CA Assignment #1

Name – Abhradip Ghosh **SR. No. - 21059**

1.a)

We genetated the values of performance counters using PAPI. The sample code is given below,

---

```
#include <stdio.h>
#include <stdlib.h>
#include "papi.h" /* This needs to be included every time you use PAPI */

#define NUM_EVENTS 4
#define ERROR_RETURN(retval) { fprintf(stderr, "Error %d %s:line %d: \n", retval,__FILE__,__LINE__);  exit(retval); }
#define size 2048
int matA[size][size],matB[size][size],matC[size][size];
int main()
{
  int EventSet = PAPI_NULL;
  /*must be initialized to PAPI_NULL before calling PAPI_create_event*/
  long long values[NUM_EVENTS];
  /*This is where we store the values we read from the eventset */
  /* We use number to keep track of the number of events in the EventSet */
  int retval, number;
  /* Event codes*/
  int event_codes[NUM_EVENTS]={PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_LD_INS,PAPI_SR_INS};
  char errstring[PAPI_MAX_STR_LEN];
  /*initializing matrices*/
  for(int i=0;i<size;i++){
    for(int j=0;j<size;j++){
    matA[i][j]=0;
    matB[i][j]=0;
    matC[i][j]=0;
```

```
  }
  }
  if((retval = PAPI_library_init(PAPI_VER_CURRENT)) != PAPI_VER_CURRENT )
    ERROR_RETURN(retval)
  /* Creating the eventset */
  if ( (retval = PAPI_create_eventset(&EventSet)) != PAPI_OK)
    ERROR_RETURN(retval);
  /* Add the array of events PAPI_TOT_INS and PAPI_TOT_CYC to the eventset*/
if ((retval=PAPI_add_events(EventSet, event_codes, NUM_EVENTS)) != PAPI_OK)
ERROR_RETURN(retval);
  /* Start counting */
  if ( (retval = PAPI_start(EventSet)) != PAPI_OK)
    ERROR_RETURN(retval);
  /* you can replace your code here */
  for(int i=0;i<size;i++){
    for(int j=0;j<size;j++){
      for(int k=0;k<size;k++){
        matC[i][j]+=matA[i][k]*matB[k][j];
      }
    }
  }
  /* Stop counting and store the values into the array */
  if ( (retval = PAPI_stop(EventSet, values)) != PAPI_OK)
    ERROR_RETURN(retval);
  /*write output to file*/
  FILE *fPtr;
  fPtr = fopen("loop1.txt", "a");
  fprintf(fPtr,"total instructions executed %lld \n", values[0] );
  fprintf(fPtr,"total cycles executed %lld \n",values[1]);
  fprintf(fPtr,"total load instructions %lld \n",values[2]);
  fprintf(fPtr,"total store instructions %lld \n",values[3]);
  fclose(fPtr);
  /* free the resources used by PAPI */
  PAPI_shutdown();
```

```
exit(0);
```

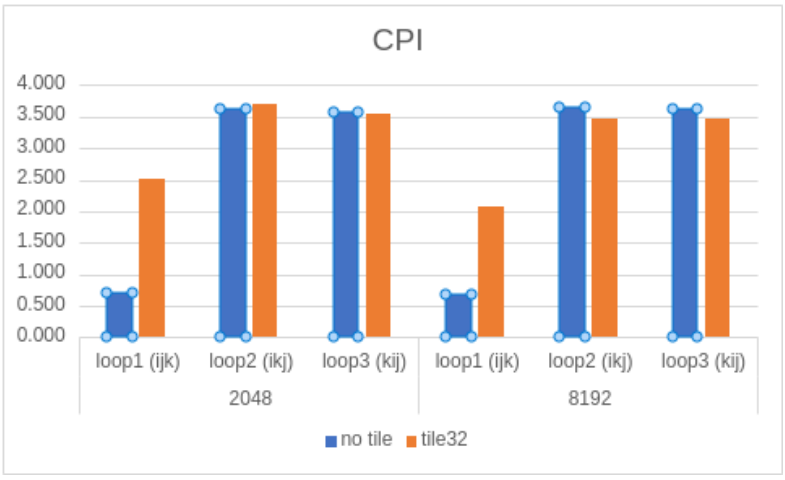
The structure of loop is as follows.

```
Loop A   -->   for(...){
Loop B   -->       for(...){
Loop C   -->           for(){
                           mat1[i][j] += mat2[i][k] + mat3[k][j]
                       }
                   }
               }
```

- For loop(ijk) loop A iterates on variable i, loop B iterates on variable j, loop C iterates on variable k.
- For loop(ikj) loop A iterates on variable i, loop B iterates on variable k, loop C iterates on variable j.
- For loop(kij) loop A iterates on variable k, loop B iterates on variable i, loop C iterates on variable j.

For a matrix mat[n][n] having n rows and n columns, the matrices are stored in row major format so, mat[1][1], mat[1][2], …..mat[1][n] are stored in consecutive memory locations in virtual memory and the next row is stored after that. One row in physical memory are stored in consecutive memory if it fits onto one page, or else it is broken into multiple pages and the part which is in a page is contiguous.

If matrix is of size 2048, then size of row is 2048*4B = 8KB, so it will be span across two physical pages.



For loop(ijk)

Variable (i) used to access a row of mat1 is used in loop A and variable (j) used to access a column of mat1 is used in loop B, so mat1 is accessed row by row manner from column 0 to 2048, we are moving to next element when the iteration of loop k completes, so when an element of mat1 is accessed for the first time it is again accessed multiple time then we move to next column, thus showing good temporal and spatial locality.

Variable (i) used to access a row of mat2 is used in loop A and variable (k) used to access a column of mat2 is used in loop C, so a whole row of mat2 is accessed in one iteration of loop B which spans across 2 pages again that same row is accessed in next iteration of loop B after a iteration of loop A completes then we move to next row of mat2. So, it doesn't show good temporal locality because if we access an element mat2[i][k], we will next access mat2[i][k+1] and we will access mat2[i][k] in next iteration of loop B.

Variable (k) used to access a row of mat3 is used in loop C and variable (j) used to access a column of mat3 is used in loop B, so in every iteration of loop C we are accessing a next row which is in different page so, it will show high TLB misses and cache misses. This is evident from the fact that this version of loop has the highest TLB misses and cache misses. As in other versions of loop we are not accessing matrix in this row wise manner.

Without Tile this version shows CPI of (0.714) and with Tile of 32 it decreases to (2.502), this difference is most compared other versions as mat3 accesses different rows which are in different pages in each iteration of innermost loop.

For loop(ikj)
Variable (i) used to access a row of mat1 is used in loop A and variable (j) used to access a column of mat1 is used in loop C, so we are accessing all elements of whole one row in mat1 in one iteration of loop B then in next iteration we are moving to next row of mat1. Access pattern of this matrix is like access pattern of mat2 in loop(ijk).

Variable (i) used to access a row of mat2 is used in loop A and variable (k) used to access a column of mat2 is used in loop B, so its access pattern is like mat1 in loop(ijk) which shows good temporal and spatial locality.

Variable (k) used to access a row of mat3 is used in loop B and variable (j) used to access a column of mat3 is used in loop C, so in every iteration of loop C we are moving from one column to next in same row. In one iteration of loop B, we are accessing one whole row then we are moving to the next row in the next iteration of loop B.

This version shows the least TLB misses among the other three. As

For loop(kij)
Variable (i) used to access a row of mat2 is used in loop A and variable (j) used to access a column of mat1 is used in loop C, its access pattern is like mat3 in loop(ikj).

