Optimization of FFT Parallel Algorithm on Multi-Core CPUs

Fang Ai Dong

School of Information and Engineering, Suzhou University, Suzhou, 234000, China

Abstract -- Fast Fourier Transformation is widely applied in fields of digital signal processing and communication. Multi-core CPU is increasingly common, and based on the parallelism of FFT algorithm, the butterfly algorithm can be flexibly decomposed. We can improve the computational efficiency of FFT via investigating the allocation of parallel blocks and nested relations to optimize the algorithm, and reasonably allocating threads to realize parallel computing of multi-core CPU. The guided task scheduling model is used before the optimization of the relocation algorithm in parallel. The effect is better than using Static mode and Dynamic mode. Whether for the pre-modification of the butterfly algorithm or for the modified butterfly algorithm or not, the performance of parallel programs is significantly better than others. Parallel realization of DIT-R2-2D-FFT algorithm performance has been significantly improved.

Keywords - FFT; multi-core CPU; OpenMP; parallel computing

I. INTRODUCTION

In the field of digital signal processing and image processing, the Discrete Fourier Transformation (DFT) algorithm has the important status and spread to all fields of science quickly after its Fast Algorithm FFT (Fast Fourier Transform Algorithm, Fast Fourier Transform) is put forward by Turkey and Cooley. FFT algorithm has been greatly improved in the aspects of structure covering algorithm data storage way, instruction execution. With the development of the CPU, GPU, and special-purpose DSP technology, the parallel algorithm based on multi-core and multi-processor show its powerful computing efficiency, therefore they are widely used in real time, efficient, scientific computing, the FFT parallel algorithm implementation technology is research deeply. Parallel processor structure and parallel processing algorithm are the basic method to realize parallelism, reasonable design of parallel algorithm is an important assurance for efficient operation. Based on multi-core CPU and two-dimensional fast Fourier transform of parallel implementation and its optimization, discusses the parallel computing in process and distribution of parallel block in the nested relation which has influence on the efficiency of the algorithm.

1965, Cooley and Tukey put forward the calculation method of the discrete Fourier transform rapidly marks a significant milestone in DFT calculation. It is basically divided into two categories, according to DIT (Decimation - in – Time) method and DIF (Decimation - in - Frequency,) method. Because time extraction calculation has more symmetrical neat FFT butterfly unit than frequency sampling calculation, this paper chooses extraction based on time for a brief introduction of the FFT algorithm and optimize FFT algorithm.
Figure 2 shows the iterative process of N points FFT according to time extraction \( N = v^2 \).

(3) the two-dimensional discrete Fourier transform and its separation properties

The two-dimensional DFT for the separation of two step process of one dimension DFT can be represented vividly as Figure 3. For example, the application of image processing, can be understood as DFT transformation to a two-dimensional images, each column / line of the image as one dimension DFT respectively; then each line / column of the obtained results for one dimension DFT respectively.

Figure 3 shows DFT calculations by two-step two-dimensional DFT.

The fast Fourier transform of intrinsic parallelism

By a FFT algorithm processes the FFT algorithm with high partition features, the algorithm has more than can be carried out in parallel operation.

(1) butterfly computation of parallelism

In the process of the whole butterfly operation, a node in the calculation of different stages do two DFT calculations with different node, the butterfly operation has strict successively relationships from left to right different levels. Butterfly operation process at each level, to participate in two DFT calculations are independent of each other, between nodes within the same level of butterfly computation, multiple points to two points DFT can be calculated by the multiple units in parallel processing respectively, set up a sync point at each level, all of DFT in the same level within two processing is completed, unified into the next level of butterfly computation for a new round of parallel DFT calculations with two points.

(2) the parallelism existing in the move operation

Adjusted the order of the input node in the move operation, there is also a similar parallelism. Define a move operation: transfer position of the node of \( x(k) \), \( k \). Determine whether \( k_i \) is greater than \( k \), change the serial number as and two nodes if \( k_i \) is greater than \( k \), otherwise do not make any operation. When \( k \) traverse the entire range of \([0, N-1]\), it can complete all nodes position adjustment without repeat.

After judgment which is greater, if \( k_i \) less than or equal to \( k \), it does not perform any action and won't read and write the corresponding node position, so when two independent of operation adjust the position of \( x(k) \) and \( x(k_i) \) respectively at the same time, there will be only one of the operation variables of two position switch position, and won’t produce reading and writing conflict. Above of all, if there are multiple units do move operations on different position of the node. In parallel, it still can get the correct results.

