International Journal of Computer Applications Technology and Research
Volume 2– Issue 4, 455 - 458, 2013, ISSN: 2319–8656

Dynamically Adapting Software Components for the Grid

B. Dhanalakshmi
Aryabahata Institute Of Technology and Science
Jawaharlal Nehru Technological University
Hyderabad, India

Abstract: The surfacing of dynamic execution environments such as 'grids' forces scientific applications to take dynamicity. Dynamic adaptation of Grid Components in Grid Computing is a critical issue for the design of framework for dynamic adaptation towards self-adaptable software development components for the grid. This paper carries the systematic design of dynamic adaptation framework with the effective implementation of the structure of adaptable component. i.e. incorporating the layered architecture environment with the concept of dynamicity.

Keywords: grid; adaptation; resources; entities; component specific level; component independent level.

1. INTRODUCTION

A grid [4] is a type of parallel and distributed system, which aims at exploiting the ability to share and aggregate distributed computational capabilities. Number of resources may differ from processor to processor or processor to grid [4] while performing scientific applications. Therefore, adopting the software components for the programming model enables security and portability on different resource infrastructures.

This adaptability framework project model can be used to adapt the software components at run time to varying conditions. This framework report gives the transparency of adaptability in scientific and distributed applications by giving the framework impact and its requirement. This gives the ability of software to autonomously react and repair non convenient events observed during program execution without any intervention by the programmers.

Adaptability ensures that the application continuously executes the best configuration depending on the actual execution environment.

2. PROPOSED SCHEME

In this scheme, the applications involve several services like information services, resources reservation/allocation, file transfer and job launching and monitoring, which are executed on different environments. Grid components may change in processor availability, network availability, resource sharing between applications, administration tasks, failures etc...These environments constitute a disseminated, heterogeneous, highly dynamic communication structure that makes the applications as adaptive software includes different mechanisms to modify the behavior of application or components dynamically. This scheme suggests a layered approach model to put together self-adaptive entities:

The central stage level defines the mandatory functionalities for adaptive entities, while upper stage levels define the structure to bring together primitive or composite components adaptation [11].

With the allocated resources, we may define the best way that is used by scientific applications modify themselves depending on their actual execution environment. This framework gives the ability of software to *autonomously react to* and *repair non convenient events observed* during program execution without any intervention by the programmers /users. In the autonomous computing, adaptation is further characterized as activity performed by code, acts to events, performs suitable actions, identifies wrong behavior etc..

This paper presents a framework intended to help developers in the task of designing dynamically adaptable components, which puts the emphasis on an experimental evaluation of the cost of using such a framework.

3. DYNAMIC ADAPTATION

In order to achieve an adaptation, a component needs to be able to get information about the system (the component itself and its environment), to make a decision according to some optimization rules and to modify or replace some parts of its code. Any scientific system amalgamates its modifications in a crystal clear way for its end users. These modifications include adapting to variable run-time conditions, masking failures, performance measures and the evolution of scientific application components. 'Dynamic adaptation' [1] coats different techniques for managing all these modifications in the execution environment.

Dynamic adaptation [5] is classified into three dimensions named kinds, characteristics and techniques. These dimensions are introduced because they answer the frequent questions of administrators and developers of the application. This classification is the result of our investigation of existing adaptable platforms.

For the sake of reusability, it is highly desirable to separate the adaptation engine from the content of the component. I capture adaptability within a framework as Dynamic Adaptation for Components (Dynaco [1]). Associated to a component's content, it forms a dynamically adaptable component.

This paper presents a framework intended to help developers in the task of designing dynamically adaptable components,

which puts the emphasis on an experimental evaluation of the cost of using such a framework.

4. DYNAMIC ADAPTATION

Applications are of 2 types as, Resource-Aware: describe resources options, select resources and then run and Dynamic Adaptive: collect runtime information, consider/decide to change resources, select resources and run.

