INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 9, Number 2, Pages 232–246

HYBRID PERFORMANCE MODELING AND ANALYZING OF PARALLEL SYSTEMS

BIN CHENG, WEIQIN TONG, AND XINGANG WANG

Abstract. Performance is a key feature of parallel system. However, there is a great gap between the peak performance and performance attainable by a practical application. The model-based performance evaluation may be used to support the performance-oriented program development for parallel system. In this paper a hybrid TCPN model is proposed to describe the parallel program and the resources respectively. This method can bring less effect to modify the program structure because of running environment changes. And the performance engineering activities based on this model ranges from performance prediction in early development stages, performance analysis in the coding phase, to locate the performance bottleneck and modify it. After the correctness verification of the TCPN model, a reachable graph can be got. Then the further performance-tuning can be done by summing the execution time of corresponding action in the critical path.

Key words. Timed Coloured Petri Net, parallel system, formal method

1. Introduction

Compared to the traditional development process of sequential software where performance issues are insufficiently considered one is now convinced that performance evaluation is a critical factor in the upcoming parallel software development methodology. But performance orientation in the development process of parallel software is motivated by outlining the misconception of current approaches where performance activities come in at the very end of the development, mainly in terms of measurement or monitoring after the implementation phase. At that time major part of the development work is already done, and performance pitfalls are very hard to repair. So a development process for parallel programs that launches performance engineering in the early design phase is needed.

In this paper we put forward a hybrid approach to support for functional and temporal specification which can specify various aspects of parallel system like software, such as control flow, data flow, communication, synchronization and so on, and hardware, such as processors, memory, communication media etc. It is simple but expressive graphical means. This novel method can develop parallel software by performance engineering during the whole parallel development cycle. It means the performance analysis begins at the early design phases and goes on until the completion of the application, not work during or after the running procedure like some test tools now. Otherwise, it happens often that a correct program is a program with lower performance and leads to the huge expenses to modify program.

The rest of this paper is organized as follows: after the briefly introduction of the problem that need to be solved in Section 1, influencing factors of parallel system performance will be analyzed in Section 2 and a time model of parallel program is built. PRM model based on Timed Coloured Petri Net is proposed. The formal models of parallel program and executive environment are defined in detail. There is

Received by the editors December 31, 2010 .
2000 *Mathematics Subject Classification.* 35R35, 49J40, 60G40.
This research was supported in part by Shanghai Leading Academic Discipline Project, Project Number: J50103.

an example in Section 3 which demonstrates the building process and the analyzing of the TCPN model. Section 4 concludes this paper and introduces the future work.

2. Hybrid Performance Modeling and Analyzing

Formal methods are frequently applied in industries to build mission-critical systems where insufficient verification can cause human injury or large-scale financial loss. The topic of using the formal method to identify the correctness verification and performance analysis of the parallel program gains a lot of attention [5,6].

Petri nets provide the foundation of the graphical notation and the basic primitives for modeling parallel, communication, and synchronization. It is a strict formal tool of mathematical modeling which can transform the Petri nets model into mathematical problems or simulation models. Then the qualitative and quantitative analysis of the system performance can be easy. However, the basic Petri nets only record the number of token not the individual character. And excessive description about the individual changes makes too many nodes of the Petri nets to characterize the complex process of the system and decreases the abstract capacity. To solve these problems many authors propose extensions of the basic Petri nets model.

Several authors have extended the basic Petri net with coloured tokens. In these models tokens have a value, often referred to as 'colour'. There are several reasons for such an extension, such as uncoloured Petri nets describe real systems tend to be complex and extremely large, tokens often represent objects or resources which have attributes in the modeled system. These 'coloured' Petri nets allow the modeler to make much more succinct and manageable descriptions.

Petri nets execute transition firings instantaneously, i.e. there is no time to be consumed, which is certainly sufficient for reasoning about the quality of system behavior, such as synchronization. To make the Petri nets formalism adequate also for quantitative, i.e. performance, analysis, finite timing of activities can be expressed by associating a time concept to places, transitions, tokens and any combination of them.

Compared to another formal methods, only Timed and Coloured Petri Net(TCPN) is suitable to model the large and complex parallel systems and more effective to analyze synchronization, communication, performance and reduce the possibility of state explosion.

2.1. PRM Model based on TCPN. It's figured that the performance of parallel systems is not only determined by the performance of the hardware itself but also by the structure of the parallel program and the assignment of program parts to resources in this paper. The actual performance is determined by the interdependencies between hardware performance and the requirements of parallel programs, i.e. the proper utilization of hardware performance by the program. It means the performance measures of interest are the run time of the parallel program and the degree of hardware utilization.

With PRM[1] a modeling technique has been given considering hardware resources, parallel program and mapping between them as the performance influencing factors for the prediction of performance of parallel computations running on parallel hardware. The separation of the three specification elements should enable to vary each of them, as far as possible, independently from the other specifications. This approach can realize various mapping between the parallel program and the resource with minimal additional effort and compare the performance of program running on different resource.

The performance analysis procedure of parallel system based on performance model is illustrated in FIG. 1. In the Program Model, each task of f_i the program is described in terms of the operations performed and expressed as a function of the program character parameters P=$(f_1, f_2...f_n)$. In the Resource Model, each machine resource is described by their performance character r_i, such as the communication bandwidth, computing speed, storage speed, etc.

The most important layer of this model is the mapping between the task f_i of parallel program and the resource r_i. The formal description of program model, resource model and mapping can be converted a timed coloured Petri net model(TCPN) and analyzed. It is a simulation process. Its output is the execution time of the program, the utilization of the hardware resource, speedup and other performance measures. Analysis the performance report can tell the programmer where the bottleneck is and how to modify the program or improve the hardware architecture. The model that combines Program Model with Resource Model is referred to as hybrid model.

