Autonomic Workflow Management in the Grid

Guangsheng Zhang[1, 2], Changjun Jiang[1], Jing Sha[1], and Ping Sun[1]

[1] College of Electronic & Information Engineering Tongji University
201804 Shanghai, P.R. China
Zhanggstide@hotmail.com
[2] The Public Security Bureau of Jinan 250001 Jinan, Shandong, P.R. China

Abstract. The autonomic workflow engine enables workflow to be dynamically specified and adapted using Event-Condition-Action rules. A new approach to autonomic execution of workflow processes based on matrix-based formulation was adopted that, together with the Petri net marking transition equation, provides a full dynamical description of a workflow system. We have built a workflow that models Traffic-information-grid-based Mobile Navigation Service to illustrate the quality of self-management using DEC through awareness of the computational environment.

Keywords: Grid, Workflow, Petri net, Discrete-event controller.

1 Introduction

In order to support complex scientific experiments and applications, grid services including computational devices, data, applications, and scientific instruments need to be orchestrated and composed while managing the application workflow operations within Grid environments. Thus grid workflow faces many uncertain factors such as unavailability, incomplete information and local policy changes. The complexity of workflow management in a grid context has sparked a great deal of interest in autonomic systems designed to run on the grid platform.

Autonomic computing [1] is touted as the means to realizing grid workflow systems capable of managing themselves with minimum human intervention. Bae J et al [2] propose a new approach to the automatic execution of business processes using ECA rules that can be automatically triggered by an active database. [3]presents a novel approach of using ECA rules to realize the workflow modeling and implementation for service composition. However, there are still many challenges that must be addressed by the research community. At runtime, as the execution proceeds, new components might be added to the application workflow and/or old components might be deleted. How can we dynamically add, delete, and change the algorithm used to implement each component at runtime?

Another issue in grid workflow is the balance between centralized and decentralized control.There are already many distributed business workflow management systems in the market. In [4], the Grid Workflow Execution Services is the dynamic and interactive execution of distributed workflows which coordinates the

Y. Shi et al. (Eds.): ICCS 2007, Part III, LNCS 4489, pp. 220–227, 2007.

composition and execution process of grid workflows. As far as we know, most workflow systems employ a centralized structure (see GridAnt and Condor). Dynamic SWMS [5] includes a human user in the runtime loop of a flow execution, and allows an engine to flexibly orchestrate a workflow according to the human decision and the runtime states of the environment. Some significant results in a supervisory control have also been obtained using Petri nets(PN) [6]. However, PN based workflow is hard to model uncertainty. There is a lack of supervisory-control techniques that can dynamically sequence different processes in the Autonomic Workflow Engine(AWE) according to the scenario and reformulate the process if some of the grid services fail.

In this paper, we propose the use of the matrix-based DEC as a central planner to produce high level missions for AWE. The DEC allows for tailored combinations of centralized and decentralized control by using the properties of block-matrix structures. This paper presents the AWE that enables self-managing Grid applications. It extends the service-based grid computing paradigm to relax static (defined at the time of instantiation) application requirements and system/application behaviors and allows them to be dynamically specified via ECA rules. **A** new approach to autonomic execution of workflow processes based on matrix-based formulation was adopted that, together with the PN marking transition equation, provides a full dynamical description of a grid workflow system.

The rest of the paper is organized as follows. Section 2 describes the design and implementation of the AWE architecture. Section 3 illustrates self-managing behaviors enabled by DEC using Traffic-information-grid-based Mobile Navigation Service(TigMNS). Section 4 presents a conclusion.

2 Autonomic Workflow Engine Architecture (AWEA)

The AWEA focuses on setting up the application execution environment and then maintaining its requirements at runtime. As shown in Fig.1, the AWEA main modules include Workflow Runtime Manager, Autonomic Service Component(ASC) and Policy engine. The users can specify the requirements using an Application Strategies Editor. Each application(or mission) can be expressed in terms of the Policy engine which creates and manipulates policies. These policies are stored in the policy repository so they can be reused if the user comes back later in order to avoid renegotiation and redefinition of previously generated policies.

