CSE 520: Computer Architecture 2 – Assignment 1

Name : Viraj Savaliya

ASU Id: 1217678787

➢ ***Cpu info:***

- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 1024K
- L3 cache: 16896K
- Thread(s) per core: 2
- Core(s) per socket: 10

Note: All the output data related to the benchmarks results can be found in an excel sheet named Comp Arch-assgn1.

I. 1ST BENCHMARK

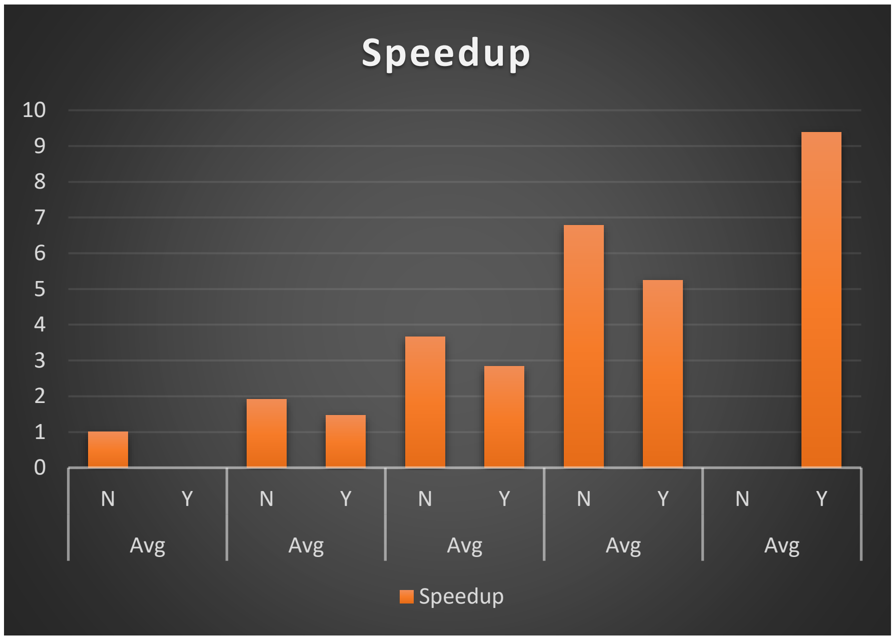

Fig 1.1: Speedup

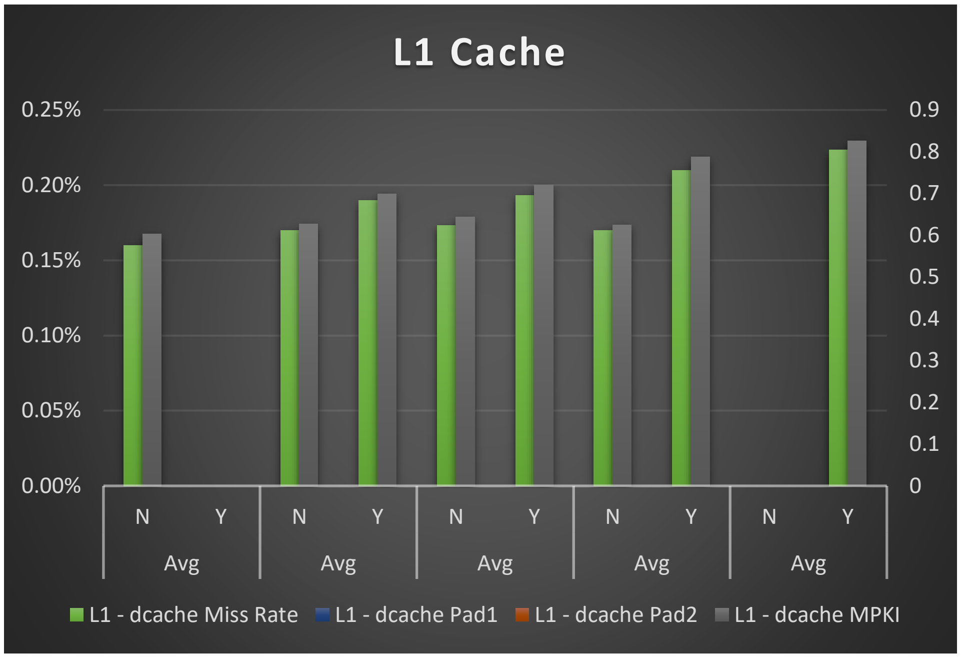

Fig 1.2: L1 Cache

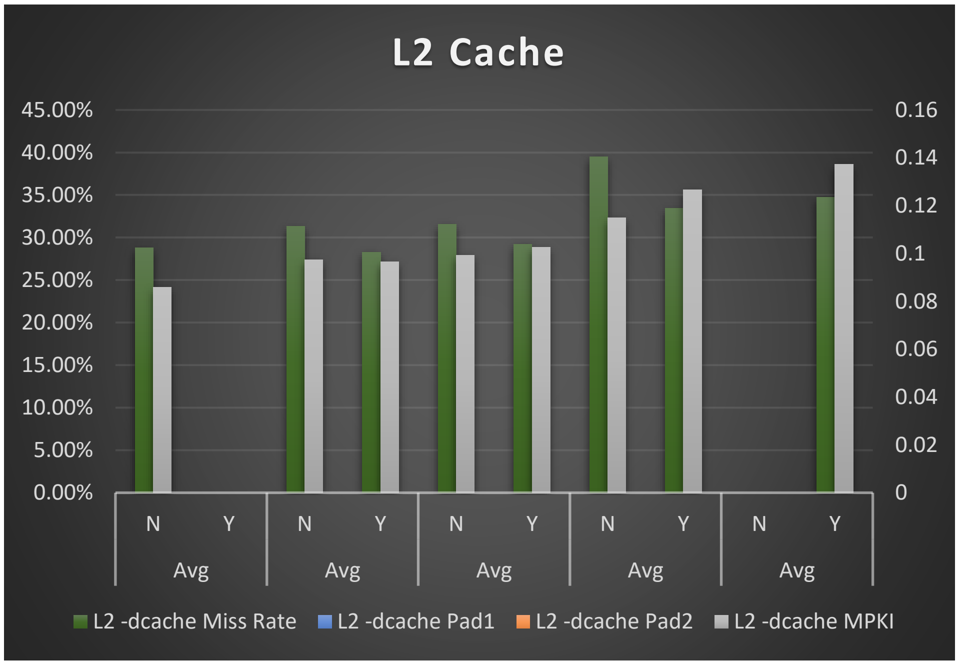

Fig 1.3: L2 Cache

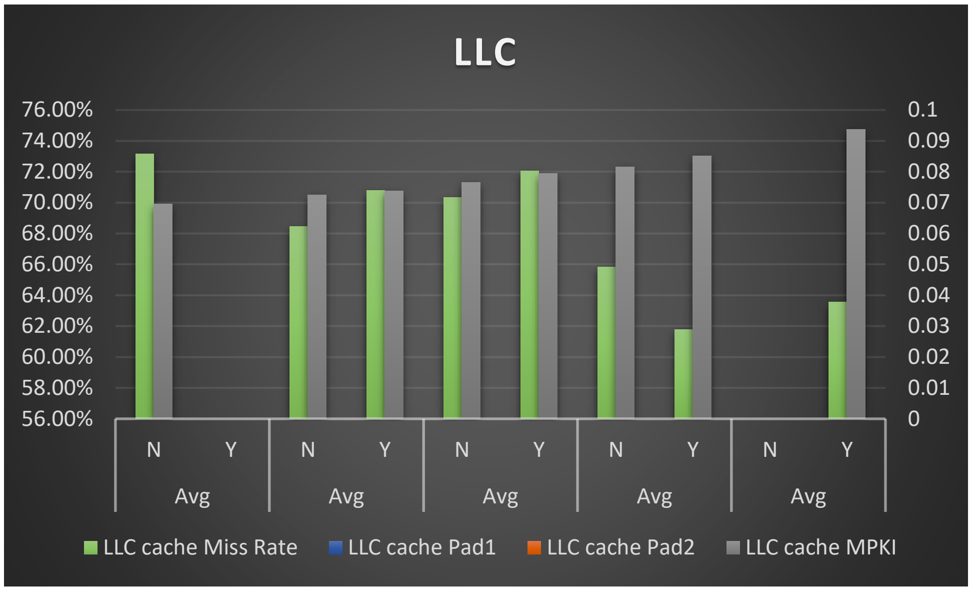

Fig 1.4: LLC

- **Observations:**

1) Speed per thread increases as we go on increasing threads in both cases (i.e. with and without hyperthreading). The Speed increases linearly i.e. as we increase the number of threads the speed also increases at the same rate. The reason behind the above argument can be as more threads are sharing the same workload.