(3) the partition of lines and columns of the two-dimensional DFT

Two-dimensional DFT ranks of can be decomposed into two one dimension in the direction of DFT in implementation, as well as the two-dimensional node data of each row (or each column) for DFT, between line and line (or columns between the columns). Therefore, from the point of view of 2-d DFT realization, the calculation of DFT of row (column) node as a computing module, used multiple cell and DFT of the multiple set of nodes are calculated respectively. Line in the direction of DFT calculations are completed, unified into the column on the direction of DFT calculations; finally get the correct calculation results of two-dimensional DFT.

(4) other operations of parallelism

In the process of the calculation of FFT, the basic operation is also the existence of parallelism. Such as complex multiplication, the product of the real part and imaginary part can parallel computing; two DFT calculations also in a similar situation. Of course, for this kind of small particles of parallel block, to make a way to create a thread for the parallel implementation is inappropriate. Usually it can be used to accelerate the operation instructions to complete this particular calculation, as Cheng-Jun Li [5, 6] use Intel - SIMD instructions.

II. MULTI-CORE CPU ON THE PARALLEL IMPLEMENTATION OF FFT AND ITS OPTIMIZATION

OpenMP is a compiled for multiprocessing machine in Shared memory parallel programs designed for the application programming interface, with standard Fortran, the combination of C and C ++ for synchronous load tasks such as Shared variables, reasonable distribution, provides effective support, and has the characteristics of simple general, development fast [10-11].
OpenMP data parallel programming language

When using OpenMP of FFT algorithm for parallel processing in the process of we mainly use the following command and functions:

1) #pragma omp for [clause [ clause...]] New - line

For loop iteration must be independent, so no matter how the iteration order of execution, or what exactly thread execution by part which iteration of the loop, loop results will be the same. If a thread into variables, another thread to read the variable, then transfer cycle correlation, program will also generate incorrect results.

2) the NUM_THREADS ()

This clause can be closely followed for instruction, is used to specify the number of threads in the created thread group.

3) omp_set_nested () function

This function is used to enable or close nested parallel region. When nested parallel area is enabled, as long as it is need to create a multithreading instruction, so the thread pool capacity permits always create or allocate a new thread. When nested parallel area is closed, only the main thread to meet need to create multithreaded instructions will accordingly create or allocate a new thread.

The multicore cpus on the parallel implementation of FFT and its optimization used FOR instruction of parallel modules

We move from the site operations, butterfly operations and procession of partition of two-dimensional FFT to do parallel processing.

(1) move parallel implementation of operation

The following table to move both before and after optimization algorithm, but the algorithm analysis can be found that the optimized function is not suitable for use for instruction of OpenMP parallel processing.

<table>
<thead>
<tr>
<th>Before the serial optimization</th>
<th>After the serial optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>for(i=1;i&lt;=N; i++) {</td>
<td>for(i=1;i&lt;=N-2; i++) {</td>
</tr>
<tr>
<td>if((bitAddA = Rev_d(i, m)) &gt; i)</td>
<td>if((i&lt;j){</td>
</tr>
<tr>
<td>temp = Data[i];</td>
<td>temp = Data[j];</td>
</tr>
<tr>
<td>Data[j]=Data[bitAddA];</td>
<td>Data[j]=Data[i];</td>
</tr>
<tr>
<td>Data[bitAddA] = temp;</td>
<td>Data[i]=temp;</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

Reason is that in the optimized BitReverse () function in the for loop, the variable j as you progress through the loop iteration (I, in turn, varying from 1 to N - 2), the change of its value is affected by the variable I and k; And former one iteration of the j variables will affect the results of a new round of iterations. This shows that the existing in the circulation loop transfer correlation, in violation of the principle of the for loop iteration must be independent. And initial BitReverse () function does not exist the problem, and only need to add the following statement before for statement:

# pragma omp parallel for private (i, bitAddA, temp) num_threads(ThreadNum) schedule(type [, the chunk])

Therefore, in the general base-band algorithm should not only seriously in the parallel implementation of the parallelism of algorithm analysis theory itself, but also in terms of specific code implementation should pay attention to the rationality of the algorithm. Not every kind of the optimized algorithm can directly use simple compilation statements which can be converted into parallel algorithm, only meet for loop iteration must be the principle of independence, to eliminate possible internal variables loop transfer correlation, can correctly use for instruction will be paralleled algorithm.