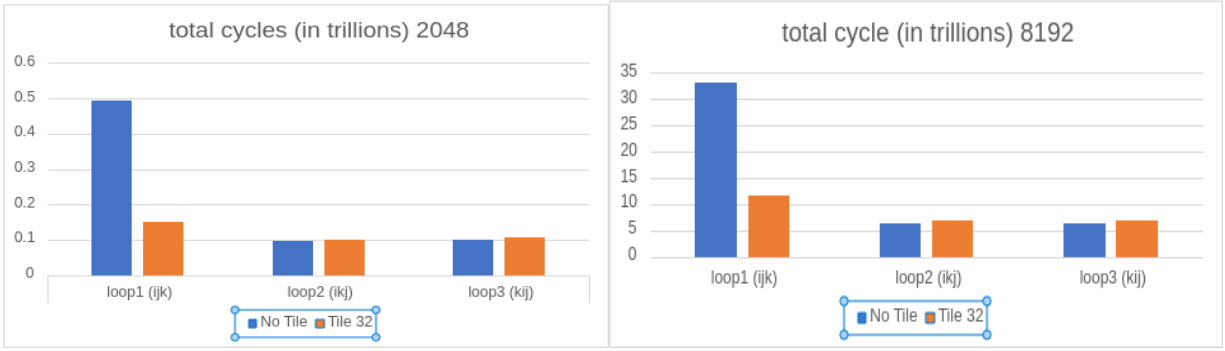
Variable (i) used to access a row of mat2 is used in loop B and variable (k) used to access a column of mat2 is used in loop A, so we access the same element of mat2 while loop C runs. And then we move to the next row during the next iteration of loop B which is on a different page. Thus, it will increase the TLB misses compared to loop(ikj), and will slightly increase the cache misses.

Variable (k) used to access a row of mat3 is used in loop A and variable (j) used to access a column of mat3 is used in loop C, so its access pattern is like access pattern mat1 in loop(ikj).

Thus, this version shows slightly worse than loop(ikj) version and loop(ijk) is the worst.

Based on this analysis we can say that loop(ikj) is the best version then loop(kij) and loop(ijk) is the worst version.

CPU cycles -



When moving to Tiled version we can see that TLB misses goes down in every version of loop as TLB has limited entries and in Tiled version we are working on a Tile then we are moving to another tile.

Cache misses decreases if there is good locality on the portion of tile that fits into cache.Thats why for some versions cache misses increases. In version (ikj) mat1 we are reading and writing on same element in loop C but in other versions we are moving to other elements.

	TLB load data misses	
	2048	8192

	loop1 (ijk)	loop2 (ikj)	loop3 (kij)	loop1 (ijk)	loop2 (ikj)	loop3 (kij)
no tile	8652965539	8457246	12694845	553746652572	541606215	610905151
tile32	7933046	311433	619059	590278062	25629949	31948210

Total data for matrices of size 2048 is given below,

	2048					
	loop1 (ijk)		loop2 (ikj)		loop3 (kij)	
	no tile	tile32	no tile	tile32	no tile	tile32
total instructions executed	352216750341	372691739121	352216700809	372691735193	352216700975	372691735787
total cycles executed	493373998201	148944477926	97033817209	100582038691	98530871402	105037422680
CPI	0.714	2.502	3.630	3.705	3.575	3.548
total load instructions	111677999317	121920633177	111681688758	121921244801	111681615446	121920779414
total l1 load misses	10571737887	8405220038	203176343	321268257	69188706	575957722
total store instructions	171882619330	177340499662	171882619330	177340499662	171882619330	177340499662
total l1 store misses	1722114	581829	32467	78280	2328	61587
total l1 total misses	11007138584	8434933373	541062164	993804862	552119261	1371274180
total l1 data misses	11006679482	8434849414	541032776	993750152	552010808	1371122258
total l1 instruction misses	459102	83959	29388	54710	108453	151922
total l2 load data misses	8628918625	2708354877	14179467	37984635	14772504	91196942
total l2 store data misses	2385967	82883	7912	724	1052	10326
total l2 cache accesses	10640717660	8409606672	186870672	315830779	67828224	573383321
total l2 cache reads	10647309162	8407666091	196894373	317811471	70996392	40985673
total l2 cache writes	820094	1078556	56544	30930	3968	573043391

total l2 cache misses	25765028342	9470861403	1189870863	698968787	1219041343	126196
total l3 cache access	25778930170	9705337984	1176899396	837043004	1219188912	1049784351
total l3 misses	21444890523	451798452	808598080	28680764	859057893	888817542
total l3 load misses	252690537	1154008	7797692	315073	11874715	474912
total tlb data misses	8652965539	7933046	8457246	311433	12694845	619059
total tlb instruction misses	43954	1508	1830	343	4936	2055

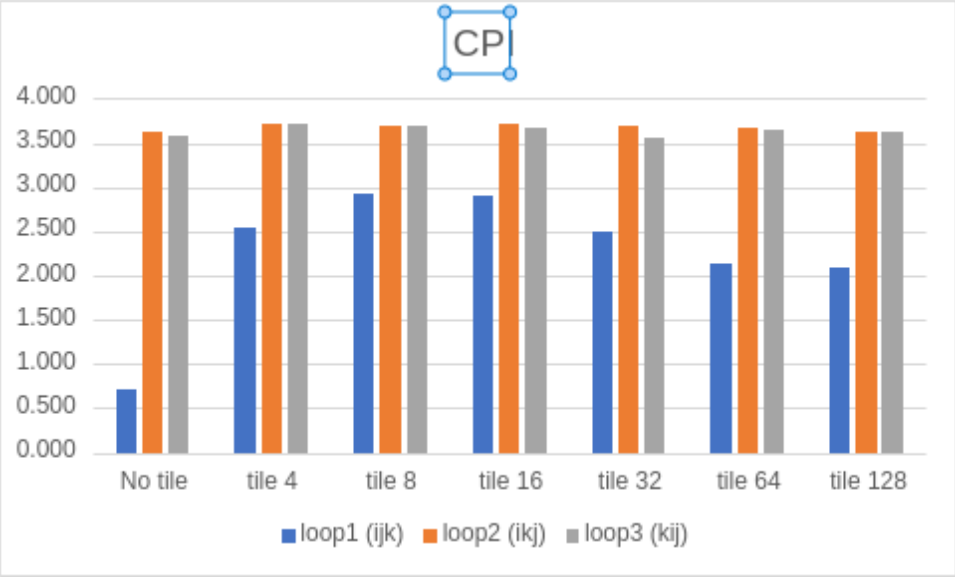
Total data for matrices of size 2048 is given below,

	8192					
	loop1 (ijk)		loop2 (ikj)		loop3 (kij)	
	no tile	tile32	no tile	tile32	no tile	tile32
total instructions executed	22540462581777	23852270080306	22540458717404	23852269726195	22540458729556	23852269823739
total cycles executed	331066127537751	115211314983329	619102511397 4	688671458975 8	621140650956 0	689463486212 7
CPI	0.681	2.070	3.641	3.464	3.629	3.460
total load instructions	71467968959 54	78028971316 20	71470212071 32	78029560447 05	71470231547 69	78029528272 34
total l1 load misses	10859708206 38	11349787940 26	9586698282	20784154187	7639944976	369726645 80
total store instructions	10996458619 30	52784894697 7	10996458619 30	11349787940 26	10996458619 30	11349787940 26
total l1 store misses	17552248	13432979	2642590	3233574	2655160	9148040
total l1 total misses	11390638122 18	53090059739 0	74711499767	58369290127	74742158860	79337774195
total l1 data misses	11390160928 50	53089559590 0	74706744407	58363822824	74736772118	79334661092
total l1 instruction misses	47719368	5001490	4755360	5467303	5386742	3113103
total l2 load data misses	57157868398 2	47135236874 4	573779341	18894255646	654880326	337784828 26

total l2 store data misses	292178791	815210	23340	87566	41371	5799403
total l2 cache accesses	1086159698146	527390929965	9063323851	20657421989	8378426760	35204107873
total l2 cache reads	1087008866915	525720536464	9018202744	20801456642	8857567091	36874492141
total l2 cache writes	26189172	28224224	3129520	50561452	9067990	30235156
total l2 cache misses	2776228903368	2166369162823	768247895008	212407608992	777947579006	250423038126
total l3 cache access	2737542673222	2170179083856	758583508338	213680398298	774764142041	257599293636
total l3 misses	1633342118022	3582536432 0	560969088 28	6380038284	554125531 86	6663866419
total l3 load misses	1061919179 0	177556567	538185215	108856978	603715060	85255860
total tlb data misses	553746652572	590278062	541606215	25629949	610905151	31948210
total tlb instruction misses	463803	291208	62822	91524	79930	38923

L1 total misses is reducing for tiled version of loop(ijk) and loop(ijk) but it is increasing for loop(kij).

b) Comparing different tile sizes of matrix size 2048 -



Execution time for 2048-

	loop1 (ijk)	loop2 (ikj)	loop3 (kij)
No tile	226.628	32.936	32.943
tile 4	51.729	34.663	34.883
tile 8	41.829	33.350	33.560
tile 16	40.677	32.791	33.181
tile 32	43.620	32.635	33.234
tile 64	52.978	32.802	33.031
tile 128	56.746	32.689	32.750

Based on the data we can say that for loop(ijk) tile 16 is best, for loop(ikj) it performs worst with increasing tile size because of increasing cache misses based on how we access mat1 in it but after tile size 16 it performs better because of less TLB misses, for it tile size 32 is best, for loop(kij) similar trends exists it has lower cache locality but since it accesses matrix row wise it has low TLB misses.