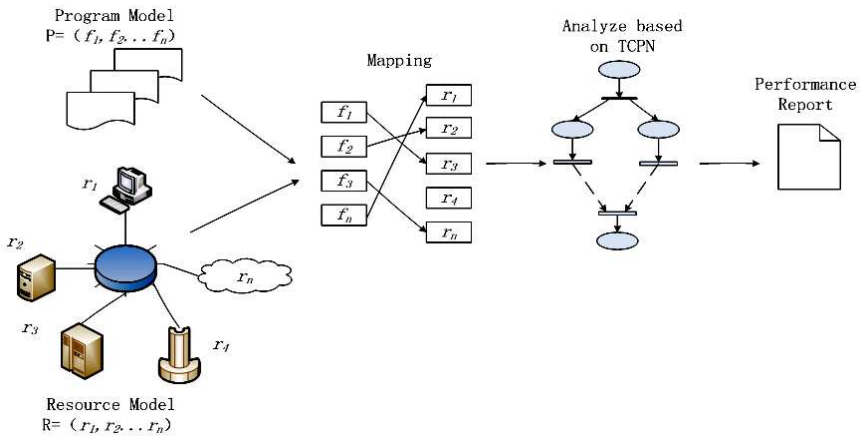

FIG. 1. Performance analysis procedure of parallel system based on performance model

2.2. Static Structure of TCPN. Definition 1 S={P-TCPN, R-TCPN, M} is the model of parallel systems. P-TCPN is the model of parallel program based on TCPN. R-TCPN is the model of resource based on TCPN. And M is the mapping between P-TCPN and R-TCPN.

Definition 2 P-TCPN $=(P_p, T_p, A_p, C, \Phi, FT)$ satisfying the following requirements:

(1) P_p is a finite set of places. A place contains zero or more tokens. A token has 3 attributes $< p, v, tm >$ to denote a token in place p with value v and time tm. v is the colour of the token and tm is the timestamp that token arrives at the p. P_{start} and P_{end} are $E_1.E_{p1}$ and $En.E_{pm}$ denote the start place and the end place of P-TCPN respectively.

(2) T_p is a finite set of transitions. $P_p \cap T_p = \emptyset$.

(3) A_p is a set of flow relations, $A_p \subseteq (P_p \times T_p) \cup T_p \times P_p)$. It shows the track of tokens flowing in the net.

(4) C is a colour function. It is a multi-set. Each $p_i \in P_p$ has a set of colours attached to it, i.e. $v \in C(p_i)$.

(5) Φ is the set of functionalities of T_p. $\forall t_i \in T_p$ finishes partial function of the program, which is $\Phi(t_i)$.

(6) FT is the set of time functions of T_p. $FT(t_i)$ is the time delay to realize $\Phi(t_i)$ and also describes the interval that t_i occupies the related resources.

P-TCPN oriented parallel application model is used to specify the structure, functionality and operation character. Each $p_i \in P_p$ is a state of an object and each $t_i \in T_p$ is a change or an event. The token is the object or the condition of p_i. $\forall t_i \in T_p$ is either a delay transition or an instant transition, which is represented by box or bar. If a transition spends some time on consuming tokens from the input places and producing new tokens to the output places when it is enabled, it is a delay transition, such as the task execution time of processor, message communication, and so on. On the contrary if a transition consumes tokens from the input places and produces new tokens to the output places immediately when it is enabled, it is an instant transition, such as logical decision.

An event or a change is ready to get active, if its corresponding transition t is enabled; it gets active as the corresponding transition starts firing, and remains active for the firing duration. The transition behaves according to a predefined functionality and terminates by releasing tokens to output places, then making subsequent transitions ready to get enabled/active. The P-TCPN can be arranged to be executed in sequence, parallel, alternatively or iteratively, like FIG.2. To support hierarchical specification, event compositions can be folded to form a single, compound event, graphically represented by a single transition (box), by aggregation of the tasks and the execution time of all the events constituted in it.

FIG.2 describes the structure of a simple parallel program. P_s,t_s, P_e and t_e compose the parallel process. P_{10} and $P_{20}$are the initial states of the two parallel program bodies. The transitions represented by boxes are compound transitions. t_{seq1} and t_{seq2} are the sequential composition and they aggregate a set of sequential processes to a single compound process as a matter of abstraction. The execution time of t_{seq1} or t_{seq2} is cumulated by all primitive transitions in it. P_{11} and t_{rep} compose the *loop* structure. P_{20}, t_{b1}, t_{b2} and P_{21} compose the alternative structure. t_s, P_{com} and t_r compose the communication structure. t_s is the sender, P_{com} is the buffer and t_r is the receiver.

After correct verification of the P-TCPN model the expected performance of the parallel application under development can be analyzed. This requires inclusion of hardware performance characteristics and program to hardware mapping information into the model. We assume the hardware resource which is the running environment of parallel application is a pool of resources full of memory, processing elements and communication devices. Their connectivity and interactivity as well as the potential performance are modeled by R-TCPN.

Definition 3 R-TCPN=$(\sum_r, P_r, T_r, A_r)$ satisfying the following requirements:

(1) $\sum_r$ is a finite set of resources. $\sum_r = (r_1, r_2 \ldots r_n)$.

(2) P_r is a finite set of home places for the resources $r \in \sum_r$. r is idle and available if there is a resource token in its home place. The token has 2 attributes $< p, v >$, describing the token's home place p and the physical characteristics v separately.

(3) T_r is a finite set of transitions.

