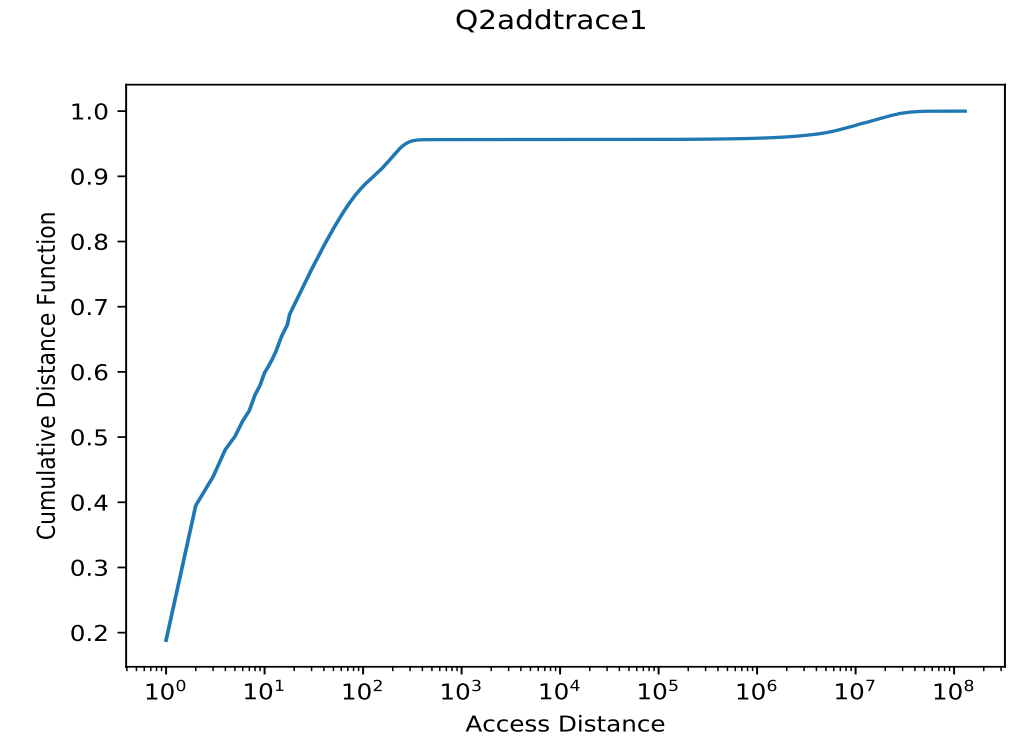
PART 4: Sharing profile analysis

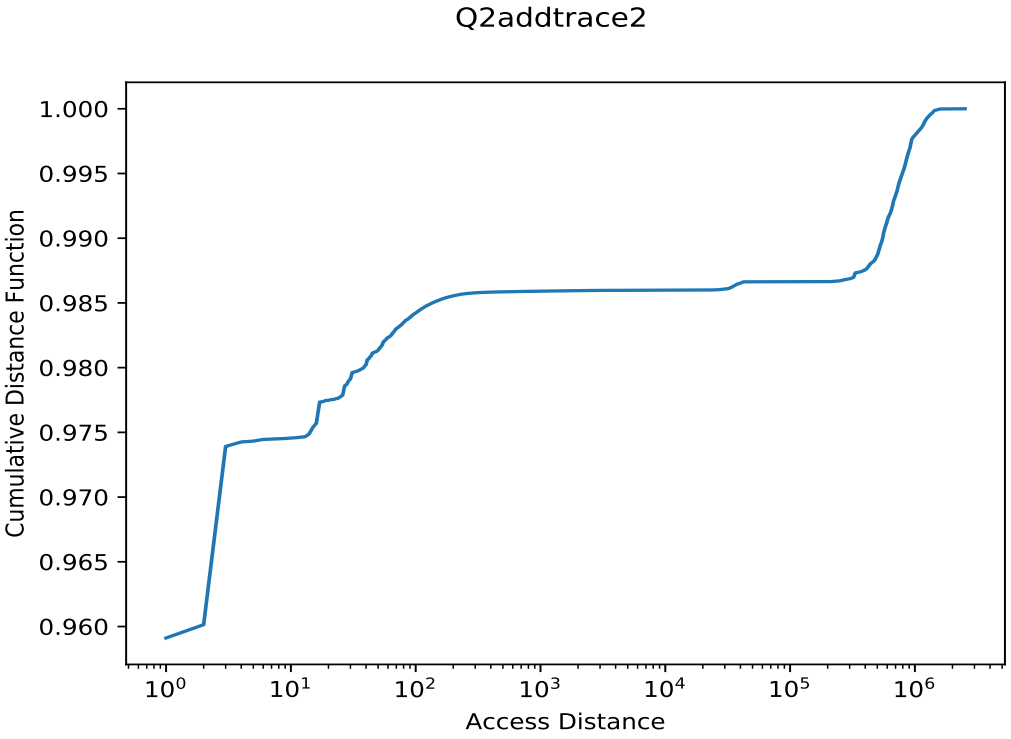
The sharing profile for each of the 4 target programs is given below. The trace corresponding to the highlighted values in part1 were selected for the result analysis.

