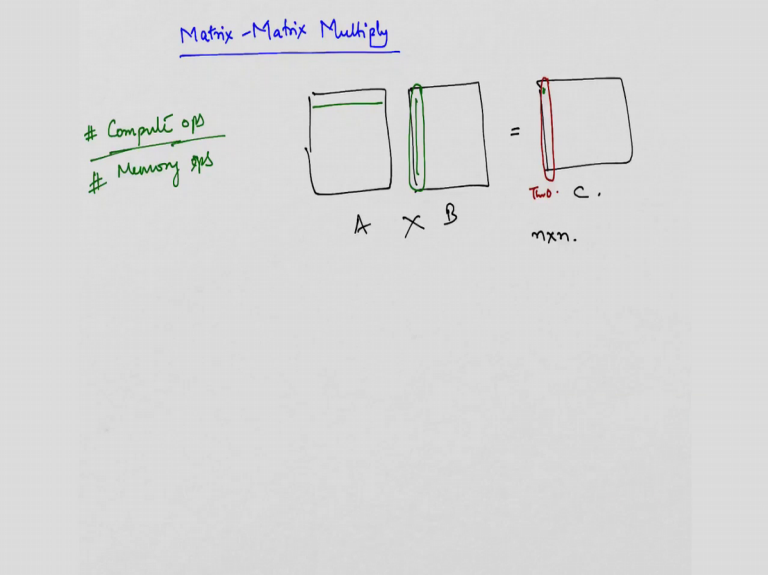
Introduction to Parallel Programming in OpenMP
Dr. Yogish Sabharwal
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 26
Matrix-Matrix operations (Matrix-Matrix Multiply)

Now let us go to Matrix-Multiply right.

(Refer Slide Time: 00:04)



So, what do I want to do? I have two matrices A B, I want to compute A times B and store it in C. For simplicity right now I am going to assume that they are all n cross n how should I distribute this.

Student: Each row of the final matrix (Refer Time: 00:38).

Each row of the final matrix is given to.

Student: A particular thread.

A particular thread let us see how that works. So, I give this to thread 0 first, let us talk about this approach then we will go to that one what does thread 0 need to fill up these entries? The entire a and the first column of b right that is a matrix vector product.

Student: (Refer Time: 01:06).

A times the first column of B will be the first column of C right. So, essentially what I am doing is the matrix vector product right, I take this entire matrix, I multiply it with this column and I generate the column of c.

So, what are the issues here? One issue is C is not being accessed contiguously that is why because I only need to update c at the end, I do not need to keep on updating it right is this point clear. So, I multiply this whole row with this column then I generate one element of C. So, there are you know the order of n multiply add being done before I do one update of C. So, c not being contiguous it is an issue, but not such a big issue, but b not being contiguous is an issue right because I have to access this b again and again and again right and it is not contiguous. So, I am not getting the benefit of data locality right.

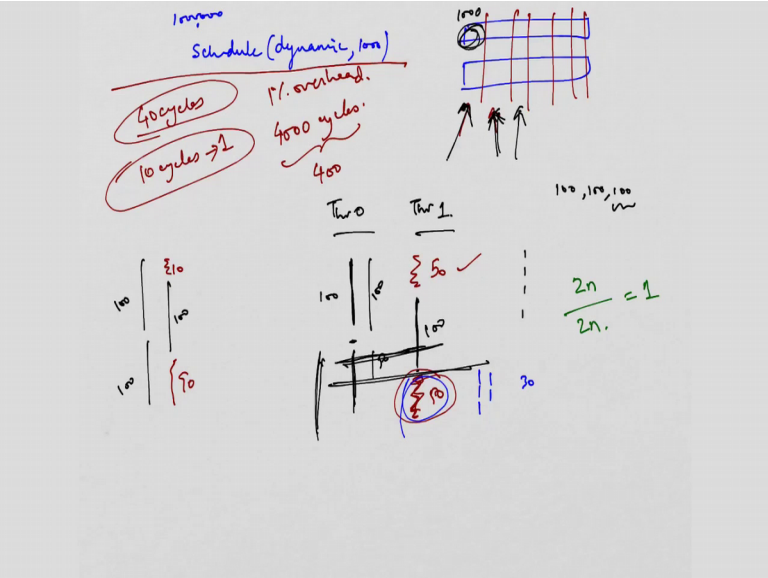
Student: We can make b (Refer Time: 02:13) column major (Refer Time: 02:14).

