Performance Evaluation - Single and Multi-Core Matrix Multiplication Algorithms

(1st Lab Work)

Parallel and Distributed Computing 2024/25 – LEIC

L.EIC028-12

Students

Beatriz Ferreira up202205612
Letícia Coelho up202108877
Victor Matos up202102358

21st march 2024

Index

1. Problem Description

In this project, we will analyse the performance differences between several matrix multiplication approaches. We will explore how building an algorithm with smarter cache usage affects execution time and how performance is increased by using parallel programming approaches through the use of the OpenMP C++ library.

2. Algorithms

Matrix multiplication performance is influenced by memory access patterns. To analyze this, we implemented and compared different approaches using **C++** and **Python**.

We started with a **basic row-by-column multiplication** in all two languages, followed by an **element-wise multiplication approach** in **C++** and **Python**. Finally, a **block-oriented optimization** was implemented in **C++**. Performance was measured across various matrix sizes, with larger tests conducted in C++ to evaluate scalability. The **Performance API (PAPI)** was used to gather execution metrics, and optimizations were applied where applicable.

The following sections describe each algorithm in detail.

2.1 Basic Matrix Multiplication

```
for ( i=0; i<m_ar; i++) {
      for ( j=0; j<m_br; j++) {
            temp = 0;
            for ( k=0; k<m_ar; k++) {
                  temp += pha[i*m_ar+k] * phb[k*m_br+j];
            }
            phc[i*m_ar+j] = temp;
      }
}
```

Image 1 - Basic Matrix Multiplication (Row-by-Column) in C++

This implementation follows the traditional row-by-column approach for matrix multiplication. Each element of the resulting matrix is computed by iterating through the rows of matrix **A** and multiplying them with the corresponding columns of matrix **B**.

2.2 Line Matrix Multiplication

```
for (i = 0; i < m_ar; i++) {
      for (k = 0; k < m_ar; k++) {
            temp = pha[i * m_ar + k];
            for (j = 0; j < m_br; j++) {
                  phc[i * m_ar + j] += temp * phb[k * m_br + j];
            }
      }
}
```

Image 2 - Line Matrix Multiplication in C++

This implementation optimizes the traditional row-by-column approach by reusing computed values from matrix **A** more efficiently, reducing redundant memory accesses. Instead of

computing each element of the result matrix individually by iterating over rows and columns, this approach processes **one row at a time**, multiplying it with all columns of matrix **B** before moving to the next row.

2.2.1 Parallel Nested Loops with OpenMP (*i-loop* parallelized)

```
#pragma omp parallel for
for (int i = 0; i < m_ar; i++) {
   for (int k = 0; k < m_ar; k++) {
      double temp = pha[i * m_ar + k];
      for (int j = 0; j < m_br; j++) {
         phc[i * m_ar + j] += temp * phb[k * m_br + j];
      }
   }
}
```

Image 3 - Parallel Nested Loops with OpenMP in C++

This implementation applies OpenMP parallelization to the **outermost loop (`i-loop`)**, distributing the iterations across multiple threads. Each thread processes a different row of matrix **A**, performing all multiplications for that row in parallel. The num_threads parameter was also used to study the difference of the usage of different thread counts.

2.2.2 Parallel Nested Loops with OpenMP (*j-loop* parallelized)

```
#pragma omp parallel
for (int i = 0; i < m_ar; i++) {
   for (int k = 0; k < m_ar; k++) {
      double temp = pha[i * m_ar + k];

      #pragma omp for
      for (int j = 0; j < m_br; j++) {
         phc[i * m_ar + j] += temp * phb[k * m_br + j];
      }
   }
}
```

Image 4 - Parallel Nested Loops with OpenMP in C++

This version applies **nested parallelization**, where the **outermost `i-loop`** is parallelized, and an additional `#pragma omp for` is applied to the **innermost `j-loop`**. This means that, within each thread, multiple threads are also distributing the work inside the row-wise multiplication process. Similarly to the algorithm above, num_threads was also used for similar effects.

2.3 Block Matrix Multiplication

```
for (int blockY = 0; blockY < blocksPerRow; ++blockY) {
  for (int blockX = 0; blockX < blocksPerRow; ++blockX) {
    for (int block = 0; block < blocksPerRow; ++block) {
      for (int i = 0; i < bkSize; ++i) {
        for (int n = 0; n < bkSize; ++n) {
          for (int j = 0; j < bkSize; ++j) {
            phc[blockIndex + (i*m_ar+j)] += pha[blockAindex + (i*m_ar+n)] * phb[blockBindex + (n*m_ar+j)];
```

Image 5 - Block Matrix Multiplication in C++

This implementation optimizes matrix multiplication by **dividing the matrices into smaller sub-blocks**. Instead of processing entire rows or columns at once, computations are performed on blocks, which helps improve **cache efficiency** and **memory locality**. By keeping blocks small enough to fit in **cache memory**, we reduce memory access latency and enhance computational performance.

3. Performance Metrics

To assess the performance of the implemented matrix multiplication algorithms, we analyzed key metrics such as **execution time, cache misses, and floating-point operations per second (FLOP/s)**. These metrics provide insights into the efficiency of different approaches and programming languages, helping to evaluate the impact of memory hierarchy and parallelization.

Performance Measurement Using PAPI

For the **C++ implementations**, we used the **Performance Application Programming Interface (PAPI)** to measure **hardware performance counters**, specifically:

- **L1 and L2 Data Cache Misses**: Indicates how well memory is being reused (**lower cache misses generally result in better performance** due to reduced memory access latency).
- **Execution Time (seconds)**: The total time taken to execute each algorithm.

3.1 Execution Time and Multi-Core Evaluation

The algorithms were implemented in **C++ and Python**, and execution time was measured for both. The **parallelized versions** of the algorithms using **OpenMP** were also evaluated to measure performance improvements due to **multi-core execution**.

To ensure accuracy, all measurements were conducted on the **same system**, running **Ubuntu 22.04**, with an **Intel i5 7200 processor (3.10 GHz)** featuring:

Cache	Total Size	Associativity
L1 Data Cache	64 KiB	8-Way
L1 Information Cache	64 KiB	8-Way
L2 Cache	512 KiB	4-Way
L3 Cache	3 MiB	12-Way

Each experiment was repeated **five times**, and the average execution time was reported to minimize variations due to system background processes.