Loop(ijk) of size 2048 :

	2048						
	ijk (loop1)						
	no tile	tile4	tile8	tile16	tile32	tile64	tile128
total instructions executed	352216750341	402923471718	384030945837	376233389597	372691739121	371003315289	370178843158
total cycles executed	493373998201	158528088815	130741191927	129193228165	148944477926	172827486328	176513973328
CPI	0.714	2.542	2.937	2.912	2.502	2.147	2.097
total load instructions	111677999317	137036975483	127590668321	123691933531	121920633177	121076958102	120664135850
total l1 load misses	1057173788 7	1052826032	4300672975	8014227620	8405220038	8507195506	8565482415
total store instructions	1718826193 0	2281753911 4	1962947434 6	1832494724 2	1773404996 2	1745256663 4	1731514426 6
total l1 store misses	1722114	193461	1461023	1134852	581829	157995	97532
total l1 data misses	1100667948 2	1206919073	4371961359	8082107972	8434849414	8525065139	8577604535
total l1 instruction misses	459102	97063	79738	64286	83959	42192	33935
total l1 total misses	1100713858 4	1207016136	4372041097	8082172258	8434933373	8525107331	8577638470
total l2 load data misses	8628918625	554139493	174595268	440292061	2708354877	7311398559	8086205161
total l2 store data misses	2385967	12713	141529	37309	82883	63567	36286
total l2 cache accesses	106407176 60	1063028030	4318189671	8028453866	8409606672	8508289734	8563617982

total l2 cache reads	10647309162	1055471845	4315050565	8023232816	8407666091	8507106588	8565768975
total l2 cache misses	25765028342	1953752677	940188637	946245903	9470861403	19721816465	22357854047
total l2 cache writes	820094	363508	3492046	2304160	1078556	271734	118916
total l3 misses	21444890523	1528470842	748592907	182803954	451798452	32860727	26214113
total l3 cache access	25778930170	1801307732	935838987	961102655	9705337984	19845967406	22363847502
total l3 load misses	252690537	119233925	17829865	2709843	1154008	330930	430402
total tlb data misses	8652965539	540763211	131213659	31616515	7933046	1825439	724920
total tlb instruction misses	43954	926	1856	1302	1508	330	591

Loop(ikj) of size 2048 :

	2048						
	ikj (loop2)						
	no tile	tile4	tile8	tile16	tile32	tile64	tile128
total instructions executed	352216700809	402923467453	384030943514	376233387247	372691735193	371003309477	370178837046
total cycles executed	97033817209	108501380499	103645865924	101230418372	100582038691	101014935575	102035040078
CPI	3.630	3.714	3.705	3.717	3.705	3.673	3.628
total load instructions	111681688758	137037060548	127590898015	123692165125	121921244801	121077138879	120664886031
total l1 load misses	203176343	159327018	812326152	535684627	321268257	216879660	138656279
total store instructions	171882619330	228175391146	196294743466	183249472462	177340499624	174525666346	173151442646
total l1 store misses	32467	122999	1778892	9584	78280	103460	14841
total l1 data misses	541032776	440791822	1690450194	1328492033	993750152	735008532	648053745
total l1 instruction misses	29388	28709	34637	31826	54710	32333	17473
total l1 total misses	541062164	440820531	1690484831	1328523859	993804862	735040865	648071218
total l2 load data misses	14179467	4043946	5968686	23172115	37984635	105628841	86076738
total l2 store data misses	7912	5250	9957	421	724	5929	15885
total l2 cache accesses	186870672	165483086	817593504	532991827	315830779	214109893	142690025
total l2 cache reads	196894373	161767426	808860409	539741581	317811471	215518498	143281999

total l2 cache misses	1189870863	312096723	243246323	319364684	698968787	1576708749	2169560573
total l2 cache writes	56544	319062	4114098	220558	30930	287510	87726
total l3 misses	808598080	185121195	90707922	50347166	28680764	24330721	57797453
total l3 cache access	1176899396	316470710	245134915	390124073	837043004	1581754592	2183282108
total l3 load misses	7797692	1921439	1027028	537851	315073	232079	387785
total tlb data misses	8457246	2096727	1092122	587500	311433	159407	91020
total tlb instruction misses	1830	983	319	367	343	137	173

Loop(kij) of 2048:

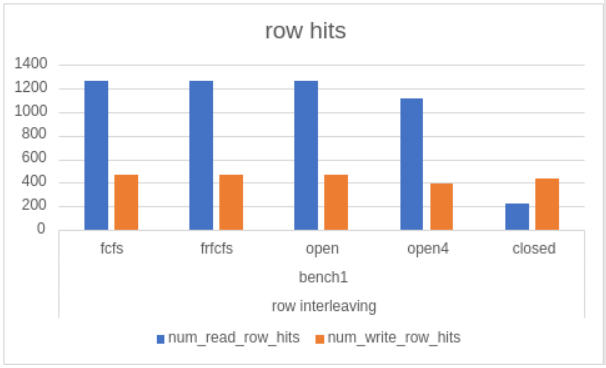
	2048						
	kij (loop3)						
	no tile	tile4	tile8	tile16	tile32	tile64	tile128
total instructions executed	352216700975	402923467453	384030943816	376233387424	372691735787	371003309612	370178837012
total cycles executed	985308714002	108737680494	104067667751	102298000086	105037422680	101582433530	102196999887
CPI	3.575	3.705	3.690	3.678	3.548	3.652	3.622
total load instructions	111681615446	137037073730	127590856977	123692116269	121920779414	121077135953	120664825379
total l1 load misses	69188706	214011150	972793197	1086196821	575957722	318879280	181628260
total store instructions	171882619330	228175391144	196294743466	183249472422	177340499624	174525666344	173151442666
total l1 store misses	2328	2308942	2729313	125799	61587	97347	52857
total l1 data misses	552010808	510775024	1907489699	2088465836	1371122258	975504499	788272305
total l1 instruction misses	108453	42590	45399	31225	151922	27185	18205
total l1 total misses	552119261	510817614	1907535098	2088497061	1371274180	975531684	788290510
total l2 load data misses	14772504	5281676	9788851	49828155	91196942	197595264	145139381
total l2 store data misses	1052	22609	17053	13858	10326	13825	737
total l2 cache accesses	67828224	204840639	953905624	1087507651	573383321	320184558	180774343
total l2 cache reads	70996392	205467604	949178938	1086479896	40985673	317773776	174003740
total l2 cache misses	1219041343	358157941	254482824	497061812	126196	2017977534	2467662042

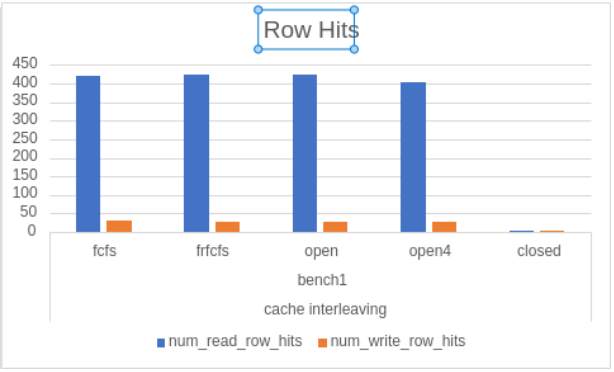
total l2 cache writes	3968	4827580	8300492	864966	573043391	222136	139426
total l3 misses	859057893	229389147	114360057	65181106	888817542	26426339	41314205
total l3 cache access	1219188912	361889651	283572666	451127314	1049784351	1999499958	2474037180
total l3 load misses	11874715	2975305	1534530	785694	474912	316331	422282
total tlb data misses	12694845	3167467	1604648	1337042	619059	261512	134771
total tlb instruction misses	4936	657	738	333	2055	102	5088

