J. Parallel Distrib. Comput. 84 (2015) 51–64

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

An integrated approach to workflow mapping and task scheduling for delay minimization in distributed environments

Daqing Yun [a], Chase Qishi Wu [b,*], Yi Gu [c]

[a] Department of Computer Science, University of Memphis, Memphis, TN 38152, United States
[b] Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, United States
[c] Department of Computer Science, Middle Tennessee State University, Murfreesboro, TN 37132, United States

HIGHLIGHTS

- We study workflow execution dynamics in distributed environments.
- We formulate an optimization problem on workflow mapping and task scheduling.
- We propose an integrated solution to maximize workflow performance.
- The proposed solution is evaluated through simulations and experiments.

ARTICLE INFO

Article history:
Received 13 February 2015
Received in revised form
18 June 2015
Accepted 7 July 2015
Available online 17 July 2015

Keywords:
Scientific workflows
Workflow mapping
On-node scheduling
End-to-end delay

ABSTRACT

Many scientific applications feature large-scale workflows consisting of computing modules that must be strategically deployed and executed in distributed environments. The end-to-end performance of such scientific workflows depends on both the mapping scheme that determines module assignment, and the scheduling policy that determines resource allocation if multiple modules are mapped to the same node. These two aspects of workflow optimization are traditionally treated as two separated topics, and the interactions between them have not been fully explored by any existing efforts. As the scale of scientific workflows and the complexity of network environments rapidly increase, each individual aspect of performance optimization alone can only meet with limited success. We conduct an in-depth investigation into workflow execution dynamics in distributed environments and formulate a generic problem that considers both workflow mapping and task scheduling to minimize the end-to-end delay of workflows. We propose an integrated solution, referred to as Mapping and Scheduling Interaction (MSI), to improve the workflow performance. The efficacy of MSI is illustrated by both extensive simulations and proof-of-concept experiments using real-life scientific workflows for climate modeling on a PC cluster.

1. Introduction

The processing and analysis of simulation or experimental datasets generated in next-generation e-Science require the construction and execution of domain-specific workflows in distributed network environments, such as clusters, grids, or clouds for collaborative research and discovery [23,33]. Such scientific workflows typically consist of many interdependent computing modules,[1] and are managed and executed by either special- or general-purpose workflow engines such as HTCondor/DAGMan [18,26,20], Kepler [35], Pegasus [42,19], Triana [16], Askalon [24] and Sedna [45]. In general, a workflow system first assigns component modules to a set of networked nodes (i.e. mapping[2]) in the deployment phase and then decides the order or priority of module execution (i.e. scheduling) at runtime.

The workflow mapping problem is well known to be NP-complete and non-approximable for planar and bipartite workflow graphs [25], and has been extensively investigated in the literature to minimize the *End-to-end Delay* (ED) or makespan of a work-

* Corresponding author.
E-mail addresses: dyun@memphis.edu (D. Yun), chase.wu@njit.edu (C.Q. Wu), yi.gu@mtsu.edu (Y. Gu).

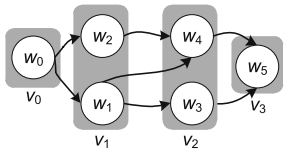
[1] Workflow modules are also referred to as tasks/subtasks, activities, stages, jobs, or transformations in different contexts.

[2] Workflow mapping is occasionally referred to as a scheduling problem in the literature. In this paper, we use the term "mapping" to differentiate it from the workflow on-node scheduling problem under study within the same framework.

http://dx.doi.org/10.1016/j.jpdc.2015.07.004
0743-7315/

Module	CR	Mapped To (PP)
w_0	0	v_0 (---)
w_1	40	v_1 (10)
w_2	10	v_1 (10)
w_3	20	v_2 (10)
w_4	10	v_2 (10)
w_5	0	v_3 (---)

Workflow parameters



Workflow structure & mapping scheme

Fig. 1. A simple example illustrating the interactions between mapping and scheduling.

flow. No matter which mapping scheme is applied, it is often inevitable to assign multiple modules to the same node (i.e. node reuse) for better utilization of limited computing resources, leading to possible concurrent module execution. In such cases, the node's computing resources are generally allocated by kernel-level CPU scheduling policies, such as the round-robin algorithm to ensure fine-grained *fair-share* (FS). Similarly, a network link's bandwidth is also fairly shared by multiple independent data transfers that take place concurrently over the same link through the use of the widely deployed TCP or TCP-friendly transport methods. Such system-inherent fair-share scheduling mechanisms could reduce the development and implementation efforts of workflow systems, but may not always yield the best workflow performance, especially in distributed environments. The scheduling effect was considered in some recent efforts including [52], where a *Critical Path*[3]*-based Priority Scheduling* (CPPS) algorithm is proposed to improve the end-to-end performance of a workflow under a given mapping scheme that is initially calculated based on FS scheduling.

