International Journal of Security and Its Applications
Vol.8, No.1 (2014), pp.45-56
http://dx.doi.org/10.14257/ijsia.2014.8.1.05

# A TPSAC Model and Its Application to Mechanical Cloud Simulation

Changyu Liu [1,3,*], Shoubin Dong [1], Huiling Li [2], Bin Lu [1] and Alex Hauptmann [3]

[1] *School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China*
[2] *State Key Laboratory of Pulp and Paper Engineering, South China University of Technology, Guangzhou 510640, China*
[3] *School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA*
[*] *chinoli@cs.cmu.edu, sbdong@scut.edu.cn, huiling_li@126.com, lbhqu@163.com, alex@cs.cmu.edu*

***Abstract***

*As a further development of the simulation grid, the cloud simulation platform is a new kind of network modeling as well as simulation platforms, and one of the hottest research directions in the cloud computing. While brings about the users with a lot of convenience, the cloud simulation shows also many severe security issues with its own characteristics, which can't be solved effectively by the traditional access control strategies. According to the traditional role based access control(RBAC) model, this paper proposed a tree proxy-based and service-oriented access control(TPSAC) model. In the TPSAC model, a multilevel inherited meta permission and a multi-tree child-sibling linked list were adopted to separate the permission loading function and the permission distribution function to achieve a multi-granularity and quantized access control with the cloud simulation. A verification experiment on the CloudSim simulation platform was conducted then to demonstrate that the TPSAC model achieved the desired result.*

***Keywords:*** *access control; cloud simulation; cloud computing; TPSAC; CloudSim*

## 1. Introduction

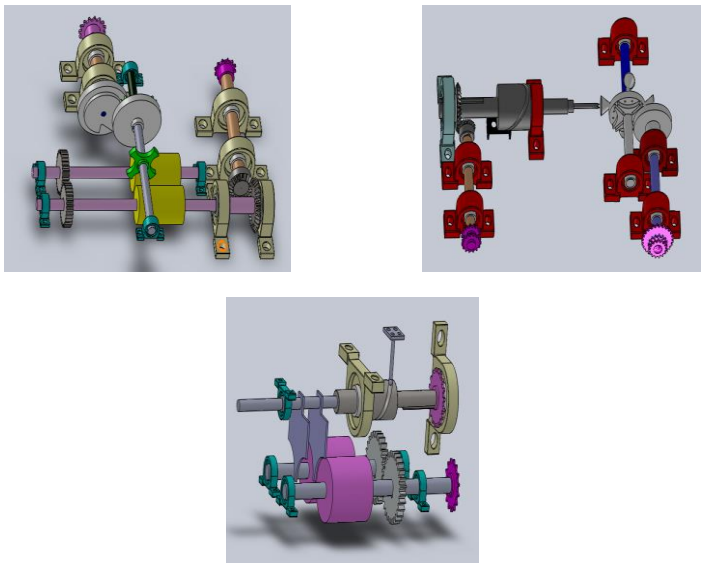
The access control technology came firstly as a security protection approach for the dataset in large computer systems in 1980s. In the following thirty years, a various of new access control strategies were proposed, of which the most widely used were discretionary access control(DAC), mandatory access control(MAC) and role-based access control(RBAC) [1]. Their fundamental objectives then were to prevent the illegal user from logging into the specific system and the legal user from using the system resource illegally. As a critical technology for achieving a safe system operation, the access control technology is a solution to the system safety and a key approach for guaranteeing the information confidentiality and integrity. The research on the access control is attracting an increasing attention in the field of computer science.

The cloud computing was a developed and commercial result of the distributed computing, the parallel computing and the grid computing [2]. By dividing a big computing task into several small ones, the cloud computing carried out them on a massive distributed computers instead that of on several local computers or remote servers. As to the secure cloud services, it was a important aspect for the cloud computing to consider how to carry out a security certificate as well as a access control management [3]. Some scholars proposed many correspondent methods of user authentication and access control according to its characteristics. Yassin *et. al.*, [4] put forward an authentication technology of the cloud computing based on a public key and a mobility.

ISSN: 1738-9976 IJSIA

Achieving a user identification [5] with the access control module in the Hadoop platform, Chang *et. al.*, [6] present a access control method for the cloud computing with the fingerprint as well as the face image. Abdulrahman *et. al.*, [7] bring a distributed access control architecture for cloud computing based on the synergy theory of the tight and loose coupling. Celesti *et. al.*, [8] gave a reference infrastructure for the identity management in the cloud computing. Lin *et. al.*, [9] described a BLP and Biba based access control model to distribute different users with a corresponding permission according to their behavior characteristics. However, these solutions all had some limitations on the user access control of a complex environment in the cloud computing.

As one of the hottest applications of the cloud computing, the cloud simulation was proposed currently as a new modeling as well as simulation mode and had attracted rapidly a wide attention from both industrial and academic communities. There are two types of new demands [10] in the development of the cloud simulation, which are: 1) Due to the expanding and increasingly complex scale as well as the structure, a distributed, heterogeneous, synergistic, interoperable and reusable simulation system is in an urgent need. 2) The people hope to be able to obtain smoothly the needed simulation service from the network at any time and any place. Based on the fact that the traditional access control strategy couldn't satisfy efficiently these emerging demands, this paper proposed firstly a tree proxy-based and service-oriented access control model (TPSAC) by transforming a Grid-HLA based Grid Simulation platform into a Cloud Simulation platform. Then combining with several scene models of the production line [11, 12] (See Figure 1) and the CloudSim cloud simulation platform [13], this paper gave also an experimental result.