3.1.1 FLOP Calculation and Performance Estimation

For performance evaluation, the **floating-point operations per second (FLOP/s)** metric was computed using:

$$FLOPS = \frac{2 \times (matrix\ size)^3}{execution\ time}$$

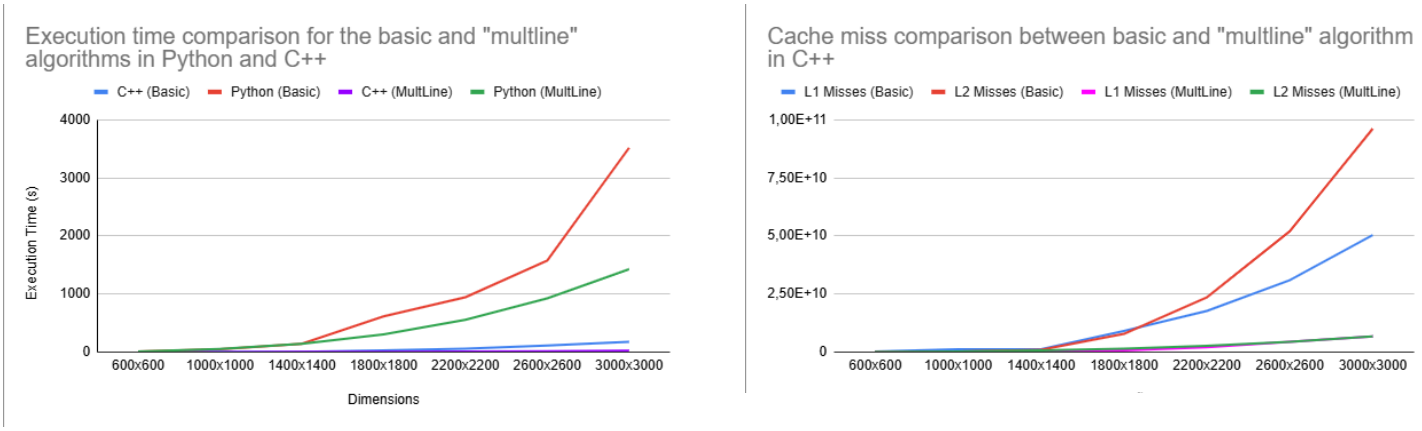
3.1.2 Speedup and Efficiency in Parallel Implementations

To quantify the performance gain from parallel execution, we computed:

$$Speedup = \frac{T_{sequential}}{T_{parallel}} \qquad Efficiency = \frac{Speedup}{\text{number of logical processors}}$$

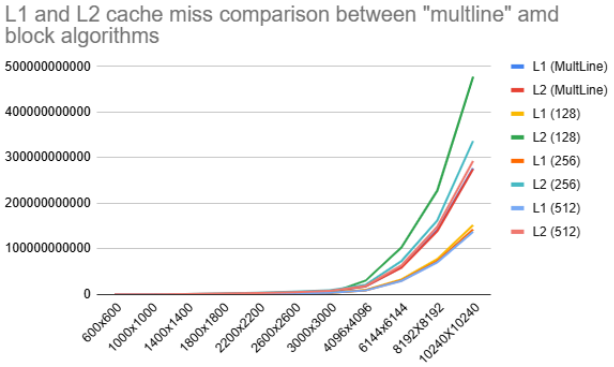
4. Results and Analysis

4.1 Basic vs Line Multiplication



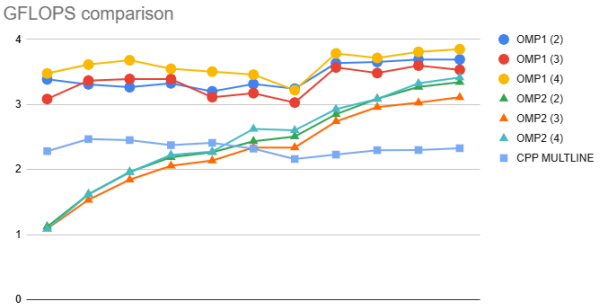
By looking at the first graph we can see that the execution time in Python is drastically higher than that of the one experienced when using C++. This was expected, as Python is an interpreted programming language while C++ is compiled: there is additional overhead due to code being “translated” at runtime. We can also tell that the line multiplication algorithm is significantly faster than the basic one originally provided by the teachers.
In relation to cache misses, expectedly the basic algorithm experiences more of them. It justifies the relative slowness in execution and shows the effects of poor cache management when building an algorithm.

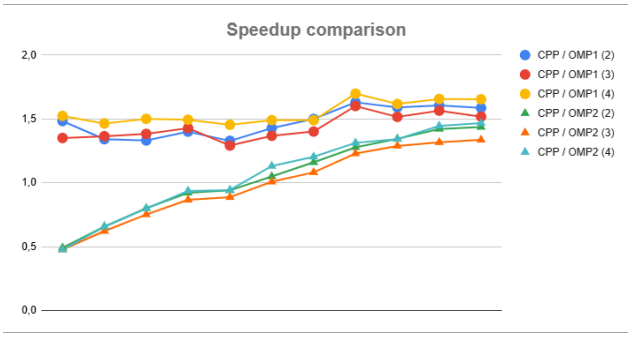
4.2 Line Multiplication vs Block Algorithm



When it comes to execution time, the conclusion was that both algorithms perform quite similarly when analysed in a line graph. The slight timing difference mainly lies in how these algorithms handle cache resources - as you can see, the block approach has slightly more L2 cache misses and slightly less L1 cache misses than the line multiplication approach, with the 512 block resulting in the lowest number of L1 cache misses overall.

4.3 Single vs Multi Core Line Multiplication





The computer used for testing only has two physical cores so a maximum of 4 threads was used. The number of FLOPS stayed stable for regular line multiplication and OMP1 and increased over time for OMP2 - this is explained by the need for a larger overhead of thread creation and synchronization, since we're parallelizing the innermost loop in this approach, which ends up being unsustainable especially in smaller matrices. Overall we can tell that the highest speedup was achieved by the 4 threaded OMP1 approach. We were also able to conclude that using 3 threads ends up being less efficient than using 4 or even 2 threads, since there is an workload imbalance between cores that calls for a significant context switching overhead for workload assignment.

