Scheduling Scientific Workflows Elastically for Cloud Computing

Cui Lin
California State University, Fresno
clin@csufresno.edu

Shiyong Lu
Wayne State University
shiyong@wayne.edu

***Abstract*—Most existing workflow scheduling algorithms only consider a computing environment in which the number of compute resources is bounded. Compute resources in such an environment usually cannot be provisioned or released on demand of the size of a workflow, and these resources are not released to the environment until an execution of the workflow completes. To address the problem, we firstly formalize a model of a Cloud environment and a workflow graph representation for such an environment. Then, we propose the SHEFT workflow scheduling algorithm to schedule a workflow elastically on a Cloud computing environment. Our preliminary experiments show that SHEFT not only outperforms several representative workflow scheduling algorithms in optimizing workflow execution time, but also enables resources to scale elastically at runtime.**

I. Introduction

Due to the complexity of scientific processes, scientific workflows have become increasingly compute and data intensive [1]. These scientific workflows are often required to be executed in a distributed high-end computing environment, such as recently emerging Cloud computing environments. A Cloud computing environment can provide workflows several features that are distinct from other computing environments: (1) Compute resources in Cloud are exposed as services that provide a standardized interface for services to access over the network; (2) The number and type of compute resources assigned to a workflow are determined by service requests; (3) The number of resources assigned to a workflow can be dynamically changed at runtime, so workflow compute resources can be *elastically* scaled on demand; (4) Not all requested compute resources need to be assigned at the beginning of a workflow execution. Resources can be assigned only when they are needed.

In such a Cloud environment, the number of assigned resources to a workflow can be elastically scaled by service requests. Even though there have been much work on workflow scheduling in the literature, most existing solutions address the problem by assigning a workflow to a bounded number of resources. The number of resources cannot be automatically determined on demand of the size of a workflow, and the resources assigned to a workflow usually are not released until the workflow completes an execution. As a result, resources assigned to a workflow sometimes may become insufficient for the execution of a workflow, which leads to a long execution duration; on another occasion, many resources might become idle for a long time during workflow execution, which leads to a waste of resources and budgets. To address these scheduling problems, we firstly formalize a model of a Cloud computing environment and a workflow graph representation for such an environment, followed by a formalization of the workflow scheduling problem. Then the SHEFT workflow scheduling algorithm is proposed to schedule workflows in a Cloud computing environment. Our preliminary experiments show that SHEFT not only outperforms HEFT [2], one of representative workflow scheduling algorithms in optimizing workflow execution time, but also enables resources to scale elastically during workflow execution.

II. Proposed Scheduling Algorithms

We firstly model a Cloud computing environment by partitioning all resources into a number of clusters. Resources with the same computing capability are grouped into one cluster. Resources within one cluster usually share the same network communication, so they have the same data transfer rate with each other within this cluster. Also, resources within one cluster have the same data transfer rate to resources in another cluster. The model accommodates both heterogeneous and homogeneous computing environments, in terms of computing capability and data communication.

Next, we formalize a scientific workflow that consists of a set of tasks and a set of data dependencies between these tasks. Then a weighted directed acyclic graph is modeled to represent a workflow in a computing environment, in which the vertices of the graph represent a set of tasks, the edges of the graph represent a set of data dependencies, the communication cost is determined by the weight of edge in the graph, and the task computation cost is determined by the weight of vertex in the graph.

Based on the above models, several important notions for scheduling workflows can be formalized as follows: The *earliest ready time* of a task T_i, denoted by $ERT(T_i, R_m)$, is the earliest time when all predecessors of T_i have completed their executions and all input data have arrived at a resource R_m. The earliest start time of T_i on a resource R_m, denoted by $EST(T_i, R_m)$, can be set as the earliest time that R_m is available for a task execution ($EST(T_i, R_m) \geq ERT(T_i, R_m)$). A workflow makespan ($WMS$), the total

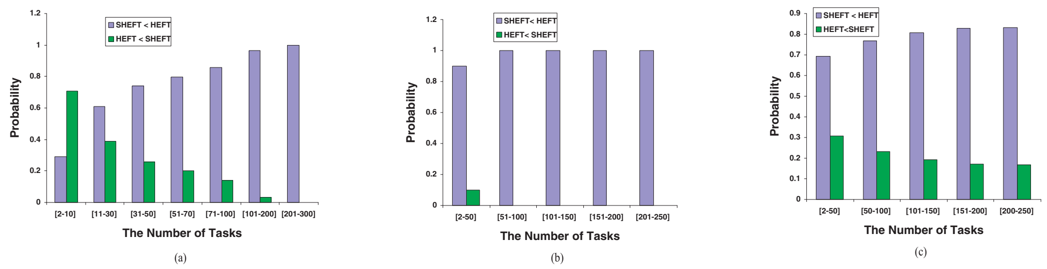

Figure 1. Scheduling 50,000 randomly generated workflows for the number of tasks in each range. The comparison of scheduling results between SHEFT and HEFT on (a) large-scale workflows; (b) compute-intensive workflows; (c) data-intensive workflows.

