REPORT

ASSIGNMENT 1

Format:

1. Three benchmark programs, parsec fluidanimate, parsec bodytrack and starbench H264dec are run and L1-dcache loads, misses, L2 cache demand data reads, misses, LLC loads, misses, instruction count, cpu cycles and execution time are collected through perf profiling command.
2. All the benchmarks and results are collected in a particular folder for each benchmark. Parsec fluidanimate data is collected in test-case1 which has 4 scripts to calculate all the datas required. Data files collected for each iteration are collected in the following manner,
   - ❖ LLC loads,misses for Fluidanimate:
     - ➢ 1st iteration-LLCA1
     - ➢ 2nd iteration-LLCB1
     - ➢ 3rd iteration-LLCC1
   - ❖ A, B,C specifies each of the 3 iterations and a subscript "1"after A specifies the benchmark number:
     - ➢ 1-Parsec Fluidanimate
     - ➢ 2-Parsec Bodytrack
     - ➢ 3-Starbench H264dec

   e.g: L1A2- First iteration of L1 d-cache loads, misses for parsec bodytrack.
   L2C3- Third iteration of L2 cache loads, misses for starbench H264dec.
   CPUB1- Second iteration of CPU instructions, cycles for parsec fluidanimate.

Cache Clearing/Replacement:

1. LLC cache size: 16.5 MB
2. Total bytes: 16.5 x 10^6 bytes
3. Data type used to allocate unsigned long long, which takes 4 bytes of memory.
4. Size of array to be declared: ((16.5 x 10^6)/4) =4125000
5. Since cpu has 10 cores with 2 threads on each core and LLC is shared by all the cores, so for memory allocated for each thread running on each cpu = 4125000/10= 412500 so, size of the array should be more than 412500.
   - A multithreaded pthread program is created which takes multiple thread values in the command line argument to run the threads parallely. Each thread allocates a memory of 4125000 of unsigned long long type and a for loop of the given argument is ran which sums each term in a sum variable along with a rand output to store a different result in the memory on each runs for each thread. Srand function is used which creates a different random number for each specified time.
   - In the folder the cache clearing program is specified as treadmul which is called in the script file each time between two different perf events.
   - Each thread in the treadmul program is ran on each of the twenty threads of the ten cpu cores.
   - Instead of running on ten threads on ten cores of the cpu, running twenty threads on each of the ten cpu cores gave better and higher cache miss results.

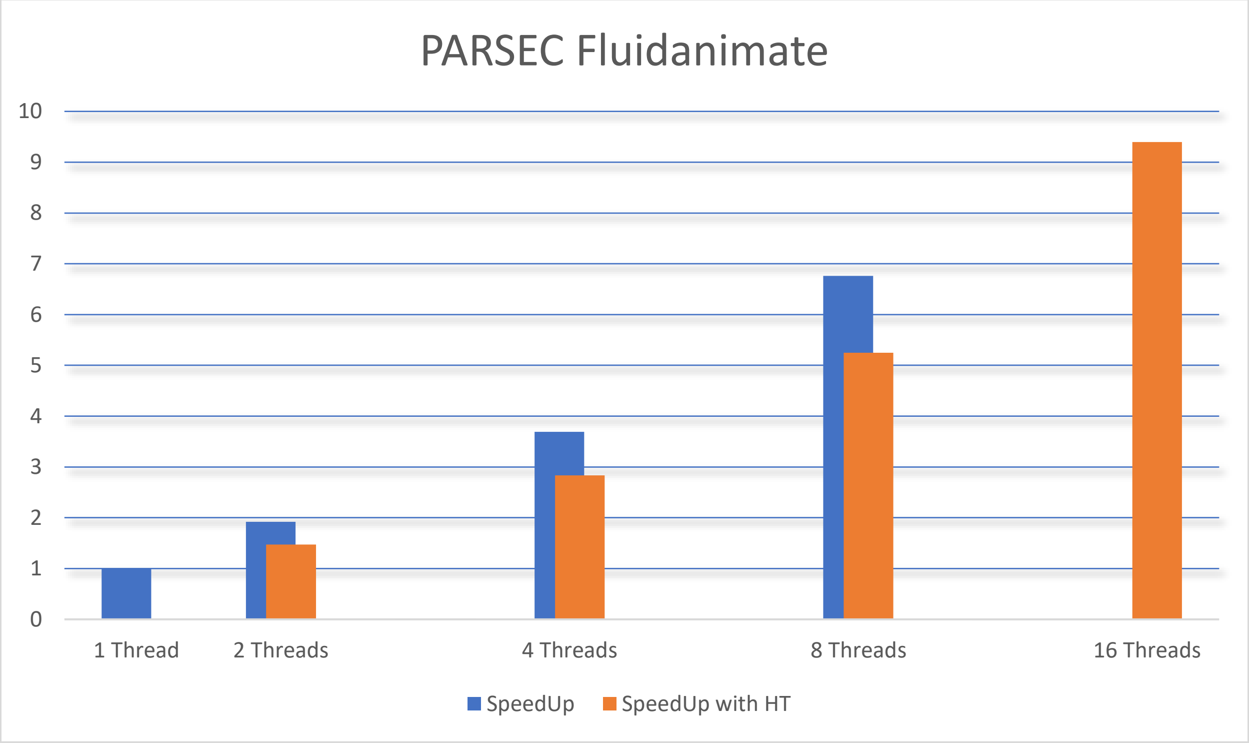
CPU Specifications:

1. Number of cores: 10
2. Number of threads: 20
3. Processor base frequency: 3.30 GHz
4. Cache:16.5 MB LLC

PERFORMANCE ANALYSIS:

