

SDSC PA1 Report

Josyula V N Taraka Abhishek - 200010021

M V Karthik - 200010030

September 2022

Contents

1	Analysis of Matmul	2
1.1	Initialisation	2
1.2	Implementation:	2
1.3	Compiation:	2
1.4	Usage:	2
1.5	Idea behind deciding for row major or column major	2
1.6	matmul1 :	3
1.7	matmul2 :	3
1.8	matmul3 :	3
1.9	Results:	3
2	Analysing Matmul using papi	4
2.1	Analysing the system	4
2.2	What we are trying to do:	5
2.3	L3 Cache Hit:	5

List of Tables

1	Matmul1	4
2	Matmul2	4
3	Matmul3	4

List of Figures

1	PAPI L3 cache error	5
---	-------------------------------	---

1 Analysis of Matmul

1.1 Initialisation

Each matrix is a 2D array of type `double**`. Each matrix has been allocated on heap using `new`.

Matrix C is zero initialized. Matrix A and B is initialized to 1 in all elements. At the end C is $\text{dim} \times \text{I}$ where I is unit matrix. This is our validation for each matmul algorithm.

The source code for each matmul is available in src folder, header files in the inc folder. The final executable is present in bin folder

1.2 Implementation:

- matmul1 is implementation of ijk loop
- matmul2 is implementation of kij loop
- matmul3 is implementation of row blocking
- -O3 optimisation flag is used

1.3 Compilation:

In matmul folder:

- "\$ make" to compile all versions
- "\$ make matmul1" to compile ith version
- "\$ make clean" to remove executables.

1.4 Usage:

- "\$matmul1 dim" or "\$matmul1"
- "\$matmul2 dim" or "\$matmul2"
- "\$matmul2 dim blk_size" or "\$matmul2"

1.5 Idea behind deciding for row major or column major

We implement the most frequently accessed matrix, which is the matrix looped over in the inner loops as row major and other one as col major.

1.6 matmul1 :

After Initialization we implement

- A into row major
- B into col major
- C into row major

because ijk::multiplication algorithm is After analysing the loop: $C[i][*] += A[i][*] + B[*][k]$
so we bring A,C to row major.

1.7 matmul2 :

After Initialization we implement

- A into col major
- B into row major
- C into row major

because kij::multiplication algorithm is After analysing the loop: $C[i][*] += A[*][k] + B[k][*]$
so we bring C, B to col major

1.8 matmul3 :

After Initialization we implement

- A into col major
- B into row major
- C into row major

because rowblocked::multiplication algorithm is After analysing the loop:
 $C[blk][*] += A[*][k] + B[blk][*]$
so we bring B,C to col major

1.9 Results:

We took the average of data which we ran 5 times for each matmul version and block size.

Table 1: Matmul1

N	L1 misses	L2 misses	Total instructions	total cycles	Ratio	Time taken(ms)
1024	135129770	3096791	6449802052	5336823593	43.6354	1839.57
2048	1152251601	123346992	51569049602	45364427359	9.34155	14943.3
4096	18520046741	1751680399	412436292236	369518241777	10.5727	121940

Table 2: Matmul2

N	L1 misses	L2 misses	Total instructions	total cycles	Ratio	Time taken(ms)
1024	135059008	101788005	1994429497	1376419503	1.32687	527.328
2048	1216912215	827955632	15493882504	21232318184	1.46978	7568.92
4096	17321695385	6865740819	122106896635	210350581223	2.52292	78369.8

Table 3: Matmul3

N	K	L1 misses	L2 misses	Total instructions	total cycles	Ratio	Time taken(ms)
1024	16	135854221	122301831	1992017466	1268881521	1.110872	502
1024	32	136961017	50019142	1992787866	1119727456	2.747684	404
1024	64	137908286	3321429	1994328876	919614800	41.76642	338
1024	128	141140312	2526496	1997410969	882501977	55.876	307
2048	16	1193306947	1024481178	15482738230	10712611061	1.16482	3751
2048	32	1196873404	1029252195	15484281957	10891386622	1.16291	3858
2048	64	1204127809	1035948994	15487369695	11049130743	1.162554	4133
2048	128	1234519212	515624658	15493525855	9990540558	2.395434	3789
4096	16	17346405898	3830359866	122057861052	98298726236	4.553096	35555
4096	32	17350508913	3799142823	122060993240	99339917260	4.571132	37094
4096	64	17334383977	4552133933	122067114936	99553272971	3.814104	36024
4096	128	17339324083	4749205158	122079615428	101533861423	3.651496	37215