5. Conclusions

In conclusion, this project allowed us to fully grasp the extent to which successful cache management and smartly built algorithms affect the performance of certain tasks. Furthermore we had the opportunity to try out some parallel programming approaches, which were foreign to us up until this point.
It will definitely serve as a reminder for the remainder of our academic and professional career that we should always take into consideration some of the lower level aspects of the programs we are running in order to optimize performance.

References

freeCodeCamp. **Interpreted vs Compiled Programming Languages: What's the Difference?.** 2020. Accessed on 20 March 2025. <https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/>

Annexes

A1. Basic Algorithm

A1.1. C++ - execution time (s)

Dimensions	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	0,439693	0,419459	0,391727	0,42769	0,407302	0,4171742
1000x1000	2,352758	2,207778	2,310205	2,198498	2,214327	2,2567132
1400x1400	5,890811	6,01915	5,878399	6,22058	5,739422	5,9496724
1800x1800	29,268526	28,574936	29,469361	27,501497	27,289729	28,4208098
2200x2200	59,098902	58,8099	59,10075	58,822947	58,517598	58,8700194
2600x2600	111,589645	109,523708	107,390359	112,599718	108,725018	109,9656896
3000x3000	186,188844	171,556685	171,954471	179,419925	168,992275	175,62244

A1.2. Python - execution time (s)

Dimensions	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	9,639553	9,604367	9,67931	9,973183	9,994624	9,7782074
1000x1000	47,732815	48,035698	47,761805	49,535921	48,063445	48,2259368
1400x1400	152,441989	141,919836	146,396799	138,014873	138,019214	143,3585422
1800x1800	635,329777	637,423009	587,58044	577,64713	624,95063	612,5861972
2200x2200	946,874424	967,744301	872,949756	951,401237	975,825248	942,9589932
2600x2600	1628,028695	1587,41334	1546,093184	1530,391618	1589,310286	1576,247425
3000x3000	3519,678407	3574,332479	3541,415252	3476,996507	3473,67521	3517,219571

A1.3. C++ - cache misses L1

Dimensions	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	244170603	244545924	244690993	243957461	244298561	244332708,4
1000x1000	1227149425	1228541869	1229378949	1228506174	1228208050	1228356893
1400x1400	1227149425	1228541869	1229378949	1228506174	1228208050	1228356893
1800x1800	9068659941	9086327414	9087193201	9087330712	9069318795	9079766013

2200x2200	17659290853	17660327097	17647178484	17636370466	17635257412	17647684862
2600x2600	30929545853	30882096670	30882829862	30884477028	30883622949	30892514472
3000x3000	50306908398	50321200773	50320309731	50309596231	50341456050	50319894237

A1.4. C++ - cache misses L2

Dimensions	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	38375124	39428196	38763910	38692234	37894911	38630875
1000x1000	233482975	266676576	242436027	239857155	229334703	242357487,2
1400x1400	670316965	1329592129	1155520751	1272211041	641124775	1013753132
1800x1800	5592660964	8483799979	8503403610	8018560927	8427590392	7805203174
2200x2200	23488625800	24027791827	23718687713	23239861384	23304198911	23555833127
2600x2600	52428066623	52160409699	51483154059	51758056597	51987566146	51963450625
3000x3000	95370773532	98281602694	95864069723	95853165032	95099173586	96093756913

A2. Line Multiplication (Single Core)

A2.1. C++ - execution time (s)

Dimensions	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	0,166654	0,197872	0,167535	0,214487	0,19953	0,1892156
1000x1000	0,807852	0,812936	0,805502	0,815078	0,811128	0,8104992
1400x1400	2,218192	2,23107	2,219219	2,226751	2,292973	2,237641
1800x1800	4,728846	4,725345	4,711708	5,159114	5,229171	4,9108368
2200x2200	8,929431	8,779584	8,79437	8,797028	8,869663	8,8340152
2600x2600	14,801709	15,750524	14,95215	15,376528	14,856572	15,1474966
3000x3000	22,996702	25,769195	25,620781	26,036189	24,454995	24,9755724
4096x4096	59,10882	61,951388	61,754143	62,294359	63,088056	61,6393532
6144x6144	201,872146	201,034928	201,035997	201,76821	203,788372	201,8999306
8192x8192	472,392815	482,11446	478,736591	478,431713	478,953047	478,1257252

10240x10240	920,239415	930,133287	922,528146	919,882397	919,78131	922,512911

A2.2. Python - execution time (s)

Dimensions	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	10,584092	10,650233	10,676091	10,502165	10,631115	10,6087392
1000x1000	52,336577	50,405668	50,983617	50,570455	50,601985	50,9796604
1400x1400	140,599531	141,502348	143,052117	141,578753	141,417281	141,630006
1800x1800	302,866577	302,840571	303,926355	303,256146	305,527085	303,6833468
2200x2200	555,43571	554,718959	555,463223	556,335064	559,351865	556,2609642
2600x2600	923,982903	921,010894	925,529727	925,723786	923,55403	923,960268
3000x3000	1417,615939	1416,803158	1418,985761	1429,85886	1446,47089	1425,946922

A2.3. C++ - cache misses L1

Dimensions	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	27152125	27154161	27158034	27154541	27154725	27154717,2
1000x1000	125938814	125921227	125929517	126061555	125940697	125958362
1400x1400	347422556	347425491	347210798	348058431	348969448	347817344,8
1800x1800	752769186	750336376	748647236	770724158	779314230	760358237,2
2200x2200	2093795361	208379845 7	208425018 6	208387622 6	208722652 6	2086589351
2600x2600	4413336677	4411203820	4413165045	4412358366	4413051858	4412623153
3000x3000	6780399279	6777442980	6777215625	6776807529	6778009074	6777974897
4096x4096	17712072300	17715384631	17684276513	17684977896	17698601296	17699062527
6144x6144	59794964679	59809167132	59817491303	59767826502	59787858126	59795461548
8192X8192	141726928358	141679985501	141119100769	141152776476	141115214768	141358801174
10240X10240	276629570401	276393085500	276644828881	276767536358	276745646400	276636133508