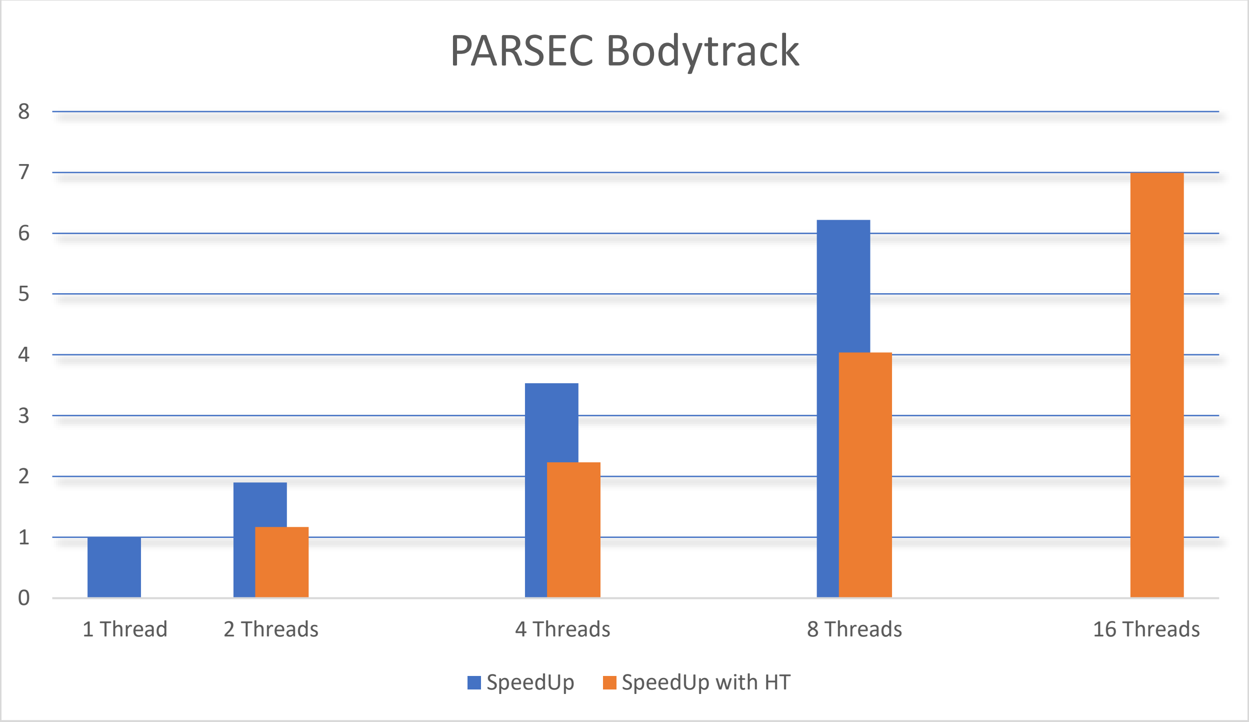
Execution Time SpeedUp

PARSEC FLUIDANIMATE:



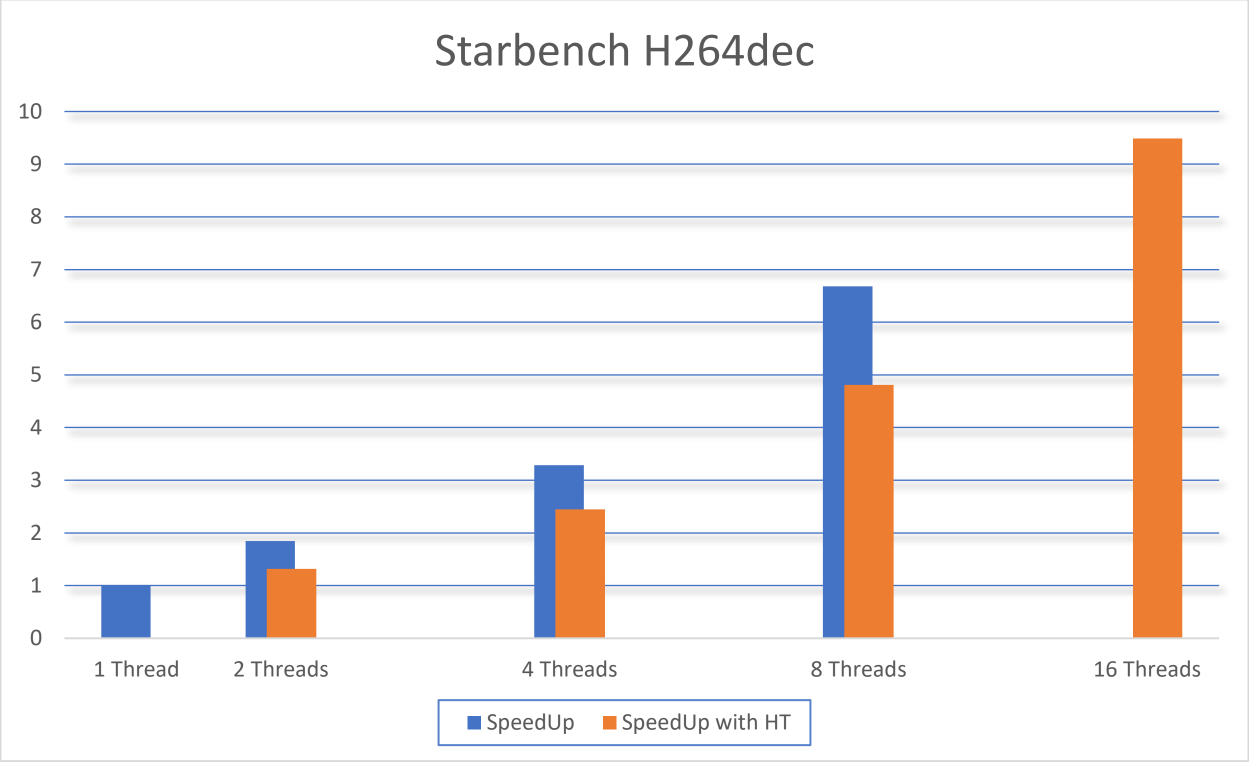
- By using the concept of parallelism, as we increase the number of threads to run on different CPUs' speedup increases, as more number of CPUs' can run the same benchmark parallelly at the same time in different CPU cores.
- Hyperthreading provides a second set of registers on a single physical core. This allows computer to treat a physical core as two logical cores. As threads runs, some of the internal architectures called as execution units are idle during run time hence, due to hyperthreading these idle execution units can process a thread thus, execution units can process instructions from two threads simultaneously resulting in lesser execution time. So, 2 ,4, 8 and 16 threads with hyperthreading gives a better speedup than 1, 2, 4 and 8 threads without hyperthreading respectively.

