Abstract
In order to improve the simulation ability, efficient communication ability and real-time capability of large scale complex product prototype, a task scheduling strategy based on multi core cluster hierarchy characteristics and node conflict cost is designed based on the support and framework constraints of High Performance Computing-Run-Time Infrastructure (HPC-RTI), and the scheduling performance, algorithm performance and communication overhead are compared and tested. The application results show that this strategy not only optimizes the scheduling length, but also significantly reduces the utilization of processing nodes. Secondly, in the task of a large number of multi-core cluster system used to produce better performance, with the communication calculation ratio CCR value increases, The proposed algorithm exhibits better scheduling performance than other algorithms.

Key words: Distributed Collaborative, Task Scheduling, High Performance Computing, Run-Time Infrastructure.

1. INTRODUCTION
The simulation verification system of large-scale and complex digital prototyping products needs high-end computing power, high-efficiency communication capability and high real-time capability (Zhang and Zhang, 2013). Especially the real-time system for some aircraft control system, air dynamics simulation environment difficult of real time requirements (Zhang and Xue, 2014). In recent years, with the rapid development of multi-core computing, reflective memory and other technologies, high performance RTI communication mechanism based on shared memory SHM (shared memory) is becoming a research hotspot.

Running Support Environment RTI is the core component based on HLA simulation system. In recent years, SHM-based high-performance RTI technology has been a group of experts and scholars in the study. Such as (Daisuke Tsujinishi and Shigeo Abe, 2005) open source RTI software - based on the increase in shared memory communication optimization strategy; Richard Fujimoto and other symmetric multi-core processor SMP (Symmetric Multi-Processor) environment for the FDK increase the shared memory communication packets and the corresponding time management algorithm (Grogan, Paul and de Weck, 2015). With the continuous development of information technology, the current solution to solve the distributed simulation technology using cloud computing as a basic simulation platform, cloud computing is a software resources to provide services and hardware resources sharing framework technology, through the provision of software as a service model provides users with more convenient and powerful comp. The cloud manufacturing is a kind of service-oriented, highly virtualized resource and intelligent networked manufacturing model based on knowledge sharing and Internet of things. On the model of traditional Web service requestor, service provider and registration center, (CSD), cloud service provider (CSP), cloud service system (Hiraishi and Kunihiko, 2015), and cloud service middleware. Cloud manufacturing can realize the unified management of design, production, processing, experiment, simulation, management and integration of complex aerospace products in the whole life cycle.

In summary, the current research on high-performance simulation support environment mainly focuses on the optimization of communication mechanism, and there is little research on computing environment of multi-core and reflective memory network, which is the breakthrough of this paper. The traditional simulation task scheduling problem mainly considers the communication cost between nodes. In the multi-core cluster environment (Tiplea and Furtcio, 2015), because of its hierarchical structure has the conflict cost of multi-core nodes increases obviously with the increase of CPU core (Lee and Logan, 2007). Therefore, this paper studies the task scheduling algorithm considering the characteristics of multi-core cluster hierarchies and the cost of intranode conflict under the support and frame constraints of HPC-RTI, and studies how to build the solution model of task simulation in multi-core cluster environment (Li, 2015).
2. ANALYSIS OF DISTRIBUTED TASK SCHEDULING MODEL

The task management function module of digital prototyping collaborative modeling is mainly used to complete tasks such as task creation, maintenance, modeling task decomposition, modeling task assignment and control of modeling and simulation process.

In this paper, the parallel program is represented by a quadruple, \( G = (T, E, R, W) \), where:

1. \( T \) is the set of vertex tasks \( \{ T_i \} \), \( T_i \) is the task number corresponding to the vertex;
2. \( E \) is a set of directed edges \( \{ E_y \} \). \( E_y \) denotes the edge between the tasks \( T_i \) and \( T_j \), \( E_y \in E \) means that the task \( T_j \) can be executed only after the execution of the task \( T_i \), and that \( T_i \) is the predecessor of \( T_j \) and \( T_j \) is the successor of \( T_i \);
3. \( R \) is the set of the computing time of the vertex task \( \{ R(T_i) \} \), \( R(T_i) \) is an additional information of the vertex, indicating the execution time of the task \( T_i \);
4. \( W \) is the set of communication overheads between tasks \( \{ W(E_y) \} \). \( W(E_y) \) is another side additional information. Which represents the communication overhead between task \( T_i \) and task \( T_j \);

\[ \text{pred}(T_i) = \{ T_j \mid E_y \in E \} \text{, succ}(T_i) = \{ T_j \mid E_y \in E \} \], where \( | \text{pred}(T_i) | \text{, } | \text{succ}(T_i) | \) denote the number of \( T_i \) predecessor and successor tasks, respectively.