2 Analysing Matmul using papi

2.1 Analysing the system

Using "\$ lscpu" command we can see that cares2 system has:

- L1 cache size of 32 kilobytes. And
- L2 cache size of 256 kilobytes.

Let us assume that bulk of data is stored in L2 memory for sake of abstraction and simplicity.

Now consider matmul3 let us take dimension of matrix to be d and block size to be b. Using the optimal cache size discussed in class.

The relation $p = q = r = \sqrt{n/3}$.

We get the relation: $d = b = \sqrt{\left(\frac{256*1000}{24}\right)} = 110$ So, if d*b is closer to what we derived above we can expect maximum performance.

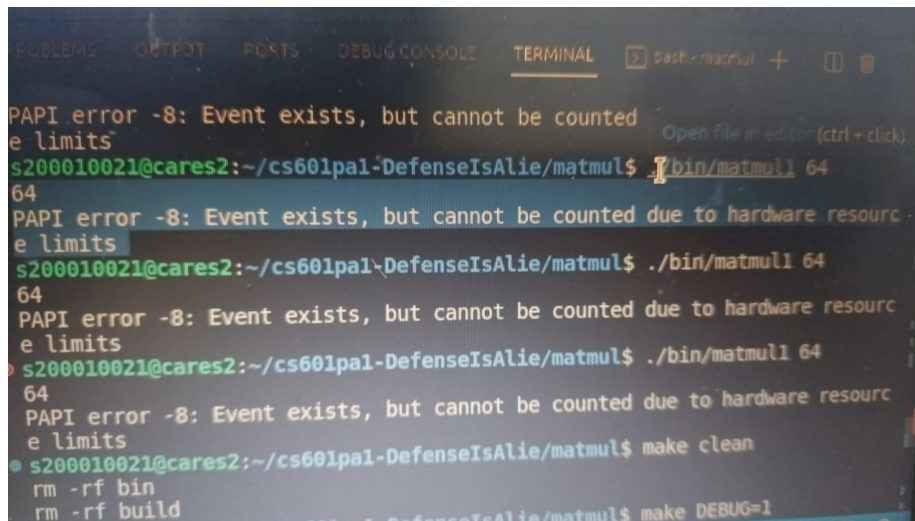
2.2 What we are trying to do:

In short what we are trying to do in terms of "PAPI" language is minimize the L2 cache misses(i.e. increase computational intensity). This helps remove the memory bottle neck and approach our goal of making matmul take less time.

2.3 L3 Cache Hit:

L3 cache hit counter is not supported by Hard Ware. If we add a papi event to count L3 cache. It will throw hardware error.

- Add PAPI event \rightarrow *PAPI_L3_TCM*
- Error \rightarrow *Hardware error.*



```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL [?] bash - matmul + [ ] [X]
PAPI error -8: Event exists, but cannot be counted
e limits
s200010021@cares2:~/cs601pal-DefenseIsAlie/matmul$ ./bin/matmul 64
64
PAPI error -8: Event exists, but cannot be counted due to hardware resourc
e limits
s200010021@cares2:~/cs601pal-DefenseIsAlie/matmul$ ./bin/matmul 64
64
PAPI error -8: Event exists, but cannot be counted due to hardware resourc
e limits
s200010021@cares2:~/cs601pal-DefenseIsAlie/matmul$ ./bin/matmul 64
64
PAPI error -8: Event exists, but cannot be counted due to hardware resourc
e limits
s200010021@cares2:~/cs601pal-DefenseIsAlie/matmul$ make clean
rm -rf bin
rm -rf build
s200010021@cares2:~/cs601pal-DefenseIsAlie/matmul$ make DEBUG=1
```

Figure 1: PAPI L3 cache error