(2) butterfly computing palatalization

One-dimensional DIT DFT DIT_R2_1D base 2 FFT to realize the function () at the core of the code section such as the following for loop shown below:

```c
for(r = 1; r <= m; r++)
{
    length = (int)pow(2.0, r);
    interval = length/2;
    for(i = 0; j <= interval-1; j++)
    {
        nk = pow(2.0, m-r);j;
        alpha = (2*PI/N)*nk;
        WN_nk.real = cos(alpha);
        WN_nk.imag = -sin(alpha);
        for(i = j; i <= N-1; i = i+length)
        {
            Data[i+interval] = Multi-
                            -ply-complex(Data[i+interval], WN_nk);
            DFT_2(Data+i, Data+i+interval);
        }
    }
}
```

Where the variable r specifies the current level of butterfly operation, scope for. J, I for two loop iteration variable reflects the specific process of butterfly computation. Can be found between the two loops nested relation, and the inner loop of the scope of the dependent variable (I) is dependent variable j varies according to the external cycle. J variable scope will change because of the change of the interval value.

※ a nested loop for load balancing problem in parallel

For parallel algorithm based on cyclic performance, the key is reasonable will allow loop iteration to the thread, the thread's load balance, namely the computing tasks to each thread distribution will be basically the same, to avoid excessive spare some threads, some threads overly busy phenomenon, lead to inadequate computing resource utilization. And a nested loop is more often appear load imbalance phenomenon is one of the areas, especially inner loop and outer loop iterative range counter variable related cases.

Code above in the algorithm of parallel processing in
the nested loop won't appear the problem of load imbalance, the reasons are as follows: if the outer for loop before adding #pragma omp for instruction, namely the for loop of j variables corresponding parallel computing, then corresponds to a value of j, circulation within the internal loop iteration calculation can only because is different with the number of times. As a matter of fact, for any given j value, at the same level of butterfly operation, the number of inner loop is the same, the reason is that the change of the i variable stride for the length. That is, the first level of butterfly computation set point DFT calculations, j variable reflects the inner loop of DFT is calculated by two node in each point DFT operation group which two, while I variable determines the DFT calculations in the inner loop of two belong to which group in the set of nodes. When variables are determined, the number of I and j values are set at the same time, and respectively. So at the same level of butterfly computation inside, arbitrary values of the variable j, I variable number of values are the same, the same amount of calculation, the corresponding cycle to ensure that the calculation of load balancing.

 encountering the effects of variable scope change

The scope of the variable J interval () value of the change and change. Assuming that the current program is running in a quad-core CPU, is when the interval of 1, j variable has only two values, at this time can only be assigned two threads to the corresponding iterative parallel computing, thus wasted two cell computing resources; Only butterfly operation immediately after entering the second, j variables have more than four values, in order to make full use of four nuclear parallel calculating all iterations. To make the following improvement the above algorithm can solve this problem:

```c
for(r = 1; r <= m; r++) {
    length = (int)pow(2.0, r);
    interval = length/2;
    nk = pow(2.0, m-r)*(2*PI/N);
    // #pragma omp parallel for private(i, address, alpha, WN_nk)
    num_threads(ThreadNum) schedule(type[chunk])
    for(i = 0; i <= N/2-1; i++) {
        alpha = nk*(i%interval);
        WN_nk.real = cos(alpha);
        WN_nk.imag = -sin(alpha);
        address = i*interval*length+i%interval;
        Data[address+interval] = Multiply-
                            Complex(Data[address+interval], WN_nk);
        DFT_2(Data+address, Data+address+interval);
    }
}
```

Butterfly operation at this point, each level has a stable and independent of each other in time loop iteration, which can be annotated the OpenMP statements to complete the for loop parallel computing. Before it is worth noting that the improvement of loop iteration algorithm, the parallel granularity is also changing, and decreases with the increase of the variable j values range, before approaching the improved the algorithm of parallel particle size. And from the Angle of the algorithm efficiency of parallel block granularity, granularity parallel block, the greater the efficiency of the algorithm will be slightly higher, so the former also has a good side, so you need to set up a reasonable compromise.