PARSEC BODYTRACK:



- The bodytrack benchmark is a computer vision application which tracks a 3D image of a human body through analyzing a sequence of images. Amount of parallelism in bodytrack is reduced by the number of vertical images passed in the sequence during the stage of image processing. Hence, the speedup of bodytrack is comparatively lower than Fluidanimate and H264.

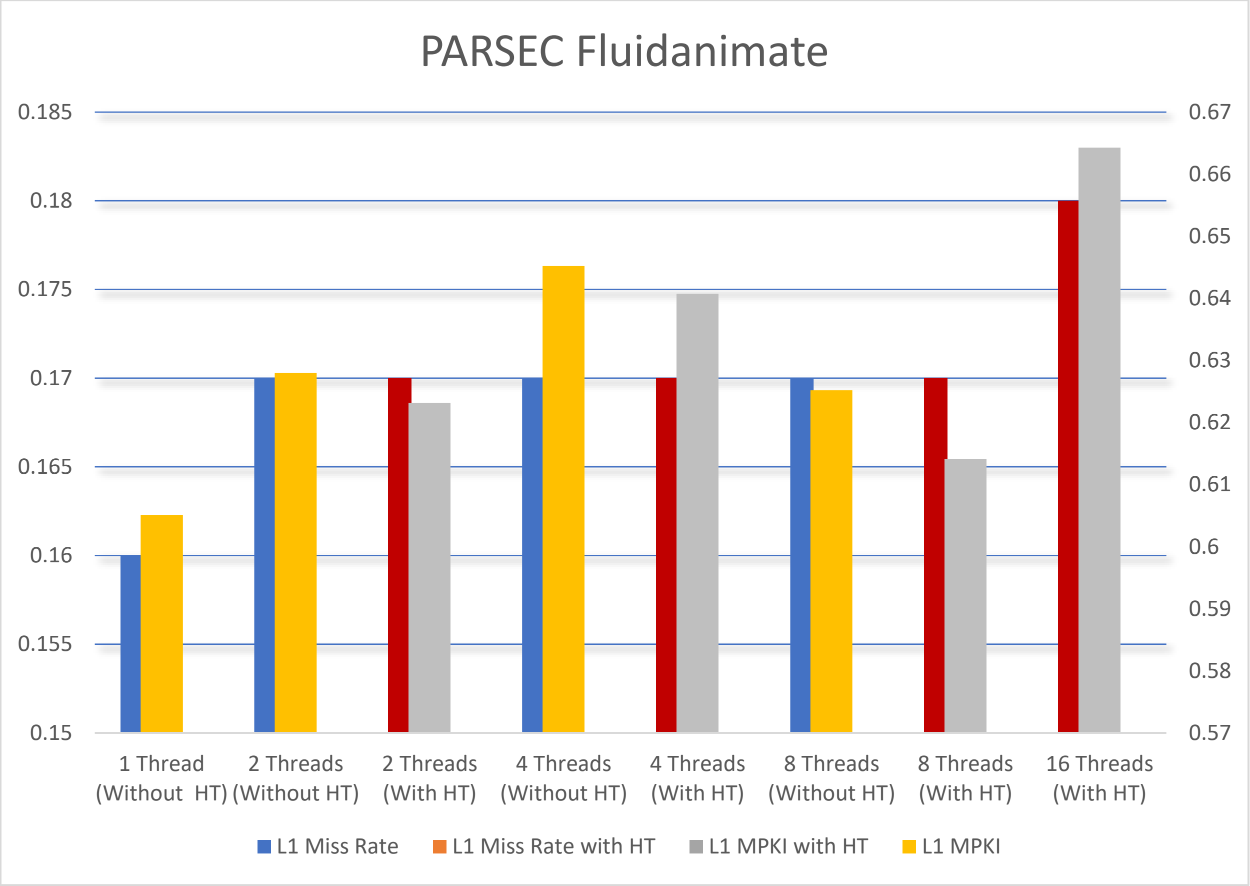
STARBENCH H264dec:



- Speedup is similar to Fluidanimate.

