WORKFLOW ENGINE FOR CLOUDS

By SURAJ PANDEY, DILEBAN KARUNAMOORTHY, and RAJKUMAR BUYYA

Prepared by: Dr. Faramarz Safi
Islamic Azad University, Najafabad Branch,
Esfahan, Iran.

Task Computing

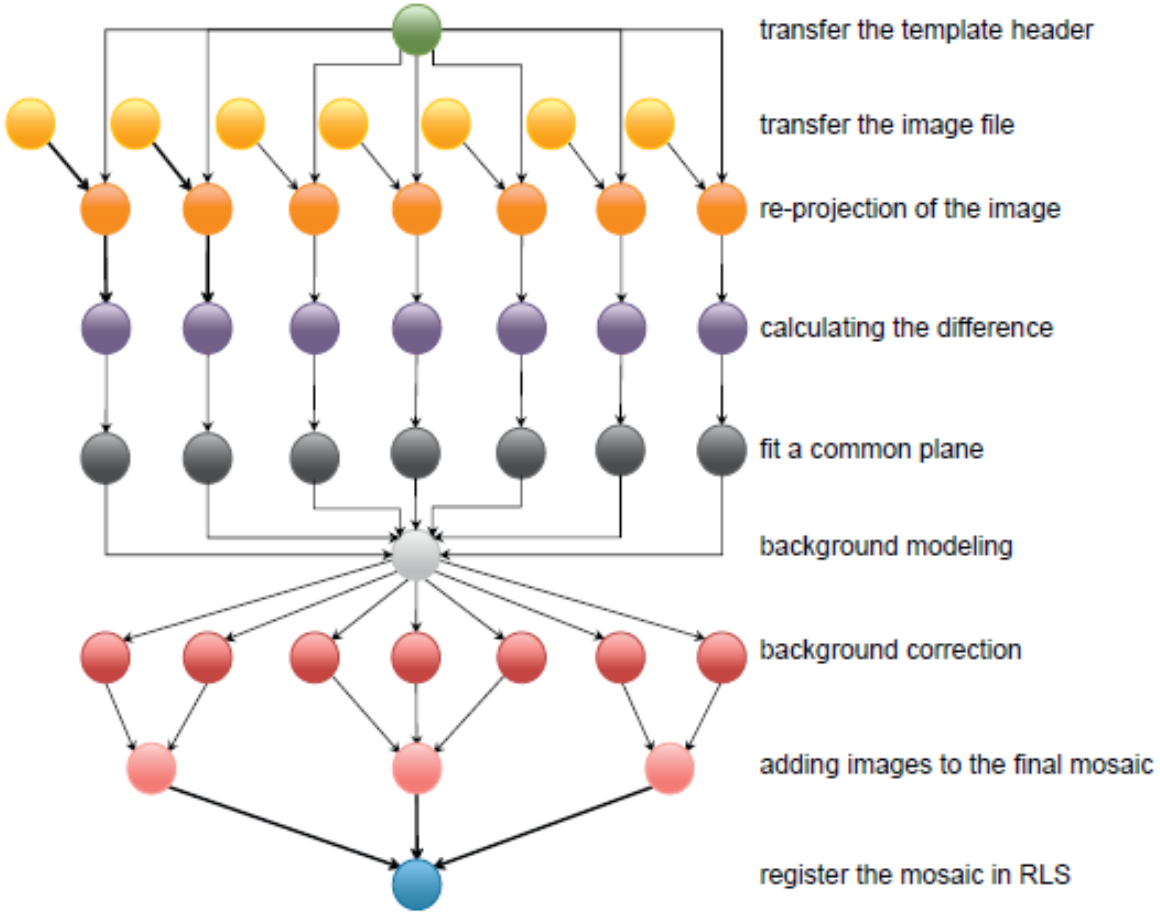
- Task computing is a wide area of distributed system programming encompassing several different models of architecting distributed applications, which ,eventually, are based on the same fundamental abstraction: the task.
- A task generally represents a program, which might require input files and produce output files as a result to fits execution. Applications are then constituted of a collection of tasks. These are submitted for execution and their output data are collected at the end of their execution.
- The way tasks are generated, the order in which they are executed, or whether they need to exchange data, differentiate the application models that fall under the umbrella of **task programming (Task Computing)**.
- This chapter characterizes the abstraction of a task and provides a brief overview of the distributed application models that are based on the task abstraction.