completion time of a workflow, is determined by the *task finish time* (TFT) of the last task of a workflow to be executed. Therefore, the scheduling problem can be formally stated as follows: Given a workflow in a computing environment, the workflow can be represented by a weighted directed acyclic graph. A workflow schedule is to assign each task of a workflow to a compute resource in a Cloud environment and order the execution of these tasks, such that a workflow makespan can be minimized.

Our solution to the scheduling problem consists of two phases: a *task prioritizing* phase and a *resource selection* phase. In the task prioritizing phase, we propose a task prioritizing algorithm to rank the order of tasks by its priority rank. The algorithm forms a list of prioritized tasks, denoted as $List_{Priority}$.

In the resource selection phase, we propose the SHEFT algorithm (Scalable-Heterogeneous-Earliest-Finish-Time algorithm) to schedule workflows for a Cloud computing environment. SHEFT is an extension of the HEFT algorithm [2], which is applied for mapping a workflow application to a bounded number of processors. At the beginning of the scheduling procedure, any resources can be assigned to a task, and a task with the highest priority rank from $List_{Priority}$ is selected to be scheduled. For each task T_i, the earliest start time $EST(T_i, R_k)$ and earliest finish time $EFT(T_i, R_k)$ on each assigned resource R_k are calculated. The resource that produces the minimized task finish time ($minTFT$) is assigned to a temporary variable R_S. If there is at least one resource R_S that is available for T_i by T_i's $ERT(T_i, R_S)$, then R_S is assigned to T_i, and the available time of R_S is reset to $minTFT$. Otherwise, the decision is automatically determined by the following cases: if $EFT(T_i, R_n)$ is earlier than $minTFT$, then R_n can be assigned to T_i, and will be available to be assigned to other tasks after $EFT(T_i, R_n)$. The number of workflow resources are elastically scaled out in this case; However, if $EFT(T_i, R_n)$ on any new resource is later than $minTFT$, then T_i is assigned to R_S. After each task assignment, a resource R_k that has been kept idle longer than a given threshold t_{idle} will be released from the assigned workflow resources. In this case, the number of workflow resources can be elastically scaled in.

To evaluate our proposed SHEFT algorithm, we firstly simulate a Cloud computing environment with the following input parameters: compute resources $|R_E| = 100$, cluster number $|C_E| = 3$, idle threshold $t_{idle} = 60$, all mappings between resource and clusters, and all data transfer rates between clusters that are randomly generated. Then we develop a workflow generator to randomly generate workflow graphs given the following input parameters: the minimum and maximum numbers of the depth of a graph, the minimum and the maximum numbers of vertices at each level, the weight of each vertex, and the weight of each edge.

To investigate the average performance for large-scale workflows, compute-intensive workflows and data-intensive workflows, our developed workflow generator randomly generates $50,000$ workflow graphs. The communication costs and computation costs of these workflows are also randomly generated within reasonable ranges. Then, we schedule these workflows by the HEFT and SHEFT algorithms, and compare workflow makespan by the two algorithms as the size of the workflows increases. The total number that one algorithm outperforms another is counted, divided by the total number of experiments is considered as the probability of this algorithm outperforms the other algorithm. The statistical results in Figure 1 show that SHEFT outperforms HEFT as the size of workflows increases in all cases.

III. Conclusions

To schedule scientific workflows for Cloud computing, we formalized the model of a Cloud computing environment and a scientific workflow for the environment. Based on the models, we proposed the SHEFT workflow scheduling algorithm to schedule workflows given the elastically changing compute resources. Our preliminary experiments showed that SHEFT not only outperforms HEFT in optimizing workflow execution time, but also enables resources to scale elastically during workflow execution.

References

[1] C. Lin and et al., "A reference architecture for scientific workflow management systems and the VIEW SOA solution," *IEEE T. Services Computing*, vol. 2, no. 1, pp. 79–92, 2009.

[2] H. Topcuoglu and et al., "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, pp. 260–274, 2002.