A generic adaptability framework for decomposition of adaptability in 4 steps: Observe the execution environment as it evolves, Decide that the component should adapt, Plan how to achieve the adaptation, Schedule and execute planned actions.

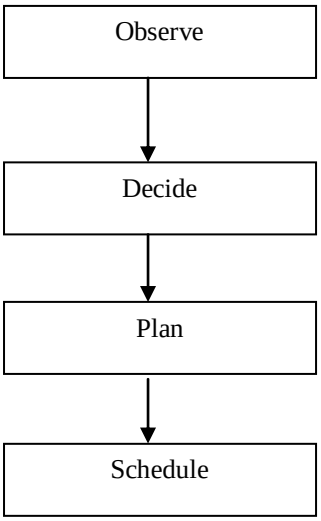


Figure 1. Proposed Design Architecture

4.1 Architectural Design

This divides adaptability into some number of sub-functionalities as Able to observe characteristics of the environment in order to trigger adaptability; When a change is detected, the framework has to decide an adaptation strategy according to observed measures; Once a strategy has been decided, the framework has to plan actions to implement it;

At last; planned actions have to be executed synchronously with the execution of applicative code.

This model exhibits the functional description for the adaptation process with the entities Decider, Planner and Executor. The environmental changes received as events will affect the decider and produces as a strategic plan for dynamic adaptation.

The *Planner* derives list of actions from the strategy in order to achieve the different steps of the process of adaptation. The *executor* implements the different steps of adaptation to modify the component.

Software components that are used in the adaptability framework are separated into some number of functional "boxes" disseminated into 3 levels as shown in figure 1. At the functional level, the service provides an expected implementation of the component is expected to do. If the component was not dynamically adaptable, then it would have the service.

The *component-independent level* [1] contains all mechanisms that can be defined independently of the content of the service functional box. The decider box is the start point of any adaptation. It decides whether the component should be adapted or not.

The *component-specific level* [1] holds the specializations of the adaptation framework for the developers. The specified framework consent the developer to focus the decider for the needs of its component. It describes how decisions can be made. The plan templates describe how the planner can build plans depending on the requested reaction and on the current execution environment.

4.2 Structural Design

This splits adaptability into four sub-functionalities as Able to observe characteristics of the environment in order to trigger adaptability; When a change is detected, the framework has to decide an *adaptation strategy* [5] according to observed measures; Once a strategy has been decided, the framework has to plan actions to implement it;

At last; planned actions have to be executed synchronously with the execution of applicative code.