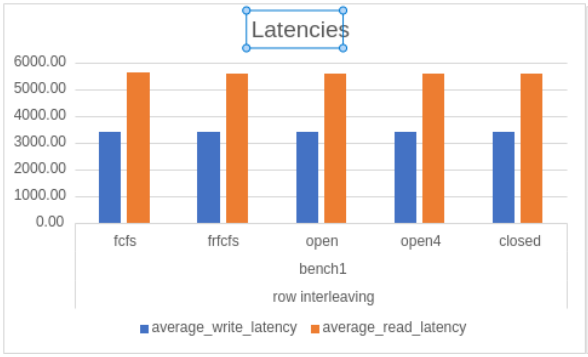
2)

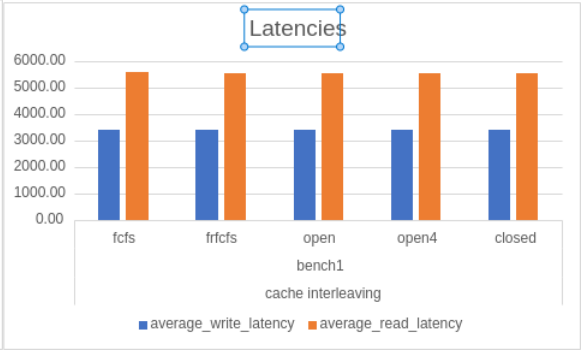
Bench 1 ->

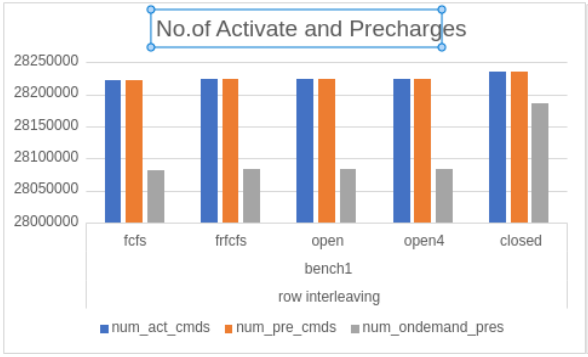
We allocate memory of size 2^32, size of row is 2^19, there are 2^14 columns, there are 2^19 banks. The main function is to select a bank and select a column which are 2^6 addresses apart iterate on it row wise which are 2^19 addresses apart, thus this benchmark shows less no of row hits compared to others. It has the lowest bandwidth.

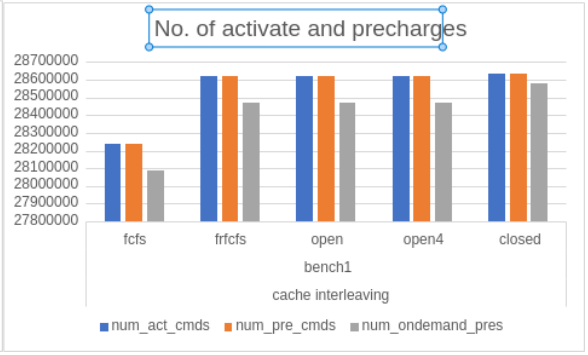






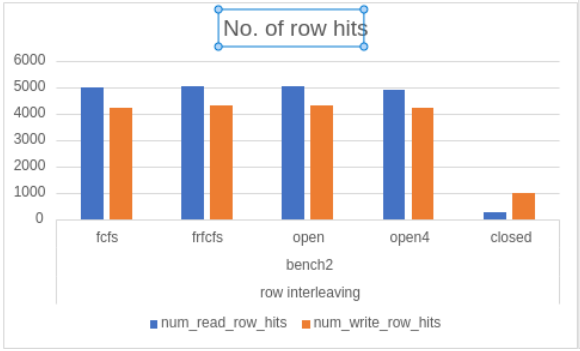


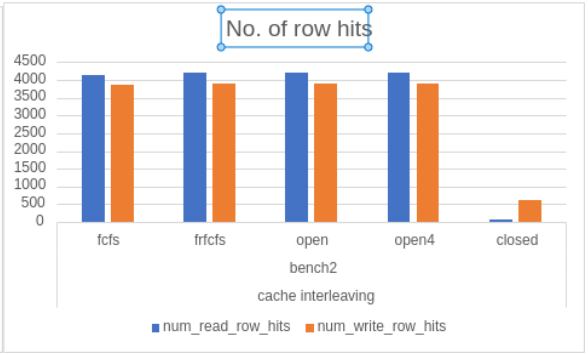


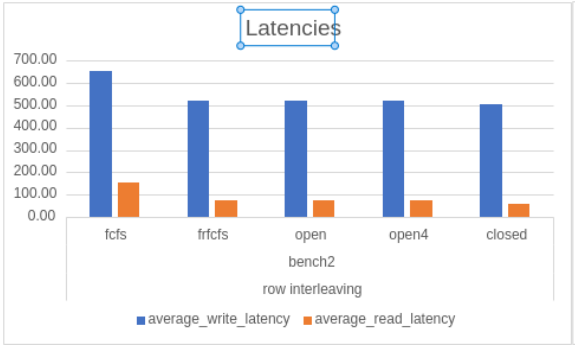


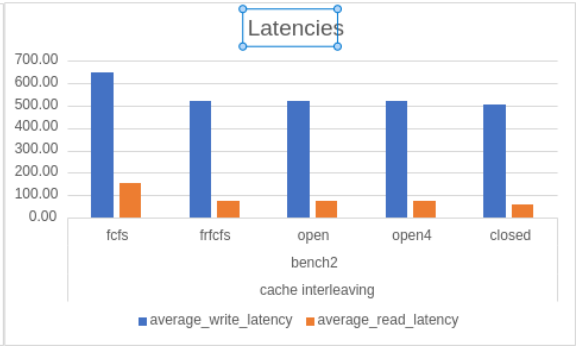
Bench 2 ->

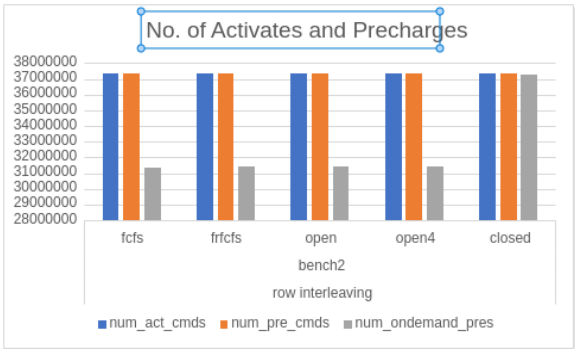
In this benchmark we are accessing the memory location completely randomly, so we are getting higher row hits compared to bench1 and average bandwidth for it is higher.