(a)Material Separation Federate (b)Cracking Federate (c)Pitting Federate

**Figure 1. Physical Simulation Modeling**

## 2. The TPSAC Model

### 2.1. Basic Concepts

*Definition 1* (Permission Package Set, PPS). The permission package(PPS) is one of the basic concepts in our model. It is a dynamic permission attribution of the proxy-based tree AccBTree. The PPS is defined as $PPS=\{(TY, BP, PV) / BP \in \{Bitmap_i\}, PV \in \{Pv_j\}, TY\&Bitmap_i \in \{0,1\}, 1 \le i \le 32, 1 \le j \le 31\}$, where: (1) *TY* is a type identifier that was defined by a binary bit to indicate the package type. (2) *BP* is a permission bitmap that each bit

represents a meta permission except for $B_1$. If $B_1$ =0, this PPS package uses only the basic bitmap, otherwise the extended bitmap. If $B_i$ =1, it will activate the data field bit $P_{i-1}$ bitmap. (3) *PV* is a value sequence, which contains a weight and a lifetime, of the meta permission set that is in the same order as with that of the part 2 bitmap. In the value sequence, each field that with 8 bits, uses 2 condensed BCD codes to express the meta permission value. For example, $V_1$= $(0001\ 1111)_2$=$(1F)_{BCD}$.

*Definition 2* (Meta Permission Sets，*MPS*）Because the system resource *OBS* = *FO* ∪*DO*，where *FO* is the function object and *DO* is the data object, the total meta permission set *MPS* = {（*obj*，*op*，*i*）/*obj* ∈*OBS*，*op* ∈*OPS*，*i* ∈*N*}，where *MetaPerm* =（*obj*，*op*，*i*） is a meta permission and a elementary unit during authorization, *OPS* is the operation set, and *I* is the lifetime or the weight that reflects the credibility and reliability of its role. As already involved the information of *op* and *i* in the PPS package, the meta permission set can be also expressed as, *MPS* = {（*obj*，*pps*）/*obj* ∈*OBS*，*pps* ∈*PPS*}。After getting rid of the overlapping permission, the total valid permission set *P* = {*p* / ∀$p_i$ ∈*MPS*∧∀$p_j$ ∈*MPS*∧*i*≠*j*→$p_i$∩$p_j$= Ø }。

*Definition 3* (Proxy-based Tree Set，*T*) The proxy-based tree *AccBTree* ,which encloses the role based meta permission set , is a dynamic object that generated when a role was activated by a user. It can be descript by a quadruple< *R*, *PPS*，*ST*，*Lifetime*>, where *R* is the activated role set, *ST* is the tree status set, *Lifetime* is the tree lifetime.

*Definition 4* (Function *load*，*destroy*，*state*). Let *D*= {*t* / *t* ∈*T* }. Then, define *load (t)* as the tree loading time, *destroy (t)* as the tree destroying time with *Lifetime (t)= destroy (t) - load (t)*, and *state (t)* ∈*{0,1,2,3}*as four tree states which represent separately the ready, running, suspended and the final state. Then, it meets the following rules according to the principle of least privilege.

*Rule 1* ∀*t*(*t* ∈*D*∧(*load (t)*=0)∧(*load (t) = destory (t)*)→(*state(t)*=0)
*Rule 2* ∀*t*(*t* ∈*D*∧(*load (t)*=0)∧(*load (t) != destory (t)*)→(*state(t)*=1)
*Rule 3* ∀*t*(*t* ∈*D*∧(*load (t)*>0)∧(*load (t) = destory (t)*)→(*state(t)*=3))
*Rule 4* ∀*t*(*t* ∈*D*∧(*load (t)*>0)∧(*load (t) != destory (t)*)→((*state(t)*=1) ∨(*state(t)*=2))

*Definition 5* (*Service*，S). The service is a important concept in the SOA architecture. It is a kind logic unit that is able to provide a specific and reusable business function to some other application systems. Loose coupling is one of the main service attributes.

*Definition 6* (*User Assignment*). Let *D*= {*u* / *u* ∈*U* }, *E*= { *r* / *r* ∈*R* }. Then define *UA*= {(*u, r*) } as a *User Assignment* relationship on *D*->*E*.

*Definition 7* (*Permission Assignment*). Let *D*= { *p* / *p* ∈*MPS* }, *E*= { *r* / *r* ∈*R* }. Then define *PA* = {(*r, p*) } as a *Permission Assignment* relationship on *D*->*E*. Because *MPS* = {（*obj*，*pps*）/*obj* ∈*OBS*，*pps* ∈*PPS*}, *PA* ={（*r*, *obj*，*pps*）| *r* ∈*R*，*obj* ∈*OBS*，*pps* ∈*PPS* } and *OBS* = *FO* ∪*DO*.

*Definition 8* (*Tree Assignment*). Let *D*= { *t* / *t* ∈*T* },*E*= { *s* / *s* ∈*S* }. Then define *TA* = {(*t, s*) } as a *Tree Assignment* relationship on *D*->*E*.