2) The reason for the speed can be so because as cache is shared between 2 logical threads when you do hyperthreading, speedup is not same as that you expect when running 2 threads without hyperthreading because the workload might not be exactly divided by half and hence one thread might take a little more time than the other one.
3) From L1 to L2 to LLC, the miss rate increase can be explained by the principle of locality.
4) MPKI for LLC is better i.e. less and serving the data request more when compared to L1 and L2. The reason I believe is that LLC has bigger capacity cache than L1 and L2 (LLC > L2 > L1).
5) In L1 cache, when we enable hyperthreading the miss rate and MPKI is increasing since both the logical threads in the same physical core need to share same L1dcache and L1icache.
6) As we increase the number of threads, more threads try to hit LLC as it is shared by everyone and hence MPKI is increasing with increase in number of threads.
7) In L1 cache, the miss rate and MPKI remains almost constant even if we increase the number of threads for both the cases (i.e. with hyperthreading or without hyperthreading) because L1 cache is smallest in size and will have very closely related data.
8) In L2 cache the miss rate decreases for hyperthreading but in L1 cache the miss rate increases as we enable hyperthreading. The reason can be that threads running on the same physical core might share some data/instructions in common which is closely associated but not as closely associated with the data/instructions in L1.
9) In L1 and L2 cache, the relation between the miss rate and MPKI for all cases of different number of threads in hyperthreading and without hyperthreading remains the same. The reason can be because every physical core will have its own L1 and L2 cache. The effect cannot be seen in LLC as LLC is shared by every core.
10) The MPKI value increases in both cases (i.e. with hyperthreading and without hyperthreading) as the number of threads are increased because more threads will try to access the same dataset which is not much related.