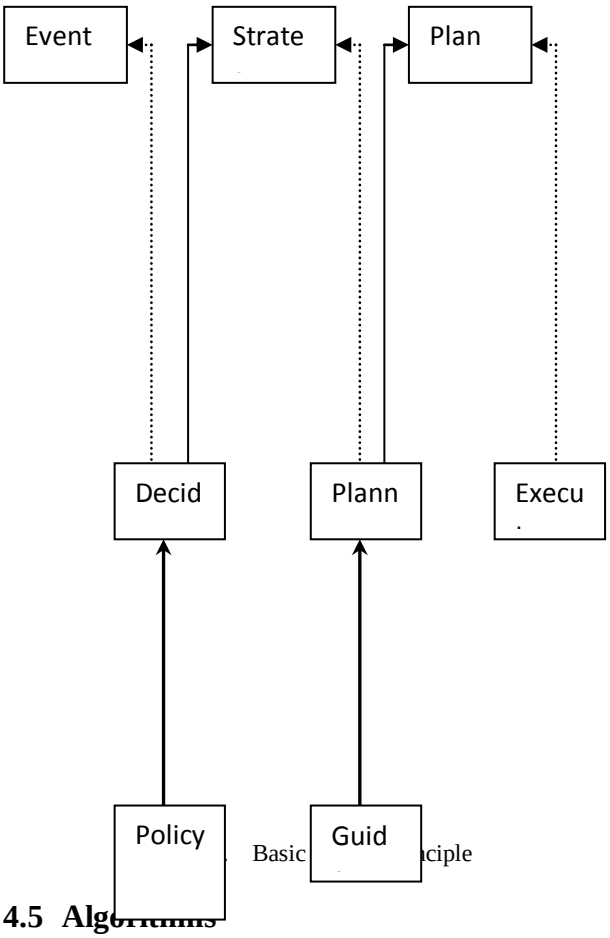
4.3 Events Generation

The Decision making process[2] of dynamic adaptation given by the policy procedure which will give the information about the decisions and strategies that are used to change the component's behavior depending on the execution environment.

Monitors [13] are the entities that are used to create the events and these events are helpful in the monitoring of components execution. All these events are observed by decider or monitor.

4.4 Adapting the Components

Subsequent to adaptation plan, next is the executor's turn, which is regarded as a virtual machine [13] that will monitor the control flow instructions with in the adaptation plan depending on the execution schedule. To do so executor depends on the adaptation points [2] this will have the information about component states. The component states are constrained by integrity and consistency requirements.



Basic nciple

4.5 Algorithms

Following are the two algorithms that are used in this framework.

4.5.1 Converter

The first algorithm, the *converter* converts a given component to particular interface by deterministically matching the given component/interface pair to an adapter on demand. The algorithm will rely on the executable notations for interfaces, adapters and components, e.g., it will have to find out at runtime which edges a given component supports and which borders a given adapter maps to.

4.5.2 Binder

The second algorithm, the *binder* combines the given interface with an implementation instance, which is considered as a component or a component enclosed by adaptors.

4.6 Autonomic Implementation of the Schema

The schema notation of this framework is as below.

do

{

decide:

what has to adapt

what has to discard

how to adapt

on trigger:

decide possible implementations for the policy

commit:

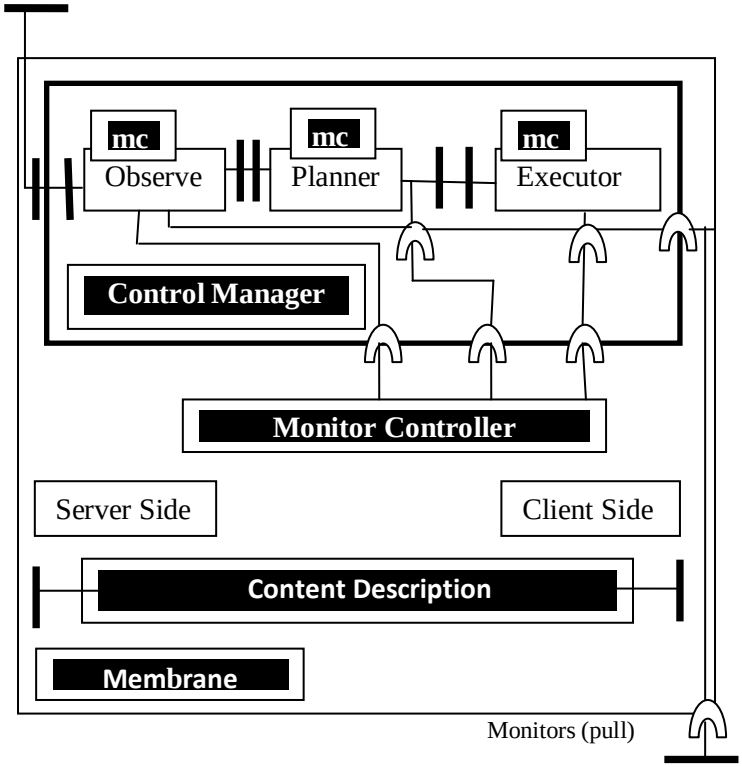
plan for predefined implementation

plan adaptive code mechanisms

schedule all the executable operations

} while(!end)

Monitors (push)



Monitors (pull)

Figure 3. Structure of an adaptable component using this Framework [13]

4.7 Realization as a Framework Model

In the Framework model realization, the two contents are defined as: The *content description* realizes the component state functionalities; the *membrane* [2] is a possessor consisting of non-functional services that manage the component's behavior. In an adaptation plan, dynamic execution environments depend on the behavior of dynamically adaptable components. The *Control Manager* is responsible for the existed components. A modification controller (mc) realizes the adaptable component's actions.