(4) A_r is a set of flow relations, $A_r \subseteq (P_r \times T_r) \cup (T_r \times P_r)$. $\forall a_i \in A_r$is the direction of resource flows and describes the interactions between resources.

Every resource in the parallel system is modeled by a token in the home places of R-TCPN. It means the resource is idle if the corresponding token is in the place.

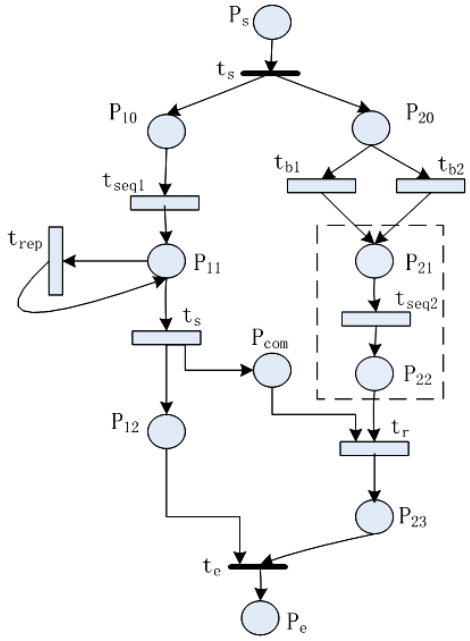

FIG. 2. Structure of a simple parallel program

A_r describes the direction of resource flows and help to model interactions and interdependencies between resources.

Definition 4 $M \subseteq (P_r \times T_p) \cup (T_p \times P_r)$ is the mapping between P-TCPN and R-TCPN.

(1) $P_r \times T_p$is the set of arcs leading from resource home to processes transitions. It is resource assignment.

(2) $T_p \times P_r$ is the set of arcs which is opposite to $P_r \times T_p$. It will happen only if the time of resources occupied by process is equal to the time of allowable service, or the process releases the resources after finishing the task.

S={P-TCPN, R-TCPN, M} is the combination of a P-TCPN and a R-TCPN by a mapping M to a single, integrated net model. It is the specification of the parallel application at the algorithm structure level. Assigning the tokens in home places to transitions of the P-TCPN is allowed only if the events expressed in the transition are offered as services by the resource. FIG.3. shows a simplified assignment of the dashed box in FIG.2 to the hardware resources.

Consider the *state* shown in FIG.3(a), home place P_{free} of R-TCPN and place P_{21} of P-TCPN contain a token respectively. Transition t_{seq2} is *enabled* because there are enough tokens on each of its input places. Assume that t_{seq2} is a compound transition and is composed of t_{seq2-1}, t_{seq2-2} and t_{seq2-3} like FIG.3(b). And $P_{free} = (P_{freeP}, P_{freeM})$ is the set of the idle resources which can serve the transitions $t_{seq2} = (t_{seq2-1}, t_{seq2-2}, t_{seq2-3})$. Each sub-transition is a primitive operation which have deterministic resource requirement. t_{seq2-1} and t_{seq2-2} require the same resource of type P(processor) and P_{freeP} is the resource of this type. So $P_{freeP}$is allowed to be mapped to the two transitions like the directed arc from P_{freeP} to t_{seq2-1}.

Firing a transition means consuming tokens from the input places and producing tokens on the output places. If, at the same *enabling time*, more than one transition

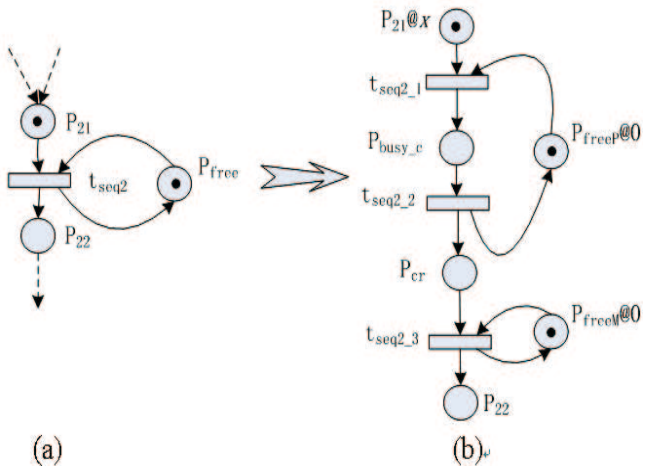

FIG. 3. PRM for the mapping of processor and memory

is enabled, then any of the several *enabled* transitions may be the next to fire, i.e. it is a non-deterministic sequence.

The number and the values of tokens produced by the firing of a transition may depend upon the values of the consumed tokens. And the enabling time of a transition is the maximum timestamp of the tokens to be consumed. The relation between the multi-set of consumed tokens and the multi-set of produced tokens is described by the transition function Φ.

Function $\Phi(t_{seq2-1})$ specifies transition t_{seq2-1} in the PRM shown in FIG.3(b):

$$dom(\Phi(t_{seq2-1})) = \{ < P_{21}, Data > + < P_{freeP}, P > | Data \in C(P_{21}), P \in \sum_r \}$$
$$\Phi(t_{seq2-1})(\langle P_{21}, Data\rangle + \langle P_{freeP}, P\rangle) = \langle P_{busy-c}, \langle Data^{'}, P\rangle\rangle$$