Workflow applications with task dependencies

- Workflow applications are characterized by a collection of tasks that exhibit dependencies among them. Such dependencies, which are mostly data dependencies (i.e., the output of one task is a pre- requisite of another task), determine the way in which the applications are scheduled as well as where they are scheduled. Concerns in this case are related to providing a feasible sequencing of tasks and to optimizing the placement of tasks so that the movement of data is minimized.
- The workflow depicted in the Figure describes the general process for composing a mosaic; the labels on the right describe the different tasks that have to be performed to compose a mosaic. In the case presented in the diagram, a mosaic is composed of seven images. The entire process can take advantage of a distributed infrastructure for its execution, since there are several operations that can be performed in parallel. For each of the image files, the following process has to be performed: image file transfer, reprojection, calculation of the difference, and common plane placement. Therefore, each of the images can be processed in parallel for these tasks. Here is where a distributed infrastructure helps in executing workflows.

What is a workflow?

A sample Montage Scientific Workflow



What is a workflow?

- The term workflow has along tradition in the business community, where the term is used to describe a composition of services that altogether accomplish a business process. As defined by the Workflow Management Coalition, a workflow is the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant (a resource; human or machine) to another for action, according to a set of procedural rules [64].
- The concept of workflow as a structured execution of tasks that have dependencies on each other has demonstrated itself to be useful for expressing many scientific experiments and gave birth to the idea of scientific workflow. Many scientific experiments are a combination of problem-solving components, which, connected in a particular order, define the specific nature of the experiment.
- When such experiments exhibit a natural parallelism and need to execute a large number of operations or deal with huge quantities of data, it makes sense to execute them on a distributed infrastructure.
- In the case of scientific workflows, the process is identified by an application to run, the elements that are passed among participants are mostly tasks and data, and the participants are mostly computing or storage nodes.

What is a workflow?

- The set of procedural rules is defined by a workflow definition scheme that guides the scheduling of the application. A scientific workflow generally involves data management, analysis, simulation, and middleware supporting the execution of the workflow. A scientific workflow is generally expressed by a directed acyclic graph (DAG), which defines the dependencies among tasks or operations.
- The nodes on the DAG represent the tasks to be executed in a workflow application; the arcs connecting the nodes identify the dependencies among tasks and the data paths that connect the tasks.
- The most common dependency that is realized through a DAG is data dependency, which means that the output filesof a task (or some of them) constitute the input files of another task. This dependency is represented as an arc originating from the node that identifies the first task and terminating in the node that identifies these task.

What is a workflow?

- The DAG in Figure 7.6 describes a sample Montage workflow. Montage is a tool kit for assembling images into mosaics; it has been specially designed to support astronomers in composing the images taken from different telescopes or points of view into a coherent image.The toolkit provides several applications for manipulating images and composing them together; some of the applications perform background re-projection, perspective transformation, and brightness and color.