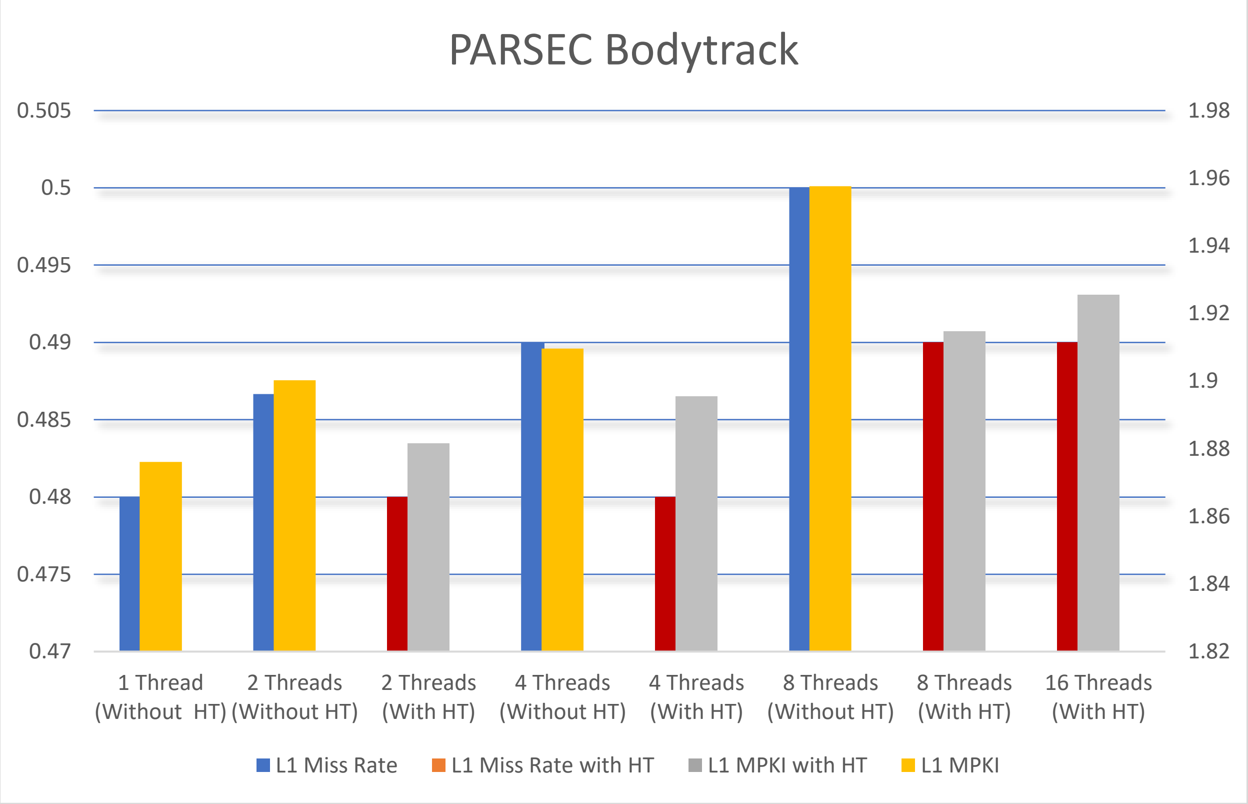
L1 D-CACHE:

PARSEC FLUIDANIMATE:



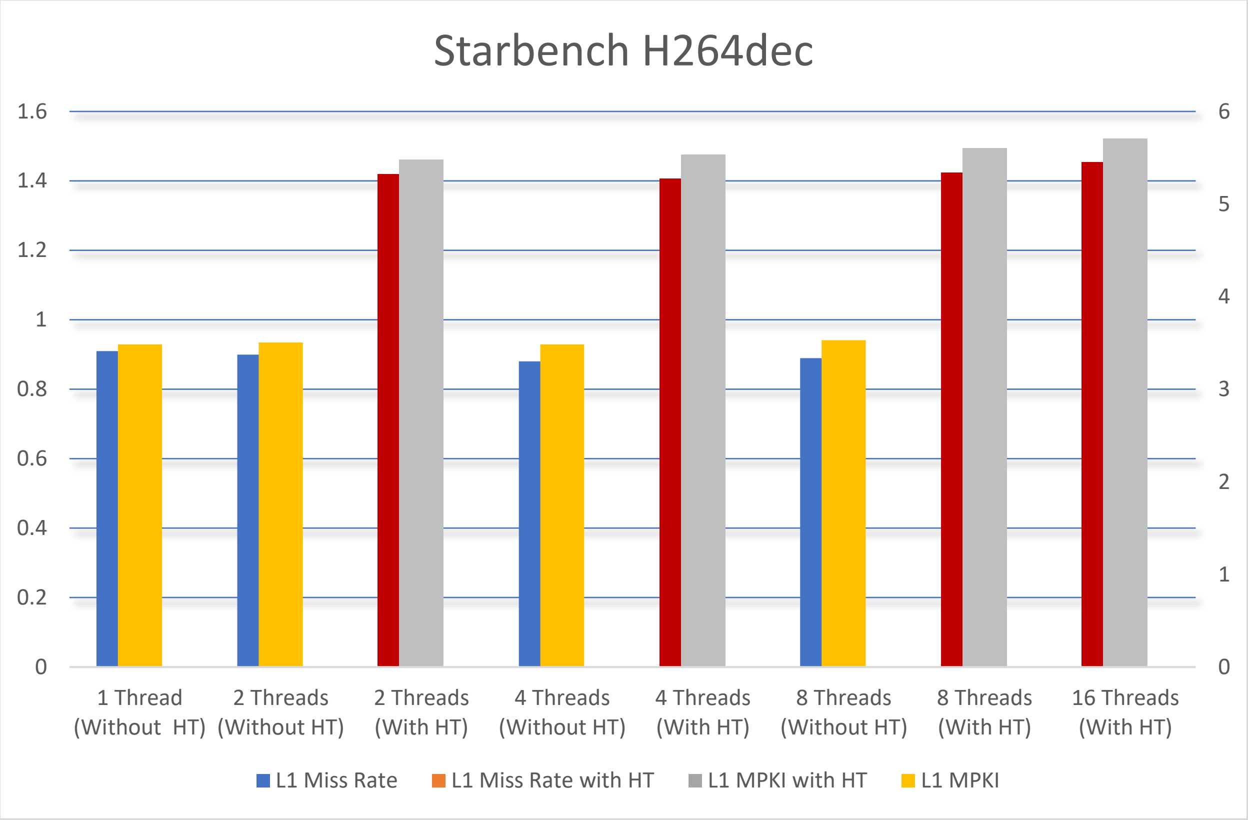
- Low miss rates are due to the fact that L1 cache has high locality. Which results in more hits and faster performance.
- Misses per 1000 instructions are high because L1 cache is smaller than L2 and LLC.