II. 2ND BENCHMARK

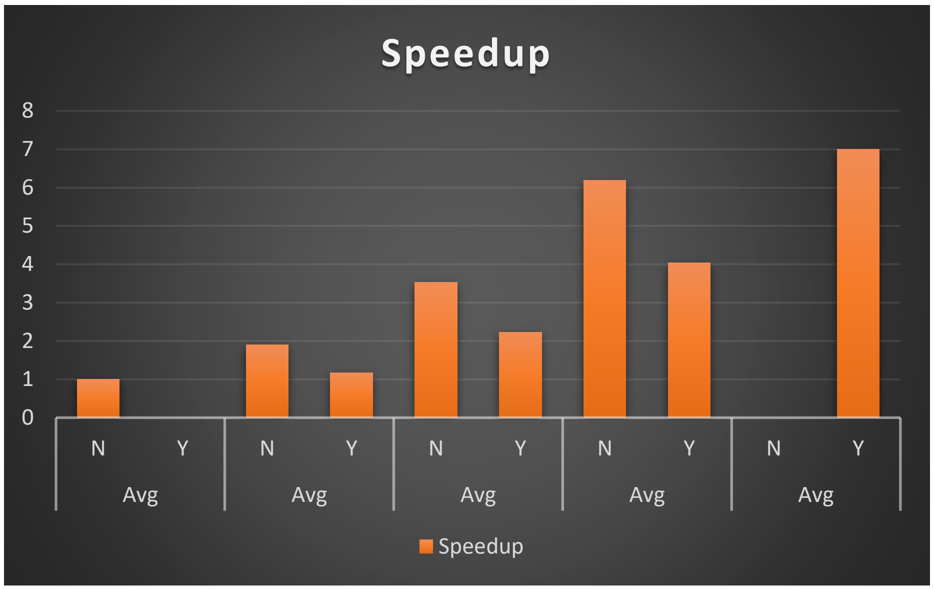

Fig 2.1: Speedup

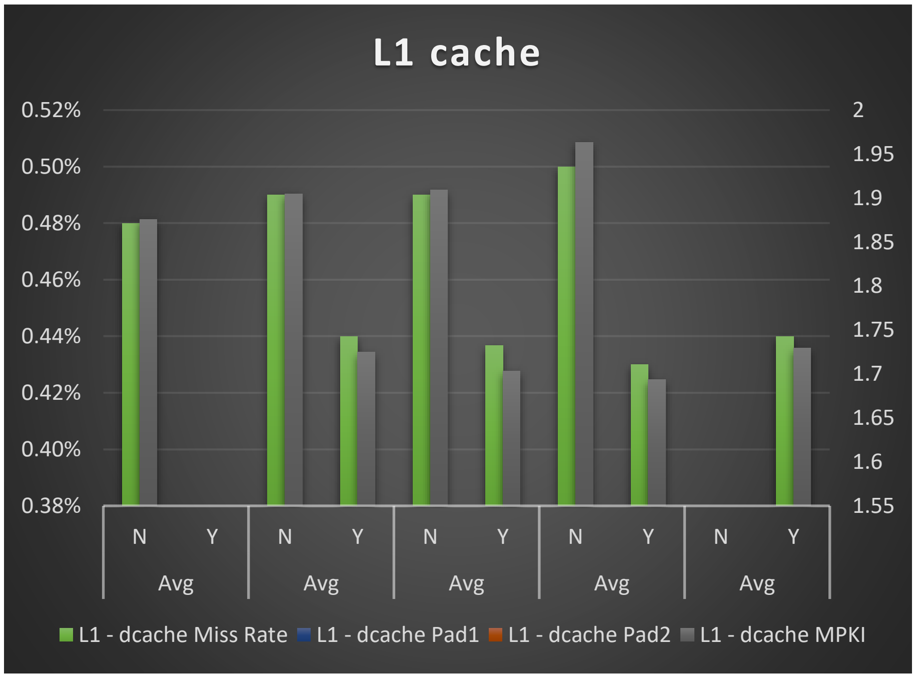

Fig 2.2: L1 Cache

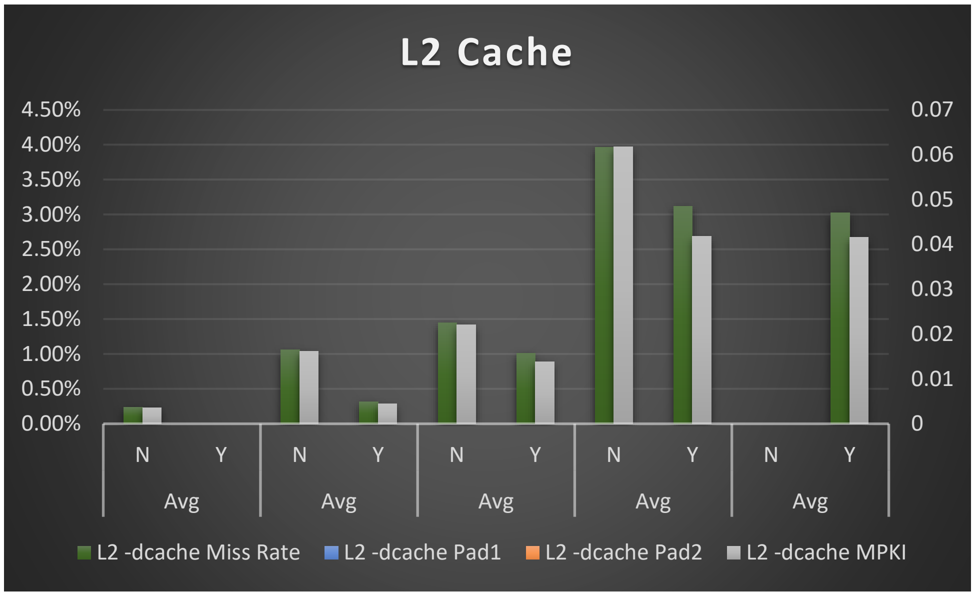

Fig 2.3: L2 Cache

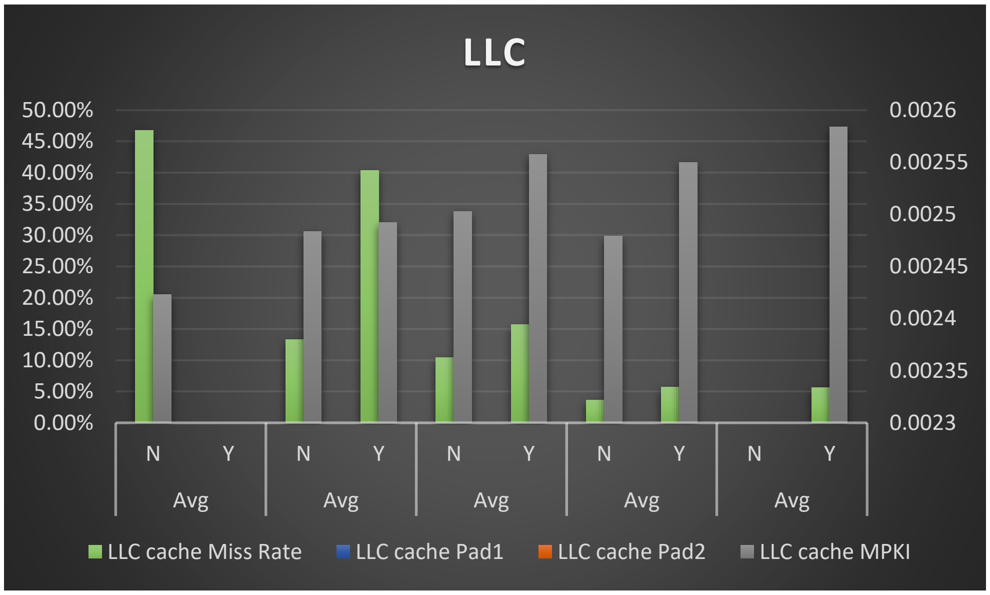

Fig 2.4: LLC

- **Observations:**

1) Speed per thread increases as we go on increasing threads in both cases (i.e. with and without hyperthreading). The Speed increases linearly i.e. as we increase the number of threads the speed also increases at the same rate. The reason behind the above argument can be as more threads are sharing the same workload.
2) The reason for the speed can be so because as cache is shared between 2 logical threads when you do hyperthreading, speedup is not same as that you expect when running 2 threads without hyperthreading because the workload might not be exactly divided by half and hence one thread might take a little more time than the other one.
3) From L1 to L2 to LLC, the miss rate increase can be explained by the principle of locality.
4) MPKI for LLC is better i.e. less and serving the data request more when compared to L1 and L2. The reason I believe is that LLC has bigger capacity cache than L1 and L2 (LLC > L2 > L1).
5) In L1 cache, when we allow hyperthreading the miss rate and MPKI decreases and the reason can be that the data/instructions are more closely related for two threads running on the same physical core.

6) Similarly, almost the same trend can be observed as of point 5 of miss rate and MPKI in L2 cache. So, we can say that the principle of locality works well for benchmark 2.
7) In L1 cache, the miss rate and MPKI remains almost constant even if we increase the number of threads for either of the cases i.e. with hyperthreading or without hyperthreading because L1 cache is smallest in size and will have very closely related data.
11) In L1 and L2 cache, the relation between the miss rate and MPKI for all cases of different number of threads in hyperthreading and without hyperthreading remains the same. The reason can be because every physical core will have its own L1 and L2 cache. The effect cannot be seen in LLC as LLC is shared by every core.
8) In LLC, the miss rate increases but the MPKI decreases when hyperthreading is allowed. The reason I believe so is as the requests received by LLC might be sparse but when we talk with respect to the instructions of the program code, they might be more reusable and hence MPKI is decreasing with hyperthreading.
9) The above argument can be strengthened by another observation of LLC cache that as number of threads goes on increasing in either cases of allowing hyperthreading or not allowing it the pattern observed is that the miss rate increases with increase in number of threads.