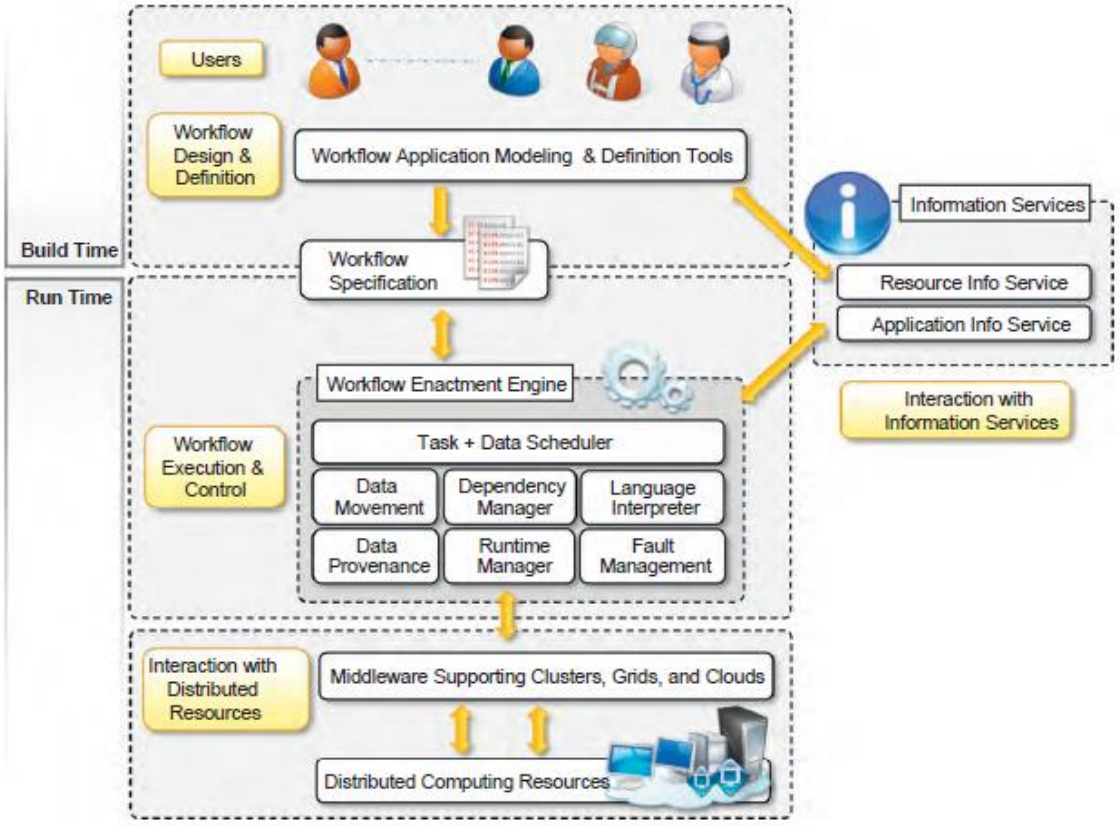
What is a workflow?

- There might be another reason for executing workflows on a distributed infrastructure: It might be convenient to move the computation on a specific node because of data locality issues.
- For example, if an operation needs to access specific resources that are only available on a specific node, that operation cannot be performed elsewhere, whereas the rest of the operations might not have the same requirements.
- A scientific experiment might involve the use of several problem-solving components that might require the use of specific instrumentation; in this case all the tasks that have these constraints need to be executed where the instrumentation is available, thus creating a distributed execution of a process that is not parallel in principle.

Workflow technologies

- Business- oriented computing workflows are defined as compositions of services, and there are specific languages and standards for the definition of workflows, such as Business Process Execution Language (BPEL) [65]. In the case of scientific computing there is no common ground for defining workflows, but several solutions and workflow languages coexist[66].
- Despite such differences, it is possible to identify an abstract reference model for a workflow management system[67], as depicted in Figure7.7.
- Design tools allow users to visually compose a workflow application. This specification is normally stored in the form of an XML document based on a specific workflow language and constitutes the input of the workflow engine, which controls the execution of the workflow by leveraging a distributed infrastructure.
- In most cases, the workflow engine is a client-side component that might interact directly with resources or with one or several middleware components for executing the workflow. Some frameworks can natively support the execution of workflow applications by providing a scheduler capable of directly processing the workflow specification.