*Definition 9* (*GList*). Generalized List (*GList*) is a kind of popularized linear list. Let *LS*=（$\alpha_1$,$\alpha_2$,...$\alpha_n$）, where *LS* is the name of the generalized list, *n* is the list length, $\alpha_i$ is the individual element with lowercase letters or the sub generalized list with uppercase letters. For example, let *D=(a,C)* and *C=(b,(c,d,e))* be two generalized list， then *D=(a,(b,(c,d,e)))*.

### 2.2. The Model Definition

In order to satisfy fully the integrity constraints of task flows and enhance system security, the user permission is not only subject to the performed role but also subject to

the actual task status. When a role is activated by one user, it activates accordingly a dynamic and temporary proxy-based tree *AccBTree*. As a atomic function in SOA, the *AccBTree* could be created parametrically when executing a task and be released when terminating a task. Thus, the system could recycle the previous assigned user permission. Figure 2 shows a frame of the *TPSAC* model.

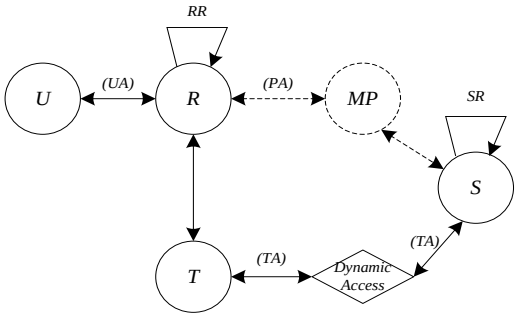



**Figure 2. TPSAC Model**

*Definition 10* (TPSAC Model). *TPSAC=<U,R,MP,T,S>*,where U is the user set, R is the role set, MP is the meta permission set, T is the Proxy-based tree set, S is the service set, UA is the *User Assignment*, PA is the *Permission Assignment* and TA is the *Tree Assignment*. Then RR is the compound and inherited role relation, and SR is the constrained service relation.

The process when a role obtains a permission service is also the one when a proxy-based tree is spanned with its status changing. The status and the constraint condition of this tree determine whether a service is allowed. There are three phases in TPSAC, which are a global static permission assignment, a proxy-based tree spanning and a dynamic permission adjustment.

(1) Global static permission assignment. The main function of this stage contains a role distribution for users and a predefinition of role permission. The distribution function Perm_Assign(r) is used to correlate UA and PA.

(2) Proxy-based tree spanning. In TPSAC model, there in no immediate relation between the role and the permission. When a user activates a role, the system generates a instance of AccBTree. This function is defined as Create_Tree(r, s).

(3) Dynamic permission adjustment. This stage includes a change to the meta permission by Change_Meta_Value(Vi, Value) and to the tree status by Chang_Tree_State(stValue). Let MT（u, p）denote that u has the related right to carry out p, then its necessary condition for being performed is MT(u, $p) \rightarrow \exists r(u \in UA(r) \cap p \in PA(r) \cap t \in T \cap state(t)=1 \cap (setValue<v_i))$.

### 2.3. The Model Advantages

Compared with the service oriented access control model of paper [14], the TPSAC model has the following advantages, as: (1) Use a proxy-based service tree to support a multilevel inherited role. In the TPSAC model, the role related meta permission was encapsulated in the PPS package of a proxy-based binary tree. So, the role inheritance corresponds to that of tree. (2) Use a meta permission to support a quantized access control. In the practical application, the permission has not only the attribute of quality, but that of quality. For example, as to the presidential election in the political field, the Constitution stipulates clearly that every citizen aged eighteen or over has the right to vote, but the election is only valid when there are more than half of them involved in this voting. The present of meta permission cloud support well this quantized access control. (3) Use a

compressed PPS package to support a multilevel secure access control. Because there is one PPS package that has 1+32+8*31=281 bits（71bits for BCD code）for 31 predefined meta permissions of each role in the TPSAC model, the compression degree is very high. Under the premise of supporting the principle of least privilege and the principle of separation of duties, these encapsulated and packaged permissions could enhance a better system security.

## 3 The TPSAC based Cloud Simulation

### 3.1. The Simulation Frame

The High Level Architecture(HLA) was proposed by the United States Department of Defense [15]. It is a technical specification for the distributed simulation system and can be split into a HLA regulation, a interface specification and a object model template. In HLA, a federation was consisted of simulation systems(federates), supported software and related components [16]. There are three levels in the web service of HLA, which are the federate level, the RTI level and the federate/RTI level [17]. Due to the difficulty in achieving a web service based RTI, this TPSAC system gives an incomplete expansion on RTI level by introducing a grid service. The extended service includes a GC-RTI Grid service and a GFM-RTI Handle service. Then the user federate can go through firewall by calling the grid based CRC service. The permission service contains a authorization service from the *AuthorityDistr SYS* and a authentication service from the *TAccessBTree*. Figure 3 shows a TPSAC and Grid-HLA based cloud simulation frame.