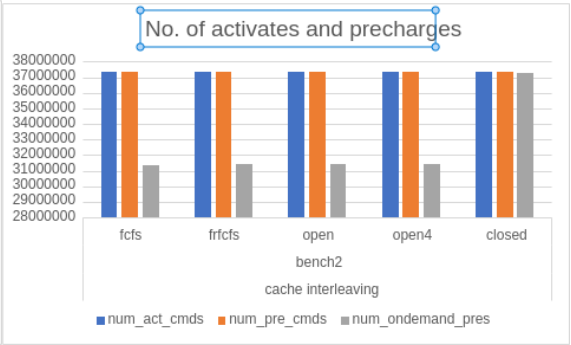






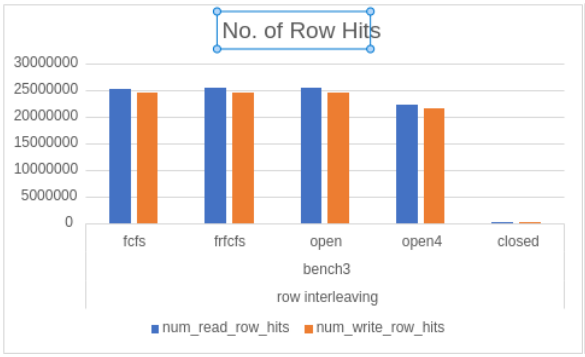


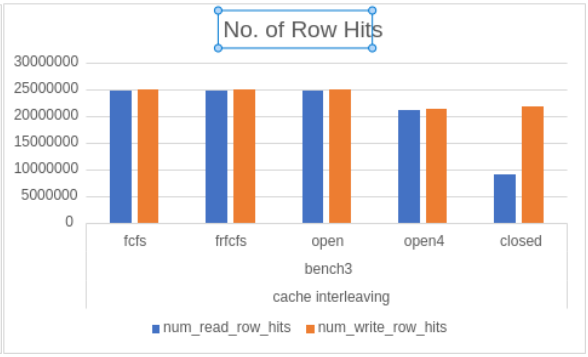


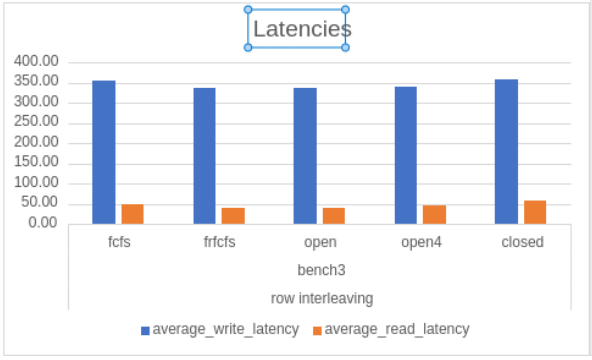


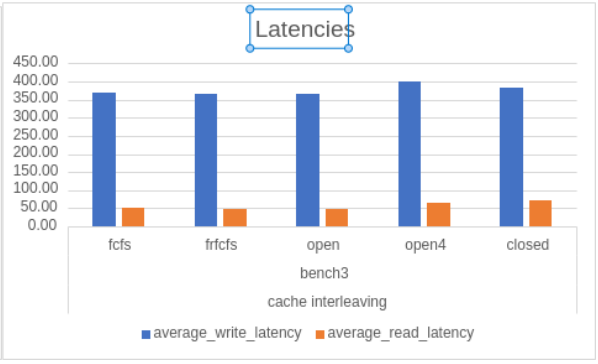
Bench 3 ->

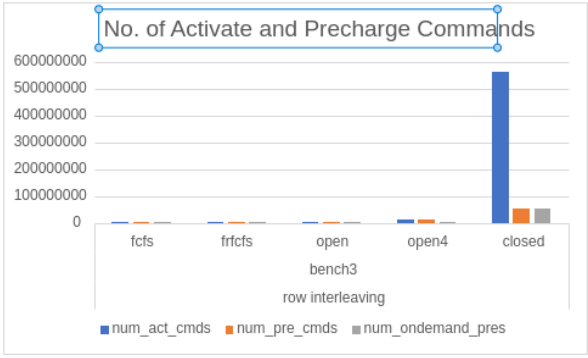
In this benchmark we are fixing a row, selecting a column and accessing it bank wise, this benchmark shows highest row hits as stride length is not high as in bench1. It has the highest bandwidth.

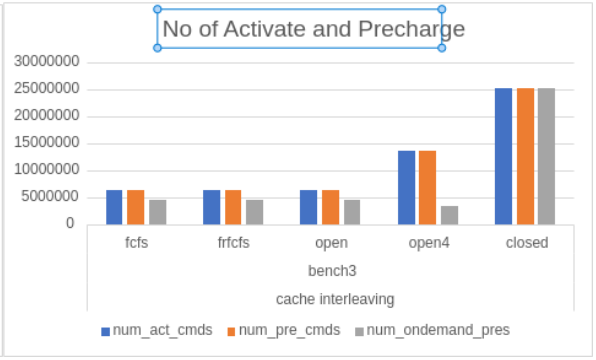












FCFS mostly performs the worst cause we can't exploit bank level parallelism with it.
FR-FCFS performs better than FCFS as we select the command which satisfies the timing and resource constraints.
In open we don't precharge the bank untill an access to new row comes and there are no pending hits for the current open row.
In closed we precharge the bank as soon as there is no pending hit to the open row, it helps when it is unlikely that an access to that row will come in future.
We observe that in all scheduling policy row hit count is most in open policy and less in open4 and least in closed policy. Similarly, the number of ondemand precharges is least on open more in open4 and most in closed.

Bench1 (Row Interleaving)-

	row interleaving				
	bench1				
	fcfs	frfcfs	open	open4	closed
num_cycles	2000000000	2000000000	2000000000	2000000000	2000000000
epoch_num	1500	1500	1500	1500	1500
num_write_buf_hits	0	0	0	0	0
num_write_cmds	14086380	14087032	14087032	14087089	14093762
num_reads_done	14085364	14086031	14086031	14086080	14092856
hbm_dual_cmds	0	0	0	0	0
num_ref_cmds	384689	384689	384689	384689	384689
num_read_row_hits	1266	1266	1266	1119	226
num_read_cmds	14085296	14085963	14085963	14086012	14092788
num_writes_done	14086400	14087057	14087057	14087105	14093781
num_write_row_hits	470	470	470	396	437
num_act_cmds	28222776	28224250	28224250	28224481	28236321
num_pre_cmds	28222775	28224249	28224249	28224480	28236304
num_ondemand_pres	28082331	28083867	28083867	28083809	28185886
rank_active_cycles.0	128765	128765	128765	125075	812
rank_active_cycles.1	1399528066	1399548432	1399548432	1399551709	509513817
all_bank_idle_cycles.0	1999871235	1999871235	1999871235	1999874925	1999999188
all_bank_idle_cycles.1	600471934	600451568	600451568	600448291	1490486183
interarrival_latency[-0]	0	0	0	0	0
interarrival_latency[0-9]	13646019	13647685	13647685	13647759	13654220
interarrival_latency[10-19]	69	81	81	104	85
interarrival_latency[20-29]	668	862	862	836	850
interarrival_latency[30-39]	651	756	756	722	754
interarrival_latency[40-49]	509	642	642	639	624
interarrival_latency[50-59]	1530987	1531054	1531054	1531596	1531652
interarrival_latency[60-69]	5704286	5703586	5703586	5703029	5706404
interarrival_latency[70-79]	6751743	6751428	6751428	6751595	6755002
interarrival_latency[80-89]	48	59	59	61	77
interarrival_latency[90-99]	30	31	31	30	30
interarrival_latency[100-]	536801	536936	536936	536847	536976
write_latency[-0]	0	0	0	0	0
write_latency[0-19]	0	0	0	0	0
write_latency[20-39]	0	2	2	1	1
write_latency[40-59]	29	37	37	6	5
write_latency[60-79]	18	24	24	6	40
write_latency[80-99]	19	12	12	5	22
write_latency[100-119]	25	21	21	2	14
write_latency[120-139]	11	13	13	17	16
write_latency[140-159]	19	19	19	9	14
write_latency[160-179]	18	17	17	22	19
write_latency[180-199]	13	11	11	30	14
write_latency[200-]	14086228	14086876	14086876	14086991	14093617
read_latency[-0]	0	0	0	0	0
read_latency[0-19]	0	0	0	0	0

read_latency[20-39]	1188	1196	1196	1041	88
read_latency[40-59]	640	668	668	805	1422
read_latency[60-79]	14	17	17	16	312
read_latency[80-99]	8	5	5	13	59
read_latency[100-119]	10	33	33	32	38
read_latency[120-139]	5	5	5	5	4
read_latency[140-159]	3	20	20	23	13
read_latency[160-179]	5	12	12	13	26
read_latency[180-199]	6	13	13	9	7
read_latency[200-]	14083485	14084062	14084062	14084123	14090887
ref_energy	3.51826E+11	3.52E+11	3.52E+11	3.52E+11	3.51826E+11
refb_energy	0	0	0	0	0
write_energy	46518900000	4.65E+10	4.65E+10	4.65E+10	46543200000
read_energy	54087500000	5.41E+10	5.41E+10	5.41E+10	54116300000
act_energy	1.36282E+11	1.36E+11	1.36E+11	1.36E+11	1.36347E+11
sref_energy.0	0	0	0	0	0
sref_energy.1	0	0	0	0	0
pre_stb_energy.0	6.71957E+11	6.72E+11	6.72E+11	6.72E+11	6.72E+11
pre_stb_energy.1	2.01759E+11	2.02E+11	2.02E+11	2.02E+11	5.00803E+11
act_stb_energy.0	56862600	5.69E+07	5.69E+07	5.52E+07	358579
act_stb_energy.1	6.18032E+11	6.18E+11	6.18E+11	6.18E+11	2.25001E+11
average_write_latency	3430.70	3430.70	3430.70	3430.10	3428.90
average_read_latency	5633.11	5632.93	5632.93	5633.01	5629.64
average_interarrival	70.993	70.9896	70.9896	70.9894	70.9555
total_energy	2.08052E+12	2.08E+12	2.08E+12	2.08E+12	1.98664E+12
average_power	1040.26	1040.27	1040.27	1040.27	993.319
average_bandwidth	1.202	1.20205	1.20205	1.20206	1.20263