The *executor* [13] execute actions depending on the plans given by the planner. Server sides (push model) and client

sides (pull model) are applicable to the adaptable component while executing them.

Modifications are possible by pushing and pulling the adaptation methods by which the model implements the adaptability concept on components.

5. CONCLUSION

This framework analyses how to design dynamic adaptability support for scientific applications. It is independent of formalisms and technologies. Also evaluate the proposed model as well as the possibilities to write generic adaptation policies at the collection and application levels.

This framework for adaptability is independent of the applications like numerical algorithms, transaction systems etc…

Still lot of problems to be investigated / solved (adaptation policies, performance models ...). For the future work, the activity of generalizing the approach is considered. i.e., generic definition of global adaptation points should be implemented.

6. ACKNOWLEDGMENTS

My sincere thanks to the experts, who have contributed towards development of this framework.

7. REFERENCES

[1] Jérémy Buisson, Françoise André, and Jean-Louis Pazat. Dynamic adaptation for grid computing. In P.M.A. Sloot, A.G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, editors, Advances in Grid Computing- EGC 2005 (European Grid Conference, Amsterdam, The Netherlands, February 14-16**, 2005**, Revised Selected Papers), volume 3470 of LNCS, pages 538–547, Amsterdam, February **2005.** Springer-Verlag.

[2] Je´re´my Buisson, Franc¸oise Andre´, and Jean-Louis Pazat. Performance and practicability of dynamic adaptation of parallel computing : an experience feedback from Dynaco. Publication interne 2006 Projeeects Paris.

[3] Jérémy Buisson, Françoise André, and Jean-Louis Pazat. Enforcing consistency during the adaptation of a parallel component. In The 4th International Symposium on Parallel and Distributed Computing, July **2005**.Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[4] Greg Burns, Raja Daoud, and James Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceed- ings of Supercomputing Symposium*, pages 379–386, **1994**.

[5] Introduction to Grid Computing – A IBM’s red book for details about Grid Computing is also useful for installation of Globus Tool Kit 4.

[6] Jérémy Buisson, Françoise André and Jean-Louis Pazat. A framework for dynamic adaptation of parallel components**.** In *ParCo 2005*, Málaga, Spain, 13-16 September **2005**.

[7] Pierre-Charles David and Thomas Ledoux. Towards a framework for self-adaptive component-based applications. In *DAIS’03*, volume 2893 of *LNCS*. Springer-Verlag, November **2003.**

[8] Brian Ensink and Vikram Adve. Coordinating adaptations in distributed systems. In *24th International Conference on Distributed Computing Systems*, pages 446–455, March **2004**.

[9] Brian Ensink, Joel Stanley, and Vikram Adve. Program control language: a programming language for adaptive distributed applications. *Journal of Parallel and Distributed Computing*, 63(11):1082–1104, November **2003**.

[10] Introduction to Grid Computing – A IBM’s red book for details about Grid Computing is also useful for installation of Globus Tool Kit 4.

[11] Research group on “Performance models and adaptivity”: http://www.di.unipi.it/~marcod/WP3homepage/RG_adaptivity/index.html

[12] Segarra, M.T. ; Dept. of Comput. Sci., IT/TELECOM-Bretagne, Brest ; Andre, F.Autonomic and Autonomous Systems, 2009. ICAS '09. Fifth International Conference on « Building a context-aware ambient assisted living application using a self adaptive distributed model

[13] http://hal.archives-ouvertes.fr/docs/00/05/76/49/PDF/PI-1782.pdf

.