If \( | \text{pred}(T_i) | = 2 \), then \( T_i \) is the join node. If \( | \text{pred}(T_i) | = 0 \), \( T_i \) is the starting task. If \( | \text{succ}(T_i) | = 0 \), then \( T_i \) is the termination task, denoted by \( T_j \); If \( | \text{succ}(T_i) | \geq 2 \), then \( T_i \) is the fork node. The Fork-Join structure is a basic model structure for parallel processing, and most of the task graphs can be simplified to a combination of Join and Fork structures.

Since a set of ordered tasks can be represented as a DAG graph, each node in the graph corresponds to a simulation verification task or task node. Fir st, process allocation and thread allocation are carried out. Therefore, this article makes the following agreement:

1. The simulation verification task is executed in non-preemptive mode, that is, the simulation tasks with precedence constraints are only executed when the predecessor task is completed and the generated data is transferred to the computing resource of the successor task.
2. Simulation task transmission time only with the transmission between the two tasks. If two simulation verification tasks are executed on the same computing resource, the transmission delay is zero; otherwise, a constant;
3. The simulation tasks allocated to the same computing resource, the execution time can not overlap, that is, at the same time on the same computing resources, only one simulation task can be run.
4. In the DAG diagram there is only one starting task node, one terminating task node.

Figure 1 is a DAG diagram showing the virtual node T0 and T9.

![Example of a DAG diagram with two virtual nodes](image)

Because the computation cost of virtual nodes and the communication cost with other tasks are defined as 0. Therefore, this paper adds a virtual node to a DAG graph with multiple start nodes or multiple termination nodes to simplify the operation and has no effect on the original DAG scheduling length.
Multi-core cluster task scheduling process is the first process assigned to the processor node, and then the threads in the process assigned to each processing core nodes (Xu, 2012). The goal of scheduling is to minimize the overall scheduling length of parallel programs on the basis of balanced load balancing.

3. DISTRIBUTED COOPERATIVE TASK SCHEDULING ALGORITHM

3.1 Requirement Analysis and Design of Modeling Task Management Function

The functional description of a Modeling Project Task Management (MPTM) can be described as:

\[ MPTM = (MPCM, MPMM, MTRM, MTAM) \]  

(1)

Where,

(1) **MPCM**: Modeling Project Created Management is based on user task requirements to build performance prototype collaborative modeling tasks. The main attributes of the modeling project are: project name, project number, project design unit, project category, project status, project creator, project creation time, project completion time, project last modification.

(2) **MPMM**: Modeling Project Maintenance Management is a process of modifying, updating, etc., of a constructed modeling project.

(3) **MTRM**: Modeling Project Resolved Management is based on the hierarchical structure model of the prototype system. The modeling of the complex product performance prototype is divided into molecular system. Each subsystem can be divided into modeling subtasks with small coupling degree.

(4) **MTAM**: Modeling Project Assigned Management is composed of different disciplines in the field of modeling staff with different modeling tasks to complete a modeling goal, in the performance of multiple tasks for the serial, parallel and cross-coupling process (Chi, 2010). Therefore, after the decomposition of modeling tasks, task allocation is needed to ensure that sub tasks are allocated to the appropriate collaborative modeling team at the right time. Modeling task allocation and task design ability requirements are related to the ability of the modeler. This paper uses the personnel integrated capability matrix to describe the task capability allocation factors (Wang, 2013).

Set a modeling task set is \( T = \{T_1, T_2, ..., T_n\} \), the collection of modelers is \( Man = \{E_1, E_2, ..., E_m\} \), the task assignment problem is to look for a mapping from \( T \) to \( Man \), assigning the right task to the right person. Personnel Allocation Factors Available Personnel Allocation Matrix:

\[
[P]_{m \times n} = \begin{bmatrix}
P_{11} & P_{12} & \cdots & P_{1n} \\
P_{21} & P_{22} & \cdots & P_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
P_{m1} & P_{m2} & \cdots & P_{mn}
\end{bmatrix}
\]  

(2)

\([P]_{m \times n}\) is used to quantitatively describe the extent to which the modeler is qualified for each task.