Bench2 (Row Interleaving)-

	row interleaving				
	bench2				
	fcfs	frfcfs	open	open4	closed
num_cycles	2000000000	2000000000	2000000000	2000000000	2000000000
epoch_num	1500	1500	1500	1500	1500
num_write_buf_hits	0	0	0	0	0
num_write_cmds	18633577	18633577	18633577	18633577	18633578
num_reads_done	18637915	18637915	18637915	18637915	18637915
hbm_dual_cmds	0	0	0	0	0
num_ref_cmds	384689	384689	384689	384689	384689
num_read_row_hits	5038	5071	5071	4926	285
num_read_cmds	18637915	18637915	18637915	18637915	18637915
num_writes_done	18633582	18633582	18633582	18633582	18633582
num_write_row_hits	4221	4323	4323	4253	1016
num_act_cmds	37326468	37328070	37328070	37328289	37336214
num_pre_cmds	37326446	37328048	37328048	37328263	37336214
num_ondemand_pres	31365845	31394291	31394291	31394293	37270191
rank_active_cycles.0	1872589600	1872686701	1872686701	1872684619	491915289
rank_active_cycles.1	2288973	2287457	2287457	2279769	9811
all_bank_idle_cycles.0	127410400	127313299	127313299	127315381	1508084711

all_bank_idle_cycles.1	1997711027	1997712543	1997712543	1997720231	1999990189
interarrival_latency[-0]	0	0	0	0	0
interarrival_latency[0-9]	0	0	0	0	0
interarrival_latency[10-19]	243	243	243	243	243
interarrival_latency[20-29]	1653	1653	1653	1653	1653
interarrival_latency[30-39]	659	659	659	659	659
interarrival_latency[40-49]	4711895	4711895	4711895	4711895	4711895
interarrival_latency[50-59]	30429552	30429552	30429552	30429552	30429552
interarrival_latency[60-69]	1381568	1381568	1381568	1381568	1381568
interarrival_latency[70-79]	232472	232472	232472	232472	232472
interarrival_latency[80-89]	377597	377597	377597	377597	377597
interarrival_latency[90-99]	87653	87653	87653	87653	87653
interarrival_latency[100-]	48205	48205	48205	48205	48205
write_latency[-0]	0	0	0	0	0
write_latency[0-19]	0	0	0	0	0
write_latency[20-39]	1	348	348	348	0
write_latency[40-59]	41	157899	157899	157867	429135
write_latency[60-79]	38	500873	500873	500856	968813
write_latency[80-99]	36	842507	842507	842497	258566
write_latency[100-119]	21	290168	290168	290150	388078
write_latency[120-139]	16	434234	434234	434240	461132
write_latency[140-159]	20	425674	425674	425671	417574
write_latency[160-179]	15	411169	411169	411175	515874
write_latency[180-199]	106	515035	515035	515053	333897
write_latency[200-]	18633283	15055670	15055670	15055720	14860509
read_latency[-0]	0	0	0	0	0
read_latency[0-19]	0	0	0	0	0
read_latency[20-39]	3129	4734	4734	4588	97
read_latency[40-59]	8682054	16088895	16088895	16089026	17073234
read_latency[60-79]	715651	1093042	1093042	1093046	425634
read_latency[80-99]	599370	294551	294551	294556	153642
read_latency[100-119]	603012	154359	154359	154360	106488
read_latency[120-139]	602158	86830	86830	86830	64868
read_latency[140-159]	600080	55559	55559	55560	54573
read_latency[160-179]	605721	48767	48767	48767	47447
read_latency[180-199]	593311	45790	45790	45792	37832
read_latency[200-]	5633429	765388	765388	765390	674100
ref_energy	3.52E+11	3.52E+11	3.52E+11	3.52E+11	3.52E+11
refb_energy	0	0	0	0	0
write_energy	6.15E+10	6.15E+10	6.15E+10	6.15E+10	6.15E+10
read_energy	7.16E+10	7.16E+10	7.16E+10	7.16E+10	7.16E+10
act_energy	1.80E+11	1.80E+11	1.80E+11	1.80E+11	1.80E+11
sref_energy.0	0	0	0	0	0
sref_energy.1	0	0	0	0	0
pre_stb_energy.0	4.28E+10	4.28E+10	4.28E+10	4.28E+10	5.07E+11
pre_stb_energy.1	6.71E+11	6.71E+11	6.71E+11	6.71E+11	6.72E+11
act_stb_energy.0	8.27E+11	8.27E+11	8.27E+11	8.27E+11	2.17E+11
act_stb_energy.1	1.01E+09	1.01E+09	1.01E+09	1.01E+09	4.33E+06
average_write_latency	653.00	521.50	521.50	521.60	507.80
average_read_latency	153.593	74.0077	74.0077	74.008	56.5894
average_interarrival	53.6603	53.6603	53.6603	53.6603	53.6603

total_energy	2.21E+12	2.21E+12	2.21E+12	2.21E+12	2.06E+12
average_power	1103.58	1103.59	1103.59	1103.59	1030.58
average_bandwidth	1.59025	1.59025	1.59025	1.59025	1.59025

Bench3 (Row Interleaving)-

	row interleaving				
	bench3				
	fcfs	frfcfs	open	open4	closed
num_cycles	2000000000	2000000000	2000000000	2000000000	2000000000
epoch_num	1500	1500	1500	1500	1500
num_write_buf_hits	0	0	0	0	0
num_write_cmds	28209530	28209529	28209529	28209529	28209528
num_reads_done	28220232	28220232	28220232	28220232	28220231
hbm_dual_cmds	0	0	0	0	0
num_ref_cmds	384689	384689	384689	384689	384689
num_read_row_hits	25418713	25438133	25438133	22390574	7310
num_read_cmds	28220232	28220232	28220232	28220232	28220232
num_writes_done	28209534	28209534	28209534	28209534	28209534
num_write_row_hits	24677557	24656367	24656367	21619106	20538
num_act_cmds	6333951	6335711	6335711	12421309	565039453
num_pre_cmds	6333919	6335679	6335679	12421277	56503946
num_ondemand_pres	195805	197815	197815	174087	56401909
rank_active_cycles.0	1904120874	1904120935	1904120935	1904114462	722661106
rank_active_cycles.1	0	0	0	0	0
all_bank_idle_cycles.0	95879126	95879065	95879065	95885538	1277338894
all_bank_idle_cycles.1	2000000000	2000000000	2000000000	2000000000	2000000000
interarrival_latency[-0]	0	0	0	0	0
interarrival_latency[0-9]	0	0	0	0	0
interarrival_latency[10-19]	523	523	523	523	523
interarrival_latency[20-29]	3086	3086	3086	3086	3086
interarrival_latency[30-39]	51775577	51775577	51775577	51775577	51775577
interarrival_latency[40-49]	3403276	3403276	3403276	3403276	3403276
interarrival_latency[50-59]	913540	913540	913540	913540	913540
interarrival_latency[60-69]	251108	251108	251108	251108	251108
interarrival_latency[70-79]	58283	58283	58283	58283	58283
interarrival_latency[80-89]	13315	13315	13315	13315	13315
interarrival_latency[90-99]	3452	3452	3452	3452	3452
interarrival_latency[100-]	7607	7607	7607	7607	7607
write_latency[-0]	0	0	0	0	0
write_latency[0-19]	0	0	0	0	0
write_latency[20-39]	0	131328	131328	293580	0
write_latency[40-59]	2612791	2707361	2707361	2210284	4558
write_latency[60-79]	55331	126137	126137	303095	3388
write_latency[80-99]	1330641	1356703	1356703	1283974	2883732
write_latency[100-119]	1149091	1213031	1213031	1199447	1367562
write_latency[120-139]	324909	372271	372271	523765	223016
write_latency[140-159]	1211145	1249797	1249797	1125245	1345186
write_latency[160-179]	799292	865424	865424	903030	1040978
write_latency[180-199]	732667	783592	783592	798492	568462