The Workflow Runtime Manager can be further divided into several parts: Online Monitoring and Analysis(OMA), DEC, Event server and Autonomic Execution. The OMA performs online *monitoring* to collect the status and state information using ASC sensors and *analyzes* ASC behaviors and detects any anomalies or state changes [1]. The DEC generates the appropriate control and management *plans*. The plan generation process involves triggering the appropriate rule from the list of rules (for the appropriate service) in the knowledge repository.

An ASC is the fundamental service building block .It is a simple computational cell(service agent) with encapsulated rules, constraints and mechanisms for self-management and dynamic interactions with other cells. It also extends autonomic cell defined in [1].The ASC implements two interfaces for importing/exporting the functionalities of the services component (functional port),sensing and changing the runtime state of the component(sensor port).For example in TigMNS, map service(S_2)

is a agent which produces a desired background digital map on a geographical data query formula. On receipt of a relevant event from sensor port, S_2 will be able to produce and listen for specific event types and content via local policy database to proper manage its sub-workflow processes such as data prefetching increase the display speed and to eliminate screen shake for thin terminal while PAD terminal need to filter out unsuitable ones. This is the sub-workflow autonomic management.

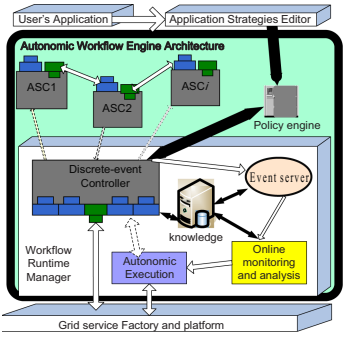

Fig. 1. AWE architecture

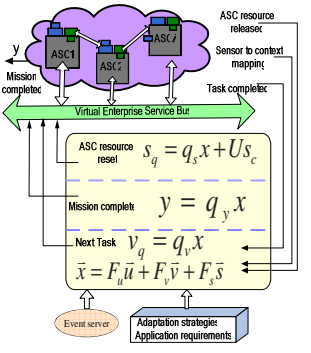

Fig. 2. System Level control architecture

2.1 Discrete-Event Controller (DEC)

A matrix-based DEC gives a very direct and efficient technique for both computer simulation and actual online supervisory control of DE systems[7,8]. In complex workflow systems, the interdependence of ASCs generates multiple events when a single change happens causing multiple rules to be triggered. The order of execution of rule actions determines the system behavior necessitating reasoning about execution order. For coordination problems of workflow systems we can write down a set of ECA rules to define **the mission planning** of the AWE, such as: **on** *event* **if** *condition* **do** *action.* The matrix-based DE controller allows one to easily represent these linguistic rules in a rigorous fashion. Let ***s*** be the vector of ASCs, ***v*** the vector of tasks that the services can perform, and ***u*** the vector of input events (occurrence of sensor detection events or content-based event notification[9], node failures, etc.). We define a mission as a prescribed sequence of tasks programmed into the Planer database. Let ***y*** be the vector of completed missions (outputs). Finally, let ***x*** be the state logical vector of the rules of the DE controller, whose entry of "1" in position ***i*** denotes that rule ***i*** of the supervisory-control policy is currently activated.

The DEC is based on formal equations that allow computation of the next tasks and ASCs to be assigned in the grid services pool. As shown in Fig.1, the DEC reads information from the Event server about sensor events, node failures, and ASCs availability. Then, it computes which tasks to start and which ASCs to assign or release, which are commands sent to the ASCs. In the following, all matrix operations are defined to be in the or/and algebra, where + denotes logical *or* and 'times' denotes logical *and*. The main equation is the controller-state equation

$$\bar{x} = F_u \bar{u} + F_v \bar{v} + F_s \bar{s} \quad (1)$$

where $\boldsymbol{F_v}$ is the task-sequencing matrix, $\boldsymbol{F_s}$ is the ASCs matrix, and $\boldsymbol{F_u}$ is the input matrix. $\boldsymbol{F_v}$ has element **(*i, j*)** set to "1" if the completion of task $\boldsymbol{v_j}$ is an immediate prerequisite for the activation of logic state $\boldsymbol{x_i}$. The $\boldsymbol{F_s}$ has element **(*i, j*)** set to "1" if the availability of service (ASC_j) is an immediate prerequisite for the activation of logic state $\boldsymbol{x_i}$. The overbar in equation (1) denotes logical negation so that tasks complete or resources released are represented by '0' entries.On the grounds of the current status of the workflow system, (1) calculates the logical vector $\boldsymbol{x}$, i.e., which rules are currently activated. The activated rules determine the commands that the DEC has to sequence in the next iteration, according to the following equations:

