The Adapta Framework for Building Self-Adaptive Distributed Applications

Márcio Sekeff, Francisco Silva

Laboratório de Sistemas Distribuídos
Departamento de Informática
Universidade Federal do Maranhão, Brasil

June, 2007
1. Introduction

2. Adapta Architecture

3. Case Study: Towards a Self-Reconfigurable Grid Middleware

4. Conclusions
Motivation

- Modern distributed applications are getting harder to develop, maintain, and configure as environmental dynamism and computational devices heterogeneity increase;

- Pervasive computing environment:
  - Variations of resources and services availability, intermittent connectivity, huge device heterogeneity and communication technologies, and issues concerning user mobility;

- Opportunistic Grid:
  - High variation on resource availability, node instability, variations on load distribution, and heterogeneity of computational devices and network technology.
Challenge

- Developers must address not only the business logic of the system, but also how to make it flexible in order to run on most platforms and adaptable to changes on its executing environment;

- Applications must be able to apply modifications on its own structure and functionality automatically, without code recompiling and human intervention;

- Configuration and management must be performed on the fly, transparently to end users and not disrupting the service being offered.
The Middleware Approach

- Middleware architectures can provide a comprehensive solution guideline and tools to the problem of building effective self-adaptive systems, easing the application developer work;

- Three fundamental questions concerning the design of self-adaptive systems:
  1. *When to adapt?* How can the system detect that it’s time to adapt so that its performance will improve and changes in the environment will not harm system correct functioning?
  2. *What to adapt?* Which parts, elements, components of the system are subject to being adapted or replaced?
  3. *How to adapt?* Which adaptive mechanism would be more beneficial to be applied given a certain system and environmental state?
Adapta Framework: Overview

- Adapta is a framework for developing self-adaptive distributed applications;
- The framework decouples the code that governs the business rules from the code responsible for adaptation, based on computational reflection;
- The meta-level receives event notifications describing environmental changes and reacts by applying reconfiguration actions to the base level objects;
- Adapta also has a runtime system that monitors the executing environment and notifies events to registered components whenever a resource availability condition is detected.
Adapta Framework: Overview

- Each adaptive application component has a reconfiguration file that contains the representation of its object model described in a XML based language;
- Application developers or administrators can alter an application component object model by simply modifying its corresponding reconfiguration file;
- Modifications to the object model are interpreted and built on runtime, using the Adaptive-Object Model (AOM) pattern.
Adapta Framework Components
Monitoring Service (MS)

- Regularly inspects the underlying hardware and the executing environment on every node using monitor objects, which are individually assigned to a single property;
- Monitors can be dynamically instantiated to introduce new monitoring requirements not known at design-time or replaced on the fly;
- Each property also has a set of operating ranges, for example, one could use the following operation ranges for monitoring the CPU load usage: [0%, 40%), [40%, 75%), and [75%, 100%]. The MS notifies a collocated Local Event Service (LES) whenever there is a change on the operating range of a property.
Adapta Framework Components

- Monitoring Service
  - significative change
  - local events
- Local Event Service
- CORBA Event Channel
  - local events
  - distributed events
  - local and distributed events
- Event Processing System
- DyReS
Local Event Service (LES)

- Notifies events to subscribed components whenever a determined resource availability condition occurs;
- Events are based on a boolean expression provided by the application developer as part of the event definition;
- UCP:Load grt 80 AND MainMemory:Available les 50;
- The boolean expression must stay true during an amount of time specified by the user, known as the *duration time*. It avoids generating notifications when temporary situations occur, such as a resource usage peak.
Event Processing System (EPS)

- EPS is a distributed event service that detects composite events from different event sources (distributed nodes);
- Example: application where component migration takes into account the CPU usage of every node across the network;
- Addresses typical distributed issues, such as message ordering, loss, or duplication;
- Notifies events to subscribed components whenever a event definition is detected.
Adapta Framework Components
Dynamic Reconfiguration Service (DyReS)

- Each adaptive component must instantiate a meta-level called DyReS;
- It is responsible for receiving event notifications (from LES or EPS) and for applying reconfiguration actions on behalf of its component;
- Implemented reconfiguration actions:
  1. Dynamic change of application parameters;
  2. Dynamic replacement of application algorithms with a well-defined state transfer protocol.
- DyReS can be extended by expert users to introduce new actions, such as component addition, removal, or replacement.
Dynamic Reconfiguration Service (DyReS)

- DyReS uses the ComponentConfiguration architecture, maintaining references to the components it depends on (called hooks), as well as references to dependent components (called clients);