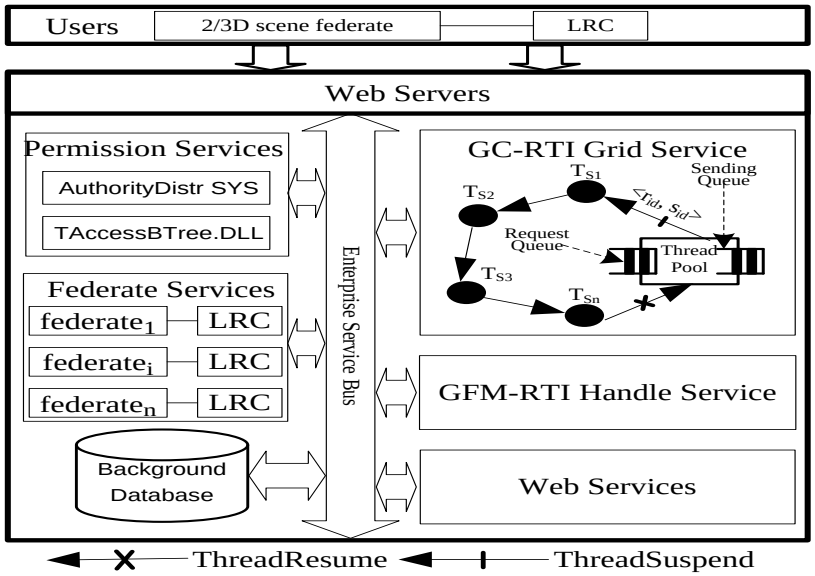
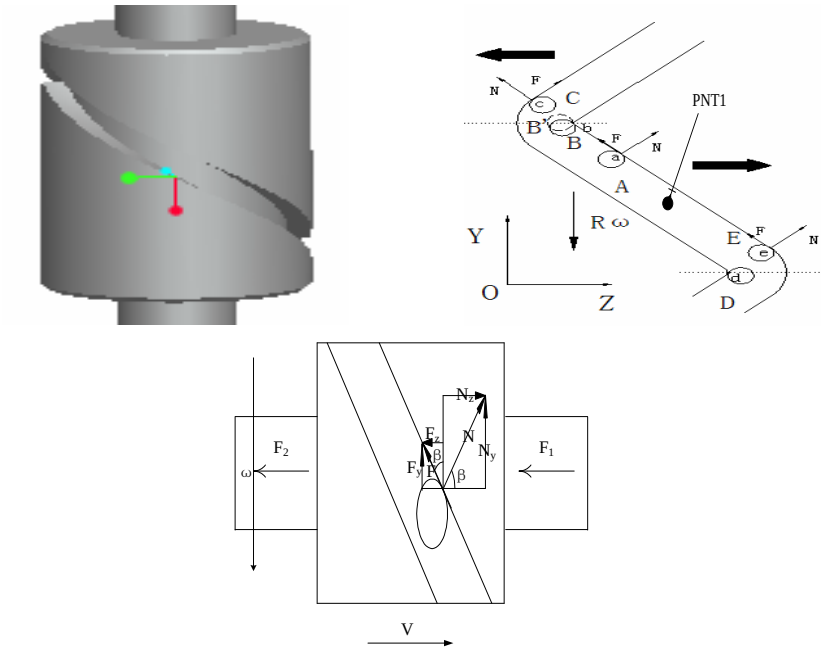



**Figure 3. The *TPSAC* and *Grid-HLA* based Cloud Simulation Frame**

### 3.2. Scene Modeling

Scene modeling [11] includes physical as well as behavioral modeling of federate. All federates in *JPL* and *FPL* are objects of physical modeling. Because of the limited space, this paper describes only scene modeling process of cracking and pitting machine. Three major physical mechanisms of cracking and pitting machine are material division, cracking and pitting. As cam mechanism is the core component of cracking and pitting machine, this paper gives only its behavioral modeling process, of which includes kinematics modeling and dynamic modeling, as shown in Figure 4, where figure *a* is the cylindrical camshaft to be analyzed, figure *b* is the kinematics model that gained after the

expansion for cylindrical camshaft in coordinate plane *YOZ*, and figure *c* is the dynamic model for cam while doing a uniform motion on *Z* axis.




(a)Cylindrical Camshaft (b)Cam Kinematics Model (c)Cam Dynamic Model

**Figure 4. Behavioral Modeling of Cracking and Pitting Machine**

In Figure *b*, when camshaft moves in opposite *Y* direction, the push rod moves in *ABCDE* sequence. Let *r* be the distance between contact point and haft, *R* be the haft radius, $\omega$ be the haft rotating speed, and *t* be the time. Then the rate equation and acceleration equation of any point *m* on camshaft can be expressed as follows:

$$\begin{cases} v_x = x' = -r\omega \sin\omega t \\ v_y = y' = r\omega \cos\omega t \\ v_z = z' = R\omega (m \in EABorCD) \\ \quad = R\omega + at (m \in BCorDE) \end{cases} \quad (1) \qquad \begin{cases} a_x = x'' = -r\omega^2 \cos\omega t \\ a_y = y'' = -r\omega^2 \sin\omega t \\ a_z = z'' = 0 (m \in EABorCD) \\ \quad = a (m \in BCorDE) \end{cases} \quad (2)$$

Where: $v_m = \sqrt{v_x^2 + v_y^2 + v_z^2}$, $a_m = \sqrt{a_x^2 + a_y^2 + a_z^2}$.