A2.4. C++ - cache misses L2

Dimensions	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	53631041	54441848	54529030	54739825	54566531	54381655
1000x1000	250624630	251865398	246026804	246581623	247044234	248428537,8
1400x1400	685642797	674535541	671529696	670000947	683349204	677011637
1800x1800	1474732612	1441819270	1469131525	1470405190	1444686339	1460154987
2200x2200	2687641360	2686607199	2696537044	2627910692	2617718259	2663282911
2600x2600	4374704052	4437473057	4374702515	4388727811	4502275520	4415576591
3000x3000	6737396997	6804616622	6790468182	6827335558	6701215835	6772206639
4096x4096	17582546234	17553782111	17558202402	17648275895	17373546872	17543270703
6144x6144	59236286867	58180901261	58204022218	59153483216	58595505263	58674039765
8192X8192	139353152647	140059648489	139567774690	137526789401	138342347536	138969942553
10240X10240	281551931080	273606965054	276029494217	271409619268	272823963343	275084394592

A3. Block Multiplication

A3.1. Execution time (s)

Dimensions	Block SIze	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	128	0,169692	0,165527	0,169001	0,167367	0,167091	0,1677356
	256	0,200018	0,195913	0,210652	0,209468	0,235808	0,2103718
	512	0,177651	0,191799	0,195314	0,267	0,177882	0,2019292
1000x1000	128	0,743163	0,817756	0,831334	0,800828	0,827871	0,8041904
	256	0,736131	0,736801	0,74509	0,734772	0,73613	0,7377848
	512	0,849685	0,83734	0,845516	0,817946	0,8363	0,837359

			8				
1400x1400	128	2,04857	2,052335	2,049205	2,036978	2,071667	2,051751
	256	2,041896	2,036405	2,028108	2,039485	2,249262	2,0790312
	512	2,20949	2,198519	2,207738	2,201811	2,216112	2,206734
1800x1800	128	4,28972	4,832219	4,291887	4,717409	4,293226	4,4848922
	256	4,288478	4,314774	4,910938	4,821712	4,295664	4,5263132
	512	5,128629	4,757467	5,464643	5,060843	5,462048	5,174726
2200x2200	128	8,187262	8,057813	7,870871	7,947953	7,931159	7,9990116
	256	7,819376	8,614781	8,371213	8,439576	8,517552	8,3524996
	512	8,827248	9,696861	9,662971	8,709997	9,466025	9,2726204
2600x2600	128	12,931972	13,595905	13,92568	13,83276	13,787916	13,6148466
	256	12,875163	13,815314	13,948038	14,028577	12,915217	13,5164618
	512	14,604651	15,654414	14,984728	15,788725	15,184353	15,2433742
3000x3000	128	19,83103	19,819097	20,679933	19,880797	20,42467	20,1271054
	256	19,735704	19,741985	20,636479	19,732203	19,872044	19,943683
	512	22,505544	23,307896	23,268751	23,436277	22,147441	22,9331818
4096x4096	128	86,216835	83,943095	79,57701	80,398629	80,792582	82,1856302
	256	70,223737	69,80046	69,827641	69,93322	70,305864	70,0181844
	512	62,128087	61,013158	61,192701	61,521632	61,063067	61,383729
6144x6144	128	210,866738	220,762766	214,215706	220,183199	218,410988	216,8878794
	256	227,231946	227,546524	226,577129	227,033124	227,252658	227,1282762
	512	202,74466	204,958	204,9668	205,0070	204,954	204,526163

		1	278	23	04	049	
8192x8192	128	810,612121	790,822141	803,088016	802,357899	803,597817	802,0955988
	256	594,163771	594,452017	615,07659	616,221892	620,752189	608,1332918
	512	518,898974	505,487237	513.088196	514,726568	511,543329	512,664027
10240x10240	128	1014,218588	1027,942598	1053,823548	1042,924869	1046,165015	1037,014924
	256	1036,154706	1030,992011	1038,562635	1038,079011	1048,243225	1038,406318
	512	952,945122	950,609856	944,101049	935,243307	943,157265	945,2113198

A3.2. Cache misses

Dimensions	Block Size	Cache	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	128	L1	31444808	31401452	31403791	28536787	31407010	30838769,6
	128	L2	28680424	24936550	31225214	28536787	27283187	28132432,4
	256	L1	29769776	29775000	29770527	29774005	29775205	29772902,6
	256	L2	68813422	67518394	65288460	66374134	66198475	66838577
	512	L1	28867378	28869502	28867926	29094839	28866893	28913307,6
	512	L2	61407484	61657570	61720179	63598132	62676612	62211995,4
1000x1000	128	L1	143505157	143521431	143521038	143528247	143516208	143518416,2
	128	L2	128072059	133781881	138882735	149511840	139913409	138032384,8
	256	L1	135130227	135087338	135099852	135091788	135092353	135100311,6
	256	L2	349669828	348734601	348475028	338943828	338899328	344944522,6
	512	L1	130223901	130324774	130345297	130330506	130348169	130314529,4

	512	L2	288900151	287159804	293054043	287472680	286199963	288557328,2
1400x1400	128	L1	392331618	391975441	391982476	391898979	392033401	392044383
	128	L2	391840245	383950671	397629470	350353799	385208045	381796446
	256	L1	372146021	372118474	372104313	372138257	37244857	372191128,4
	256	L2	906640654	935338352	933954827	906232478	940069655	924447193,2
	512	L1	358802894	358768340	358781094	358778883	358795565	358785355,2
	512	L2	809215238	793306969	793968734	809133728	809670243	803058982,4
1800x1800	128	L1	826848910	826855010	826255460	827564671	826342260	826773262,2
	128	L2	162892946	475612693	176258794	281745221	262190633	271740057,4
	256	L1	787474719	787487132	788218791	788186625	787361488	787745751
	256	L2	2015066545	1948514619	2015448471	1992644160	1947127349	1983760229
	512	L1	764449793	764114192	765285838	764381420	765254298	764697108,2
	512	L2	1741196334	1733187895	1731282252	1738462698	1735852139	1735996264
2200x2200	128	L1	1515233790	1514818320	1514574572	1514547717	1514750164	1514784913
	128	L2	1017319652	961356285	863366504	825938345	888162339	911228625
	256	L1	1445113232	1446177597	1445801933	1445957120	1446113309	1445832638
	256	L2	3706792349	3636756644	3620036920	3634688811	3601307109	3639916367
	512	L1	1399878668	1401459657	1401396476	1399743382	1400892269	1400674090
	512	L2	3198717873	3156288535	3168276998	3141836321	3189348212	3170893588
2600x2600	128	L1	2544591349	2545238814	2546046177	2546020251	2544583818	2545296082
	128	L2	1545525348	1662937357	2009978035	2116460446	2075342897	1882048817