Abstract model of a workflow system



Architectural Overview

Workflow Engine in the Cloud

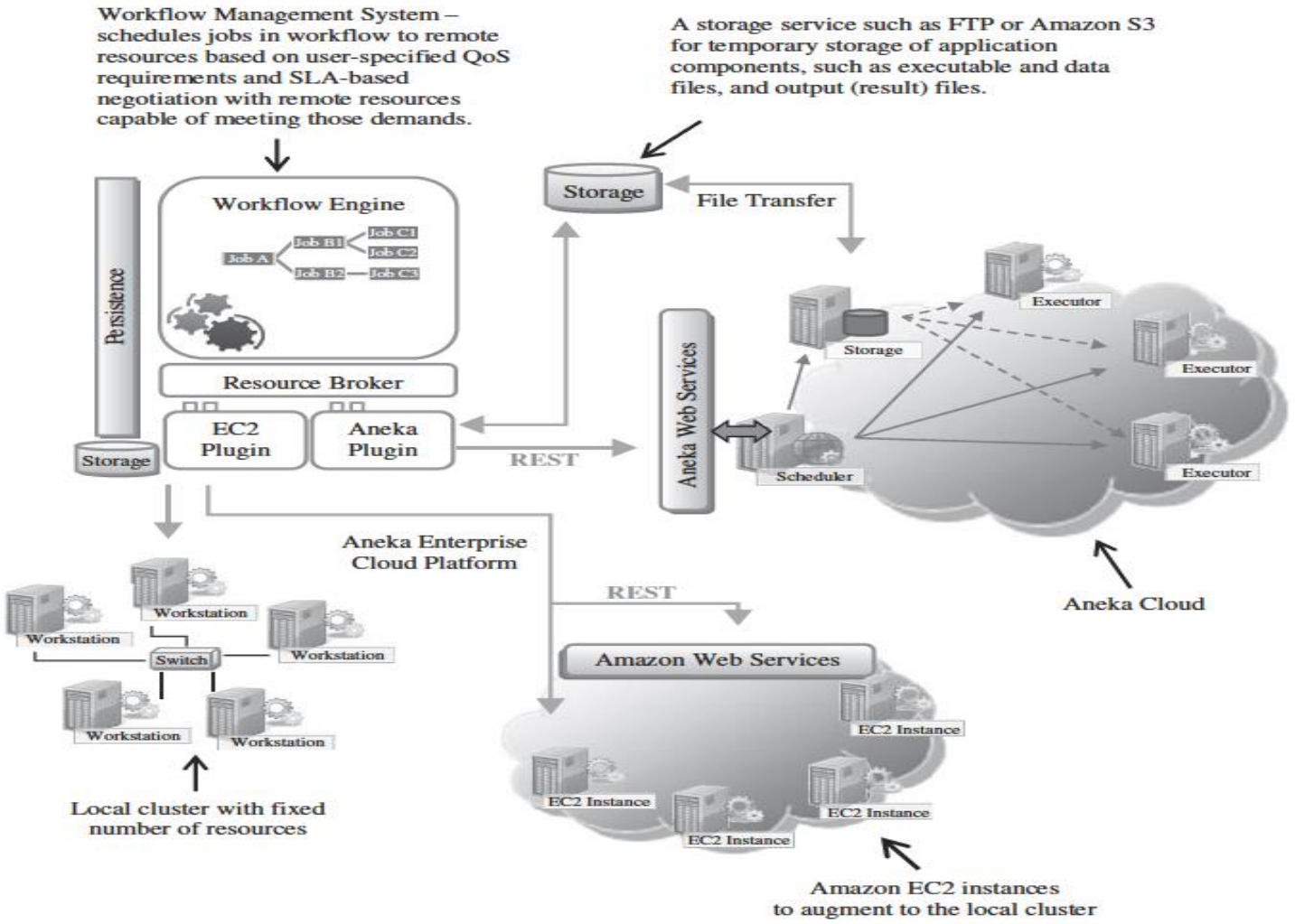


FIGURE 12.1. Workflow engine in the cloud.

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>=2 is the number of objectives and the set is the feasible set of decision vectors defined by constraint functions.

$$\begin{aligned} &\min \left(f_1(x), f_2(x), \ldots, f_k(x)\right)^T \\ &\text{s.t. } x \in X, \end{aligned}$$

WORKFLOW ENGINE FOR CLOUDS
CASE STUDY: EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATIONS

- 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**.

WORKFLOW ENGINE FOR CLOUDS

CASE STUDY: EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATIONS

Objectives

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.