That is one approach that take the transpose of b and then do row multiplications right yeah that is one option. You have the transpose overheads, how do you write an efficient matrix transpose that is another interesting problem, but we would not go there right now what else here is another issue right. So, this is important. So, typically we have to see that what is the compute to memory ratio of the operations we are doing. How many compute operations am I doing and how many memory operations am I doing.

So, if I am doing more memory operations. So, typically what happens in processes is that you have much more computational capability then you have memory bandwidth right. So, you may only be able to fetch one data element every 5 nanoseconds or something, but you can do compute computations that in every nanosecond something of that show it right.

So, you want to have more compute you want to be doing more computations then memory accesses. There is let us just go back to the other operations that we did first and then we will come back to matrix multiply. So, what is the first one we did we did a dot product.

(Refer Slide Time: 03:35)



What is the number of compute operations I do in dot product 2 n.

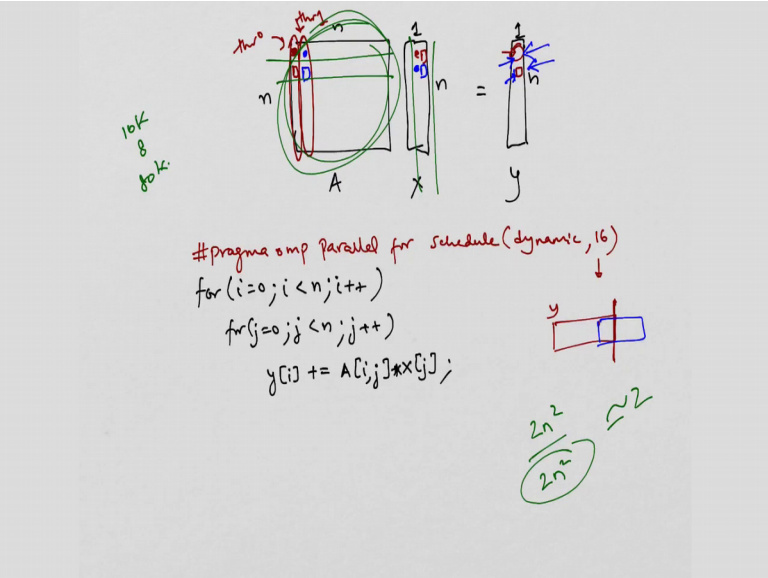
Student: (Refer Time: 03:44).

N multiplications n additions right what is the number of memory accesses I have to do? 2 n right is that clear.

Student: Hum

But I have to do 2 n memory accesses, I am only working on each element only ones right. So, I cannot get around this, this compute to memory ratio is one, the number of compute operations to number of memory operations is one yeah what about this.

(Refer Slide Time: 04:19)



What is the total number of compute operations I am doing? Matrix vector product number of compute operations, do not give me the exact just give me the give me the highest order term and the constant associated with it.

Student: (Refer Time: 04:43).

Two n square

Student: (Refer Time: 04:45).

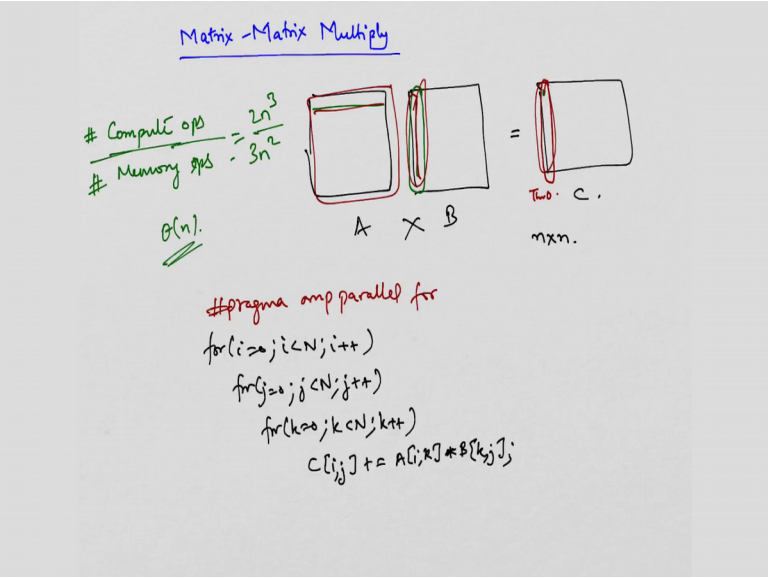
And what is the number of memory operations I am doing?