write_latency[200-]	19993663	19403885	19403885	19568617	20772646
read_latency[-0]	0	0	0	0	0
read_latency[0-19]	0	0	0	0	0
read_latency[20-39]	23762814	24167732	24167732	21236477	767
read_latency[40-59]	1516471	2293429	2293429	4769877	25107162
read_latency[60-79]	210538	153532	153532	502715	241769
read_latency[80-99]	184934	128577	128577	189889	1412530
read_latency[100-119]	180669	112760	112760	156023	112291
read_latency[120-139]	170334	88671	88671	88712	87749
read_latency[140-159]	153190	74576	74576	74592	74480
read_latency[160-179]	149409	61932	61932	61947	62011
read_latency[180-199]	145151	60296	60296	60319	59530
read_latency[200-]	1746722	1078727	1078727	1079681	1061942
ref_energy	3.52E+11	3.52E+11	3.52E+11	3.52E+11	3.52E+11
refb_energy	0	0	0	0	0
write_energy	9.32E+10	9.32E+10	9.32E+10	9.32E+10	9.32E+10
read_energy	1.08E+11	1.08E+11	1.08E+11	1.08E+11	1.08E+11
act_energy	3.06E+10	3.06E+10	3.06E+10	6.00E+10	2.73E+11
sref_energy.0	0	0	0	0	0
sref_energy.1	0	0	0	0	0
pre_stb_energy.0	3.22E+10	3.22E+10	3.22E+10	3.22E+10	4.29E+11
pre_stb_energy.1	6.72E+11	6.72E+11	6.72E+11	6.72E+11	6.72E+11
act_stb_energy.0	8.41E+11	8.41E+11	8.41E+11	8.41E+11	3.19E+11
act_stb_energy.1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
average_write_latency	355.10	337.00	337.00	341.00	360.10
average_read_latency	49.6986	40.5717	40.5717	44.6262	58.2247
average_interarrival	35.4423	35.4423	35.4423	35.4423	35.4423
total_energy	2.13E+12	2.13E+12	2.13E+12	2.16E+12	2.25E+12
average_power	1064.51	1064.51	1064.51	1079.2	1123.26
average_bandwidth	2.40767	2.40767	2.40767	2.40767	2.40767

Bench1 (Cache Interleaving)-

	cache interleaving				
	bench1				
	fcfs	frfcfs	open	open4	closed
num_cycles	2000000000	2000000000	2000000000	2000000000	2000000000
epoch_num	1500	1500	1500	1500	1500
num_write_buf_hits	0	0	0	0	0
num_write_cmds	14092746	14285032	14285032	14285032	14290157
num_reads_done	14091799	14284015	14284015	14284015	14289021
hbm_dual_cmds	0	0	0	0	0
num_ref_cmds	384689	384689	384689	384689	384689
num_read_row_hits	422	425	425	405	4
num_read_cmds	14091733	14283947	14283947	14283947	14288954
num_writes_done	14092776	14285053	14285053	14285053	14290173
num_write_row_hits	30	28	28	28	2
num_act_cmds	28236080	28621219	28621219	28621240	28631608
num_pre_cmds	28236079	28621218	28621218	28621239	28631600
num_ondemand_pres	28085442	28468482	28468482	28468482	28579105

rank_active_cycles.0	130761	121022	121022	121022	831
rank_active_cycles.1	1412255893	1416317936	1416317936	1416317937	515619914
all_bank_idle_cycles.0	1999869239	1999878978	1999878978	1999878978	1999999169
all_bank_idle_cycles.1	587744107	583682064	583682064	583682063	1484380086
interarrival_latency[-0]	0	0	0	0	0
interarrival_latency[0-9]	13653238	14097309	14097309	14097309	14106107
interarrival_latency[10-19]	3544	11368	11368	11368	9534
interarrival_latency[20-29]	1387	52970	52970	52970	51147
interarrival_latency[30-39]	9529	37315	37315	37315	35999
interarrival_latency[40-49]	528	33889	33889	33889	30670
interarrival_latency[50-59]	1532422	1525239	1525239	1525239	1530132
interarrival_latency[60-69]	5701363	5579537	5579537	5579537	5580329
interarrival_latency[70-79]	6745683	6686470	6686470	6686470	6685577
interarrival_latency[80-89]	48	2667	2667	2667	7192
interarrival_latency[90-99]	30	31	31	31	30
interarrival_latency[100-]	536835	542305	542305	542305	542510
write_latency[-0]	0	0	0	0	0
write_latency[0-19]	0	0	0	0	0
write_latency[20-39]	0	307	307	307	320
write_latency[40-59]	1	52	52	52	35
write_latency[60-79]	0	20	20	20	17
write_latency[80-99]	1	133	133	133	152
write_latency[100-119]	2	261	261	261	258
write_latency[120-139]	1	16	16	16	18
write_latency[140-159]	6	231	231	231	217
write_latency[160-179]	6	171	171	171	170
write_latency[180-199]	36	169	169	169	176
write_latency[200-]	14092693	14283672	14283672	14283672	14288794
read_latency[-0]	0	0	0	0	0
read_latency[0-19]	0	0	0	0	0
read_latency[20-39]	266	407	407	386	0
read_latency[40-59]	1197	8025	8025	8043	8751
read_latency[60-79]	26	339	339	341	27
read_latency[80-99]	11	749	749	749	754
read_latency[100-119]	21	7565	7565	7565	7594
read_latency[120-139]	21	35	35	35	36
read_latency[140-159]	18	2970	2970	2970	1637
read_latency[160-179]	21	3386	3386	3386	6468
read_latency[180-199]	17	1782	1782	1782	78
read_latency[200-]	14090201	14258757	14258757	14258758	14263676
ref_energy	3.51826E+11	3.52E+11	3.52E+11	3.52E+11	3.52E+11
refb_energy	0	0	0	0	0
write_energy	46539900000	4.72E+10	4.72E+10	4.72E+10	4.72E+10
read_energy	54112300000	5.49E+10	5.49E+10	5.49E+10	5.49E+10
act_energy	1.36346E+11	1.38E+11	1.38E+11	1.38E+11	1.38E+11
sref_energy.0	0	0	0	0	0
sref_energy.1	0	0	0	0	0
pre_stb_energy.0	6.71956E+11	6.72E+11	6.72E+11	6.72E+11	6.72E+11
pre_stb_energy.1	1.97482E+11	1.96E+11	1.96E+11	1.96E+11	4.99E+11
act_stb_energy.0	57744100	5.34E+07	5.34E+07	5.34E+07	366970
act_stb_energy.1	6.23652E+11	6.25E+11	6.25E+11	6.25E+11	2.28E+11

average_write_latency	3429.00	3401.00	3401.90	3401.90	3400.70
average_read_latency	5630.4	5561	5561	5561	5558.62
average_interarrival	70.9607	70.0056	70.0056	70.0056	69.9809
total_energy	2.08197E+12	2.09E+12	2.09E+12	2.09E+12	1.99E+12
average_power	1040.99	1042.82	1042.82	1042.82	995.297
average_bandwidth	1.20254	1.21895	1.21895	1.21895	1.21938

Bench2 (Cache Interleaving)-

	cache interleaving				
	bench2				
	fcfs	frfcfs	open	open4	closed
num_cycles	2000000000	2000000000	2000000000	2000000000	2000000000
epoch_num	1500	1500	1500	1500	1500
num_write_buf_hits	0	0	0	0	0
num_write_cmds	18633577	18633577	18633577	18633577	18633578
num_reads_done	18637915	18637915	18637915	18637915	18637915
hbm_dual_cmds	0	0	0	0	0
num_ref_cmds	384689	384689	384689	384689	384689
num_read_row_hits	4155	4201	4201	4195	61
num_read_cmds	18637915	18637915	18637915	18637915	18637915
num_writes_done	18633582	18633582	18633582	18633582	18633582
num_write_row_hits	3865	3889	3889	3889	604
num_act_cmds	37327665	37329331	37329331	37329337	37336695
num_pre_cmds	37327643	37329308	37329308	37329314	37336688
num_ondemand_pres	31366395	31395026	31395026	31395026	37270828
rank_active_cycles.0	1872590546	1872686454	1872686454	1872686454	491892124
rank_active_cycles.1	2282717	2286140	2286140	2286140	9916
all_bank_idle_cycles.0	127409454	127313546	127313546	127313546	1508107876
all_bank_idle_cycles.1	1997717283	1997713860	1997713860	1997713860	1999990084
interarrival_latency[-0]	0	0	0	0	0
interarrival_latency[0-9]	0	0	0	0	0
interarrival_latency[10-19]	243	243	243	243	243
interarrival_latency[20-29]	1653	1653	1653	1653	1653
interarrival_latency[30-39]	659	659	659	659	659
interarrival_latency[40-49]	4711895	4711895	4711895	4711895	4711895
interarrival_latency[50-59]	30429552	30429552	30429552	30429552	30429552
interarrival_latency[60-69]	1381568	1381568	1381568	1381568	1381568
interarrival_latency[70-79]	232472	232472	232472	232472	232472
interarrival_latency[80-89]	377597	377597	377597	377597	377597
interarrival_latency[90-99]	87653	87653	87653	87653	87653
interarrival_latency[100-]	48205	48205	48205	48205	48205
write_latency[-0]	0	0	0	0	0
write_latency[0-19]	0	0	0	0	0
write_latency[20-39]	0	312	312	312	0
write_latency[40-59]	8	157524	157524	157524	428075
write_latency[60-79]	23	500689	500689	500689	969459
write_latency[80-99]	18	843162	843162	843162	257869
write_latency[100-119]	8	291198	291198	291198	389494
write_latency[120-139]	7	434009	434009	434009	460978
write_latency[140-159]	12	425704	425704	425704	419655