PARSEC BODYTRACK:



- More number instructions hence more loads and misses leads to higher miss rates and MPKI than parsec.

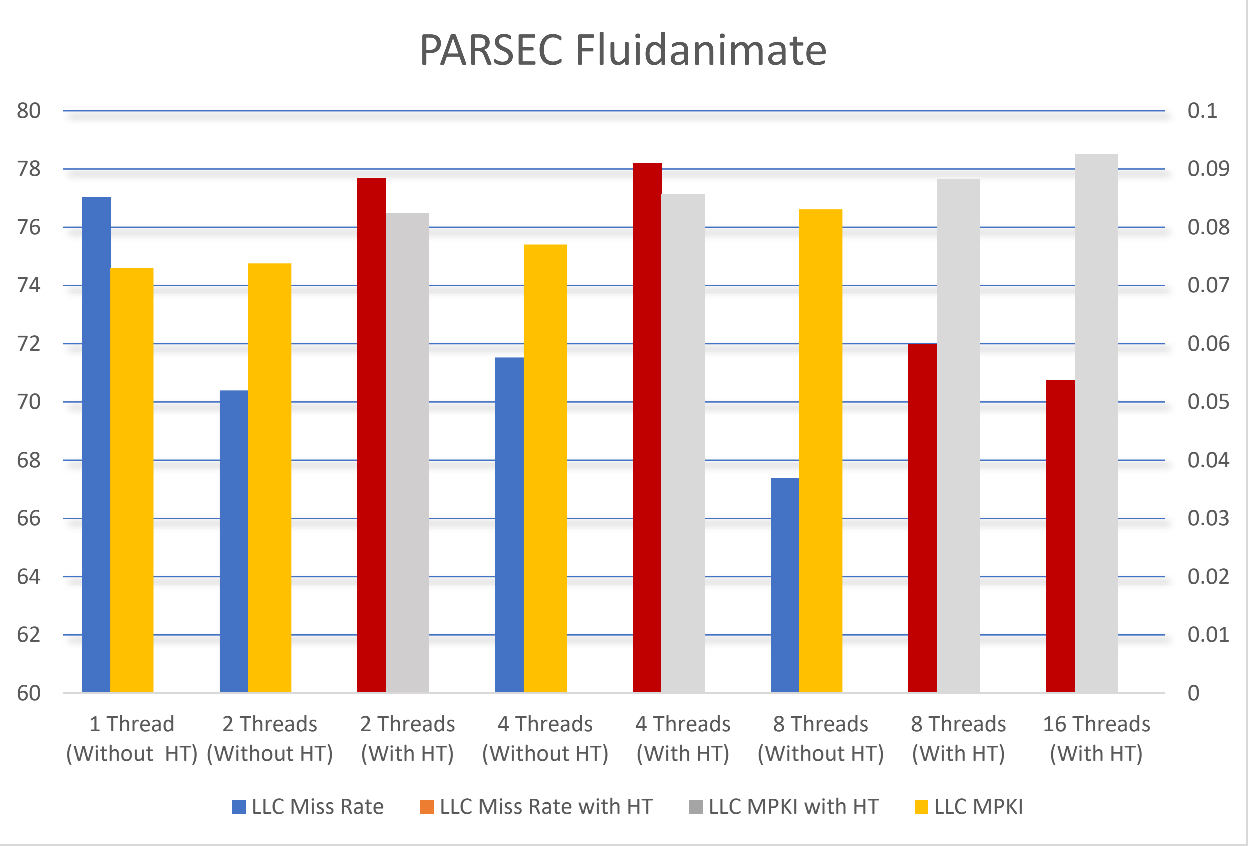
STARBENCH H264dec:



- More number instructions hence more loads and misses leads to higher miss rates and MPKI than other benchmarks.