As well recognized, the performance of scientific workflows depends on both the mapping scheme and the on-node scheduling policy, which are traditionally treated in two separate realms. As the scale of scientific workflows and the complexity of network environments rapidly increase, each individual aspect of performance optimization alone can only meet with limited success. In fact, there exists a certain level of interactions between mapping and scheduling, which could be exploited to further improve the end-to-end workflow performance. We shall use a simplified numerical example on a workflow consisting of 6 modules as shown in Fig. 1 to illustrate how mapping and scheduling can interact with each other, leading to different end-to-end performances. For simplicity, we ignore the computational requirement (CR) of the start/end module (i.e. w_0 and w_5). The initial mapping scheme maps modules w_1 and w_2 to node v_1, and maps modules w_3 and w_4 to node v_2. All the nodes have identical *processing power* (PP) of 10 units/s.

Under the given mapping scheme in Fig. 1, the execution dynamics of the entire workflow are shown in Fig. 2 based on FS scheduling. Along the *x*-axis, starting from time point 0, it takes 2 s of fair share between w_1 and w_2 on v_1 to finish w_2 and then another 3 s of exclusive execution of w_1 to finish w_1 at time point 5. At this point, w_3 and w_4 become "ready". It takes 2 s of fair share between w_3 and w_4 on v_2 to finish w_4 and another 1 s of exclusive execution of w_3 to finish w_3 at time point 8. Therefore, the end-to-end delay based on FS scheduling is 8 s on the critical path: $w_0 \rightarrow w_1 \rightarrow w_3 \rightarrow w_5$. Based on this FS schedule, by employing a naive on-node scheduling policy that always executes the critical module exclusively first in case of resource competition, we can cut down the end-to-end delay from 8 to 7 s as shown in Fig. 2. However, this new schedule leads to a different critical path, i.e. $w_0 \rightarrow w_2 \rightarrow w_4 \rightarrow w_5$. Such a shift of the critical path indicates that the original mapping scheme computed by any critical path-based mapping algorithm based on FS scheduling may no longer be the optimal one, hence requiring reexamining the mapping procedure.

Modern systems and networks provide various mechanisms to perform resource scheduling that goes beyond the system-inherent FS scheduling. For example, the allocation of system resources (mainly CPU cycles) on a host among concurrent modules or jobs could be controlled by assigning different CPU quantum values through "nice" or "renice" commands or some specialized tools such as "CGroups" [12] in Linux. Meanwhile, apart from TCP-based data transfer over default best-effort IP paths, the allocation of network resources (mainly bandwidths) could also be controlled on dedicated channels, as exemplified by high-performance networks with the capability of bandwidth reservation [41,32] and the Internet using QoS technologies such as DiffServ [5], IntServ [8], RSVP [9], and MPLS [2] to implement fine-grained bandwidth control. Therefore, it is practically feasible to perform more sophisticated control for module execution and data transfer than the default FS scheduling to achieve a higher level of workflow performance.

In this paper, we conduct an in-depth investigation into workflow execution dynamics and formulate a generic workflow optimization problem that considers both workflow mapping and scheduling. We propose an integrated solution, referred to as *Mapping and Scheduling Interaction* (MSI), to optimize workflow performance. MSI takes an iterative approach to perform the mapping-scheduling cycle in order to minimize the end-to-end delay of a distributed workflow. The performance superiority of the proposed solution is illustrated by extensive simulations and proof-of-concept experiments using domain-specific scientific workflows in real network environments.

The rest of this paper is organized as follows. Section 2 conducts a survey of related work. Section 3 formulates the workflow optimization problem. Section 4 details the design of MSI algorithm. Section 5 evaluates the algorithm performance through simulations and experiments. Section 6 concludes our work.

2. Related work

We conduct a background survey on workflow optimization, which has been the focus of research in recent years [21,52,53,15]. Generally, there are two aspects of optimizing distributed tasks to improve scientific workflow performance: (i) assigning the component modules in a workflow to suitable computing resources, referred to as workflow mapping; and (ii) deciding the execution order/priority and resource sharing policy among concurrent modules on computer nodes or processors, referred to as workflow on-node scheduling. Both aspects have been extensively studied in various contexts due to their theoretical significance and practical importance [11,36,14,13,44,38,43,3,10,37].

The existing workflow mapping algorithms can be roughly classified into the following categories: (i) Graph-based methods [40,17], which tackle the mapping problem using graph theory. The subgraph isomorphism has been proven to be NP-complete and the complexity of graph isomorphism is still unknown; (ii) List scheduling approaches [30,34], most of which employ a critical path-based procedure; (iii) Clustering algorithms [7,28], which assume an unlimited number of processors and thus are not feasible in reality; (iv) Duplication-based algorithms [1,43], most of which have a complexity of $O(n^4)$ or higher for n nodes; (v) Guided random search such as genetic algorithms [22,31], which often require additional efforts to determine an appropriate termination condition and usually does not have performance guarantee.

In early years when networked resources were still scarce, workflow modules were typically mapped to tightly coupled homogeneous systems such as multiprocessors [25,51,28,34]. As distributed platforms such as clusters, grids, and clouds are rapidly

[3] The critical path in the context of workflow optimization refers to the longest path of execution time.