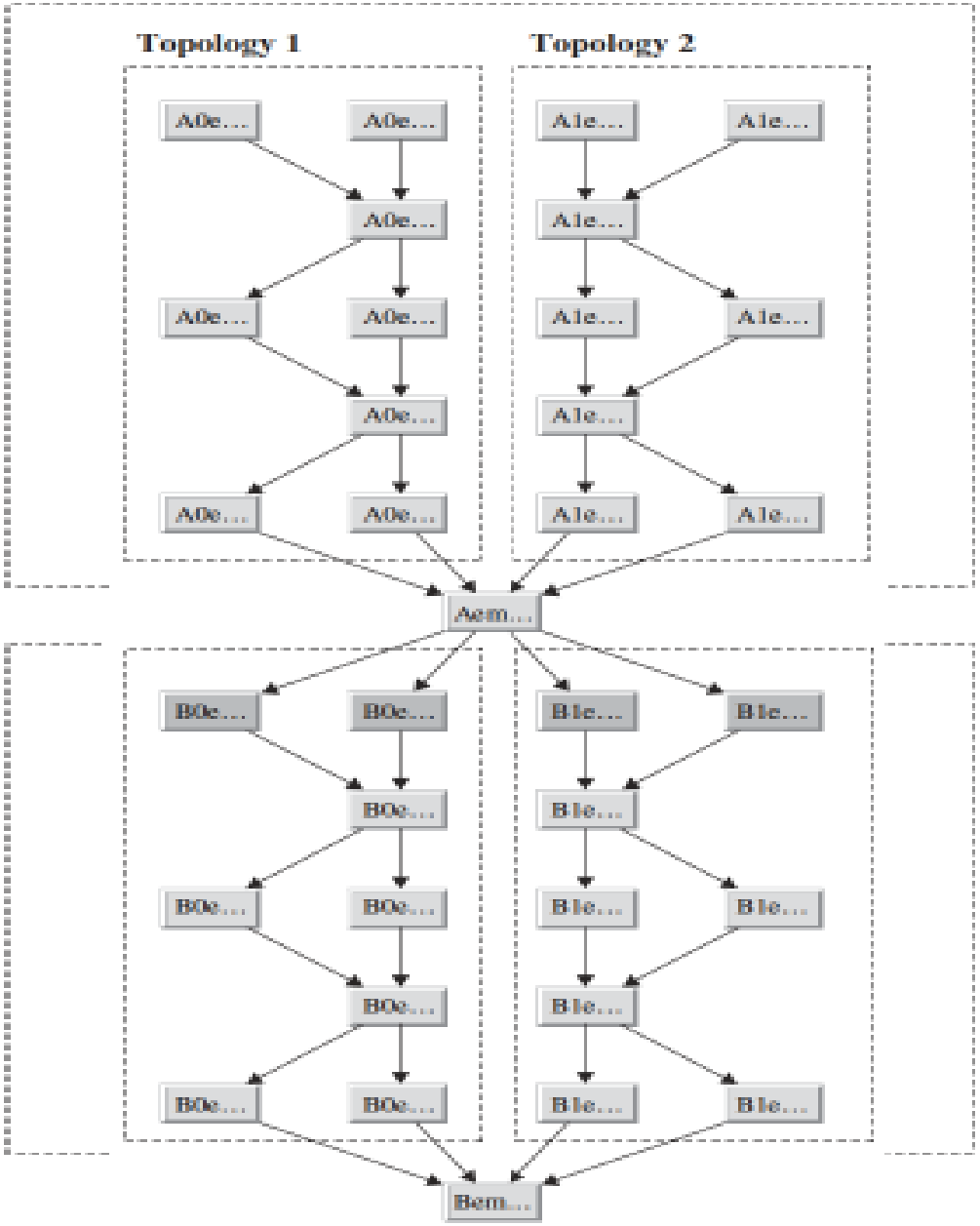


FIGURE 12.6. EMO workflow structure (boxes represent task, arrows represent data dependencies between tasks).

WORKFLOW ENGINE FOR CLOUDS

CASE STUDY: EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATIONS

Workflow Solution

- 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.”