Student: (Refer Time: 04:49).

I would argue its 2 n square maybe you can reduce it 2 n square, well you definitely have to pick up the entire a matrix and it depends on how you work on x. If you are first multiplying the first row with the entire x column, and by the time you do that your data is going to go out of the cache right. So, for the next row when you start multiplying that with x, x is not in the cache if these matrices are large if these matrices and vectors are large if n is large is this point clear. If you are working with large matrices as you generally have to do in scientific computing applications, by the time you finish one row if it has like let us say 10,000 elements, you are loading 10,000 elements that is about what 10 k elements each is 8 bytes.

So, this is about 80 k that is huge right l one caches are smaller than this 32 k, 64 k. So, you are going to end up doing about 2 n square operations, let us forget that also you have to at least pick up this entire a right you have to do n square memory loads. So, you are going to get about 2 maybe in the best case even if you are able to reuse your x properly, 2 is not good enough two compute operations for every memory load not good enough right you are still memory bound. So, dot product was also memory bound matrix vector product is also memory bound and there is nothing you can do about it these operations are known to be memory bound.

(Refer Slide Time: 06:22)



What about matrix multiply forget about the algorithm, how much data is involved first of all, what is the minimum number of loads you have to do.

Student: (Refer Time: 06:37).

Load stores.

Student: (Refer Time: 06:38).

Three n square is the number of memory operations that I have to perform right and what is the amount of computation I have to do. So, order n cube right roughly about 2 n cube, if I am able to reuse data I may be able to do something better. So, I want to be compute bound I do not want to be memory bound, I want to come up with an algorithm that is compute bound I may not be able to design and algorithm where the compute two

memory operation is of the order of n because of the limitations of the cache size right, but I want to reuse the cache size as much as possible and come up with a good enough bound so that I am compute bound. So, what is the problem with this algorithm that we just discussed? What is the last thing we were discussing? You take the entire matrix a you multiply it with the column of b and you get a column of c right that is what we were discussing right. One thread is doing this, how many load store operations is it performing and how many memory operation is it performing? It is essentially a matrix vector product.

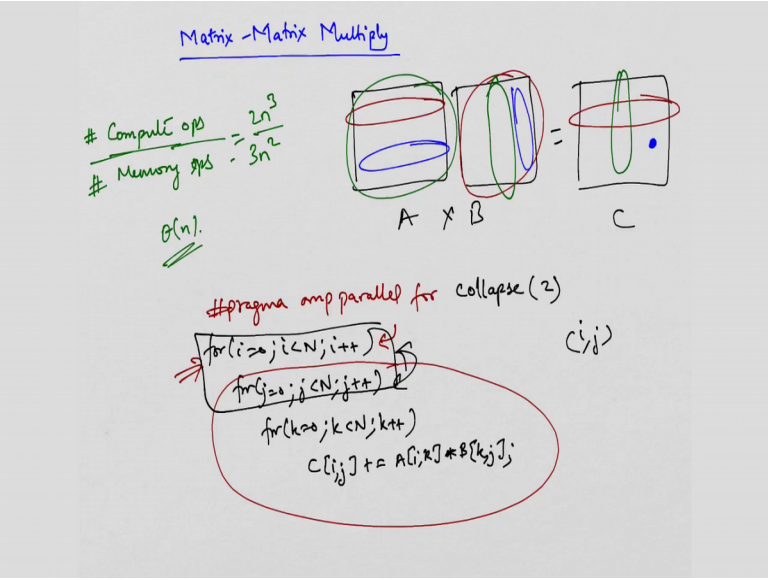
So, that ratio is going to be 2, I want a much higher ratio to be compute bound. So, that is where this scheme lacks if any ideas what can we do then.

Student: (Refer Time: 08:04).

How do we generally write the code for matrix multiply for i is equal to 0, i is less than n, i plus plus for j equal to 0, j is less than n, j plus plus right and then c ij plus equal to I am assuming I have initialized see outside right.

