Parallel Computing Method for the Non-linear Model of Displacement Measurement

Yuze Chai[1], Jinchun Hu[1,*], Longmin Chen[2] and Shengwu Du[2]
[1]State Key Laboratory of Tribology, Beijing Key Lab of Precision/Ultra-precision Manufacturing Equipments and Control, China
[2]Department of Mechanical Engineering, Tsinghua University, Beijing, China
[*]Corresponding author

***Abstract*—Non-linear model of displacement and sensor detection is usually used in multi-DOF precise displacement measurement. The computing burden of these models is often too large to satisfy the need of real-time performance of displacement measurement. This paper proposes a parallel computing method based on the task scheduling model, and uses GA to optimize task scheduling plan to enhance the real-time performance of displacement measurement.**

Keywords-nonlinear model; parallel computing; task schedule; GA

I. INTRODUCTION

Precise displacement measurement with significant multi-DOF motion is a hot research area in mechanical engineering. Guofeng Ji [1] established a non-linear model of grating interferometer detection and displacement, and solve the displacement by numerical iteration. Chen L [2] established a non-linear model of Hall sensor detection and displacement, and solve the displacement by decoupling and optimization. Non-linear model can enhance the precision of displacement measurement on the above occasion, however, with model complexity and redundancy, the computation burden of the method is too large to meet the demands of real-time performance.

We usually use parallel computing method to solve the real-time performance problem when faced with large computation burden in the field of image processing [3], audio processing [4], radar signal processing [5], etc. The common characteristic of these fields is that they need to deal with lots of data in short time, and parallel computing in these problems is to parallelize the matrix computing.

However, the data amount of the non-linear model of displacement measurement is usually not that large. Meanwhile, the models of different measuring systems also differ, which makes it hard to parallelize the measuring model from the perspective of parallel matrix computing. But if we regard the model as a task set, and the calculation steps of the model as tasks with logical order, we can design a task scheduling algorithm to get appropriate parallel computing plan for any certain model. In this way, we can realize the parallel computing for non-linear measuring model. As an NP-hard problem [6], task scheduling can be solved by GA [7]. From the above, we propose a GA-based design method for parallel task scheduling plan, to parallelize the computing process of the non-linear model in displacement measurement.

II. PROBLEM CHARACTERISTICS

Non-linear model displacement measurement is based on the continuity and inertia of the movement. We assume that the displacement change between two contiguous sampling time is sufficiently small, so that the non-linear model can be high precisely linearized. In this way, we can solve the current displacement with the result of previous time.

Set the sensor signal at moment k as Y_k, the displacement of the object as X_k, the sensor noise as $\tilde{\varepsilon}_k$, the parameters about model as C. Then

$$Y_k=F(C,X_k)+\tilde{\varepsilon}_k \quad (1)$$

Considering about the movement continuity and the sufficient high sampling frequency, X_k can be sufficiently close to the real displacement at moment k+1. We can get the approximate solution $X_{k+1}^{(i+1)}$ by iteration

$$X_{k+1}^{(i+1)}=X_{k+1}^{(i)}+\left[J^T\left(X_{k+1}^{(i)}\right)J\left(X_{k+1}^{(i)}\right)\right]^{-1}J^T\left(X_{k+1}^{(i)}\right)\left(Y_{k+1}-Y_{k+1}^{(i)}\right) \quad (2)$$

where $X_{k+1}^0=X_k$, $Y_{k+1}^0=Y_k$, $Y_{k+1}^{(i)}=F\left(C,X_{k+1}^{(i)}\right)$, $J\left(X_k^{(i)}\right)=\left.\frac{\partial F}{\partial X}\right|_{X_k^{(i)}}$.

In the solving process above, the relation between sensor signal Y and displacement X is usually complex and includes lots of trigonometric function and exponential function operations. On the other hand, we must use float-point other than fixed-point number in calculation, because the fixed-point number may produce significant precision loss in the non-linear function [8]. These reasons lead to the real-time performance problem of non-linear model displacement measurement.

Chen L [2] established the non-linear model of Hall sensors and 3-DOF displacement. We can expand the model to 6-DOF displacement measurement model as (3).

In (3), q_{ij} satisfies (4), and x_{i0}, y_{i0}, z_{i0} are coordinates in moving reference system of sensors. x, y, z, θ_x, θ_y, θ_z are displacement. B_m, τ are parameters about magnetic field.