	256	L1	2388207092	2389428301	2389733472	2389484516	2388313823	2389033441
	256	L2	5765525896	5948081676	5917271457	5865677974	5921656044	5883642609
	512	L1	2311274983	2313078885	2311848160	2313533979	2312154810	2312378163
	512	L2	5059467339	5115871723	5140701183	5088867062	5133359604	5107653382
3000x3000	128	L1	3873571192	3873476962	3875726262	3873515220	3874356160	3874129159
	128	L2	2740434242	2790803576	2874522104	2656938223	2872989333	2787137496
	256	L1	3647906882	3647942066	3649419830	3647857161	3648330570	3648291302
	256	L2	9424787125	9430000701	9127361124	9379230429	9357237376	9343723351
	512	L1	3528785195	3530488617	3530958595	3530569818	3528237257	3529807896
	512	L2	7781647836	7819509621	7802203291	7772971501	7776560576	7790578565
4096x4096	128	L1	9720265732	9805629743	9828947101	9824304460	9823611613	9800551730
	128	L2	30550639850	30396105260	30311366584	30746173163	30715040493	30543865070
	256	L1	9117358007	9115646816	9116573636	9117286207	9117456975	9116864328
	256	L2	21449495375	20734433131	21200499585	21326657980	21041272067	21150471628
	512	L1	8795040074	8784904715	8784143871	8782520134	8779541156	8785229990
	512	L2	18616164363	18536235666	18948935013	18611106022	18968174943	18736123201
6144x6144	128	L1	32831082955	32813951502	32805006265	32811264674	32820621964	32816385472
	128	L2	102323035163	103540169110	103581047041	103151420296	103129621290	103145058580
	256	L1	30786020324	30785957678	30789904906	30788066187	30786760332	30787341885
	256	L2	73422762323	73443550173	73356438168	73156410755	73157843185	73307400921
	512	L1	29641605157	29697862231	29694482728	29696753573	29698412479	29685823234

	512	L2	64190256377	64094463800	63241338565	63356600374	63809781971	63738488217
8192x8192	128	L1	77430805798	77977842306	77636704872	77652642798	77620541153	77663707385
	128	L2	229020858001	225219601937	228412932908	228305304408	228185703496	227828880150
	256	L1	731118739 45	731083931 48	730302847 47	730193782 29	729961209 52	73053210204
	256	L2	161130337931	162089546167	160885116548	163916715151	164097720711	162423887302
	512	L1	707682462 81	705285132 12	706552791 78	706894651 89	706285659 84	70654013969
	512	L2	148995935900	147506902514	149409140046	149754341774	148058779033	148745019853
10240x10240	128	L1	152030265097	152289066162	151889812347	152129883663	151998348810	152067475216
	128	L2	476605171466	476679753965	480811351217	476284361463	478800435779	477836214778
	256	L1	142301639236	142354637247	142490791914	142434195336	142514516633	142419156073
	256	L2	332724139022	335295190652	338163831088	337190505131	338372009137	336349135006
	512	L1	137336485620	137311900989	137129936497	137046745624	137080290444	137181071835
	512	L2	291589769375	293810621151	293544779850	291469438725	292833349182	292649591657

A4. Multi-Core Line Multiplication

A4.1. OMP1

A4.1.1. Execution time (s)

Dimensions	Threads	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	2	0,133173	0,126825	0,127318	0,12552	0,125019	0,127571
	3	0,14065	0,14016	0,138799	0,136753	0,143946	0,1400616
	4	0,121696	0,120065	0,120519	0,121076	0,137687	0,1242086
1000x1000	2	0,605229	0,60219	0,607021	0,610244	0,59670	0,6042788

			9			1	
	3	0,59231	0,594017	0,590376	0,597523	0,595357	0,5939166
	4	0,548989	0,565182	0,545893	0,555952	0,550534	0,55331
1400x1400	2	1,72948	1,641767	1,716196	1,643082	1,672229	1,6805508
	3	1,593967	1,603807	1,637393	1,614101	1,640456	1,6179448
	4	1,493687	1,492441	1,488367	1,493782	1,487045	1,4910644
1800x1800	2	3,590526	3,489502	3,494568	3,483293	3,478633	3,5073044
	3	3,413368	3,40402	3,527402	3,398753	3,455244	3,4397574
	4	3,245397	3,236904	3,28394	3,30898	3,35934	3,2869122
2200x2200	2	6,789767	6,71304	6,581465	6,588806	6,576794	6,6499744
	3	7,4101	6,748105	6,684831	6,469265	6,896399	6,84174
	4	6,029528	6,145899	6,066905	6,062828	6,071029	6,0752378
2600x2600	2	10,496629	10,49743	10,589845	10,685093	10,752799	10,6043592
	3	11,130943	10,450068	10,598962	12,266784	10,922482	11,0738478
	4	10,108168	10,129539	10,133121	10,167169	10,271554	10,1619102
3000x3000	2	16,109718	16,143745	16,732097	16,950232	17,318446	16,6508476
	3	17,059422	17,784951	17,778587	17,738587	18,765854	17,8254802
	4	16,46765	16,636279	16,857979	16,866889	17,004635	16,7666864
4096x4096	2	38,982241	36,96382	37,182291	38,717788	37,208012	37,8108304
	3	39,117728	38,640117	37,045604	38,613794	39,122489	38,5079464
	4	36,172141	36,248772	36,394238	36,392645	36,329071	36,3073734
6144x6144	2	126,87527	126,665	127,1059	127,0127	127,115	126,9551096