Now, if I simply add a hash pragma omp parallel here, how is it going to distribute.

(Refer Slide Time: 08:59)



So, it is going to give one row of a at a time right this is the first for loop, that is what it is going to divide right. This is the only loop that gets divided amongst the threads. So, once a thread gets a particular iteration of i, it is going to execute this entire piece of code

for that value of i right this point is clear right the only thing getting divided is the first for loop amongst the threads and. So, it is going to work over the entire B matrix multiply it with this vector of A and store it where in the corresponding row of C, and this is also vector matrix product right this is same as matrix vector product it is a vector matrix.

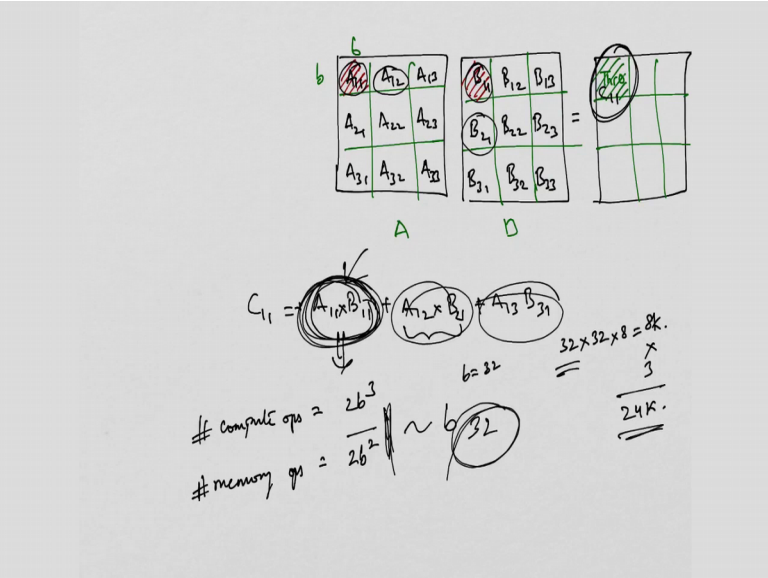
So, this is also memory bound same issue. If I swap these two loops and then I put hash pragma omp parallel for. So, the first loop is let us say j and the second loop is I then what would it be.

Student: (Refer Time: 10:00).

It would be a column of B and the entire A and you would end up producing the corresponding column of C right and we have just discussed that I want something better. Before we go to something better, there is something called the collapse clause what this does is. So, if I say collapse 2 it basically collapse these 2 loops together, the two immediately occurring loops and then I divide them up amongst the threads ok.

So, in this case each thread will get an i j pair to work on. So, what is the distribution of work, how is the work divided what is each thread getting? Each thread is getting one row of a one column of b. So, it will get one row of a one column of b and it will update that particular Cij its updating one element of c, but what is the best way of doing it we just discussed right I want to reuse the cache, because I know that there is scope because the number of compute operations is much more than the number of memory operations. But I am still constrained by the cache. So, what can I do?

(Refer Slide Time: 11:25)



Here is one way of doing it. So, let me not directly jump to blocks, let me divide this up into blocks of rows and let me divide b up into blocks of columns. Now I give these two thread 0. So, thread 0 takes care of multiplying this entire sub matrix with this sub matrix, and what does it produce? It produces this particular block right.

So, what is the problem here? So, the only problem is again cache reuse right. So, by the time I traverse this entire row and multiply with this entire row, and I traverse this and I traverse all these elements and multiply them, this first data element would have been thrown out of the cache because these vectors are so long. So, here is what I do? I just divide them into blocks let us say of size b cross b how do I choose b. So, I have to look at the cache size I have to be aware of the cache size to know that what is the b I should choose, but what happens now? Let us say that I give this block of c to thread 0 right.

So, what does it need to compute this? Well what it needs is it needs this block of A this block of B, it will multiply them together right. So, let us call these blocks something let us give them a name. So, this is A11, A12, A13, A21, A22, A23. So, these are blocks each is of size b cross b right and this is C11.