WORKFLOW ENGINE FOR CLOUDS

CASE STUDY: EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATIONS

Workflow Solution

- We see two “emomerge” tasks in the 2nd level, one task in the 6th level that merges two branches and then splits the population to two branches again, two tasks on the 8th and 10th levels, and the final task on the 12th 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.

WORKFLOW ENGINE FOR CLOUDS

CASE STUDY: EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATIONS

Deployment and Results

- **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.

WORKFLOW ENGINE FOR CLOUDS

CASE STUDY: EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATIONS Experiments

TABLE 12.1. Characteristics of Amazon Compute Resources (EC2) Used in Our Experiment

Characteristics	Aneka Master/Worker	Workflow Engine
Platform	Windows 2000 Server	Linux
CPU (type)	1 EC2 Compute Units[a] (small)	5 EC2 Compute Units[b] (medium)
Memory	1.7 GB	1.7 GB
Instance storage	160 GB	350GB
Instance location	US east 1a (19) US east 1b(20)	US east 1a
Number of instances	39	1
Price per hour	$US 0.12	$US 0.17

[a]Small instance (default) 1.7 GB of memory, 1 EC2 compute unit (1 virtual core with 1 EC2 compute unit), 160 GB of instance storage, 32 bit platform.
[b]High CPU medium instance 1.7 GB of memory, 5 EC2 compute units (2 virtual cores with 2.5 EC2 compute units each), 350 GB of instance storage, 32 bit platform.
Source: Amazon.

Experiment 1: Seven Additional EC2 Instances ere 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 instantiate **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.

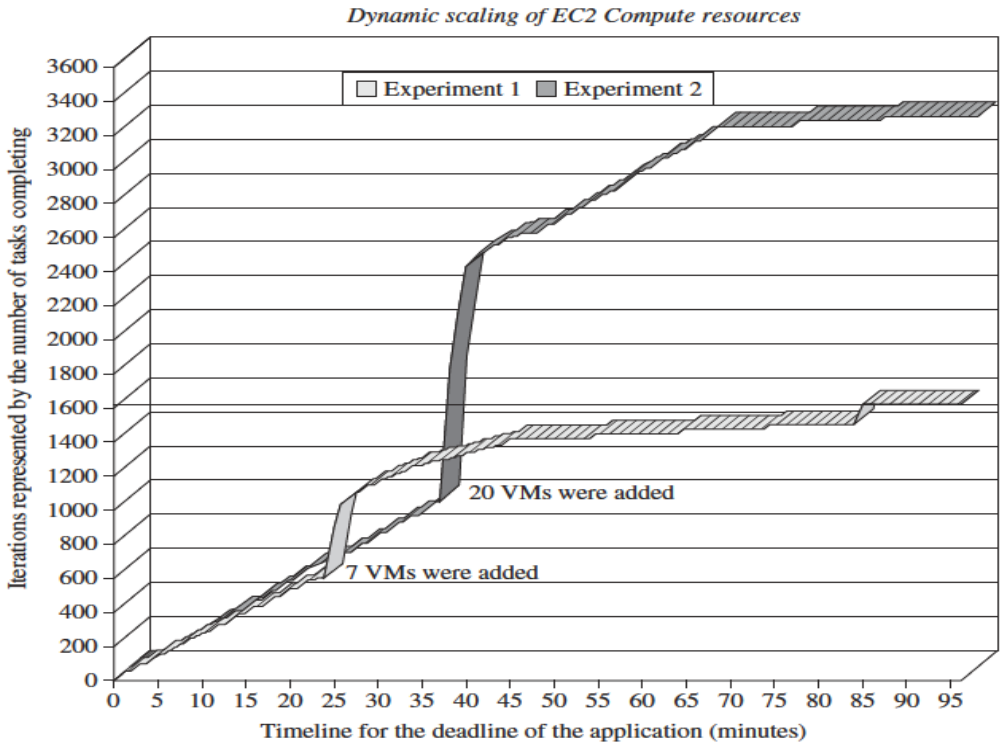


FIGURE 12.8. Number of tasks completing in time as the number of compute resources provisioned were increased at runtime.

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.