WORKFLOW ENGINE FOR CLOUDS

By SURAJ PANDEY, DILEBAN KARUNAMOORTHY, and RAJKUMAR BUYYA

Prepared by: Dr. Faramarz Safi
Islamic Azad University, Najafabad Branch, Esfahan, Iran.
Workflow Engine for clouds

Introduction

• A workflow models a process as consisting of a series of steps that simplifies the complexity of execution and management of applications.
• Scientific workflows in domains such as high-energy physics and life sciences utilize distributed resources in order to access, manage, and process a large amount of data from a higher level.
• Processing and managing such large amounts of data require the use of a distributed collection of computation and storage facilities. These resources are often limited in supply and are shared among many competing users.
• The recent progress in virtualization technologies and the rapid growth of cloud computing services have opened a new paradigm in distributed computing for utilizing existing (and often cheaper) resource pools for on demand and scalable scientific computing.
• Scientific Workflow Management Systems (WfMS) need to adapt to this new paradigm in order to leverage the benefits of cloud services.
Workflow Engine for clouds

Introduction

- Cloud services vary in the levels of abstraction and hence the type of service they present to application users.
- Infrastructure virtualization enables providers such as Amazon to offer virtual hardware for use in compute- and data-intensive workflow applications.
- Platform-as-a-Service (PaaS) clouds expose a higher-level development and runtime environment for building and deploying workflow applications on cloud infrastructures.
- Further up in the cloud stack are Software-as-a-Service providers who offer end users with standardized software solutions that could be integrated into existing workflows.
WORKFLOW MANAGEMENT SYSTEMS AND CLOUDS

- **Elasticity**: This enables workflow management systems to readily meet quality-of-service (QoS) requirements of applications, as opposed to the traditional approach that required advance reservation of resources in global multi-user grid environments.

- **SLA**: This allows workflow management systems to provide better end-to-end guarantees when meeting the service requirements of users by mapping them to service providers based on characteristics of SLAs.

- **Virtualization**: One of the benefits of using virtualized resources for workflow execution, as opposed to having direct access to the physical machine, is the reduced need for securing the physical resource from malicious code.
Architectural Overview
Workflow Engine in the Cloud

A storage service such as FTP or Amazon S3 for temporary storage of application components, such as executable and data files, and output (result) files.

FIGURE 12.1. Workflow engine in the cloud.
Architectural Overview
Workflow Engine in the Cloud

Figure 12.1 presents a high-level architectural view of a workflow management system (WFMS) utilizing cloud resources to drive the execution of a scientific workflow application.

- The workflow system comprises the workflow engine, a resource broker [13], and plug-ins for communicating with various technological platforms, such as Aneka [14] and Amazon EC2.
- User applications could only use cloud services or use cloud together with existing grid/cluster-based solutions. Figure 12.1 depicts two scenarios,
  - one where the Aneka platform is used in its entirety to complete the workflow,
  - and the other where Amazon EC2 is used to supplement a local cluster when there are insufficient resources to meet the QoS requirements of the application.
Architectural Overview
Workflow Engine in the Cloud

• Aneka [13], is a PaaS cloud and can be run on a corporate network or a dedicated cluster or can be hosted entirely on an IaaS cloud. Given limited resources in local networks, Aneka is capable of transparently provisioning additional resources by acquiring new resources in third-party cloud services such as Amazon EC2 to meet application demands.

• This relieves the WFMS from the responsibility of managing and allocating resources directly, to simply negotiating the required resources with Aneka.

• Aneka also provides a set of Web services for service negotiation, job submission, and job monitoring. The WFMS would orchestrate the workflow execution by scheduling jobs in the right sequence to the Aneka Web Services.
Multi-objective optimization
CASE STUDY: EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATIONS

Multi-objective optimization (also known as multi-objective programming, vector optimization, multi-criteria optimization, multi-attribute optimization or Pareto optimization) is an area of multiple criteria decision making, that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously.

Multi-objective optimization has been applied in many fields of science, including engineering, economics and logistics (see the section on applications for examples) where optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives.

Examples: **Minimizing weight while maximizing the strength of a particular component,** and **maximizing performance whilst minimizing fuel consumption and emission of pollutants of a vehicle** are examples of multi-objective optimization problems **involving two and three objectives,** respectively.

For a multi-objective optimization problem, there does not exist a single solution that simultaneously optimizes each objective. In that case, the objective functions are said to be conflicting, and there exists a (possibly infinite number of) Pareto optimal solutions.

A solution is called non-dominated, Pareto optimal, Pareto efficient or non-inferior, if none of the objective functions can be improved in value without impairment in some of the other objective values.

Without additional preference information, all Pareto optimal solutions can be considered mathematically equally good (as vectors cannot be ordered completely).
Solving a multi-objective optimization problem

- Researchers study multi-objective optimization problems from different viewpoints and, thus, there exist different solution philosophies and goals when setting and solving them.
- The goal may be finding a representative set of Pareto optimal solutions, and/or quantifying the trade-offs in satisfying the different objectives, and/or finding a single solution that satisfies the preferences of a human decision maker (DM).
- A multi-objective optimization problem is an optimization problem that involves multiple objective functions. In mathematical terms, a multi-objective optimization problem can be formulated as where the integer $k \geq 2$ is the number of objectives and the set $X$ is the feasible set of decision vectors defined by constraint functions.

$$\min (f_1(x), f_2(x), \ldots, f_k(x))^T$$

s.t. $x \in X$, 
This section presents a scientific application workflow based on an iterative technique for optimizing multiple search objectives, known as Evolutionary Multi-objective Optimization (EMO) [27].