The ideal I should start with 0, but its fine for now. So, how do I compute C11? C11 is A11 times B11 product of these two sub matrices, plus A12 times B21 plus A13 times B31 right if I multiply these sub matrices these pairs of sub matrices then I get my block

C11 you have to convince yourself of that its easy to check you can verify all right what have I gained by doing this.

So, let us say I do this operation on A11 and B11, I do this product before I touch A12 and B21, I do not go row wise I load this entire block I load this entire block I multiply them then I go to A12 B21 right I go block by block . So, now, in doing A11 times B11 if that entire data fits into the cache, what is the number of memory operations I have to perform and what is the number of compute operations? What is the number of computer operations?

Student: (Refer Time: 14:49).

Two b cube right and what is the number of memory ops? 2 b square or 3 b square that depends on whether my c also fits into the cache or not right I have to load these sub matrices of a and b, can I write back the sub matrix of c or not if it fits into the cash then I only incur an overhead of 2 b square. I mean I incur a cost of two b squared the number of memory operations is 2 b square right. So, you see the difference now. So, now, you are able to achieve something better right you are able to get much better compute the memory ratio. So, now, what does it depend on?

Student: (Refer Time: 15:34) earlier we were getting (Refer Time: 15:36).

We were not getting order of n that was theoretical right we did not have an algorithm to do that.

Student: (Refer Time: 15:44).

I want to achieve this that is what I was saying you cannot achieve n, because you are limited by the cache size. So, here we were making the point that maybe there exists an algorithm which can achieve this there is scope that is all I was saying. In case of vector vector operations there is no scope because the number of operations you have to do you have to fetch as many elements as the number of operations you have to do.

In case of matrix vector there is no scope, because the amount of data you have to load is proportional to the number of operations you have to do. So, there is no scope and then we said that in matrix multiply there seems to be scope, the number of compute operations I have to do is much more than the number of memory ops.

So, I want to achieve that now. So, how do how do I achieve that what is the algorithm to achieve that? I want to achieve n, but I cannot achieve n if I have infinite cache I can achieve n right I can load the entire matrices into the cache and then I will be done, but I cannot achieve n because I do not have infinite cache. So, I am bound by the cache size.

So, therefore, I said that if b is a number, such that three sub matrices of size b cross b can together fit into the cache a sub matrix b sub matrix and c sub matrix. If all three of them can in fit in the cache if there exists such a b, then I am going to achieve this ratio right. So, what is the practical value of b? So, if I choose let us say b equal to 32 right then how much data do I want to load into the cache 32 into 32 is this size of the sub matrix a and each element is 8t bytes.

So, this is about 8 k right and then there are three such matrices. So, 24 k not bad doable right you have l one caches which are typically larger than 24, 32 k, 64 k these are reasonable sizes if you have 64 k you could try to get even better.

Student: But in this case we have to use locks.

Why do you have to use locks?

Student: Because when I am giving A11 and B11 to one and A12 and b A12 and B21

right.

Student: It (Refer Time: 17:55) to another (Refer Time: 17:56).

No what I gave to thread 0 was C11 that is what I distributed amongst the thread the blocks of C. Once thread 0 is responsible for C11, it will do all these computations, but it has to rearrange the code it, it cannot be in the form that we wrote it little while back right this is not the code to achieve that it has to work this way it has to first do A11, B11 then A12, B2 and then A13, B31 right this just the order in which you write the code.

So, a lot of these things are valid even for single processors when you are writing multi core code, you have to know how to distribute the work amongst the threads right and for that you need to know what, which are the most optimal algorithms.

So, what kind of speed up have I achieved roughly about b right and if b is 32, then I have been able to achieve a compute to memory ratio of 32 which is not bad right on

most processors it could be reasonable. I should be able to cover for the memory latency with the compute time. So, typically how do you write these codes I mean we are not going to go that far, but how do you optimize this even further. So, what you do is that while the processor is computing A11 B11, it will keep issuing instructions to fetch A12 and B21 into the cache right.

But then you have to hold not 3 sub matrices, but 5 sub matrices in the cache A11 B11 the next A12, B21 and C. C11 right 5 mat sub matrices. So, then you have to choose your b accordingly, but then you can completely you know hide the memory latency behind the computations. So, I can interleave instructions in the code to fetch that data into the cache. So, there are instructions available to pre fetch data into the cache, as I said we are not going to go down to that level, but you can do all those.