The application of the model requires measuring sampling

frequency more than 20KHz. We used TMS320C6678(basic frequency 1GHz, 8 cores) as implementation platform. Under the condition of single core, the period of solving displacement is more than 1ms, which is 20 times as the period required, and the time for updating the non-linear model accounts for about 50%. This shows that the computing burden of non-linear model has seriously affected the real-time performance. In order to solve the problem, we propose the design method for parallel task scheduling plan based on GA.

$$B_i=B_m\exp\left(-\frac{2\pi}{\tau}\left(q_{31}x_{i0}+q_{32}y_{i0}+q_{33}z_{i0}+z\right)\right)\cdot\begin{pmatrix}q_{13}\cos\left(\frac{2\pi}{\tau}\left(q_{11}x_{i0}+q_{12}y_{i0}+q_{13}z_{i0}+x\right)\right)+q_{23}\cos\left(\frac{2\pi}{\tau}\left(q_{21}x_{i0}+q_{22}y_{i0}+q_{23}z_{i0}+y\right)\right)\\ -q_{33}\left(\sin\left(\frac{2\pi}{\tau}\left(q_{11}x_{i0}+q_{12}y_{i0}+q_{13}z_{i0}+x\right)\right)+\sin\left(\frac{2\pi}{\tau}\left(q_{21}x_{i0}+q_{22}y_{i0}+q_{23}z_{i0}+y\right)\right)\right)\end{pmatrix}\cdot \quad . \quad . \quad (3)$$

$$\left\{q_{ij}\right\}=\begin{bmatrix}\cos\theta_y\cos\theta_z & \sin\theta_x\sin\theta_y\cos\theta_z-\cos\theta_x\sin\theta_z & \sin\theta_x\sin\theta_z+\cos\theta_x\sin\theta_y\cos\theta_z\\ \cos\theta_y\sin\theta_z & \sin\theta_x\sin\theta_y\sin\theta_z+\cos\theta_x\cos\theta_z & \cos\theta_x\sin\theta_y\sin\theta_z-\sin\theta_x\cos\theta_z\\ -\sin\theta_y & \sin\theta_x\cos\theta_y & \cos\theta_x\cos\theta_y\end{bmatrix}. \quad .(4)$$

III. THE TASK SCHEDULING MODEL

A. *Directed Acyclic Graph Model*

Taking the computing progress of measuring model (5) as an example, we regard the progress as tasks with logical relations (as shown in Figure 1(A)).

$$F(x)=\cos(\sin x+\cos x)+\sin(\sin x-\cos x) \quad . \quad .(5)$$

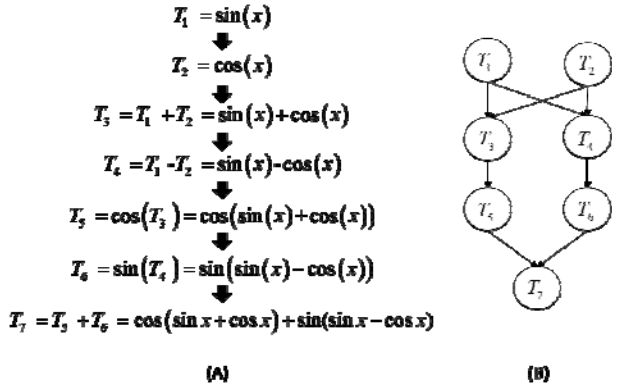

FIGURE I. (A) AN EXAMPLE OF COMPUTING PROGRESS, (B) AN EXAMPLE OF DAG MODEL

The tasks have their own processing time, and some certain tasks need the computing result of other tasks as input data. In this way, the computing progress of the measuring model can be transformed into a DAG (directed acyclic graph) as shown in Figure 1(B). A DAG can be described with a task set T and a directed edge matrix A. Where:

Task set $T=\{T_i\}$, i=1, 2, …, n. T_i denotes the ith task. Each task is a computing step of the measuring model.

Directed edge set $A=\{A_{ij}|A_{ij}\in\{0,1\}\}$, i,j=1, …, n. A_{ij} denotes the logical relations in the DAG. If $A_{ij}=1$, T_j needs the result of T_i as input and vice versa. For i=j, $A_{ij}=0$.

B. *Parallel Machine Model and Relevant Parameters*

Processor set $P=\{P_j\}$, j=1, …, m. P_j denotes the jth processor. Tasks are executed on the processors. These processors have shared memory. P and the shared memory constitute a single CPU multi-core parallel machine.