In Figure *c*, let $f_1$ be the friction coefficient between push rod and haft, $f_2$ be the friction coefficient between cam bond and bevel gear, *G* be the gravity of haft, $N_p$ be the normal pressure on the cam bond that generated by bevel gear, *L* be the distance between action point of $N_p$ and haft center, *N* be the supports reaction between push rod and haft. Then, some equations that gained directly from this figure can be expressed as follows:

$$\begin{cases} N\cos\beta - F_z = 0 \\ F_z = F_1 + F_2 + Nf_1 \sin\beta \\ F_1 = \sqrt{N^2(\cos\beta + f_1 \sin\beta)^2 + G^2}(f_1 + f_2)/2 \\ F_2 = 2N_P f_2 \end{cases} \quad (3)$$

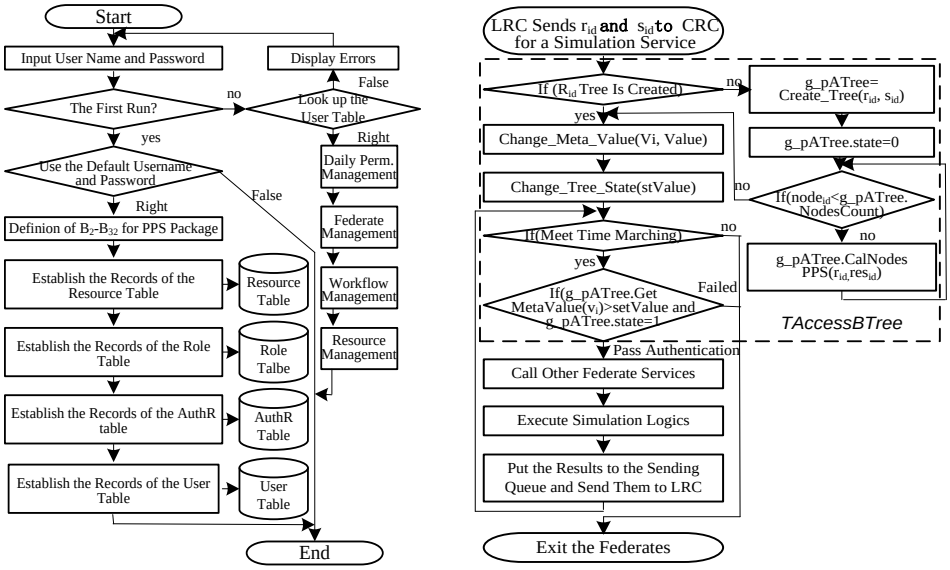
Then *N* can be obtained from equations (3), as follows:

$$N = \frac{F_1 + F_2}{\cos\beta - f_1 \sin\beta} = \frac{\frac{f_1 + f_2}{2}\sqrt{N^2(\cos\beta + f_1 \sin\beta)^2 + G^2} + \frac{f_2}{L}[N(\sin\beta + f_1 \cos\beta)R - 2N_P L]}{\cos\beta - f_1 \sin\beta} \quad (4)$$

### 3.3. The Simulation Process

Due to the limit space, this paper provides only the implementation procedure of the permission service in Figure 3, omitting that of federates, LRC and CRC. The implementation procedure includes a global static authorization service that provided by a *AuthorityDistr SYS*, and a dynamic authorization as well as authentication service by a *TAccessBTree* that created with function *Create_Tree($r_{id}$, $s_{id}$)*. The concrete implementation processes of these two services are as shown in Figure 5, where the proxy-based service tree TAccessBTree is in the dashed box.




(a)Global Static Authorization (b)Dynamic Authorization and Authentication

**Figure 5. The Simulation Process of the TPSAC Model**

### 3.4. The Simulation Algorithm

The *TPSAC* based service tree is actually a resource tree with permission information. So the *TPSAC* tree is a kind of multi-way tree. In order to locate and create conveniently a tree, this paper adopts a child-sibling list to transform the multi-way tree to a binary permission tree.

(1)Base Class Definition

*class TAccessBTree {*

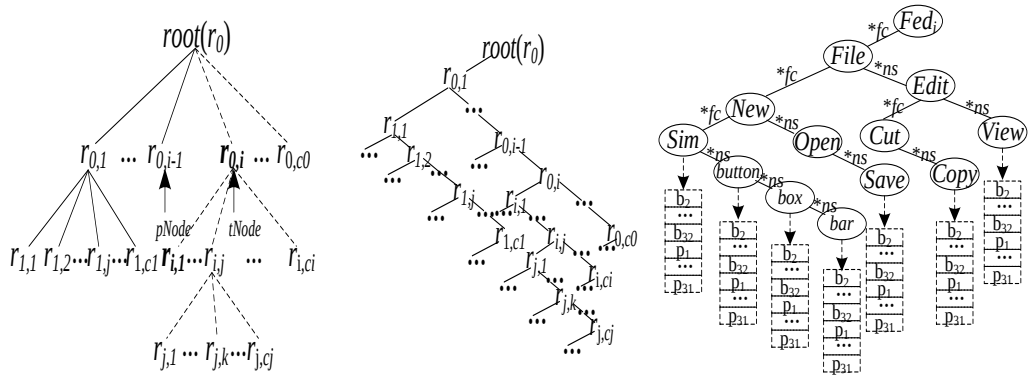
*private: static TAccessBTree *m_pBTree; int m_rID; TNode *m_pRoot; ...*