The domain of $\Phi(t_{seq2-1})$ describes the condition on which transition t_{seq2-1} is *enabled*. The enabling time of transition t_{seq2-1} depends upon the timestamps of the tokens to be consumed. If t_{seq2-1} is fired, it consumes one token from place P_{21} and one token from place P_{freeP} and it produces one token for place P_{busy-c}. FIG.4(a) is the result of firing t_{seq2-1} in the state shown in FIG.3(b). The colour of the produced token is a tuple $< Data', P >$. It means that the processor P has dealt with the data Data and a result $Data'$ is got. The resource P_{freeP} is busy and could not be occupied by other requirement. The transition t_{seq2-2} is specified as follows:

$$dom(\Phi(t_{seq2-2})) = \{ < P_{busy-c}, < Data^{'}, P >> | Data^{'} \in C(P_{21}), P \in \sum_r \}$$
$$\Phi(t_{seq2-2})(< P_{busy-c}, < Data^{'}, P >>) =< P_{cr}, Data^{''} > + < P_{freeP}, P >$$

Transition t_{seq2-2} represents the completion of using resource P and produces a new data $Data^{''}$. $t_{seq2-1}$and t_{seq2-2} are different transition and they represent different operations of the parallel algorithm. It spends some time on their execution and depends upon the function **FT**. If t_{seq2-2} occurs in the state shown in FIG.4(a), the state of the system will change into FIG.4(b). t_{seq2-2} produces two tokens $< P_{cr}, DataR >$ and $< P_{feeP}, P >$. It means the completion of a calculation job

with resource P. DataR is the result. The firing of t_{seq2-3} is similar to t_{seq2-1} and t_{seq2-2}. P_{freeM} is different resource from P_{freeP}. It describes the Memory.

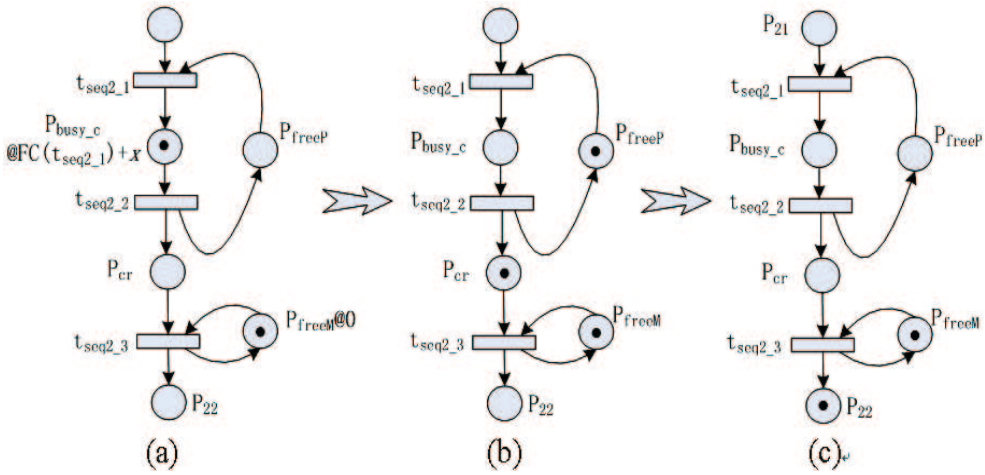

FIG. 4. Firing sequence of FIG.3(b)

2.3. Dynamic Behavior of TCPN. P-TCPN and R-TCPN specify the static structure of parallel system based on TCPN. In this section we define the behavior of P-TCPN.

(1) **Enabled Transition**

Place is the passive component, while transition is the active component. Transition changes the states. A transition is enabled if there are enough tokens, which are to be consumed, on each of its input places in P-TCPN and the required resources in R-TCPN can be assigned. Only the Enabled Transition can be fired, i.e. finish some functions of the parallel program.

(2) **Event**

An event is corresponding to some functions of the parallel program, i.e. it is $E_{pj} \in E_i$. It is a tuple $(t, p_{in}, p_{out}, time)$ and represents the firing of transition while consuming the tokens from p_{in} and adding the new tokens to p_{out} in P-TCPN. p_{in} is the input place of t and p_{out} is the output place of t. $time$ is the execution time of event. It can be got by time function FT(t). For example, the transition t_{seq2-1} in FIG.3(b) is an event. It can be expressed as $(t_{seq2-1}, P_{21}, P_{busy-c}, FT(t_{seq2-1}))$.

(3) **Enabling time**

The enabling time of a transition or an event$(t, p_{in}, p_{out}, time)$ is the maximum of all the timestamps of the tokens consumed, i.e.

$$\mathrm{ET_{ime}}(t) = \max_{(p,v,tm)\in p_{in}} tm$$

If an event is time enabled, it may occur. The enabling time of t_{seq2-1} in FIG.3(b) is x because of x=max(x, 0).

(4) **State**

A state is characterizes as a multi-set of coloured tokens each bearing a timestamp.

$$\boldsymbol{S} = (\boldsymbol{CS} \times \boldsymbol{TS})_{MS}$$
$$\forall s \in \boldsymbol{S}, s = \{(p, v, t) \mid p \in \mathbf{P}_p, v \in \mathbf{C}(p_i), t \in \boldsymbol{TS}\}$$

$\boldsymbol{S}$ is the set of all possible states. $\boldsymbol{CS}$ is a colour set of system. $\boldsymbol{TS}$ is a non-negative real, $\boldsymbol{TS} = \{x \in R \quad | x \geq 0\}$. It represents the time set. Event happening will change the system state. The state notes the static status of the parallel system. The state shown in FIG.3(b) is $< P_{21}, Data, x > + < P_{freeP}, P, 0 > + < P_{freeM}, Memory, 0 >$, that is a state with one token in P_{21} with value Data at x and one token in P_{freeP} with value P at 0 and one token in P_{freeM} with value Memory at 0. x and 0 are the timestamp of tokens.