		9	833	58	28	75	
	3	133,887349	133,481616	132,745752	131,375752	134,310556	133,160205
	4	124,703687	125,201713	124,735206	124,849723	124,72487	124,8430398
8192x8192	2	298,841363	296,884687	297,55629	297,208629	298,74309	297,8468118
	3	311,094691	300,887257	303,097357	299,113801	313,728929	305,584407
	4	286,733685	288,988119	289,757055	289,607059	288,340651	288,6853138
10240x10240	2	582,119663	581,843125	579,897847	584,277291	579,793055	581,5861962
	3	614,496456	603,560405	605,630641	607,190975	608,10679	607,7970534
	4	559,749207	558,392613	548,760862	561,861382	560,172214	557,7872556

A4.1.2. Cache misses

Dimensions	Threads	Cache	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	2	L1	13612719	13594262	13598160	13589810	13597711	13598532,4
	2	L2	27767061	27608734	27575887	27598695	27655033	27641082
	3	L1	6692937	9223009	6354216	9212395	7583135	7813138,4
	3	L2	13756862	18855231	12619679	18721672	15585448	15907778,4
	4	L1	5582989	4890612	5600231	5247375	5840198	5432281
	4	L2	11419465	9946891	11277501	10691339	11644050	10995849,2
1000x1000	2	L1	63068327	62992640	62214554	63147670	62977378	62880113,8
	2	L2	126856280	126537172	125390557	126502537	126538573	126365023,8
	3	L1	38632096	36103898	30338111	45254113	32404723	36546588,2
	3	L2	78339660	73058000	62971189	90254583	66805941	74285874,6

	4	L1	23232474	25244993	24374329	27256354	22781820	24577994
	4	L2	47333124	51479693	49672622	55653149	46572839	50142285,4
1400x1400	2	L1	178381731	173524213	176207422	173627759	173890081	175126241,2
	2	L2	353663152	350738190	356165074	346912155	341610199	349817754
	3	L1	115953689	130617825	131821208	130017582	118515650	125385190,8
	3	L2	230861827	183937854	180322270	182601368	232712971	202087258
	4	L1	114054805	152875072	149199273	143012906	148238998	141476210,8
	4	L2	128826633	167428102	162890547	157075371	161137473	155471625,2
1800x1800	2	L1	388486811	374533783	375999526	374896850	375740519	377931497,8
	2	L2	739731495	733705130	735219566	741541306	744721158	738983731
	3	L1	252043912	252327069	266028098	255121035	364119150	277927852,8
	3	L2	491334790	490827595	500400555	506939610	450475736	487995657,2
	4	L1	333417137	338646586	351762844	359927064	346113890	345973504,2
	4	L2	325405277	335656823	368233082	383182607	359091909	354313939,6
2200x2200	2	L1	1067934826	1064787815	1059835402	1058964979	1058982115	1062101027
	2	L2	1367662891	1343901046	1336381061	1371056594	1374862507	1358772820
	3	L1	846628481	774020057	819160204	791844719	763314151	798993522,4
	3	L2	940022971	843740189	916034132	892039214	918103193	901987939,8
	4	L1	631126469	654761529	643178927	659853466	650874415	647958961,2
	4	L2	629613384	690325676	654932319	688328213	683030798	669246078
2600x2600	2	L1	2209645563	2209721434	2209737100	2210022104	2209538386	2209732917

	2	L2	2205754212	2209140473	2207809742	2206747966	2219859031	2209862285
	3	L1	1433146242	1302598335	1347636323	1446121791	1426414971	1391183532
	3	L2	1404402501	1179532501	1277944137	1496671553	1457323014	1363174741
	4	L1	1101344958	1089048820	1100225621	1088465611	1069338434	1089684689
	4	L2	1194149457	1165329946	1191836541	1164961675	1104601368	1164175797
3000x3000	2	L1	3393754204	3393671982	3393452051	3393751167	3393534011	3393632683
	2	L2	3494474423	3503477654	3526852298	3443303479	3451127133	3483846997
	3	L1	2262559887	2187433833	2255688832	2265179036	2231207065	2240413731
	3	L2	2307461230	2217399728	2330782990	2406589341	2325362243	2317519106
	4	L1	1666412031	1688481159	1672833316	1694529375	1677916906	1680034557
	4	L2	1753155823	1806365926	1789231279	1839116356	1798096646	1797193206
4096x4096	2	L1	8752719079	8768154559	8769713853	8784425038	8776317309	8770265968
	2	L2	8065787720	7985128669	8097624550	8003566977	8064597294	8043341042
	3	L1	5864456194	5837607850	5610102481	5844045886	5830224713	5797287425
	3	L2	6003276443	6001615191	5313733334	5846532702	5842094263	5801450387
	4	L1	4369807291	4341843783	4371657950	4378940633	4386892722	4369828476
	4	L2	4580303128	4449281399	4551213656	4560205037	4608986183	4549997881
6144x6144	2	L1	29566295259	29589991186	29644273581	29599184737	29598602222	29599669397
	2	L2	27226769956	27499766011	27979598065	27911025756	28502486186	27823929195
	3	L1	19702459915	19763197087	19767663643	19785591067	19817679810	19767318304
	3	L2	19543297901	20379357359	20277496512	20757859700	21171178412	20425837977

	4	L1	148397 00420	148625 15590	148612 52477	148405 20523	148289 43116	14846586425
	4	L2	164004 77384	163379 46733	164888 93332	164637 78859	165177 35010	16441766264
8192x8192	2	L1	693853 32677	694061 70015	694085 25949	696049 55139	694026 87539	69441534264
	2	L2	659029 50129	659067 24229	665947 81605	678253 42070	668631 40114	66618587629
	3	L1	461436 29676	462262 51832	451232 19405	444326 67401	461078 18181	45606717299
	3	L2	503219 74190	443548 82478	489989 00874	467026 22920	514170 07595	48359077611
	4	L1	345409 53537	343795 53323	345442 54262	345176 94885	344409 23409	34484675883
	4	L2	410662 94437	414374 45822	406718 03728	406965 62233	415482 03384	41084061921
10240x10240	2	L1	137350 994016	137353 318168	137215 543092	137258 500420	137014 169688	137238505077
	2	L2	132491 162753	130815 410155	131472 450010	133469 808979	130856 562904	131821078960
	3	L1	908812 58986	909924 16828	908468 11640	913580 10505	911787 69296	91051453451
	3	L2	126664 310820	131359 770690	131827 182425	108984 862723	125874 989655	124942223263
	4	L1	684043 42201	680642 68458	681398 55366	685327 65003	683269 36632	68293633532
	4	L2	117200 084760	115575 126225	115778 546318	117756 129817	115765 433829	116415064190