*public: static TAccessBTree* getInstance(int, char*); bool LocateRes (int, PPS&*); ... };*

*class TNode { public: int id; PPS* pps; TNode *firstchild; TNode *nextsibling; ... };*

(2)The Spanning Tree Algorithm

A depth-first traversal algorithm was adopted recursively in the tree spanning function *TAccessBTree::CreateTree(TNode *root)* , as shown in figure 6. It can be seen from the figure that there were two steps in the spanning process of a binary permission tree(See Algorithm 1), which were the creation of a multi-way permission tree (See Figure 6a) and the transform from a multi-way tree to a binary tree(See Figure 6b).




(a) Multi-way Tree Creation (b) Transform to a Binary Tree (c)The Final Binary Tree

**Figure 6. The Spanning Process of a Binary TPSAC Tree**

Figure a shows a nodes relation in the depth-first traversal of a multi-way permission tree, where the dotted line indicates nodes that has not been generated yet, *tNode* denotes the nodes that will be generated soon, and *pNode* is a left neighbor brother of *tNode.* A further analysis can show that the nodes creation process must meet a constraint of the following rule 5. Figure b shows a transform that must meet a constraint of the following rule 6, from a multi-way tree to a child-sibling list binary tree. Figure c gives a final binary permission tree of one federate with this spanning algorithm, where **fc* means **firstchild*, **ns* means **nextsibling*, $b_2$-$b_{32}$ and $p_1$-$p_{31}$ are bits of PPS package.

*Rule 5* Let *root* be the resource tree or its root node, $r_i$ be the i-th nodes of *root* or a record from table *Resource*. Then, the substructure of *root* can be expressed as ($r_1$，$r_2$... $r_n$) = { $r_i$ | Parent_ID($r_i$)=ID(*Root*) }. Likewise, ($s_1$，$s_2$... $s_m$) = { $s_i$ | Parent_ID($s_i$)=ID($r_i$) }.

*Rule 6* Let $T_{ij} \in GList$ be a tree and $r_{ij}$ be its root node. Then, the former multi-way tree can be represented by a generalized list as $T_m$= ($r_0$, ($T_{0,1}$,$T_{0,2}$,…,$T_{0,C0}$)), where $T_{i,j}$ = ($r_{i,j}$, ($T_{j,1}$,$T_{j,2}$,…,$T_{j,cj}$))，$0 \le i \le c_0$，$1 \le j \le c_i$. And the corresponding binary tree can be represented as $T_b$ = ($r_0$, $T_{0,1}$), where $T_{i,j}$ = ($r_{i,j}$, ($T_{j,1}$,$T_{i,j+1}$) )，$0 \le i \le c_0$，$1 \le j \le c_i$. Between them, there is a conversion rule *T_Rule*：$T_m$->$T_b$.

*Algorithm 1 The Spanning Binary Permission Tree*

```
1 CreateTree(TNode *root)  ◁
2 TNode *pNode, *tNode;
3 m_pAdo：(r1， r2... rn) = { ri ｜Parent_ID(ri)=ID(Root) };
4 // Use m_pAdo to get the substructure of node root.
5 if (m_pAdo->RecordCount > 0){
6     m_pAdo->First();
7     for (int i=0; i<m_pAdo->RecordCount; ++i){
8        tNode = new TNode();
9        tNode->id = RES_ID(m_pAdo);
10       tNode->pps = PPS(m_pAdo);
11       /****T_Rule：Tm->Tb****/
12       if (i == 0)   root->firstchild = tNode;
13       else   pNode->nextsibling = tNode;
14       pNode = tNode;
15       CreateTree(tNode);   // Recursive Invocation
16     /****T_Rule：Tm->Tb****/
17       m_pAdo->Next();
18       }}  ▷
```

## 4. Experiment and Analysis

In the experiment part, this paper adopted a widely used cloud simulation platform CloudSim [13], which was developed by the University of Melbourne. In order to make the experimental environment to be close to the real cloud computing scene, we firstly introduced 13 federate models from paper [11]. Then, we set up a service interactive scene TPSAC Simulation in the CloudSim. After that, we configured the corresponding Service Provider(SP) and Service Consumer(SC) as SP={SCUT_CCNL_Srv1, CMU_CQ_Srv1, CMU_CQ_Srv2}, SC={USER1, USER2, USER3, USER4, USER5}. The other experiment configurations were similar as [18]. Figure 7 showed a resource utilization of Provider CMU_CQ_Srv1.