(3) parallel computing row (column) DFT

Relative to the above two parallel computing, parallel computing row (column) in the direction of multiple sets of one-dimensional DFT algorithm is more simple. If we choose to move operations, butterfly operations at the same time and the multiple sets of one-dimensional DFT calculations for parallel processing, then we will encounter the nested parallel situation. It can be taken at this time in parallel computing row (column) to multiple sets of one-dimensional DFT, only in parallel computing moved one dimension DFT internal address, butterfly computation, and nested parallel processing, we have three different algorithms respectively finished performance test experiment.

set the number of threads and task scheduling

In using OpenMP for instruction in the process of program for parallel processing can also set the number of threads in parallel computing and autonomic computing tasks assigned to each thread scheduling. When determining the number of threads will be used, mainly consider the following two points:

(1) when the loop number is less, if divided into excessive number of threads to perform, can make the total elapsed time is higher than the less threads or a thread execution, and increase energy consumption.

(2) if the thread is greater than the number of CPU on the number of nuclear and there are a lot of task switching and scheduling overhead, will also reduce the overall efficiency.

At the same time, in order to make the program on different multicore CPU computing platforms are able to calculate according to the specific conditions, make full use of computing resources, we need to dynamically adjust the number of threads for adaptive. Generally speaking, the main factors deciding the number of threads for CPU nuclear number and the number of task assigned to each thread. Therefore defines as shown in the following empirical function to set the number of threads to determine:

```c
int DecideThreadNum(int nRound, int nMinRoundPerCore) {
    int nMax_threadNum = nRound/nMinRoundPerCore;
    int tn = max_tn > G_CoreNum ? G_CoreNum : max_tn;
    if(tn < 1)
        tn = 1;
    return tn;
}
```

The nRound variables for the need to compute the
number of iterations, nMinRoundPerCore variable is hope
to each computing kernel distribution, the minimum num-
ber of jobs. The function returns the number of threads in
the relatively reasonable can be set.

In terms of task scheduling, from the above three par-
allel computation analysis, butterfly operation and one
dimensional calculation of DFT are relatively stable and
balanced, more prone to load imbalance is move opera-
tions. In the experiment, we will have three different ways
of task scheduling under the operation efficiency of the
algorithm for testing.

III. THE EXPERIMENTAL RESULTS AND DATA ANALYSIS

The several implementation methods of move operation performance comparison

From figure 4 shows, the optimized move computing algorithm has obvious advantage on the execution efficiency. Before the optimization algorithm is easy to make parallel processing, and the parallel implementation after the performance had obvious ascension. But in serial or parallel way, its efficiency is not moving as the optimized algorithm. It is important to note that in the parallel way for optimizing the move in front of the algorithm, using Guided task scheduling model, the effect is better than that of using the Static model and Dynamic model.

The concurrent nested affect the performance of parallel 2-d DFT algorithm

It can be clearly seen from test results of figure 7, only in the outer layer, namely the line(column) to the N-point DFT, you can get the best effect of accelerating algorithm from the algorithm for parallel processing.
The final multi-core CPU environment operation efficiency of the algorithm

According to the figure 8 shows, in contrast, the realization of parallel DIT - R2-2 d - FFT algorithm performance has the obvious improvement.

Figure 8 the FFT algorithm in the multi-core CPU environment performance comparison.

IV. CONCLUSIONS

This paper studied how to use OpenMP implementation two-dimensional fast Fourier transform of parallel computing under the environment of multi-core CPU, and through studying the distribution of parallel block and nested relation on algorithm optimization. Analysis and experiment show that weigh the computing and storage time, reasonable distribution of parallel block and its nested relations, can help improve the speedup of the parallel algorithm, which improve the efficiency of parallel computing.

ACKNOWLEDGEMENTS

University science research project of Anhui Province Foundation project (KJ2011B180); Anhui Province soft science research project (1502052053), Revitalization of Anhui Province university education(2013zytz074); Anhui Province Quality project Foundation project (20101077), the Adjustment and optimization for the Specialty Structure and the professional transformation, the internet of things engineering(2015zytz066) and Network Engineering (2014zytz030).

REFERENCES

[4] Liu Gongxia, Yang jing, such as the yellow turban insurrectionary. Variable-length FFT parallel rotating factor efficient algorithms and implementation. Journal of xi 'an university of electronic science and technology (natural science edition), 2009, (3) : 541-546, (In the case of journal articles)