write_latency[160-179]	7	411366	411366	411366	515286
write_latency[180-199]	107	514696	514696	514696	334638
write_latency[200-]	18633387	15054917	15054917	15054917	14858124
read_latency[-0]	0	0	0	0	0
read_latency[0-19]	0	0	0	0	0
read_latency[20-39]	2214	3892	3892	3886	1
read_latency[40-59]	8683930	16088710	16088710	16088716	17073256
read_latency[60-79]	714305	1094178	1094178	1094178	426267
read_latency[80-99]	599107	295243	295243	295243	152825
read_latency[100-119]	603227	153936	153936	153936	106883
read_latency[120-139]	602164	87003	87003	87003	64621
read_latency[140-159]	601459	55010	55010	55010	54608
read_latency[160-179]	606609	48846	48846	48846	47495
read_latency[180-199]	595116	45846	45846	45846	37839
read_latency[200-]	5629784	765251	765251	765251	674120
ref_energy	3.52E+11	3.52E+11	3.52E+11	3.52E+11	3.52E+11
refb_energy	0	0	0	0	0
write_energy	6.15E+10	6.15E+10	6.15E+10	6.15E+10	6.15E+10
read_energy	7.16E+10	7.16E+10	7.16E+10	7.16E+10	7.16E+10
act_energy	1.80E+11	1.80E+11	1.80E+11	1.80E+11	1.80E+11
sref_energy.0	0	0	0	0	0
sref_energy.1	0	0	0	0	0
pre_stb_energy.0	4.28E+10	4.28E+10	4.28E+10	4.28E+10	5.07E+11
pre_stb_energy.1	6.71E+11	6.71E+11	6.71E+11	6.71E+11	6.72E+11
act_stb_energy.0	8.27E+11	8.27E+11	8.27E+11	8.27E+11	2.17E+11
act_stb_energy.1	1.01E+09	1.01E+09	1.01E+09	1.01E+09	4.38E+06
average_write_latency	648.00	521.60	521.60	521.60	507.80
average_read_latency	153.587	74.0131	74.0131	74.0131	56.5884
average_interarrival	53.6603	53.6603	53.6603	53.6603	53.6603
total_energy	2.21E+12	2.21E+12	2.21E+12	2.21E+12	2.06E+12
average_power	1103.58	1103.59	1103.59	1103.59	1030.58
average_bandwidth	1.59025	1.59025	1.59025	1.59025	1.59025

Bench3 (Cache Interleaving)-

	cache interleaving				
	bench3				
	fcfs	frfcfs	open	open4	closed
num_cycles	2000000000	2000000000	2000000000	2000000000	2000000000
epoch_num	1500	1500	1500	1500	1500
num_write_buf_hits	0	0	0	0	0
num_write_cmds	28209529	28209529	28209529	28209530	28209529
num_reads_done	28220232	28220232	28220232	28220231	28220231
hbm_dual_cmds	0	0	0	0	0
num_ref_cmds	384689	384689	384689	384689	384689
num_read_row_hits	24954237	24801723	24799201	21239296	9122575
num_read_cmds	28220232	28220232	28220232	28220231	28220232
num_writes_done	28209534	28209534	28209534	28209534	28209534
num_write_row_hits	24997413	25176852	25182872	21437074	22009586
num_act_cmds	6489060	6462180	6458682	13777379	25343848

num_pre_cmds	6489057	6462177	6458679	13777376	25343845
num_ondemand_pres	4499720	4470621	4467123	3337524	25281529
rank_active_cycles.0	1900828424	1901759183	1901762816	1899909464	602347532
rank_active_cycles.1	0	0	0	0	0
all_bank_idle_cycles.0	99171576	98240817	98237184	100090536	1397652468
all_bank_idle_cycles.1	2000000000	2000000000	2000000000	2000000000	2000000000
interarrival_latency[-0]	0	0	0	0	0
interarrival_latency[0-9]	0	0	0	0	0
interarrival_latency[10-19]	523	523	523	523	523
interarrival_latency[20-29]	3086	3086	3086	3086	3086
interarrival_latency[30-39]	51775577	51775577	51775577	51775577	51775577
interarrival_latency[40-49]	3403276	3403276	3403276	3403276	3403276
interarrival_latency[50-59]	913540	913540	913540	913540	913540
interarrival_latency[60-69]	251108	251108	251108	251108	251108
interarrival_latency[70-79]	58283	58283	58283	58283	58283
interarrival_latency[80-89]	13315	13315	13315	13315	13315
interarrival_latency[90-99]	3452	3452	3452	3452	3452
interarrival_latency[100-]	7607	7607	7607	7607	7607
write_latency[-0]	0	0	0	0	0
write_latency[0-19]	0	7	7	21	0
write_latency[20-39]	0	319306	319402	354678	210
write_latency[40-59]	1299359	1340531	1340635	266179	193575
write_latency[60-79]	169007	246733	247800	267689	930671
write_latency[80-99]	791510	720467	721046	185400	818583
write_latency[100-119]	666723	757551	758576	668014	478347
write_latency[120-139]	417821	541490	542495	465067	996670
write_latency[140-159]	1934151	1780035	1780982	974512	650733
write_latency[160-179]	605799	651950	652921	407794	1353344
write_latency[180-199]	1112471	991670	992040	1093873	1048670
write_latency[200-]	21212688	20859789	20853625	23526303	21738726
read_latency[-0]	0	0	0	0	0
read_latency[0-19]	0	0	0	0	0
read_latency[20-39]	20200025	20550056	20552298	15652580	2931139
read_latency[40-59]	3957174	4094940	4096528	5924357	12918671
read_latency[60-79]	469075	467633	469702	980888	8059903
read_latency[80-99]	633120	537516	537947	788171	1380860
read_latency[100-119]	672300	594766	594736	877078	497652
read_latency[120-139]	265842	209372	209247	636341	588183
read_latency[140-159]	641962	575536	574397	692695	379044
read_latency[160-179]	153354	62729	60599	461271	188470
read_latency[180-199]	91394	60591	58632	516819	154480
read_latency[200-]	1135986	1067093	1066146	1690031	1121829
ref_energy	3.52E+11	3.52E+11	3.52E+11	3.52E+11	3.52E+11
refb_energy	0	0	0	0	0
write_energy	9.32E+10	9.32E+10	9.32E+10	9.32E+10	9.32E+10
read_energy	1.08E+11	1.08E+11	1.08E+11	1.08E+11	1.08E+11
act_energy	3.13E+10	3.12E+10	3.12E+10	6.65E+10	1.22E+11
sref_energy.0	0	0	0	0	0
sref_energy.1	0	0	0	0	0
pre_stb_energy.0	3.33E+10	3.30E+10	3.30E+10	3.36E+10	4.70E+11
pre_stb_energy.1	6.72E+11	6.72E+11	6.72E+11	6.72E+11	6.72E+11

act_stb_energy.0	8.39E+11	8.40E+11	8.40E+11	8.39E+11	2.66E+11
act_stb_energy.1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
average_write_latency	370.40	366.90	366.80	400.90	384.10
average_read_latency	50.921	48.5666	48.5384	66.5018	71.6253
average_interarrival	35.4423	35.4423	35.4423	35.4423	35.4423
total_energy	2.13E+12	2.13E+12	2.13E+12	2.16E+12	2.08E+12
average_power	1064.71	1064.69	1064.68	1082.25	1041.67
average_bandwidth	2.40767	2.40767	2.40767	2.40767	2.40767