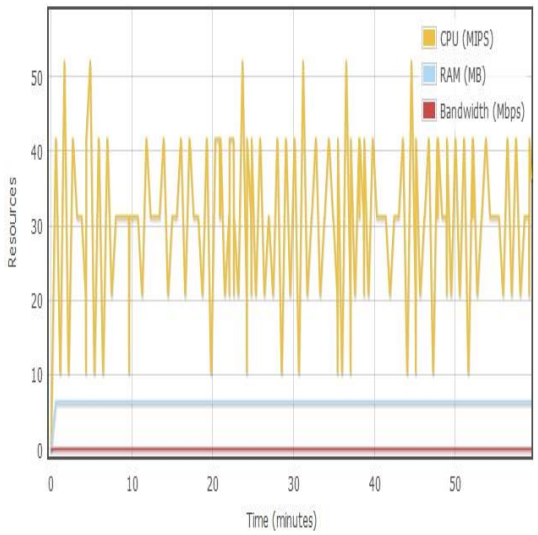
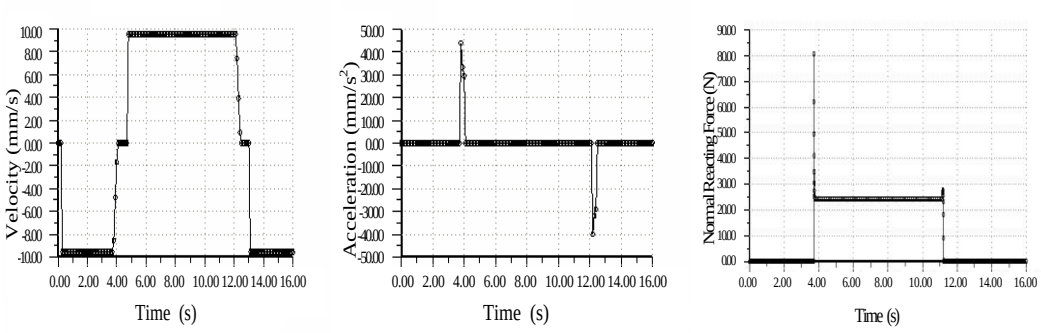



**Figure 7. The Resource Utilization of SP on CloudSim**

Kinematics simulation analysis can be performed after introducing the TPSAC model and several scene models to the CloudSim simulation platform. In fact, the federate simulation process is also the process of kinematics as well as dynamic parameter calculating. Because of the limited space, this paper gives only the simulation results of former cam behavioral model, of which include velocity, acceleration and supports reaction, as shown in figure 8. A conclusion that the whole cam design can meets the parameter requirement of $fed_{cpm}$ and our TPSAC model is more suitable for the cloud simulation can be obtained after a comparison with the result in [18].




(a)Velocity of *PNT1* Point (b)Acceleration of *PNT1* Point (c)Supports Reaction

**Figure 8. Kinematics Simulation Results of a TPSAC Federate**

## 5. Conclusion

This paper proposed and realized a tree proxy-based and service-oriented access control (TPSAC) model from the aspects of the simulation frame, the simulation process and the simulation algorithm. Currently, the TPSAC model was applied to several actual lab projects, such as the simulation scene from the virtual factory and that from the virtual campus roaming, and showed a preferable result. By introducing a quantized and dynamic binary permission tree, the TPSAC model made it more flexible to load and manage the access control process, meeting requirements from the more complex practical cloud simulation applications.

## Acknowledgment

This paper was supported by the National Natural Science Foundation of China under Grant No. 61070092. The further parallel computing part was performed on the Blacklight at the Pittsburgh Supercomputing Center (PSC). We would like to thank PSC for providing the computing resource.