$$v_q = q_v x \quad (2)$$

$$s_q = q_s x + U s_c \quad (3)$$

$$y = q_y x \quad (4)$$

Where $\boldsymbol{q_v}$: task-start matrix and has element (***i, j***) set to "1" if logic state $\boldsymbol{x_j}$ determines the activation of task ***i***. $\boldsymbol{q_y}$: output matrix and has element (***i, j***) set to "1" if the activation of logic state $\boldsymbol{x_j}$ determines the completion of mission ***i***. $\boldsymbol{q_s}$: ASC matrix and has element (***i, j***) set to "1" if the activation of logic state $\boldsymbol{x_j}$ determines the release or resetting of ASC_i ; ***U*** is any user-defined matrix of $\boldsymbol{n}$ **x** $\boldsymbol{m}$ dimensions ($\boldsymbol{m}$ can be equal to $\boldsymbol{n}$ but not smaller). $\boldsymbol{n}$ will assume the number of elements in the $\boldsymbol{s_q}$ column vector. Equation (3) can be read as follows: $\boldsymbol{s_q}$ can only be released if the relevant conditions (determined by $\boldsymbol{q_s}$) are satisfied and the relevant user-defined service(s) is/are inactive. It is worth noting that the presence of the $\boldsymbol{s_c}$ element in the latter part of the equation serves only as an added validity check since the status of $\boldsymbol{s_c}$ is already present in $\boldsymbol{x}$**.** Regardless of whether the engine has co-scheduled the entire workflow or incrementally scheduled specific services($\boldsymbol{q_s}$), the engine must know which services are currently active (executing), and which services are not yet active. For services that are active, the engine can (a) synchronously wait for them to complete, or (b) asynchronously terminate them. These can be used in any replanned workflow[9].It is noted that (2) and (4) are inspired by [8,7]. Equations (1)-(4) represent the rule base of the supervisory control of the workflow system, where the DEC equations are efficiently implemented using a software such as MATLAB [7].

2.2 Dynamical Description and Rules Termination Analysis

In complex grid systems, the interdependence of ASCs generates multiple events when a single change happens causing multiple rules to be triggered. The order of execution of rule actions determines the system behavior necessitating reasoning about execution order. However, the behavior of ECA rule set is complicated and it is difficult to analyze its termination property. We introduce a new notion called enforcement semantics that provides guarantees about rule ordering based on Conditional Color Petri net(CCPN)[10]. With CCPN, it can better express the behavior of ECA rules. The termination property of ECA rule set can be analyzed via

reachability graph or T-invariants[10]. The incidence matrix of the PN equivalent to the DE controller is obtained by defining the activity completion matrix and the activity start matrix as: $F=[F_u,F_v,F_s,F_y];S=[S'_u,S'_v,S'_s,S'_y]$. The PN incidence matrix is then given by $M=S'-F$ .A PN marking transition equation can be written as

$$M_{k+1}=M_k+M'x_k \tag{5}$$

To put the DEC eqs. into a format convenient for simulation, one may write (1) as

$$\overline{x}=F\overline{m} \tag{6}$$

The combination of the DEC (6) and PN marking eq. (5) provides a complete dynamical description of the system, which is inspired by[7,8].

3 Self-managing for TigMNS

At present, a mobile navigation system faces a lot of challenges, including: the universality and compatibility are poor; real time information service and dynamic navigation are weak; service personalization is insufficient. To overcome above problems, TigMNP was finished successfully(see Fig.3). It has four primary submodels: Personal service configuration, Mobile map generation, Individual QoS and DEC. Because different mobile terminals have different processing ability, and different wireless network has different capacity as well;this requires workflows that can be dynamically reconfigured on-the-fly on the receipt of important events from both external physical systems such as heterogenous terminals request through multiple wireless communications, and from other computational systems such as: real time transportation status on the whole route network based on STIG[11].