The state shown in FIG.3(b) is changed into $< P_{busy-c}, < Data, P >, FC(t_{seq2-1}) + x >$ after firing the transition t_{seq2-1}. $FC(t_{seq2-1})+x$ is the timestamp of the token in P_{busy-c}.

(5) **Fire**

An event$(t, p_{in}, p_{out}, time)$ may occur if it is enabled. It means the transition t fires and removes the tokens specified by p_{in} and adds the tokens specified by p_{out}. Firing an event changes the state **S** and the mark **M** of the system.

Let $M(p)$ be the number of tokens in place p. And $M[t >$ denotes t is enabled with the marking M and $M[t > M'$ means the marking M' can be obtained from M after firing t. $\Phi(t)$ is effect of firing transition t.

$$M^{'}(p) = \begin{cases} M(p) - \Phi(t) & p \text{ is the pre - set of } t \\ M(p) + \Phi(t) & p \text{ is the post - set of } t \\ M(p) & \text{otherwise} \end{cases}$$

(6) **Reachable State**

An event $(t, p_{in}, p_{out}, time)$ occurs in state s and state changes into the state s'. It is $s' = s - p_{in} + p_{out}$ and s' is directly reachable from s by the occurrence of event $e = (t, p_{in}, p_{out}, time)$. It is described by $s_1 \xrightarrow{e} s_2$. So the firing sequence of the parallel system is a sequence of states and events, $s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} s_3 \xrightarrow{e_3}$ .....and $s_i$is reachable from s_1 only if there is a firing sequence of finite length from s_1 to s_i.

Reachability analysis is used to get critical path in this paper. This technique builds a reachability graph, sometimes referred to as reachability tree. The reachability graph contains a node for each possible state and an arc for each possible state change. This technique is a very powerful method to prove the most of the properties and answer the question of the parallel system performance. However the reachability graph may become very large and often infinite. It is state explosion. So we should solve this problem before analyzing the performance of the system based on reachability graph. A reduction method which aggregates similar states into state classes will be introduced to cut down the state space in other paper.

3. TCPN Model of Construction the Finest Level of the Distributed Tree

MLFMA(multilevel fast multipole algorithm) is one of the most effective methods solving electromagnetic scattering problems from complex objects. And the parallelization of MLFMA can greatly reduce the solution time of large-scale electromagnetic scattering problems involving complicated three-dimensional objects. The basic data structure of parallel MLFMA is a distribute tree, i.e. assigning every branch of the tree to different processors. The scatterers finish the aggregation, translation and disaggregation based on the distribute tree. The bottom-to-up method is adopted to build the tree. So it is very important to build the finest level of the distributed tree. Whether the data of finest level is assigned equally or not will affect the load balancing and the parallel efficiency of the MLFMA. The

data of the finest level in the distribute tree is the Geometric Data after scatterers' triangulation, which are vertices, facets and edges.

3.1. Hierarchical TCPN Modeling. To keep the specification as clear and flexible as possible, a hierarchical model is applied throughout the parallel system specification. FIG.8 is the top model of the algorithm which is constructing the finest level of the distributed tree. It describes the information transmission among the submodels and the task execution flow. There are 3 substitution transitions in FIG.8 which are realized by relevant submodel in detail. The place between the substitution transitions shows the information interaction and every place has a colour set. The substitution transition treadjust is described in detail as FIG.9, which is the construction of finest level of the distributed tree. And Table 1 expresses the semantic of the transitions and places in FIG.9.

There are 3 processes in P-TCPN to finish the job. No.0 process is the master node and calculates the pivot elements of the sampled data from all nodes and broadcasts the result. No.1 and No.2 processes are the slave nodes. They sort local data and send the sampled data to No.0 process. After receiving the pivot elements, they divide the ordered local data into 3 parts and communicate globally. At last the data of finest level of the distributed tree are ordered locally and globally.

The substitution transition $t_{readjust}$ is built in this way:

(1) Ascertain the set of places and transitions of P-TCPN based on the definitions of $\mathbf{P}_p$ and $\mathbf{T}_p$ and the implementation process of parallel algorithm. Determine the set of places and transitions of R-TCPN based on the topology structure of the parallel environment. Table 1 can be got after this step.

```
▼Declarations
  ▼val b=5;
  ▼colset INT=int;
  ▼colset DATA=string;
  ▼colset R=string;
  ▼colset INTxDATA=product INT*DATA timed;
  ▼colset RxINT=product R*INT;
  ▼colset A = index a with 1..b;
  ▼var x,rc,n: INT;
  ▼var y,data0,data1,data2:DATA;
  ▼var df:A;
  ▼var r:R;
  ▶fun CMorton
  ▶fun sorting
  ▶fun receive
  ▶fun BC
```

FIG. 5. Data definition of the TCPN model

CPN Tools, an industrial-strength computer tool, is used to construct and analyze the TCPN model of parallel system in this paper. The behavior of the modeled system can be investigated by simulating the model. The properties can be verified by means of state space methods and model checking. FIG.5 is the data definition of the TCPN model expressed in FIG.8 by CPN language.

In CPN Tools, colour sets are defined using the CPN ML keyword **colset**, such as INT is a colour set with the integer type. The functions CMorton, sorting, receive and BC are relative to the transition functions, $\Phi(T_{01},T_{11},T_{21})$, $\Phi(T_{02})$, $\Phi(T_{03})$ relatively.

(2) Analyze and determine the relationship between $\mathbf{P}_p$ and $\mathbf{T}_p$. And build the initial model of P-TCPN, the right part of FIG.9 and R-TCPN, the left part of FIG.9.

(3) Ascertain the initial state of P-TCPN and R-TCPN, i.e. the colour set and the number of tokens, based on the rules and the status of the real system.