## References

[1] C. H. Fang, X. Z. Ye, W. Peng and Zhang, "Multi-Level and dynamic security access control for CAD models in collaborative environment", Softw., vol. 18, no. 9, **(2007)**, pp. 2295-2305.
[2] H. Wu, J. Yu and F. R. Yu, "Dynamic access control algorithm based on trust model in cloud computing", Comput. Eng. Appl., vol. 48, no. 23, **(2012)**, pp. 102-106.
[3] H. Takabi, J. B. D. Joshi and G. J. Ahn, "Security and Privacy Challenges in Cloud Computing Environments", IEEE Secur. Priv., vol. 8, no. 6, **(2010)**, pp. 24-31.
[4] A. A. Yassin, H. Jin, A. Ibrahim, W. Z. Qiang and D. Q. Zou, "Efficient Password-based Two Factors Authentication in Cloud Computing", Intl. J. Secur. Appl., vol. 6, no. 2, **(2012)**, pp. 143-148.
[5] A. J. Choudhury, P. Kumar, M. Sain, H. Lim and H. J. Lee, "A Strong User Authentication Framework for Cloud Computing", 2011 IEEE Asia-Pac. Serv. Comput. Conf. (APSCC), (2011), pp. 110-115.
[6] B. R. Chang, H. F. Tsai, C. M. Chen and C. F. Huang, "Access Control of Cloud Computing Using Rapid Face and Fingerprint Identification", 2011 2rd Intl. Conf. Innov. Bio-inspired Comput. Appl. (IBICA), **(2011)**, pp. 179-182.
[7] A. Almutairi, M. Sarfraz, S. Basalamah, W. G. Aref and A. Ghafoor, "A Distributed Access Control Architecture for Cloud Computing", IEEE Softw., vol. 29, no. 2, **(2012)**, pp. 36-44.
[8] A. Celesti, F. Tusa, M. Villari and A. Puliafito, "Security and Cloud Computing: Inter Cloud Identity Management Infrastructure", 19th IEEE Intl. W. Enabling Tech.: Infrastruct. Collab. Enterp. (WETICE), **(2010)**, pp. 263-265.
[9] G. Y. Lin, S. He, H. Huang, J. Y. Wu and W, Chen, "Access control security model based on behavior in cloud computing environment", J. Commun., vol. 33, no. 3, **(2012)**, pp. 59-66.
[10] B. H. Li, X. D. Chai and B. C. Hou, "Networked Modeling & Simulation Platform Based on Concept of Cloud Computing—Cloud Simulation Platform", J. Syst. Simul., vol. 21, no. 17, **(2009)**, pp. 5292-5299.
[11] C. Y. Liu, H. J. Wang, J. X. Chen and X. J. Zou, "Research on Service-Oriented and HLA-Based Simulation Model of Juice Production Line", Intl. Conf. Measuring Tech.&Mech. Automat. (ICMTMA), vol. 3, **(2010)**, pp. 167-170.
[12] H. J. Wang, X. J. Zou, C. Y. Liu, J. Lu and T. H. Liu, "Study on Behavior Simulation for Picking Manipulator in Virtual Environment Based on Binocular Stereo Vision", 7th Intl. Conf. Sys. Simulat. & Sci. Comput. (ICSC), pp. 27-31, **(2008)**.
[13] R. N. Calheiros, R. Ranjan and A. Beloglazov, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Softw. Pract. Exper., vol. 41, **(2011)**, pp. 23-50.
[14] F. Xu, H. G. Lai, H. Huang and L. Xie, "Service-Oriented Role-Based Access Control", Chinese J. Comput., vol. 28, no. 4, **(2005)**, pp. 686-693.
[15] Y. Zhou and J. W. Dai, "HLA simulation programming", Beijing: Publ. House Electro. Ind., **(2002)**.
[16] D. Guan and Z. X. Cai, "Service Provisioning for HLA-based Simulation on Grid", J. Syst. Simulat., vol. 20, no. 3, **(2008)**, pp. 635-642.
[17] W. Zhang, T. Zhang and Y. B. Zha, "Web Service Enabling of HLA-based Distributed Simulation", J. Natl. Univ. Defen. Tech., vol. 30, no. 5, **(2008)**, pp. 120-124.
[18] C. H. Hu, X. H. Chen, M. Wu and J. X. Liu, "A service trust negotiation and access control strategy based on SLA in cloud computing", Sci. China Inform. Sci., vol. 42, no. 3, pp. 314-332, 2012.
[19] B. Lu, C. Y. Liu and Y. H. Wang," Discovery of Community Structure in Complex Networks Based on Resistance Distance and Center Nodes", J. Comput. Inf. Syst., vol. 8, no. 23, pp. 9807-9814, **(2012)**.

[20] J. X. Chen, H. J. Wang and C. Y. Liu, "Modeling and Performance Analyzing of Helix Transmission Base on Modelica", Key Eng. Mater., vol. 455, **(2011)**, pp. 511-515.
[21] H. J. Wang, J. X. Chen, X. J. Zou and C. Y. Liu, "Structure design and multi-domain modeling for a picking banana manipulator", Adv. Mater. Res., vol. 97-101, **(2010)**, pp. 3560-3564.
[22] H. J. Wang, X. J. Zou, C. Y. Liu, T. H. Liu and J. X. Chen, "Study on a Location Method for Bio-objects in Virtual Environment Based on Neural Network and Fuzzy Reasoning", 2rd Intl. Conf. Intell. Robot. & Appl. (ICIRA), vol. 5928, **(2009)**, pp. 1004-1012.

## Authors

**Changyu Liu**, joined the Communication and Computer Network Lab of Guangdong as a PhD student in 2010 at South China University of Technology, advised by Prof. Shoubin Dong. Then, he joined the School of Computer Science at Carnegie Mellon University as a Visiting Scholar in September 2012 to work with his advisor Dr. Alexander G. Hauptmann. His research interests include Computer Vision and Multimedia Analysis.

**Shoubin Dong**, is currently a professor in School of Computer Science and Engineering at South China University of Technology. Her main research areas include high performance computing and information retrieval. She received her Ph.D. degree in 1994 from University of Science and Technology of China. She has been a visiting scholar of Carnegie Mellon University during 2001-2002. She is now the deputy director of Communication and Computer Network Lab of Guangdong.

**Huiling Li**, is currently a PhD candidate and did a master stay (2011-2013) in the State Key Laboratory of Pulp and Paper Engineering at South China University of Technology, under the guidance of Prof. Runcang Sun. The main focus of her work is the application of solid acid catalysts in the conversion of biomass into energy and high value-added chemicals.

**Bin Lu**, received his M.S. degree in Computer Science from Huaqiao University in 2005. Currently, he is a PhD candidate in School of Computer Science and Engineering at South China University of Technology, advised by Prof. Wenjun Xiao. His research interests are Complex Network and Machine Learning.

**Alex Hauptmann**, is currently a Principal Systems Scientist in the School of Computer Science at Carnegie Mellon University. He has been the leader of the Informedia Digital Library and a founder of the international advisory committee for TRECVID. He received a BA and a MA in Psychology in 1982 from the Johns Hopkins University. He received a PhD in Computer Science from the Carnegie Mellon University in 1991. His research interests are in multimedia analysis and indexing, speech recognition, interfaces to multimedia system and language in general.