	prog1.c	**prog2.c**	**prog3.c**	**prog4.c**
Private	388	384	386	8573
2-Shared	63	8255	56	57403
3-Shared	1872	16384	0	6
4-Shared	32456	40958	1	0
5-Shared	143251	5	1	0
6-Shared	244970	0	0	0
7-Shared	173831	0	0	1
8-Shared	124527	9	65545	10

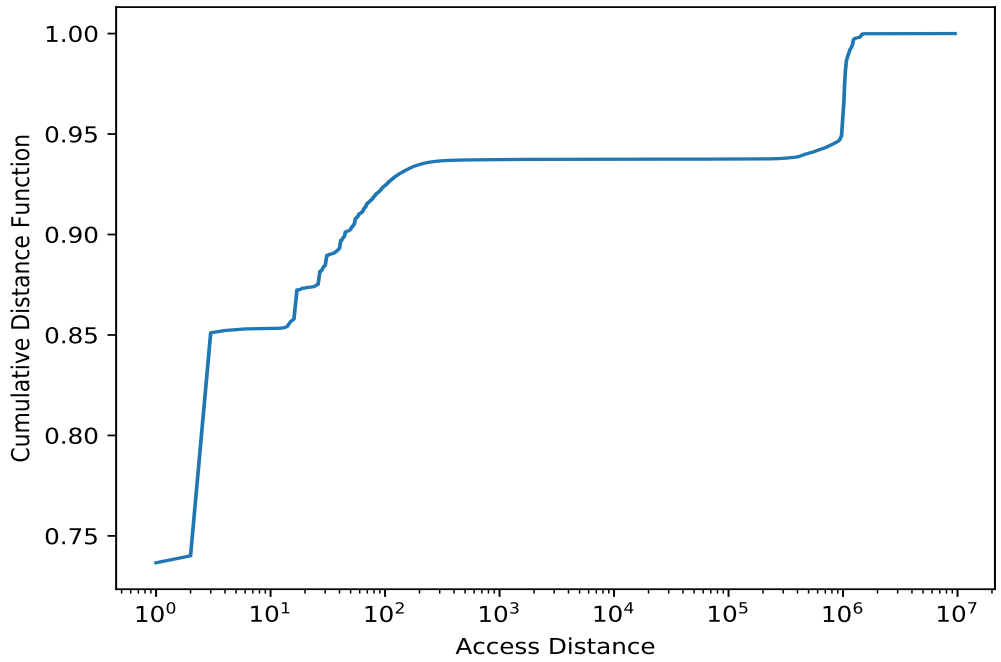
Table 2: Sharing profile analyis for 8 threads