In the parallel scheduling plan, there are two constraints on the order of task execution. First of them is the sequential logic, namely the directed edge set A. The second constraint is the precedence of the tasks allocated into the same processor. According to them we classify the related tasks of a task as DAG predecessor/successor tasks and processor predecessor/successor tasks. Set the DAG predecessor/successor task set of T_i as $Pred_D(T_i)$, $Succ_D(T_i)$, and the processor predecessor/successor task set as $Pred_P(T_i)$, $Succ_P(T_i)$.

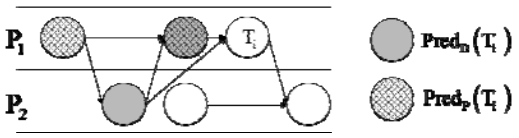

FIGURE II. AN EXAMPLE OF $Pred_D(T_i)$ AND $Pred_P(T_i)$

As shown in Figure 2, $Pred_D(T_i)$ only includes the directly related tasks of T_i in the DAG, while $Pred_P(T_i)$ includes all of the tasks before T_i in the processor itself allocated. So as the successor task sets.

According to the DAG, set the depth value of each task as $H_i=H(T_i)$. The depth value describes relative level of the tasks in the DAG. Where:

$$H_i=\begin{cases}0, & \text{if } pred_D(T_i)=\varnothing\\ 1+\max\limits_{T_j\in pred_D(T_i)}(H_j), & \text{otherwise}\end{cases} \quad . \quad .(6)$$

All of the tasks allocated in the same processor must be sorted by the ascending order of H_i, and task with smaller depth value will be first executed.

t, describe the executing parameters of T on the parallel machine, where:

Execution time set $t=\{t_i\}$, i=1, …, n. t_i denotes execution time of the ith task. For the homogeneous parallel system, the execution time of one certain task on each processor is same.

Communication time denotes the time for the process that one processor reads data from or writes data to the shared memory. The communication time for all the process are the same.

Task scheduling matrix $C_{mn}=\{c_{ji}|c_{ji}\in\{0,1\}\}$, j=1, …, m; i=1, …, n. It describes a parallel scheduling plan. If $c_{ji}=1$, T_i is allocated in P_j and vice versa. C_{mn} must satisfy the constraints on the order of task execution above.

C. *Processing Time Model*

Because tasks with sequential logic may be allocated into different processors, a task may write its result to, or read its DAG predecessor tasks' results from the shared memory. Figure 3 shows the processing time model of a single task.

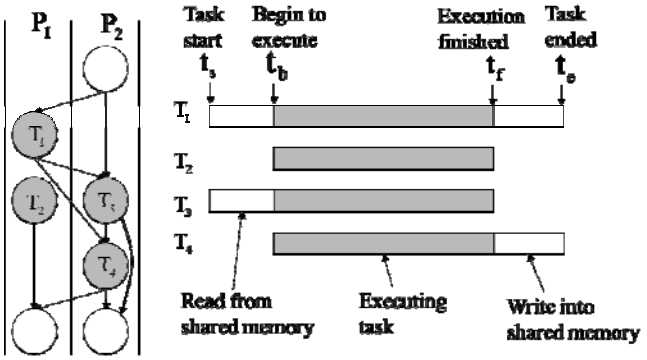

FIGURE III. PROCESSING TIME MODEL OF A SINGLE TASK.

According to the logic relation in Figure 3, we can deduce:

- The necessary and sufficient condition that T_i needn't read results from the shared memory is:

$$Pred_D(T_i)\subseteq Pred_P(T_i)\cup Pred_D(Pred_P(T_i)) \quad .(7)$$

Equation (7) means that all the DAG predecessor tasks of T_i are allocated into the processor itself allocated, or their results have been already read by the processor predecessor tasks of T_i.

- The necessary and sufficient condition of that task T_i needn't write results into the shared memory is:

$$Succ_D(T_i)\subseteq Succ_P(T_i) \quad .(8)$$

Equation (8) means that all the DAG successor tasks of T_i are allocated into the processor itself allocated.

The start time of T_i is related with the biggest end time of tasks in $Pred_D(T_i)$ and $Pred_P(T_i)$. Set the start time of T_i as $t_s(T_i)=t_{s,i}$. We can deduce that:

$$t_{s,i}=\begin{cases}0, & \text{if } pred(T_i)=\varnothing\\ \max\limits_{T_j\in pred(T_i)}(t_{e,j}), & \text{otherwise}\end{cases} \quad .(9)$$

Where $pred(T_i)=pred_D(T_i)\cup pred_P(T_i)$.