A4.1.3. GFLOPS

Dimensions	OMP1 (2)	OMP1 (3)	OMP1 (4)
600	3,386350	3,084357	3,478020
1000	3,309731	3,367476	3,614610
1400	3,265596	3,391958	3,680592
1800	3,325631	3,390937	3,548619
2200	3,202418	3,112658	3,505377
2600	3,314863	3,174326	3,459192
3000	3,243078	3,029371	3,220672
4096	3,634910	3,569106	3,785428
6144	3,653705	3,483447	3,715517

8192	3,691534	3,598062	3,808686
10240	3,692460	3,533225	3,850005

A4.1.4. Speedup

Speedup (CPP / OMP 1)		
1,483218	1,350946	1,523370
1,341267	1,364668	1,464819
1,331493	1,383014	1,500700
1,400174	1,427670	1,494058
1,328428	1,291194	1,454102
1,428422	1,367862	1,490615
1,499958	1,401116	1,489595
1,630204	1,600692	1,697709
1,590325	1,516218	1,617230
1,605274	1,564627	1,656218
1,586202	1,517798	1,653880

A4.1.5. Efficiency

Efficiency OMP 1	Logical Processors		
Dimensions	2	3	4
600	0,741609	0,450315	0,380842
1000	0,670633	0,454889	0,366205
1400	0,665746	0,461005	0,375175
1800	0,700087	0,475890	0,373514
2200	0,664214	0,430398	0,363525
2600	0,714211	0,455954	0,372654
3000	0,749979	0,467039	0,372399
4096	0,815102	0,533564	0,424427
6144	0,795163	0,505406	0,404308
8192	0,802637	0,521542	0,414054
10240	0,793101	0,505933	0,413470

A4.2. OMP2

A4.2.1. Execution time (s)

Dimensions	Threads	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	2	0,38163	0,392733	0,379815	0,379744	0,390338	0,384852
	3	0,397561	0,394668	0,400662	0,400201	0,392408	0,3971
	4	0,397735	0,399575	0,39723	0,397645	0,394297	0,3972964
1000x1000	2	1,248541	1,228382	1,233716	1,22875	1,225606	1,232999
	3	1,300675	1,31213	1,302337	1,308675	1,297564	1,3042762
	4	1,218565	1,220448	1,227221	1,259262	1,253017	1,2357026
1400x1400	2	2,792621	2,804032	2,792309	2,791545	2,793258	2,794753
	3	2,934621	3,051819	2,927747	2,973609	2,998605	2,9772802
	4	2,755612	2,773105	2,803406	2,856783	2,830516	2,8038844
1800x1800	2	5,312119	5,333321	5,321964	5,352766	5,32603	5,32924
	3	5,666931	5,745775	5,67773	5,727682	5,527767	5,669177
	4	5,273973	5,228035	5,198614	5,212823	5,306235	5,243936
2200x2200	2	10,030336	9,174896	9,179749	9,19493	9,408216	9,3976254
	3	11,010673	9,775009	9,712424	9,613366	9,708169	9,9639282
	4	9,298742	9,303683	9,205017	9,461139	9,556161	9,3649484
2600x2600	2	14,488955	14,308279	14,58885	14,35973	14,425365	14,4342358
	3	14,71801	14,920742	14,915261	15,533724	15,053726	15,0282926
	4	13,529372	13,560038	13,591233	13,149842	13,135386	13,3931742
3000x3000	2	22,581963	21,175399	21,35833	21,272864	21,266345	21,5309802

	3	23,267658	22,833769	22,930497	23,673099	22,753468	23,0916982
	4	20,035861	20,585245	20,94112	21,007768	21,170657	20,7481302
4096x4096	2	47,491802	47,321194	50,962338	47,709228	47,666066	48,2301256
	3	49,563646	50,576377	49,803411	50,081797	50,695248	50,1440958
	4	46,703251	47,654319	46,879156	46,74238	46,748251	46,9454714
6144x6144	2	149,702326	150,156668	150,273697	150,388595	150,552286	150,2147144
	3	157,139126	157,093076	156,945781	155,628854	156,713785	156,7041244
	4	150,539076	150,728752	149,802398	150,969813	150,219092	150,4518262
8192x8192	2	336,053894	335,618405	336,593993	337,412598	336,230105	336,381799
	3	363,256032	362,968359	361,744941	364,851638	362,889184	363,1420308
	4	331,77254	331,339003	329,286161	332,159449	329,241117	330,759654
10240x10240	2	641,751855	641,748247	642,251178	642,817221	642,834293	642,2805588
	3	689,663426	694,615393	690,261908	688,60193	689,272338	690,482999
	4	625,508442	625,992264	629,18624	631,184514	632,690873	628,9124666

A4.2.2. Cache misses

Dimensions	Threads	Cache	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
600x600	2	L1	17934777	17990196	17900486	17943142	17954300	17944580,2
	2	L2	46667725	46963873	46556308	46801777	46772998	46752536,2
	3	L1	13354228	10612887	13275614	12527945	10469946	12048124
	3	L2	40717795	26220832	40446769	27240282	24366039	31798343,4
	4	L1	7988640	7979790	8000542	7992331	8026984	7997657,4