3.2 Task Scheduling Algorithm

**XL** said that the multi-core cluster system scheduling length:

\[ XL = \max \{XL(P)\}, i = 1, 2, 3, ..., m \]  

(3)

Where, \( m \) represents the number of processor nodes required to allocate a given set of ordered tasks; \( XL(P) \) represents the earliest possible completion time for all tasks assigned to the processor:

\[ XL(P) = \max \{XL(C_k)\}, k = 1, 2, 3, ..., n \]  

(4)

Where, \( n \) denotes the number of processing cores in the processor \( P_i \); \( XL(C_k) \) is represents the sum of the earliest possible completion times of all the task sequences allocated to the processing core \( C_k \) of the processor \( P_i \), that can be written:

\[ XL(C_{ik}) = \sum_{T_{j} \in C_{ik}} R(T_{j}) \]  

(5)

When two tasks are assigned to different processing nodes, the communication delay between them is large. And when they are assigned to the same processing node, the communication delay between them is very small and can even be ignored. Based on the task replication algorithm is to reduce the communication overhead through this principle. In general, multi-core cluster contains two levels of processing nodes. Therefore, the
multi-core cluster system prototype simulation task scheduling algorithm must be two levels of optimization: process-to-processor node optimization, as well as the thread sequence to the processing of the nuclear node optimization; at the same time, each task execution time is determined, so in the two-stage optimization process to minimize the communication between nodes as the main purpose. This optimization problem of multi-core cluster system performance prototype simulation task scheduling length is transformed into the optimization task sequence of simulation task assigned to each processing node in the cluster.

3.3 Scheduling Strategy

General for a single CPU of CNC system, the system software structure is before and after the desktop (Xu and Xue, 2016). The program bears almost all real-time function, background program used to complete the preparatory work and management work, the task scheduling mechanism with priority preemptive scheduling and time slice rotation combination mechanism.

In this paper, we adopt the following scheduling strategy to make each node task form a new cluster:

1. Before the simulation task begins, each task node in the DAG is regarded as a cluster;
2. For the current cluster \( C_v \), if there is only one key precursor task \( u \), its corresponding cluster is \( C_u \), then the \( C_v \) will be merged into the \( C_u \) to form a new cluster: \( C_v = C_v \cup C_u \).
3. For a current task \( v \) with multiple precursor clusters, first merge \( v \) into its critical precursor cluster \( C_u \) to form a new cluster; the remaining precursors of \( v \) are then detected in turn, if a predecessor cluster is added to the new cluster \( C_v \) and the start time of the current task \( v \) can be advanced, the precursor cluster is merged into the new \( C_v \). In the process of merging two clusters, if two clusters are clusters with multiple tasks, they should first remove the redundant tasks, and then according to the priorities between the task scheduling order.

4. EXPERIMENTAL COMPARISON AND PERFORMANCE ANALYSIS OF ALGORITHMS

4.1 Scheduling Performance Test

The random graph is used as the data set for task scheduling test, and the performance of the scheduling algorithm is evaluated by comparing the latest completion time of the simulation verification task. As the clustering strategy is a key part of the algorithm, this paper first test and compares CPFD algorithm (Lei Lu and Feng Zhang, 2016), PPA algorithm and the clustering strategy proposed in this paper scheduling performance. Therefore, it is necessary to introduce the CPFD algorithm and PPA algorithm briefly: Critical Path Fast Duplication (CPFD) uses a heuristic policy to allocate scheduled tasks to the processor of the parent task and an empty processor, and calculates the corresponding earliest start time to eventually assign the task to get the earliest start time on the processor. CPFD algorithm is in the scheduling of the current task, recursively looking for its important parent task VIP (Very Important Parent). Finally, the VIP task will be copied to the current processor, so that the current task to get the earliest start time, in order to shorten the scheduling length. The scheduling results of clustering algorithm in this paper are shown in figure 2.

![Figure 2. Scheduling results of the clustering strategy algorithm in this paper](image)

As can be seen by comparing the clustering strategy of scheduling results with PPA scheduling length, but the processing node clustering strategy used is the least; and the CPFD algorithm of task scheduling in spite of shorter length, but occupy more processing nodes. Thus, this clustering strategy not only optimizes the scheduling length, but also significantly reduces the utilization of processing nodes.

4.2 Scheduling performance test

For the DAG task graphs with the number \( N_{thread} \leq 64 \), we use the "exhaustive search method" to find the optimal solution, and then compare it with the algorithm. Comparing the genetic algorithm with the normalized scheduling length of the algorithm, the task graphs with the number \( N_{thread} > 64 \) are compared.
By using the Simics simulator for the test environment, performance test of this algorithm, exhaustive search method and genetic algorithm, the three algorithms in simulation environment. The algorithm was evaluated and analyzed using 3600 randomly generated DAG maps. The number of threads in the task graph is 64-2048, which belong to 8 to 128 processes; the weight of the edge between threads is in the range of 0 to 50 randomly; the weight of the process edge is between 10 to 60. In the number of processors for the 16 to 128, and the number of processing cores for a variety of hardware platforms 8 to 128 is on the task assignment.