- The reference chain is used for synchronizing reconfiguration actions between application components;

- Example: applying a video compression algorithm to a video stream.
Autonomic Grid Motivation

- Computer grids have been used to solve problems in varied areas, empowering the conception of applications that allow combining computations, experiments, observations, and data got in real time;

- The complexity of modern grid environments have turned impracticable its configuration, maintenance and recovery in case of failures solely by human beings, due to:
  - Its dynamism, that can be observed in terms of high variation in resource availability, node instability, and workload variations in nodes and network links;
  - Its heterogeneity, by aggregating a great amount of computation and communication resources, databases and, sometimes, sensors and specific peripherals;
  - Its high scalability, by integrating computational resources spread through several administrative domains.
The AutoGrid Project

- Main goal: the development of a robust and self-managing autonomic grid system;
- It is based on the Integrade grid middleware, incorporating autonomic mechanisms to its infrastructure in order to make its configuration and administration independent from human intervention;
- Our research primarily focuses on adding to Integrade four autonomic properties: context awareness, self-healing, self-optimization, and self-configuration.
Organization of an Integrade Grid

- The basic unit: cluster, a collection of machines usually connected by a local network;
- Clusters can be organized in a hierarchy, encompassing a large number of machines;
- Each cluster contains:
  - A Cluster Manager node, responsible for managing the cluster computing resources and for inter-cluster communication;
  - Workstations, which export part of its resources to Grid users. They can be shared or dedicated machines.
AutoGrid Self-healing Mechanism

- The dynamic nature of the grid infrastructure, its high scalability, and great heterogeneity exacerbates the likelihood of errors occurrence;
- Diverse failure handling techniques can be applied:
  - Retrying: restarts an application from scratch;
  - Replication: submits the same application for execution a number of times, generating various application replicas;
  - Checkpointing: periodically saves the state of the computation in a stable storage during the failure free execution time. Upon a failure, the application restarts from the last saved point.
- Considering the objective of minimizing the application response time, experiments demonstrated that the best failure handling technique varied as we altered environment parameters such as the MTBF and the downtime.
AutoGrid Replication Mechanism

- Integrate Global Resource Manager (GRM) decides the amount of replicas to be generated for a given application submission based on the execution environment MTBF and the application mean execution time;
- It also runs the scheduler, designing different nodes for application replicas;
- The Monitoring System periodically computes the cluster MTBF based on the data stored by the Execution Manager (EM);
- An event is generated whenever a significant change is detected, updating GRM MTBF parameter;
- We measured the benefits of varying the amount of replicas with a set of simulations.
AutoGrid Self-optimization Mechanism

- The computational grid comprise a highly dynamic environment (amplified on opportunistic grids, since nodes are not dedicated);
- In this context, load balancing techniques, adaptive scheduling, and dynamic re-scheduling are important topics to be investigated;
- Dong et all refers to an adaptive solution to the scheduling problem as the one in which the algorithms and parameters used to make scheduling decisions change dynamically;
- Maheswaran et all quantifies the relative performance of scheduling heuristics. Simulations indicated that the more appropriate algorithm varied as the environment changes.
AutoGrid Self-optimization Mechanism

- The Adapta framework provides the means for dynamic switching the GRM scheduling algorithm, including a state transfer mechanism;
- We are currently altering InteGrade scheduling mechanism, allowing it to dynamically switch between three distinct algorithms: the MCT (Minimum Completion Time), max-min and min-min;
- Investigations are being made to measure the exact moment of replacement, according to the application arrival rate, resource usage, network status and other environmental properties.
Conclusions

- As dynamism and heterogeneity increase in today’s modern computing systems, application development must shift to a self-adaptive paradigm, instead of a static-based one;
- Adapta is a framework that enables the development of distributed self-adaptive applications;
- It is based on computational reflection, promoting a clear separation of concerns between the adaptable part of the application from its core (business rules), simplifying the application implementation, debugging, and maintenance;
- It provides runtime system, monitoring resource availability and notifying interested components when changes are detected on resource usage;
- It also provides an application reconfiguration engine that corresponds to the application meta-level.
Adapta set of adaptation mechanisms is not restricted and can be easily extended by experienced users through a flexible architecture;

Each Adapta component can be dynamically altered using AdaptaML, a XML-based reconfiguration language used to describe the component object model that can be modified and loaded at runtime;

Adapta framework was evaluated through concrete case studies: an autonomic grid infrastructure and an adaptive video server.