	4	L2	18417481	18253828	18473855	18479500	18986992	18522331,2
1000x1000	2	L1	75620721	75615765	75675222	75772758	75765014	75689896
	2	L2	175488089	175910957	176082169	176441778	176479449	176080488,4
	3	L1	45934412	51282643	46697259	47167571	46127937	47441964,4
	3	L2	95126360	117834253	99255200	102680147	98794273	102738046,6
	4	L1	34867026	34904169	34838203	34804528	34896031	34861991,4
	4	L2	64782999	65307426	64718851	64542914	65477510	64965940
1400x1400	2	L1	197165579	197089662	197049613	197073725	197042426	197084201
	2	L2	437889701	437176310	437131334	429076416	428636912	433982134,6
	3	L1	133430609	119705311	133318412	140324243	124707916	130297298,2
	3	L2	291572877	238725054	290314618	323902777	259952659	280893597
	4	L1	92320346	92235093	10838000	10831653	10840884	101932035,2
	4	L2	164944815	165744047	265453110	265713767	265037305	225378608,8
1800x1800	2	L1	410290155	410459461	410256888	410647841	410083332	410347535,4
	2	L2	871124028	868904082	864935621	869710770	867057534	868346407
	3	L1	287315969	259377686	287363443	258850158	282109812	275003413,6
	3	L2	628088738	472472239	634826124	460355101	615472539	562242948,2
	4	L1	223387623	223403313	223243367	223303513	223192873	223306137,8
	4	L2	541247805	540991914	540831391	541874961	541876751	541364564,4
2200x2200	2	L1	762956383	738031047	737129410	738077614	740513518	743341594,4
	2	L2	1535857146	1520649298	1525994357	1535439984	1539730307	1531534218

	3	L1	505941971	517489590	496706987	537252075	496265777	510731280
	3	L2	981508963	1051066833	892976296	103906585	876441020	968211739,4
	4	L1	384869944	369468030	369513621	385022148	375378813	376850511,2
	4	L2	907303114	786623826	784762936	901754919	837752940	843639547
2600x2600	2	L1	1211421660	1203120500	1208861290	1204814707	1208580693	1207359770
	2	L2	2427999705	2433526698	2430402276	2428151569	2423068742	2428629798
	3	L1	948954466	915981370	940603080	863186446	827918654	899328803,2
	3	L2	1747087599	1563264682	1695303061	1674656223	1743861565	1684834626
	4	L1	644737327	644435304	645480522	634107761	632855253	640323233,4
	4	L2	1504724964	1505435764	1506357732	1348038392	1345181651	1441947701
3000x3000	2	L1	1930120184	1859812952	1858569484	1848272391	1846852159	1868725434
	2	L2	3705189645	3705673249	3656428785	3639458406	3640081243	3669366266
	3	L1	1736530697	1254527857	1253922995	1708391329	1761754727	1543025521
	3	L2	2462842165	2621504586	2703924176	2315774839	2631788106	2547166774
	4	L1	905846775	957632429	1046399505	1047636463	1047008788	1000904792
	4	L2	1668152493	2029079223	2252678161	2225345878	2226998237	2080450798
4096x4096	2	L1	5204574060	5136114436	5455944146	5151830367	5145957473	5218884096
	2	L2	9181310336	8907427120	9181747211	8975390300	8973244651	9043823924
	3	L1	5988239676	3251776458	6046026871	5767509820	3965259880	5003762541
	3	L2	6064302670	6334522496	5801353321	5903714678	6295252574	6079829148
	4	L1	2811095098	2711734365	2735351785	2772040053	2821107156	2770265691

	4	L2	5003717207	5213006300	5125317866	4970852607	4930451387	5048669073
6144x6144	2	L1	30197331804	30192350349	30198544785	30198089847	30201676794	30197598716
	2	L2	29152037769	28899008807	29436596548	29148687533	29353581122	29197982356
	3	L1	18044223484	19066021533	19054507387	17160402653	17924054987	18249842009
	3	L2	20651737983	20596117234	20578774820	20500655136	20718678637	20609192762
	4	L1	15385243634	15394215465	15406704798	15309434747	15315723686	15362264466
	4	L2	16465581201	16499746578	16505463109	15756135478	15813654994	16208116272
8192x8192	2	L1	70778434314	70729280637	70783479056	70878024480	71156449771	70865133652
	2	L2	70415755889	70808735314	70135212063	69534770754	69127113918	70004317588
	3	L1	47471001831	47440562169	47402358968	47461414059	47387517226	47432570851
	3	L2	50614530874	50142160867	48906960082	49148547212	48239804742	49410400755
	4	L1	35909914770	35763074521	35762685589	35766881422	35988972708	35838305802
	4	L2	38339847757	36743041951	37708375167	36863581233	36979930693	37326955360
10240x10240	2	L1	139260755119	139043162331	139288618587	139303043929	139025062021	139184128397
	2	L2	131678043326	134153828495	130474122403	133045539070	132506920917	132371690842
	3	L1	93640583500	93484566928	93752079215	93660032919	93650833122	93637619137
	3	L2	95621427104	96673664445	95562395142	97035062093	93923750828	95763259922
	4	L1	70479437459	70449068230	70227340798	70247376418	70365247048	70353693991
	4	L2	74823954937	74115311990	74222311104	73798808896	72204918511	73833061088

A4.2.3. GFLOPS

Dimensions	OMP2 (2)	OMP2 (3)	OMP2 (4)

600	1,122509	1,087887	1,087349
1000	1,622061	1,533418	1,618512
1400	1,963680	1,843293	1,957285
1800	2,188680	2,057441	2,224283
2200	2,266104	2,137310	2,274011
2600	2,435321	2,339055	2,624621
3000	2,508014	2,338503	2,602644
4096	2,849650	2,740880	2,927630
6144	3,087956	2,960078	3,083090
8192	3,268642	3,027773	3,324201
10240	3,343529	3,110118	3,414599

A4.2.4. Speedup

Speedup (CPP / OMP 2)		
0,491658	0,476494	0,476258
0,657340	0,621417	0,655902
0,800658	0,751572	0,798050
0,921489	0,866235	0,936479
0,940026	0,886600	0,943306
1,049415	1,007932	1,130986
1,159983	1,081582	1,203751
1,278026	1,229244	1,312999
1,344076	1,288415	1,341957
1,421378	1,316636	1,445538
1,436308	1,336040	1,466838

A4.2.5. Efficiency

Efficiency OMP 2	Logical Processors		
Dimensions	2	3	4
600	0,245829	0,158831	0,119065
1000	0,328670	0,207139	0,163975
1400	0,400329	0,250524	0,199513
1800	0,460745	0,288745	0,234120
2200	0,470013	0,295533	0,235827
2600	0,524707	0,335977	0,282747
3000	0,579992	0,360527	0,300938

4096	0,639013	0,409748	0,328250
6144	0,672038	0,429472	0,335489
8192	0,710689	0,438879	0,361385
10240	0,718154	0,445347	0,366710