PART 2: Access distance analysis

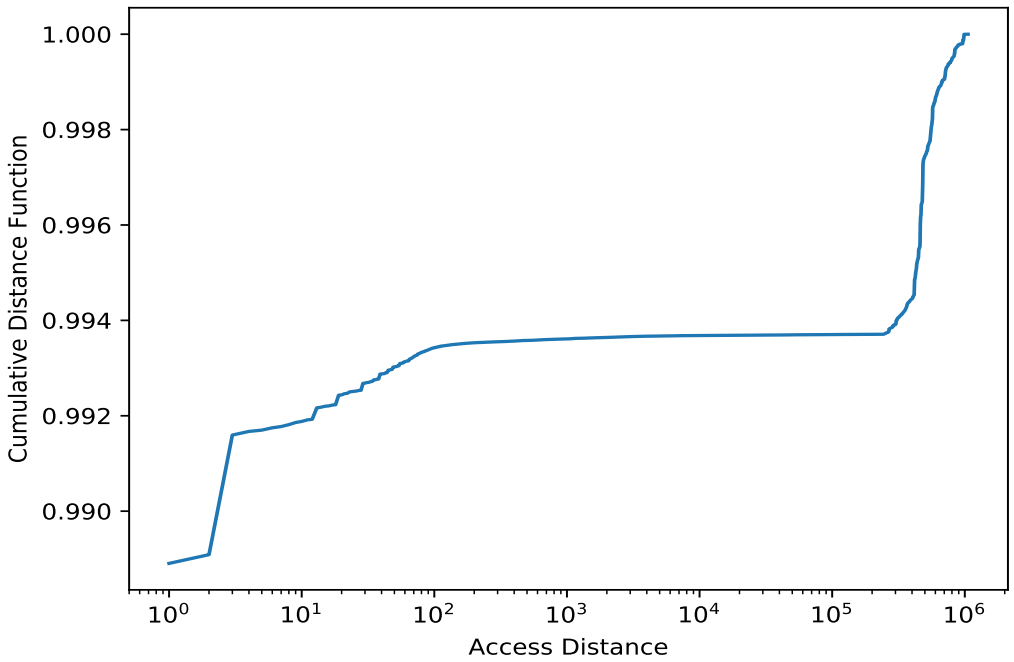




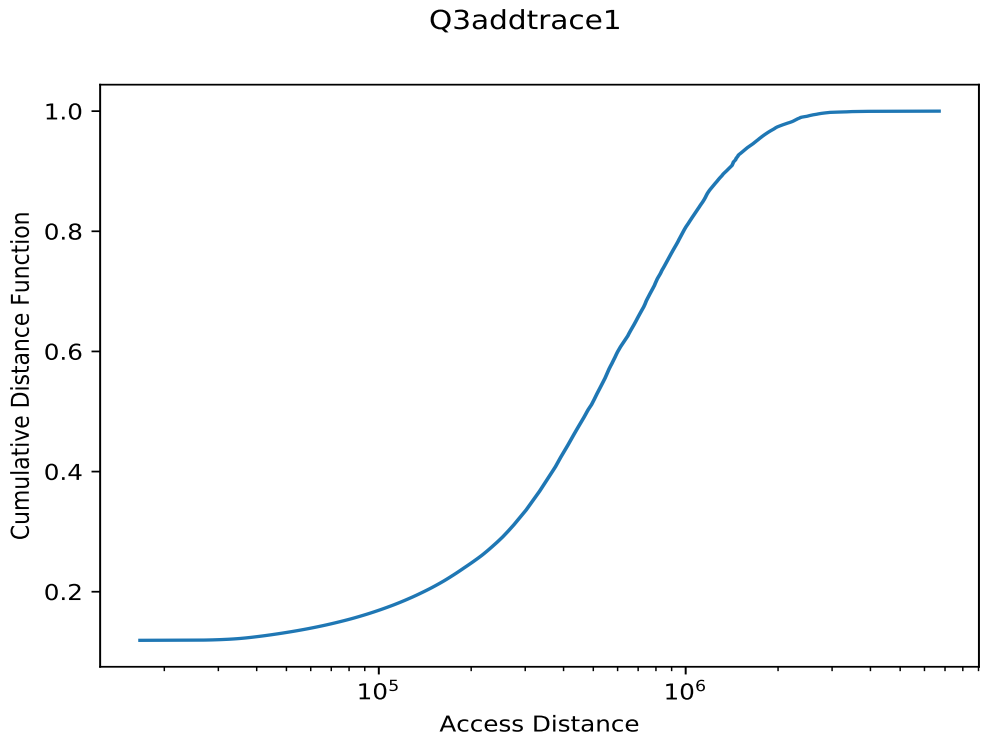
Q2addtrace3

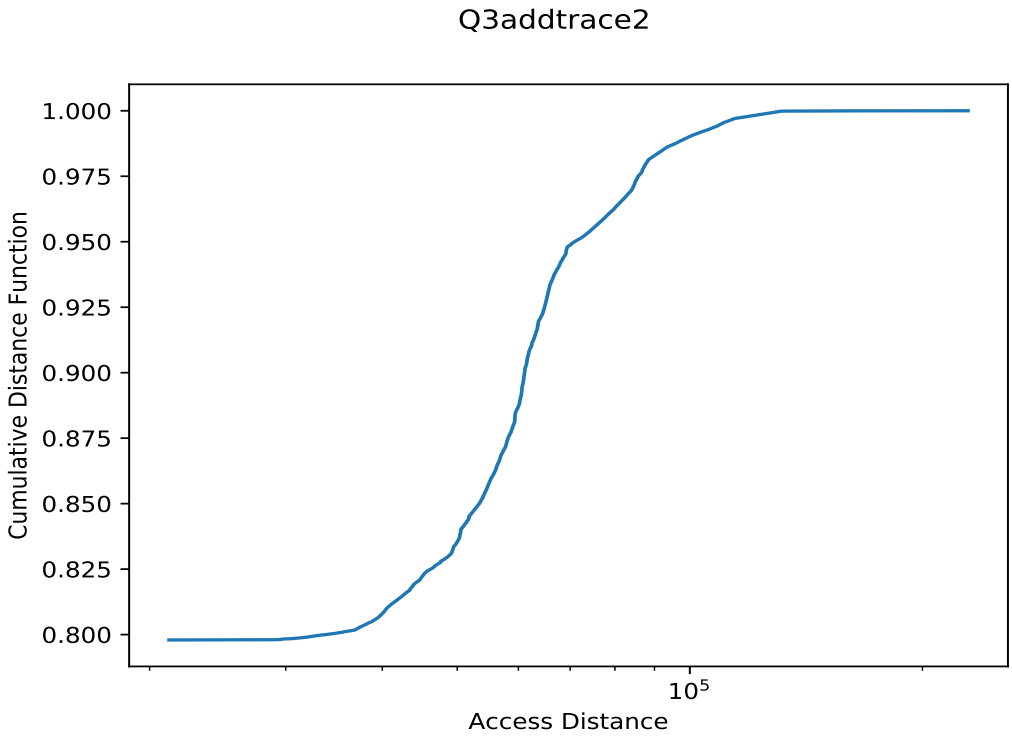


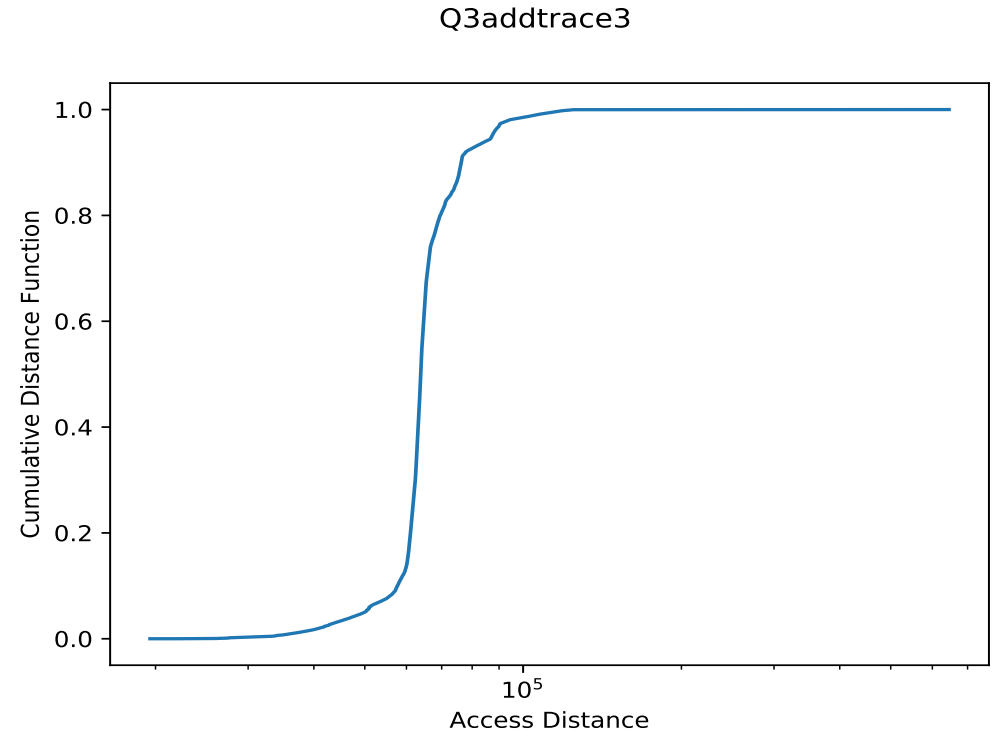
Q2addtrace4

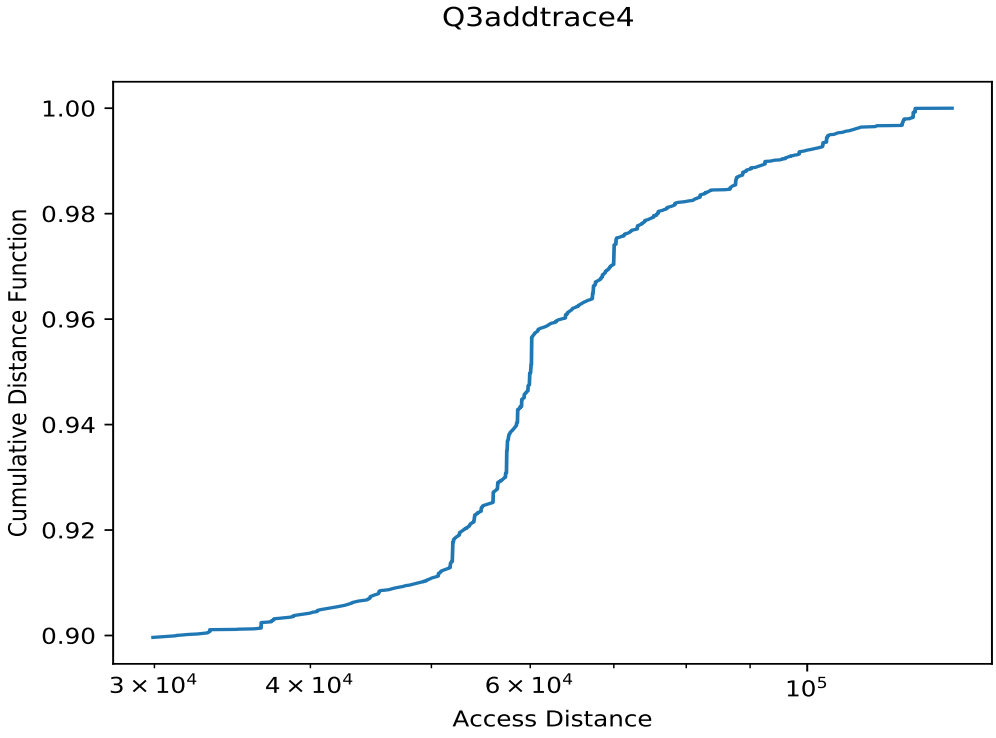


PART 3: Access distance filtered by LRU cache






Q3addtrace3
Cumulative Distance Function
1.0
0.8
0.6
0.4
0.2
0.0
10^5
Access Distance


Q3addtrace4
Cumulative Distance Function
1.00
0.98
0.96
0.94
0.92
0.90
3×10^4
4×10^4
6×10^4
10^5
Access Distance

	prog1.c	prog2.c	prog3.c	prog4.c
Hits	122297455	2295505	8862896	930888
Misses	6690582	229068	645364	130626

Table 3: Result of modelling a 2MB 16-Way cache on the traces

The cumulative distribution function for misses in counted specifically against the misses after modelling the single-level cache to the traces.

3 Discussion

Following are some of the observations on the simulation results.

3.1 Part 1

3.1.1 L2 Hits/Misses (Exclusive) == L2 Hits/Misses (NINE)