Set the end time of T_i as $t_e(T_i)=t_{e,i}$. We can deduce that:

$$t_{e,i}=t_{s,i}+t_i+\delta_r\tau+\delta_w\tau \quad .(10)$$

Where $\delta_r,\delta_w=\begin{cases}1, & \text{the task needs to writes into/ reads from the shared memory}\\ 0, & \text{otherwise}\end{cases}$

Obviously, the processing time for the whole task

$$t_e=\max_i\{t_{i,e}\} \quad .(11)$$

Based on the description above, we can express the parallel computing of the non-linear model of displacement measurement as: searching for a task scheduling plan, namely a task scheduling matrix C_{mn}, to minimize t_e.

IV. A GA-BASED SOLUTION TO TASK SCHEDULING PLAN OPTIMIZATION

The process of GA is shown in Figure 4. C_{mn} is the result to be solved, so set C_{mn} as the chromosome. Set an individual as S_i, and the chromosome $C(S_i)$ denotes a feasible solution.

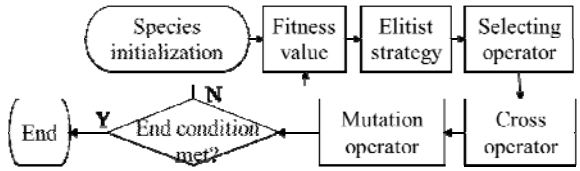

FIGURE IV. THE PROCESS OF GA

Species initialization: divide the task set T into h+1 subsets by the depth value 0~h. Allocate the tasks in a subset into m processors randomly, and sort the tasks in one processor by ascending order of depth value, then we can get a task scheduling plan C(Si).

Fitness value: we can evaluate the chromosome fitness by the time of completing all the tasks. The less time it needs, the fitter the chromosome is, so we can set the fitness formula as (12).

$$F(S_i)=\left(\sum_i t_i\right)-t_e(S_i) \quad .(12)$$

Elitist strategy: the elitist strategy that keep the

chromosome with highest fitness value from selecting operator can improve the convergence and optimize results of GA.

Selecting operator: we use fitness-proportionate selection to choose the chromosomes with better fitness. Set the population size as k, then the individual selected probability is as (13).

$$P(S_i)=F(S_i)\Big/\sum_{i=1}^{k}F(S_i) \quad .(13)$$

Cross operator: cross operator includes internal cross operator and external cross operator.

- External cross operator: generate a nonnegative integer q($0\le q\le h$) randomly. Divide C(Si) and C(Sj) into two parts by row according to q, and exchange the latter parts of the two chromosomes.
- Internal cross operator: generate a nonnegative integer q($0\le q\le h$) randomly. Choose two columns in C(Si) randomly, divide them into two parts according to q, and exchange the latter parts of the two columns. In this way, we can get a new chromosome C(Si').

Mutation operator: generate a nonnegative integer q($0\le q\le h$) randomly. Set the number of each column's tasks whose depth value equal to q as Ni. Choose a gene whose depth value equals to q from the column with the biggest Ni, and migrate it to the column with the smallest N_i. In this way, we can get a new chromosome C(S_i').

V. ALGORITHM IMPLEMENTATION

Divide the 6 DOFs magnetic field-displacement model into 54 tasks. The biggest depth value of them is 9. The executing time and communication time on TMS320C6678 is shown in Table 1, and the completed time for tasks on single core (i.e. time for serial computing) is 3860 counts.

TABLE I. THE EXECUTING TIME AND COMMUNICATION TIME

Task Type	Time
trigonometric and exponential function	200 counts
Multicore communication(for double precision float)	80 counts
other general computing	30~40 counts

According to the task scheduling model and GA optimization above, we computed the nonlinear model (3) on TMS320C6678 with 4~6 cores. The completed time is shown in Table 2.

The speedup S of parallel computing is defined as $S=T_p/T_s$, the efficiency E of parallel computing is defined as E=S/m, where T_s is time for serial computing, T_p is time for parallel computing, and m is number of cores. As shown in Table 2, with the increase of cores, time cost for computing decreases, while efficiency slightly lower. This is because with more cores, the communication time gradually increases.