In this paper, the clustering strategy is mainly focused on the communication-intensive task scheduling, because it focuses on reducing the communication overhead. The error ratio is calculated from the sum of the weights of the edges assigned to different processing nodes. The calculation formula is:

$$\text{Error ratio} = \frac{\text{Algorithm solution} \sum W(E_j) - \text{Exhaustive search for optimal solution} \sum W(E_j)}{\text{Exhaustive search for optimal solution} \sum W(E_j)} \times 100\%$$ (6)

Figure 3 and figure 4 show the optimal solution ratio and error ratio.

![Figure 3. The optimal solution ratio of this algorithm](image1)

![Figure 4. Error ratio of the algorithm in this paper](image2)

As shown in Figure 4-8, with the increase of the number of threads, the optimal solution ratio of the algorithm in this paper shows a slow descending trend. As shown in Figure 4-9, the error ratio is always controlled below 4.5%. The algorithm can find the near optimal solution.
5. CONCLUSIONS

In this paper, firstly, based on the characteristics of multi-core computing and reflective memory computing, this paper, on the basis of SHM-CERTI developed by Martin Adelantado and other scholars, can increase the performance of high performance computing platform by adding memory Sharing and researching and designing the software system structure of HPC-RTI, and deeply analyzes and studies the key issues. Then, the task scheduling algorithm considering the characteristics of multi-core cluster hierarchy and the cost of intra-node conflict is studied under HPC-RTI support and framework constraints. A solution model of task allocation problem is established in multi-core cluster environment. Finally, three sets of experiments show that the proposed algorithm is less than the other two classical algorithms, and with the increase of the communication calculation rate, the superior performance of this algorithm is better reflected.

ACKNOWLEDGEMENTS

This work is partially supported by the Defense basic research projects of #A0420131501, Funding Project for Department of Education of Shaanxi Province of China (#15JK1861, 15JK1859), Natural Science Basic Research Plan in Shaanxi Province of China (2016NY-141, 2016KJXX-62). Thanks for the help.

REFERENCES

Daisuke Tsujinishi, Shigeo Abe (2003) "Fuzzy least squares support vector machines for multiclass problems", Neural Networks, vol.16, pp.785-792.
Grogan, Paul T.; de Weck, Olivier L.(2015) “Infrastructure System Simulation Interoperability Using the High-Level Architecture”, IEEE Systems Journal, 13(12),pp.211-214.
Hiraishi, Kunihiko.(2015) “Simulating Markovian stochastic Petri Nets by difference equations with interval parameters”, Discrete Event Dynamic Systems: Theory and Applications, 25(3), pp.365-386.
Lee P M, Logan P B, Theodoropoulos P G. (2007) “DistributedSimulation of Agent-based Systems with HLA”, ACM Transactions on Modeling and Computer Simulation, 17(3), pp 1-25.
Li, Ran (2015) “Performance optimization for a class of generalized stochastic Petri nets”, Discrete Event Dynamic Systems: Theory and Applications, 15(3), pp. 387-417.
Lei Lu, Feng Zhang, Fei Yang, Jiangang Dong (2016) “Modeling and Collaborative Simulation Model Library for Performance Prototype”, Rev. Téc. Ing. Univ. Zulia, 39(4), pp. 56-63.
Tiplea, Ferucio Laurentiu (2015) “On the complexity of deciding soundness of acyclic workflow nets”, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45(9), pp. 1292-1298.
Vazquez, Carlos R.(2015) “Stochastic Hybrid Approximations of Markovian Petri Nets”, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45(9), pp. 1231-1244.
Wang, Hongwei (2013) “A distributed and interactive system to integrated design and simulation for collaborative product development”, Robotics and Computer-Integrated Manufacturing, 26(26), pp.778-789.
Xu, Heng (2012) “Implementation and simulation of ontology-based knowledge sharing model in collaborative product design support system”, International Journal of Advancements in Computing Technology, 4(13), pp. 447-453.
Yuan Xu, Huifeng Xue, Ailong Liu (2016) “A Parallel Genetic Algorithm for the Verifying Simulation of the Performance Prototype”, Rev. Téc. Ing. Univ. Zulia, 39(7),pp. 88-86.
Yuliang Chi.(2010) “Rule-based Ontological Knowledge Base for Monitoring Partners across Supply Networks”, Expert Systems with Applications, 37(2), pp. 1400-1407.
Zhang Yong-Heng, Zhang Feng (2013) “A New Time Synchronization Algorithm for Wireless Sensor Networks Based on Internet of Things”, Sensors and Transducers, 151(4), pp.95-100.
Zhang Feng, Xue Hui-feng (2014) “A method of reliability modeling based on characteristic model for performance digital mock-up”, Computer Modelling and New Technologies, 18(2), pp. 238-243.