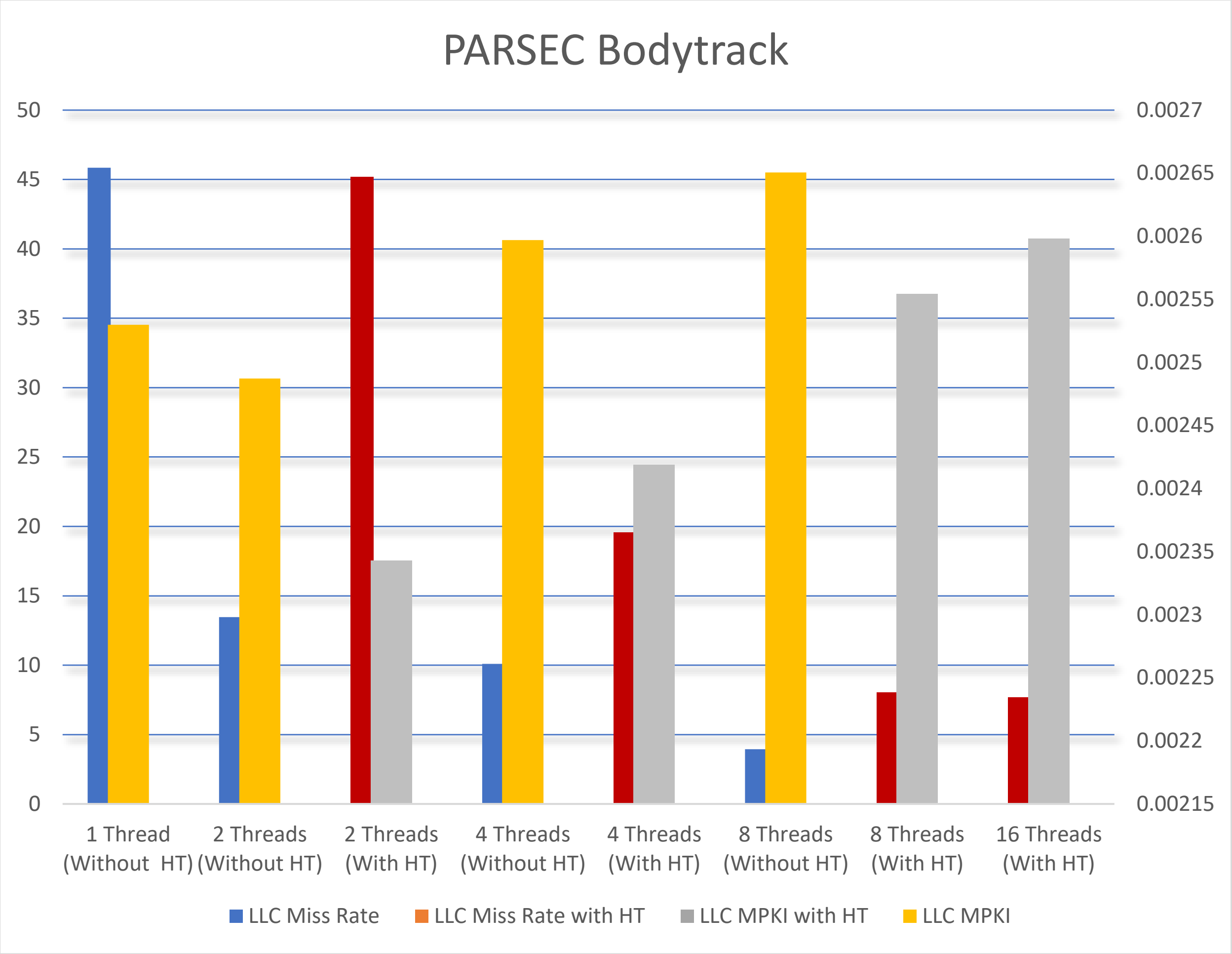
LLC CACHE:

PARSEC FLUIDANIMATE:



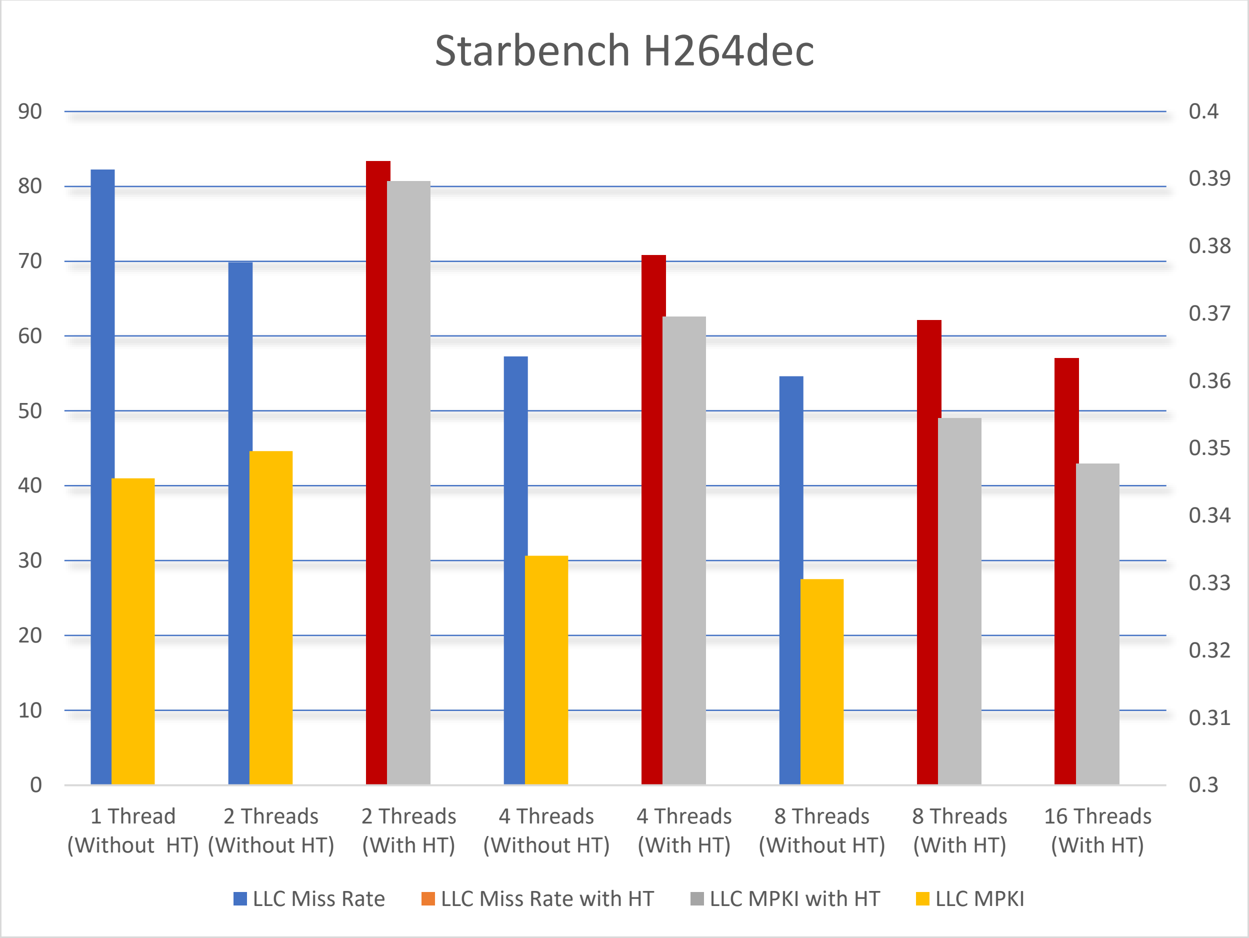
- Misses per thousand instructions are low because LLC cache is bigger than L1 and L2.
- Miss rates are higher because of poor locality than L1 and L2.