The place *P_start* in FIG.9 is the start of the parallel work and it has three tokens in the initial state. Its colour set is $INT \times DATA$ and the value(colour) of the token in this place is a two-tuples multi-set with the $INT \times DATA$ type. One of these tokens has the value $(0, data0)$ at time 0 and is expressed $1'(0, data0)@0$. The transition *T_start* fires to judge if variable x is equal to the process number.

(4) Map the elements of P_p, which have data tokens and resource requirements, to the idle elements of P_r, which have tokens too and can satisfy the resource demands.

For example, the transition function T01 is:

$$\Phi(T_{01}) = CMorton(y)$$

Firing transition T_{01} will calculate the Morton Key of data0. It requires processor to finish this job. P1 which is a place of R-TCPN has a token ("P1", 3). It means the resource is idle. Its type is processor and the clock speed is 3GHz. The service time of P1 on T_{01} is calculated by time function $FT(T_{01})$:

$$FT(T_{01}) = \frac{CA(data_0)}{CS(P_1)}$$

CA(data0) is equivalent calculation amount and **CS**(P1) equivalent calculation speed.

3.2. State Space Analysis based on TCPN Model. Simulation can be applied to execute a model of real system within a finite number steps. It is suitable for detecting errors of the model. But it is not confident adequately because it can not guarantee that the simulation covers all possible executions. State space represents all possible executions of the model. Its basic idea is to calculate all reachable markings (state) and marking changes of the TCPN model and build a reachability tree where the nodes correspond to the set of reachable markings and the arcs correspond to the transitions which lead to the marking change.

(1) Reachability

M_k is a marking from M_0 in P-TCPN and M_0 is the initial marking. It is reachable iff $\exists M_0 : M_k \in R(M_0)$. $R(M_0)$ is the set of reachable markings from M_0. Reachability can estimate whether M_k is a reachable marking during the running process of system or not. Reachability is decidable.

The nodes of the reachability tree are the markings of TCPN model and the arcs are the transitions. The algorithm of building Reachability Tree is:

a). The root node r is the initial marking M_0;

b). The node x marked by M is a leaf node

iff

$$\forall t \in T : \neg M[t > \text{ (i.e. no } t \text{ is enabled in marking M)}$$

or there is a node y between$r$ and $x$and $y \neq x$ and y is marked by M, too.

c). If node x marked by M is not a leaf node, $\forall t \in T$ enabled in marking M fires and builds a new node y marked by M'. A new arc marked by t is built from x to y.

d). M' can be calculated as following.

$$M'(p) = \begin{cases} M(p) - \Phi(t) & p \text{ is the pre - set of } t \\ M(p) + \Phi(t) & p \text{ is the post - set of } t \end{cases}$$

If there is a node z marked by M'' and $z \neq y$ from r to y and $M'' < M'$, the component j which results in $M''(p_j) < M'(p_j)$ is changed into ω.
e). Retune to step b and loop execution until $\forall t \in T : \neg M[t >$ or $\forall t \in T : \neg M[t > M'$.

The reachability tree of FIG.9 is enormous because of the complexity of the model. So we use the state space statistics produced by CPN Tools like FIG.6 to describe some basic information about the size of the reachability tree and standard behavior properties. For the model of FIG.9 there are 22671 nodes and 300203 arcs. The construction of the reachability tree took 706 seconds on a PC. The Scc Graph statistics is Strongly Connected Component Graph which is derived from the graph structure of the state space. Two nodes of reachability tree are in the same Scc if and only if they are mutually reachable, i.e., there exists a path in the reachability tree from the first node to the second node and vice versa. The structure of the Scc Graph can give useful information about the overall behavior of the model being analyzed. The result that there are fewer nodes in the Scc Graph than in the State Space shows that there exist cycles in the reachability tree.

(2) Liveness

Liveness properties in FIG.7 specify that there is a single dead marking which has the node number 17905. Dead marking is a marking in which no binding elements, namely transition, are enabled. The fact that there is only one dead marking shows the TCPN is possible to terminate the construction of finest level of the distributed tree with the correct result. FIG.7 specifies there is no dead transition, which means each transition in FIG.9 has the possibility to occur at least once, and no live transition, which means no transitions can be made enabled from the dead marking.

(3) Time

The performance model of FIG.9 is validated by comparing prediction results with measurement results for two problem sizes. We use ring and sphere which are common electromagnetic scatterer and the prediction and measurement results are shown in Table 2. The time information is accumulated after1000 cycles. n is the parameter of problem size. T_p shows the prediction results which are obtained by simulation and T_m shows the measurement results. The column that is indicated with T_s presents the simulation time. The time needed to execute real program on Ziqiang 3000 supercomputer is compared with the time needed to predict the performance model on a PC with T_m/T_s. It is observed that model-based performance evaluation was faster than the corresponding measurement-based evaluation. The parameter Error shows the prediction accuracy of the performance model.