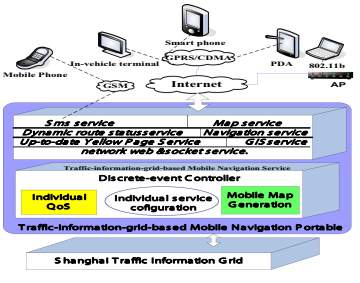

Fig. 3. Workflow Management for Traffic-information-grid-based Mobile Navigation Portable

Self-configuring behavior : self-configuring occurs on two levels. On one level, the AWE allows for a dynamic online assignment of tasks and ASCs based on Personal Service Configuration, given fixed DEC matrices $\boldsymbol{F_v}$, $\boldsymbol{F_s}$, $\boldsymbol{q_v}$, and $\boldsymbol{q_s}$. These matrices contain the task-sequencing and ASC-assignment information for fixed prescribed missions. In TigMNP, there are seven ASCs[11]:*Sms service(S_1) Map service(S_2),*

Dynamic route status service(S_3), Navigation service(S_4), Up-to-date Yellow Page Service(S_5), GIS service(S_6), and *network web &socket service(S_7).* Therefore, on a second higher level, self-configuring requires a modification of the DEC matrices as mission change, nodes fail, or additional nodes are deployed.

Self-optimizing behavior: a dynamic reallocation of the ASCs to missions can be performed by rearranging the "1"s in the matrices $\boldsymbol{F_s}$ and $\boldsymbol{q_s}$ when new missions are added. Some possible criterion could be the balance of workload of the ASCs, the minimization of circular waits and deadlock risks, the maximization of throughput (mission-completion time),Thanks to the matrix representation of the plans, these objectives can be governed via computationally efficient algorithms[7,8].

3.1 Implementation of a Mission

The procedure for implementing the supervisory-control policy consists of four different steps. (1) define the vector of *S* present in the system and the tasks they are able to perform. Here the ASC vector is $S = [s_1, s_2, s_3, s_4, s_5, s_6, s_7]$.(2) for each mission *i*, we define the vector of inputs $\boldsymbol{u^i}$,of outputs $\boldsymbol{y^i}$ and of tasks $\boldsymbol{v^i}$**,** and the task sequence of each mission. Then, we write down the ECA rules representing the supervisory coordination strategy to sequence the programmed missions (Tables 1 and2 for the mission1).(3) translate the linguistic description of the coordination rules into a more convenient matrix representation, suitable for computer implementation. (4) the formulation proposed in section 2.2 can be used to represent a CCPN. The termination property of ECA rules set can be analyzed using reachability graph[10].(5) if termination is satisfied then dispatch the command and the end else goto (2).

Table 1. Task Sequence of the Mission1

Mission1	**description**
Input(u)	S_7 connect DEC get Service Detail Code
T1	S_3 gives the traffic status report on given route
T2	S_4 provides a real-time dynamic navigation
T3	S_4 gives users a dynamic travel scheme
T4	S_5 provides GIS services to find a specific place
T5	generate answer message output y
Output(y)	Mission completed

Example: Given the Mission1: thin client with a connection to wireless network by CDMA 1x or GPRS **has request to get the best route between start point and destination.** Here, 'thin' means an application has to be installed locally and exchange information with background platform by a Socket connection[11]. We give the rule-base of Mission1 (Table 1) and the planer of services composition sequences(Table 2).We can easily write down the F_v,F_s as follows:

	T1	T2	T3	T4	T5
X_1	0	0	0	0	0
F_v=X_2	1	0	0	0	0
X_3	0	1	0	0	0
X_4	0	0	1	1	0

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
X_1	0	0	0	0	0	0	1
F_s=X_2	0	1	0	1	0	0	0
X_3	0	1	0	0	1	0	0
X_4	0	0	1	1	0	0	1

With reference to (2),(3).we also get q_s, q_v (we omit them for limited space). Fig. 4 shows the Petri Net sequence of the task with its respective autonomic service components. There is no cyclic path in the CCPN. The termination of the CCPN is guaranteed according to the algorithm in [10]. This provides a feasibility check to ensure if the task, given the request and resource assignments and the planning process, will actually be executable or not.