PARSEC BODYTRACK:



- Lower miss rates are due to the better locality of datasets of the parsec bodytrack benchmark.

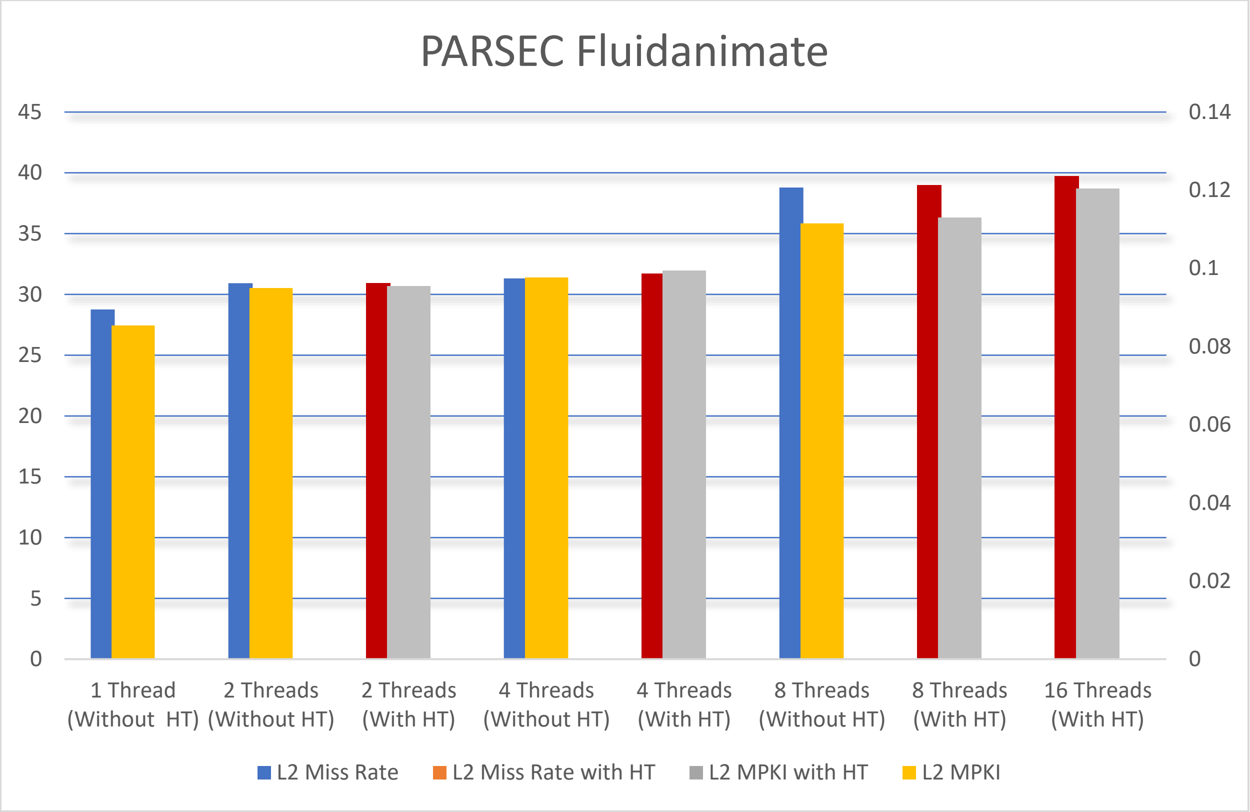
STARBENCH H264dec:



- Similar to fluidanimate.
- Miss rate decreases with number of threads because size of the LLC cache is bigger and it can bring bigger blocks of memory due to set associativity from the main memory.