EMO is a technique based on genetic algorithms. Genetic algorithms are search algorithms used for finding optimal solutions in a large space where deterministic or functional approaches are not viable.

Genetic algorithms use heuristics to find an optimal solution that is acceptable within a reasonable amount of time.

In the presence of many variables and complex heuristic functions, the time consumed in finding even an acceptable solution can be too large.

However, when multiple instances are run in parallel in a distributed setting using different variables, the required time for computation can be drastically reduced.
The following are the objectives for modeling and executing an EMO workflow on clouds:

- Design an execution model for EMO, expressed in the form of a workflow, such that multiple distributed resources can be utilized.
- Parallelize the execution of EMO tasks for reducing the total completion time.
- Dynamically provision compute resources needed for timely completion of the application when the number of tasks increases.
- Repeatedly carry out similar experiments when required.
- Manage application execution, handle faults, and store the final results for analysis.
FIGURE 12.6. EMO workflow structure (boxes represent task, arrows represent data dependencies between tasks).
In order to parallelize the execution of EMO, we construct a workflow model for systematically executing the tasks.

In our case study, the EMO application consists of five different topologies, upon which the iteration is done. These topologies are defined in five different binary files.

Each file becomes the input files for the top level tasks (A0emo1, A0emo, . . . ).

We create a separate branch for each topology file. There are two branches, which get merged on level 6. The tasks at the root level operate on the topologies to create new population, which is then merged by the task named “emomerge.”
we see two “emomerge” tasks in the 2\textsuperscript{nd} level, one task in the 6\textsuperscript{th} level that merges two branches and then splits the population to two branches again, two tasks on the 8\textsuperscript{th} and 10\textsuperscript{th} levels, and the final task on the 12\textsuperscript{th} level.

In the example figure, each topology is iterated two times in a branch before getting merged. The merged population is then split. This split is done two times in the figure. The tasks labeled B0e and B1e (depicted as darker shade in Figure 12.6) is the start of second iteration.
EMO Application. We use ZDT2 [27] as a test function for the objective function.

The workflow for this problem is depicted in Figure 12.6. In our experiments, we carry out 10 iterations within a branch for 5 different topologies.

We merge and split the results of each of these branches 10 times. For this scenario, the workflow constituted of a total of 6010 tasks. We varied the tasks by changing the number of merges from 5 to 10.

In doing so, the structure and the characteristics of the tasks in the workflow would remain unchanged. This is necessary for comparing the execution time when the number of task increases from 1600 to 6000 when we alter the number of merges from 5 to 10.
WORKFLOW ENGINE FOR CLOUDS
CASE STUDY: EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATIONS
Deployment and Results

• **Output of EMO Application.** After running the EMO workflow, we expect to see optimized values for the two objectives given by the ZDT2 test function.

• Figure 12.7 depicts the graph that plots the front obtained after iterating the EMO workflow depicted in Figure 12.6. The front at Level 2 is not the optimal.

• After first iteration, the front is optimized. Iteration 2 does not significantly change the front, hence the overlapping of the data for Iteration 1 and 2.

• **Experimental Results When Using Clouds.** Because the EMO workflow is an iterative approach, increasing the number of iterations would increase the quality of optimization in the results. Analogously, the greater the number of tasks completing in the workflow, the greater the number of iterations, hence the better the optimization.

• Because the iterations can be carried out for an arbitrarily large number of times, it is usually a best practice to limit the time for the overall calculation. Thus, in our experiment we set the deadline to be 95 minutes. We then analyze the number of tasks completing within the first 95 minutes in two classes of experiments.
Experiment 1: Seven Additional EC2 Instances Were Added.

In this experiment, we started executing the tasks in the EMO workflow initially using 20 EC2 compute resources (one node for workflow engine, one node for Aneka master, 18 Aneka worker nodes). We instantiated 7 more small instances to increase the total number of resources to 25. They were available for use after 25 minutes of execution. At the end of 95 minutes, a total of 1612 tasks were completed.

Experiment 2: Twenty Additional EC2 Instances Were Added.

In this experiment, we started executing the tasks in the EMO workflow using 20 EC2 compute resources, similar to Experiment 1. We instantiated 20 more EC2 instances after noticing the linear increase in task completion rate. These instances however were available for use after 40 minutes of execution. At the end of 95 minutes, a total of 3221 tasks were completed.
Analysis of Results

• In both experiments, the initial task completion rate increased linearly until we started more instances.
• As the number of resources was increased, the rate of task completing were increased drastically.

FIGURE 12.8. Number of tasks completing in time as the number of compute resources provisioned were increased at runtime.
FUTURE RESEARCH DIRECTIONS

• some visions and inherent difficulties faced by practitioners when using various cloud services. Drawing upon these visions, we list below some future research directions in the form of broad research directions:

• How to facilitate inter-cloud operations in terms of coherent data exchange, task migration, and load balancing for workflow application.
• When and where to provision cloud resources so that workflow applications can meet their deadline constraints and also remain within their budget.
• How to balance the use of cloud and local resources so that workflow applications can meet their objectives.
• How to match workflow application requirements to any service provider’s capabilities when there are numerous vendors with similar capabilities in a cloud.