This is observed because at any point in time, the addresses that are present in case of Exclusive policy are same as that of NINE, albeit in different configuration. We can analyze all possibilities to prove this:

- **L2 Hit:** Nothing changes here.
- **L3 Hit:** The block gets added to L2 Cache. Although the block is invalidated from L3 in exclusive case.
- **L3 Miss:** The block is brought from memory and added to both L2 and L3 cache.

Hence, in all the cases, if the configuration is same after *kth* steps, it will be same after *(k+1)th* step.

3.1.2 L3 Misses: Inclusive > ($\approx$) NINE >> Exclusive

At any given point, L3 cache in case of Exclusive policy will have most absolute space (due to non-similarity of data in L2 and L3). This explains why L3 misses in Exclusive are significantly lesser.
There isn't much difference in L3 misses for Inclusive and NINE cases, although Inclusive is slighly more. This can be attributed to the fact that eviction from L3 also evicts from L2 for Inclusive policy, which is not the case with NINE. So, an access to a block evicted from L3 would encounter a hit in L2 for NINE but would unnecessarily miss in both L2, L3 in Inclusive case.

3.1.3 L2 Misses: Inclusive > Exclusive == NINE

This is a serious problem with Inclusive policy, as mentioned in problem statement. If a block recieves constant hits in L2 cache, its rank in the LRU indexing will be quite high. As a result, the block will normally never be evicted from L2. But its age in L3 will drop significantly and might even become victim to an eviction. Following the Inclusive policy, it will also have to be removed from L2 which would unnecessarily increase L2 misses.

3.2 Part 2

3.2.1 Method of categorizing FA L3 misses

Cold: Number of unique block references.
Capacity: FA-L3 misses - Cold misses.
Conflict: 16-way L3 misses - FA-L3 misses.

1. Number of capacity misses for LRU eviction policy are more than that of Belady's policy since Belady is the most optimal solution for block replacement.

2. Number of conflict misses for *sphinx3* trace are negative (-179886). For some uncommon cases, the approach of calculating conflict misses from cold and capacity of fully-associative cache can result in a negative number [**?**].