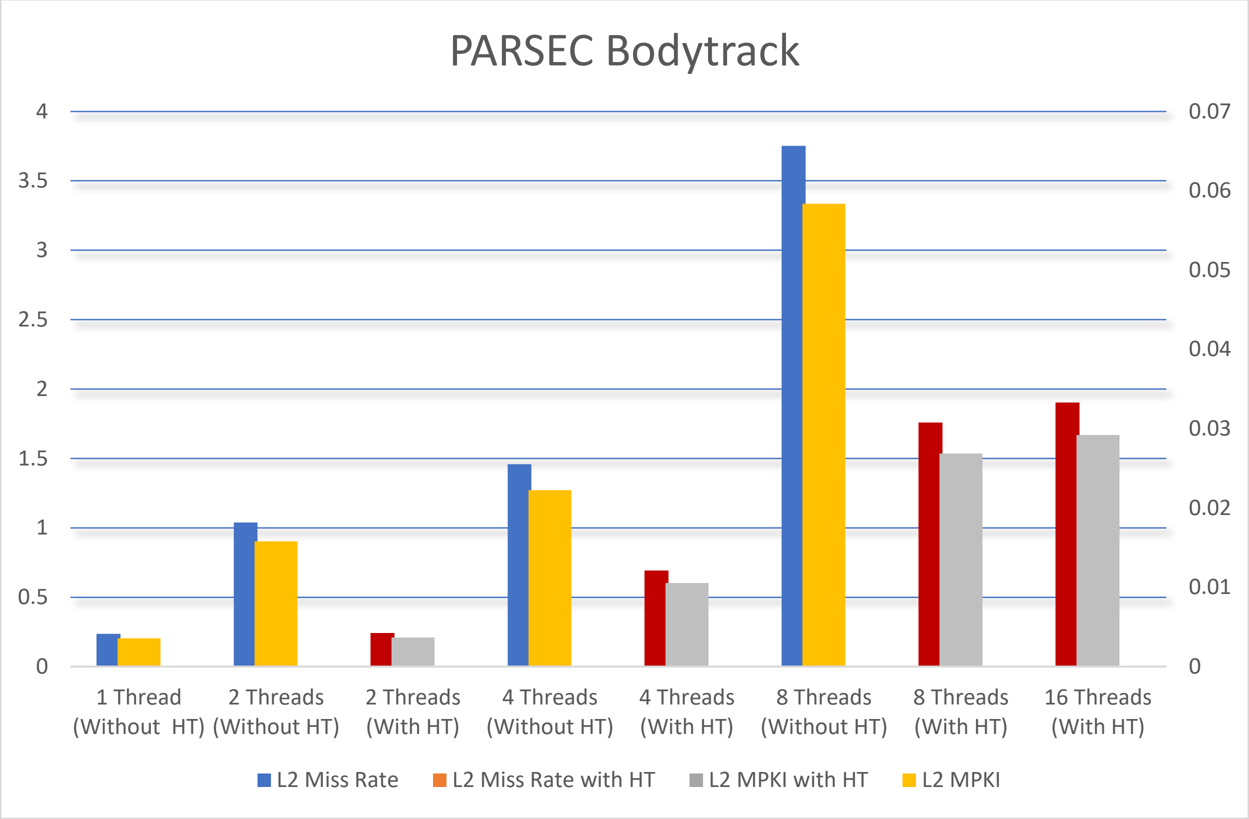
L2 CACHE:

PARSEC FLUIDANIMATE:



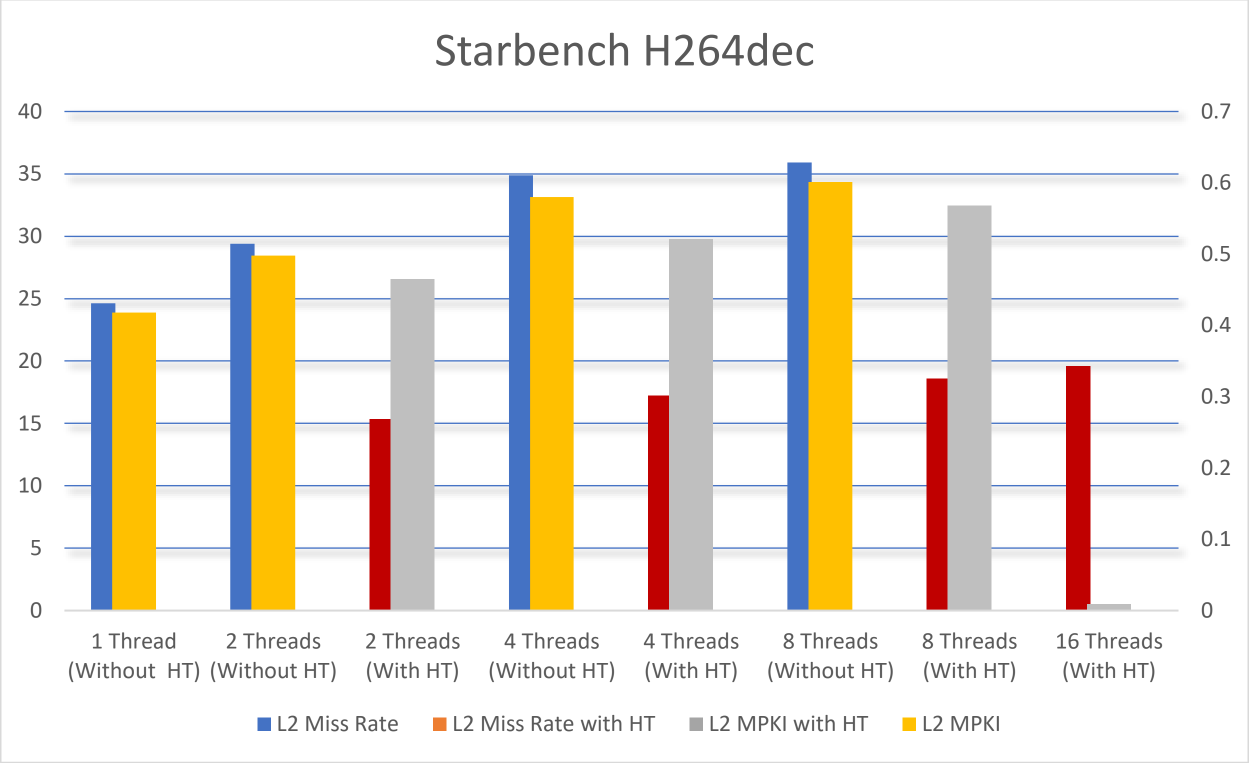
- Miss rate and MPKI in L1 and L2 increases with threads because more threads competes for the same number of instructions resulting in more misses for every benchmark.

PARSEC BODYTRACK:



- Miss rate and MPKI decreases with hyperthreading because due to hyperthreading execution units can process instructions from two threads simultaneously resulting in lesser execution time and less miss rates and MPKI.

STARBENCH H264dec:



- Miss rates of the L2 cache is in between the miss rates of L1 and LLC cache because the size of the L2 cache is lesser than LLC but greater than L1 cache.