4. Conclusions

This paper deals with performance oriented development of parallel programs and proposes a formal design of the application at the algorithm structure level. Considering that the performance of parallel systems is not only determined by the performance of the hardware, but also by the structure of the parallel program, a hybrid TCPN model proposed in this paper describes resource and parallel program respectively. The TCPN process templates along with a hierarchy concept are put forward to serve as functional and temporal specification formalism for parallel program development. And the hierarchical decomposability allows investigations on various levels of abstraction and naturally eliminates the problem of complexity of state space based analysis in TCPN models. The TCPN model not only characterizes the functions of the real system, but also contains the performance

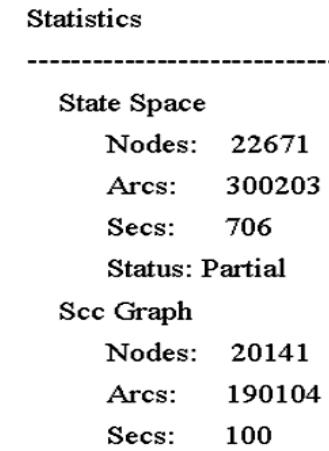

FIG. 6. Statistics of the state space report

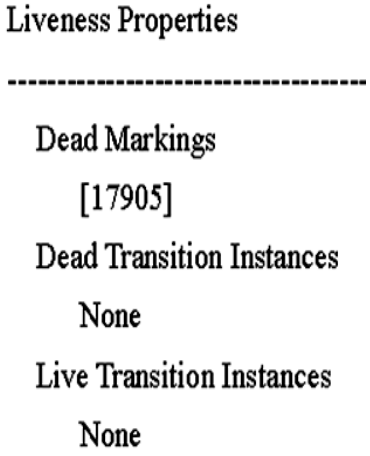

FIG. 7. Liveness properties of state space report of FIG.9

information. It is an executable model and construction such a model can lead to a more complete specification of the system design and make it possible to create a systematic investigation of scenarios which can significantly decrease the number of design errors. It makes it possible to analyze the performance of the system in the early phases, rather than in performance measurement of fully implemented applications.

References

[1] . Ferscha. Modelling Mappings of Parallel Computations onto Parallel Architectures with the PRM-Net Model. In C. Girault and M. Cosnard, editors, Proc. of the IFIP WG 10.3 Working Conf. on Decentralized Systems, 1990.

[2] . Balbo, G. Chiola, S.C. Bruell, and P. Chen. An Example of Modelling and Evaluation of a Concurrent Program using Coloured Stochastic Petri Nets: Lamport's Fast Mutual Exclusion Algorithm. IEEE Transactions on Parallel and Distributed Systems, 1992.

TABLE 1. Semantic of the transitions and places in FIG.8.

Place	**Semantic**
P_{01},P_{11},P_{21}	Entry of different processes
P_{02},P_{12},P_{22}	Status before communication
P_{03}	Ordered sampled data with pivot elements
P_{04},P_{13},P_{23}	Local data with pivot elements
P_{05},P_{14},P_{24}	Ordered local data
P_{a0},P_{a1},P_{a2}	Sending buffer of processes
P_{b0},P_{b1},P_{b2}	Receiving buffer of processes

Transition	**Semantic**
T_{01},T_{11},T_{21}	Calculate the *Morton Key* and quick sorting locally
T_{02}	Receive data from No.1 and No.2 processes
T_{12},T_{22}	Send data to No. 0 process
T_{03}	Broadcast the pivot elements to all processes
T_{04},T_{13},T_{23}	Divide the local data into 3 parts based on the pivot elements
$Tnet_i$	Communicate

TABLE 2. Prediction and Measurement Time of Ring and Sphere

Ring n=3221				
Nodes	Tp(s)	Tm(s)	Ts(s)	Error(%)
1	11.42	11.85	0.46	3.63
2	9.18	9.98	0.36	8.02
4	7.36	8.21	0.42	10.35
8	6.02	6.94	0.51	13.26

Sphere n=3072				
Nodes	Tp(s)	Tm(s)	Ts(s)	Error(%)
1	11.27	11.5	0.39	2
2	8.57	9.17	0.32	6.54
4	6.44	7.15	0.32	9.93
8	5.18	5.87	0.43	11.75

[3] . Ferscha and G. Kotsis. Optimum Interconnection Topologies for the Compute-Aggregate-Broadcast Operation on a Transputer Network. In Proceedings of the TRANSPUTER '92 Conference, 1992.
[4] . Ferscha. A Petri Net Approach for Performance Oriented Parallel Program Design. Journal of Parallel and Distributed Computing, 1992.
[5] lrich Herzog. Formal methods for performance evaluation. Springer Lectures On Formal Methods And Performance Analysis, 2002.