TABLE II. PARALLEL COMPUTING RESULTS ON TMS320C6678

Core Numbers	Time	Speedup	Efficiency
4	2770 counts	1.39	0.348
5	2260 counts	1.71	0.341
6	1890 counts	2.04	0.340

On the occasion of 6 cores, we can get an optimized task scheduling plan as shown in figure 5. Under this plan, the completed time is 1890 counts, while it is 3860 in serial computing. So the completed time decreased to 49.0% using parallel computing.

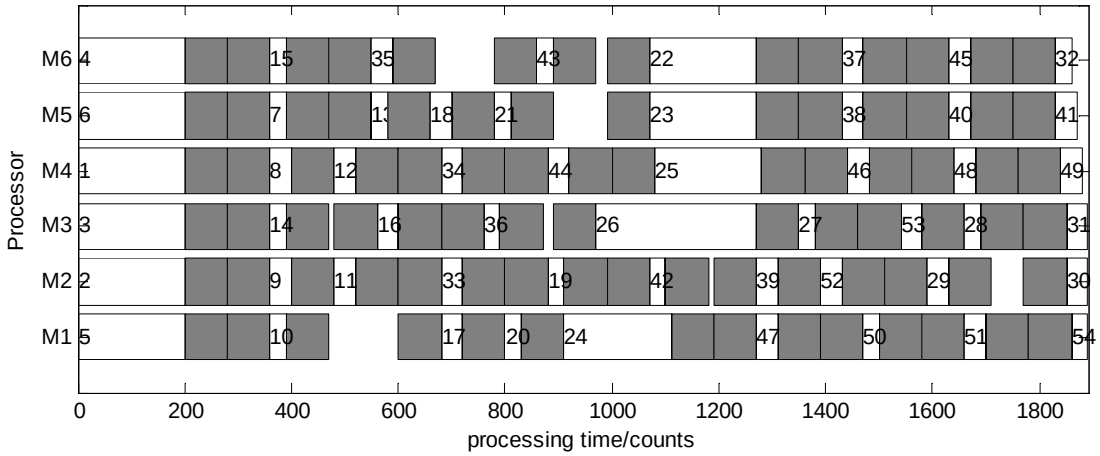

FIGURE V. AN OPTIMIZED TASK SCHEDULING PLAN ON 6 CORES

VI. CONCLUSIONS

This paper proposed a design method of parallel computing for non-linear model in displacement measurement. For the purpose of enhancing real-time performance, the method converts the original non-linear model into a task set with logical relations, and uses GA to optimize the scheduling plan of the task set on the parallel machine. In this way, we decrease the model computing time of magnetic information based 6-DOF displacement measurement down to 49.0%.

ACKNOWLEDGMENT

The research is supported by National Natural Science Foundation of China under Grant 51175296. The authors would like to thank Yipeng Ji for his contributions to this work.

REFERENCES

[1] Guofeng Ji, Jinchun Hu, Yu Zhu, et al. A Novel Method to Reduce Accumulative Switching Errors in Multi-Sensor Incremental Measurement Systems. ECS Transactions, 2014, 60(1): 863-868.

[2] Chen L, Hu J, Zhu Y, et al. A novel planar 3D magnetic position measurement methodology using linear Hall sensors[C]//Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on. IEEE, 2012: 776-781

[3] Olmedo E, Calleja J D L, Benitez A, et al. Point to point processing of digital images using parallel computing[J]. International Journal of Computer Science Issues, 2012, 9(3).

[4] Jiang G X, Chen W. Online parallel dynamic time warping algorithm for speech recognition in embedded system[J]. Application Research of Computers, 2010.

[5] Klilou A, Belkouch S, Elleaume P, et al. Real-time parallel implementation of Pulse-Doppler radar signal processing chain on a massively parallel machine based on multi-core DSP and Serial RapidIO interconnect[J]. Eurasip Journal on Applied Signal Processing, 2014, 2014(1).

[6] Chen J, Lee C Y. General multiprocessor task scheduling[J]. Naval Research Logistics, 1999, 46(1): 57-74.

[7] Sathappan OL, Chitra P, Venkatesh P, Prabhu M. Modified genetic algorithm for multiobjective task scheduling on heterogeneous computing system. IJITCC, 2011, 1(2):146–158

[8] Fang C F, Rutenbar R A, Chen T. Fast, Accurate Static Analysis for Fixed-Point Finite-Precision Effects in DSP Designs[C]// Computer-Aided Design, International Conference on IEEE Computer Society, 2003:275-275.