III. <u>3RD BENCHMARK</u>

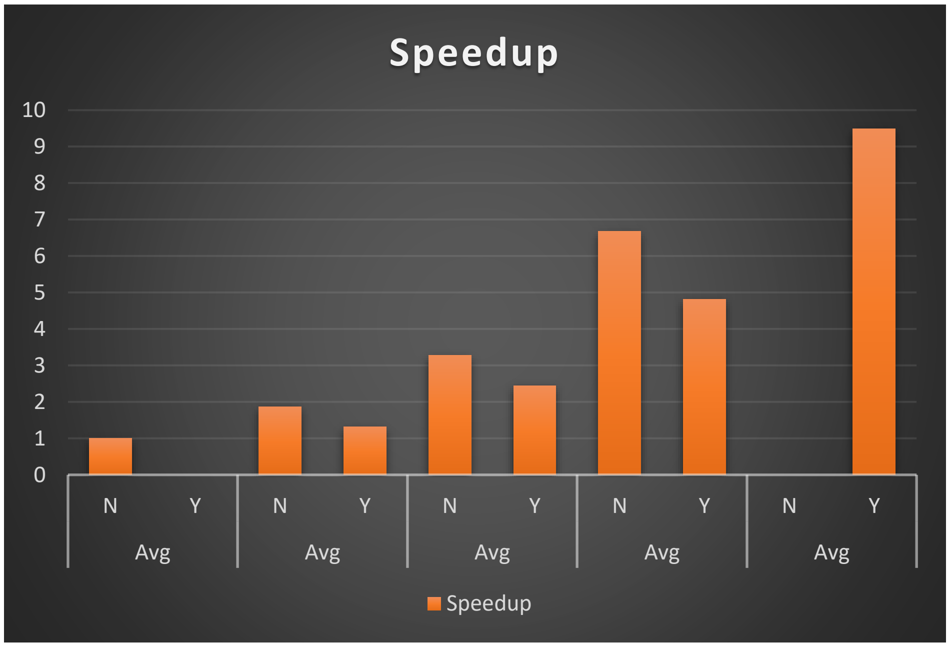

Fig 3.1: Speedup

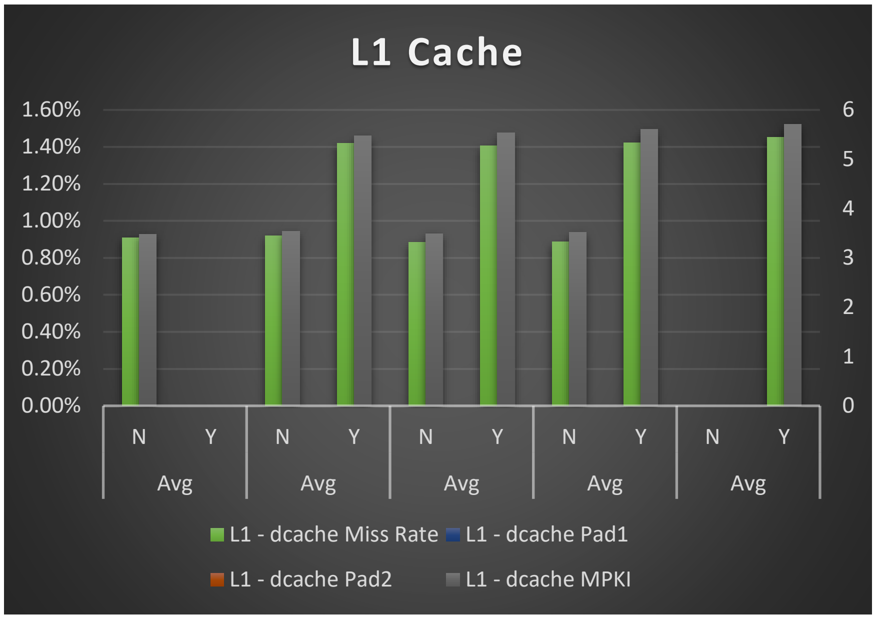

Fig 3.2: L1 Cache

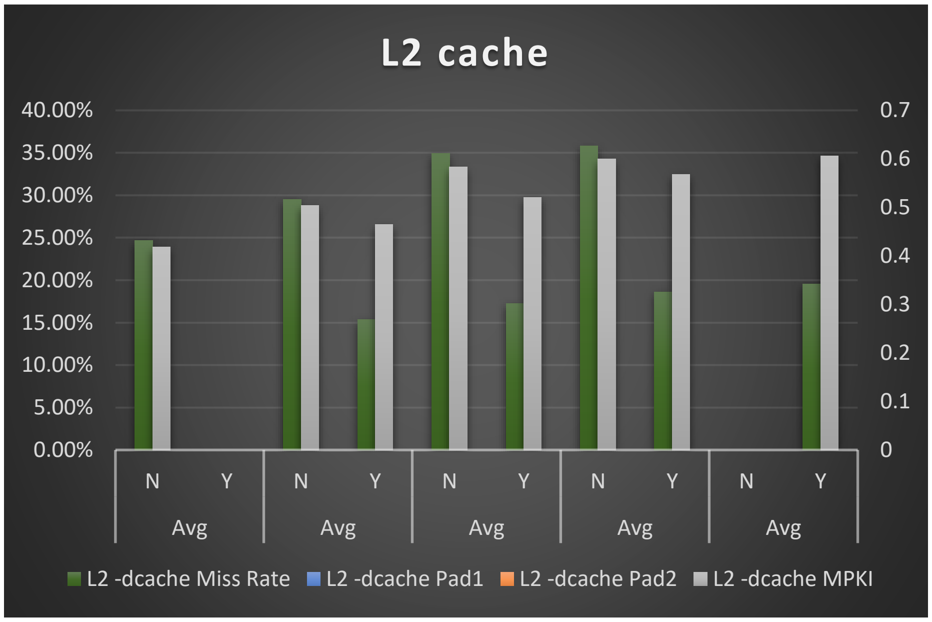

Fig 3.3: L2 Cache

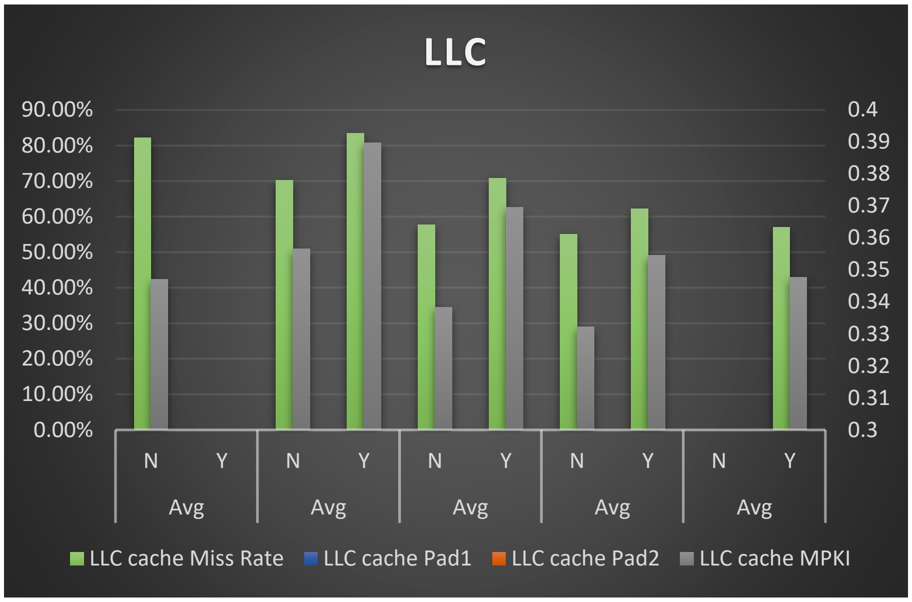

Fig 3.4: LLC

- **Observations:**

1) Speed per thread increases as we go on increasing threads in both cases (i.e. with and without hyperthreading). The Speed increases linearly i.e. as we increase the number of threads the speed also increases at the same rate. The reason behind the above argument can be as more threads are sharing the same workload.
2) The reason for the speed can be so because as cache is shared between 2 logical threads when you do hyperthreading, speedup is not same as that you expect when running 2 threads without hyperthreading because the workload might not be exactly divided by half and hence one thread might take a little more time than the other one.
3) From L1 to L2 to LLC, the miss rate increase can be explained by the principle of locality.
4) MPKI for LLC is better i.e. less and serving the data request more when compared to L1 and L2. The reason I believe is that LLC has bigger capacity cache than L1 and L2 (LLC > L2 > L1).
5) In L1 cache, when we enable hyperthreading the miss rate and MPKI increases since both the logical threads in the same physical core need to share same L1dcache and L1icache.
6) In L1 cache, the miss rate and MPKI remains almost constant even if we increase the number of threads for either of the cases i.e. with hyperthreading or without hyperthreading because L1 cache is smallest in size and will have very closely related data.
7) In L1 and L2 cache, the relation between the miss rate and MPKI for all cases of different number of threads in hyperthreading and without hyperthreading remains the same. The reason can be because every physical core will have its own L1 and L2 cache. The effect cannot be seen in LLC as LLC is shared by every core.
8) In L1 cache, the miss rate increases as we enable hyperthreading but in L2 cache the miss rate decreases for hyperthreading. The reason can be that threads running on the same physical core might share some data/instructions in common which is closely associated but not as closely associated with the data/instructions in L1.
9) In LLC, miss rate and MPKI are increasing when we enable hyperthreading because when 2 threads are accessing on the same core, they might have more associativity in data.
10) In LLC, the miss rate decreases when the number of threads is increased for both the cases i.e. with hyperthreading and without hyperthreading. The reason might be as more addresses might be reused.