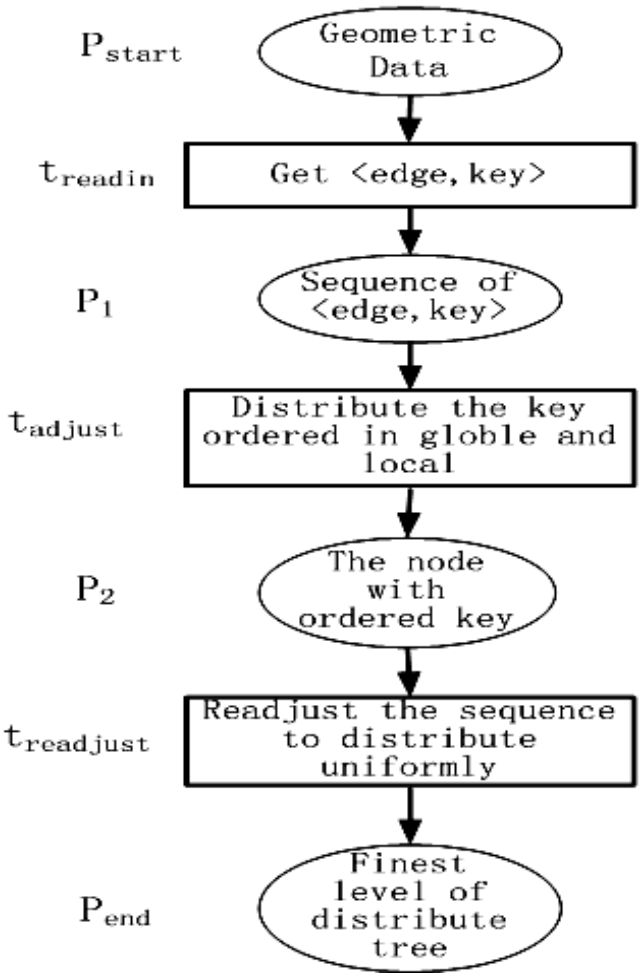

FIG. 8. Top model of the algorithm

[6] ouis Gesbert, Fr'ed'eric Loulergue. Semantics of an Exception Mechanism for Bulk Synchronous Parallel ML. Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2007.
[7] . Carter, W.B.Gardner. A Formal CSP Framework for Message-Passing HPC Programming. 2006 Canadian Conference on Electrical and Computer Engineering, CCECE'06, 2007.
[8] ornkhom, Panupong. Security Analysis of Micali's Fair Contract Signing Protocol by Using Coloured Petri Nets. Proc. 9th ACIS Int. Conf. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2008.
[9] A Min, LAN Jing-chuan, HUANG Jian-guo. Simulation and Design about Deadlock-free Problem in Parallel Test. Journal of System Simulation,2008.
[10] hen Yue, Meng Xiao-feng, Bian Ze-qiang. Model of parallel TPS management mechanism based on Petri net. Computer Engineering and Design,2008.
[11] ui Dan, Wang L isheng, Ye Qing. Petri model and Validation for MPI Program. Computer Applications and Software,2007.

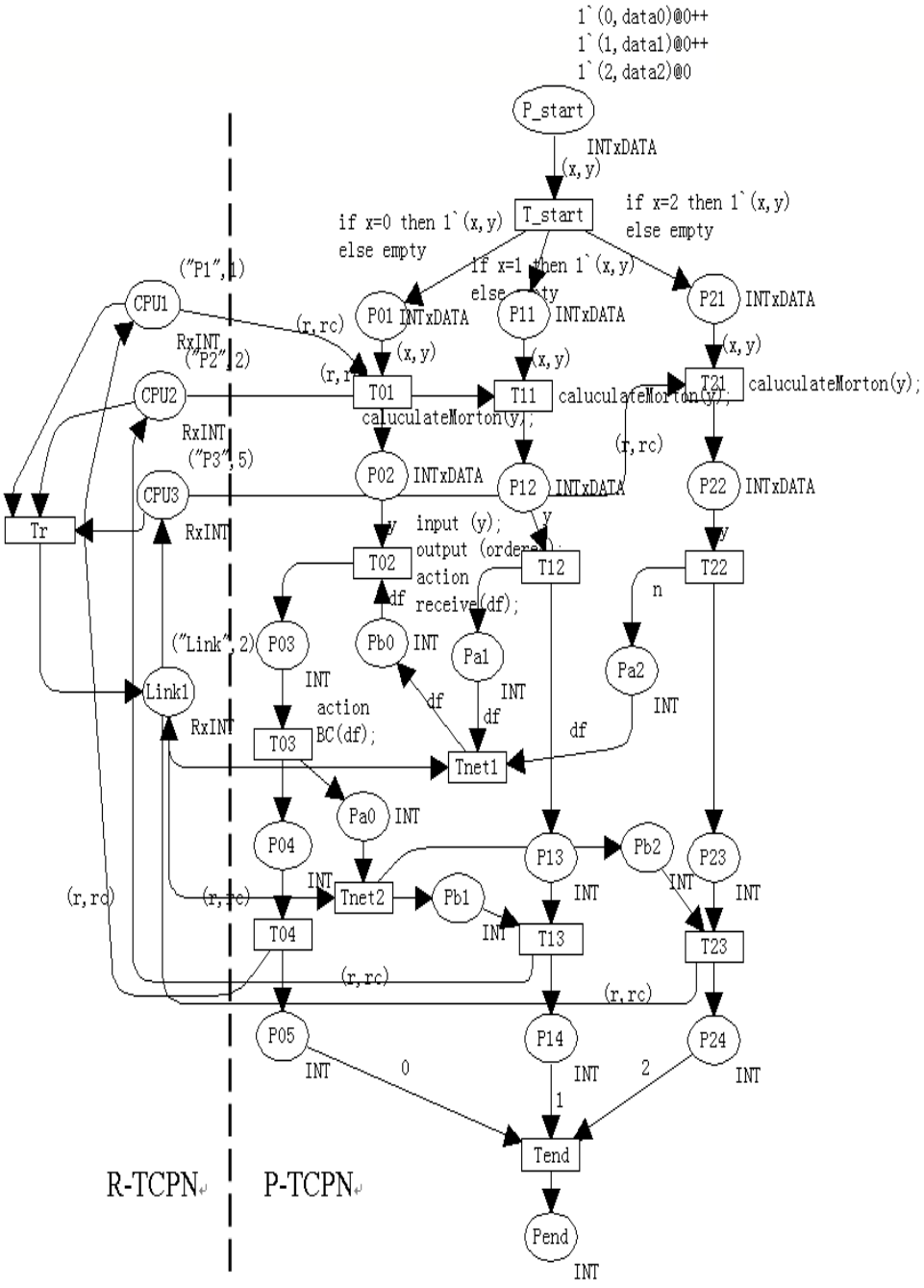

FIG. 9. Construction of the finest level of distributed tree based on TCPN

1. School of Computer Engineering and Science; Shanghai University;Shanghai 200072;China
2. College of Mathematics, Physics and Information Engineering;Zhejiang Normal University; Zhejiang Jinhua 321000; China
E-mail: cb@shu.edu.cn

School of Computer Engineering and Science; Shanghai University;Shanghai 200072;China
E-mail: wqtong@mail.shu.edu.cn and wxg@zjut.edu.cn