Table 2. Mission1-Rule Base

Rule	Mission1-operation sequence
X_1	On u occurs if S_7 available then act T1
X_2	On T1 completed if S_4 and S_2 available then act T2 and T3
X_3	On T2 completed if S_5 available then act T4
X_4	On T3 and T4 completed if S_7 connected then act T5

Table 3. Testing results of thin clinets and SMS service using Autonomic Workflow Engine

	Service content	*Response time (s)*
thin clients	get the best route between start point and destination	10.5
SMS service	query for the real-time traffic status about the route segment	5.6

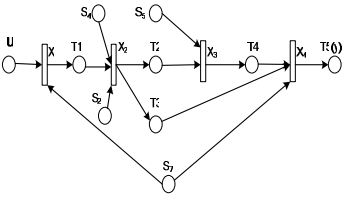

Fig. 4. CCPN representation of the mission1

3.2 Actual Running Effect

With the integration of our model to the Grid architecture, both sensor ports and functional ports of the autonomic service cell are implemented as WSDL documents. The publication/subscription structure is used for interactions between DEC and ASCs. The details of service selection and mappings of ECA rules to workflows are abstracted from the user and handled by the system itself.Fig.5 shows the result of the Mission1using AWE. When the event server(Individual Service Configuration)detects the thin terminal changes,it generates a new SDD code for DEC. Thus the AWE reconfigures the workflow (here deactive the GIS service) and dispatch S_2. To adapt to new context of service composition, the map server agent are customized for PDA (see Fig.6). Table 3 is the testing result of service answering speed for these two applications and better performance is achieved than [11].

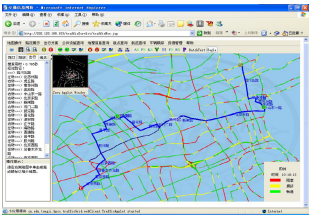

Fig. 5. Query result for thin client

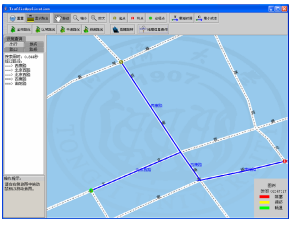

Fig. 6. Query result for PDA

4 Conclusion

The architecture enables grid workflow to be dynamically specified and adapted using ECA rules. In this architecture, enactment services can be added or removed on demand. Therefore, the whole engine will have high performance and high availability. Given the presence of such shared resources, simultaneous activation of conflicting rules may arise. That is, several tasks (possibly from different missions) may simultaneously request the same set of resources.

Acknowledgments. This work is support partially by projects of National Basic Research Program of China (973 Program) (2003CB317002, 2004CB318001-03), National Natural Science Funds (60534060, 60473094).

References

1. S. Hariri., et al., The Autonomic Computing Paradigm, Cluster Computing. 9(1), (2006) 5-17.
2. Bae J, Bae H, Kang SH, Kim Y., Automatic control of workflow processes using ECA rules. IEEE Trans. on Knowledge and Data Engineering,,16(8), (2004) 1010-1023.
3. L.Chen, et al., ECA Rule-Based Workflow Modeling and Implementation for Service Composition, IEICE Transactions on Information and Systems, E89-D(2) (2006) 624-630.
4. http://www.gridworkflow.org/kwfgrid/gwes/docs
5. Z.Zhao, et al., Dynamic Workflow in a Grid Enabled Problem Solving Environment, In proc. of CIT (2005) 339-345 .
6. Falk Neubauer, Andreas Hoheisel, Joachim Geiler., Workflow-based Grid Applications. Future Generation Computer Systems, Volume 22, Number 1-2, pp. 6-15.
7. J. Mireles, F. Lewis, Intelligent material handling: Development and implementation of a matrix-based discrete event controller, IEEE Trans.Ind. Electron., 48(6) (2001) 1087–1097.
8. V. Giordano et al, Supervisory control of mobile sensor networks: math formulation, simulation, implementation, IEEE Transactions on Systems, Man and Cybernetics, Part B,36(4) (2006) 806-819.
9. Craig A. Lee, et al, The Event-Driven Workflows towards A Posteriori Computing, Future Generation Grids, Springer-Verlag, (2006):3-29.
10. L. Baba-hamed, et al., Termination Analysis Approach and Implementation, Journal of Applied Sciences.,6 (8)(2006) 1738-1744.
11. FANG Yu, et al. A Mobile Navigation Service Platform Based on Traffic Information Grid. International Journal of Services Operations and Informatics, Vol.1 (2006) 23-37.