Future Generation Computer Systems 74 (2017) 51–62

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A Critical Path File Location (CPFL) algorithm for data-aware multiworkflow scheduling on HPC clusters

César Acevedo *, Porfidio Hernández, Antonio Espinosa, Víctor Méndez

Computer Architecture & Operating Systems Department (CAOS), Universitat Autònoma de Barcelona (UAB), Bellaterra (Barcelona), Spain

HIGHLIGHTS

- Multiworkflow scheduling strategy on a cluster is proposed.
- Critical path with data-aware.
- Scheduling is proposed to improve makespan of bioinformatic workflows.
- Simulator engine extension to scale on to a bigger cluster infrastructure and new storage hierarchy.

ARTICLE INFO

Article history:
Received 6 September 2016
Received in revised form
12 April 2017
Accepted 13 April 2017
Available online 24 April 2017

Keywords:
Multiworkflows
Cluster
Scheduler
Simulation
Critical path
Data processing

ABSTRACT

A representative set of workflows found in bioinformatics pipelines must deal with large data sets. Most scientific workflows are defined as Direct Acyclic Graphs (DAGs). Despite DAGs are useful to understand dependence relationships, they do not provide any information about input, output and temporal data files. This information about the location of files of data intensive applications helps to avoid performance issues.

This paper presents a multiworkflow store-aware scheduler in a cluster environment called Critical Path File Location (CPFL) policy where the access time to disk is more relevant than network, as an extension of the classical list scheduling policies. Our purpose is to find the best location of data files in a hierarchical storage system.

The resulting algorithm is tested in an HPC cluster and in a simulated cluster scenario with bioinformatics synthetic workflows, and largely used benchmarks like Montage and Epigenomics. The resulting simulator is tuned and validated with the first test results from the real infrastructure. The evaluation of our proposal shows promising results up to 70% on benchmarks in real HPC clusters using 128 cores and up to 69% of makespan improvement on simulated 512 cores clusters with a deviation between 0.9% and 3% regarding the real HPC cluster.

1. Introduction

Current scientific applications must deal with large data sets, usually demanding large amounts of computation and communication times. Schedulers are responsible for allocating applications to processors and ensure the execution precedence. Applications in a workflow are usually represented as a node in a graph.

Direct Acyclic Graphs (DAGs) are a good way of modeling task dependency relationships like those typically found in a workflow. Despite their wide usage to represent application stages, DAGs lack relevant information on how to deal with data files. That is, they do not show any detail on how input, output, and temporal data files are transferred to actual computational nodes where the applications run.

Fig. 1 shows an example of the many combinations that can be found when considering the possible locations of data files and a common computational cluster system architecture. Cluster nodes have their own memory hierarchy and their own secondary storage subsystem where we can find hard drive disks and other smaller but faster general purpose storage device. Also, nodes are usually connected to a distributed file system via a fast interconnection network. From the point of view of the data handling, workflow stages need to manage input, output, and many temporal files [1]. It becomes a challenge to determine which is the best location for all the files needed for the different computation steps defined in the workflow to get the best performance of the system.

* Corresponding author.
E-mail address: cesar.acevedo@caos.uab.es (C. Acevedo).

http://dx.doi.org/10.1016/j.future.2017.04.025

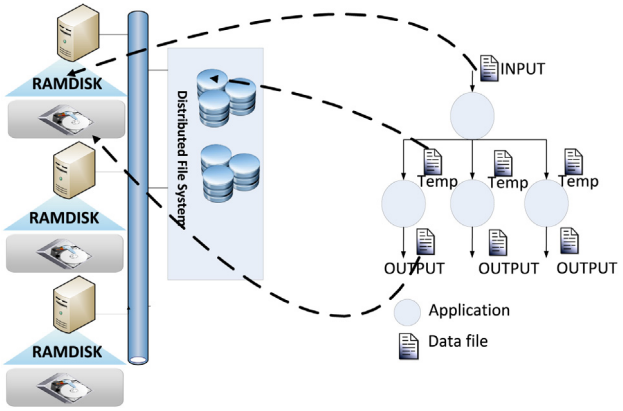

Fig. 1. Application graph and its relationship with data localization.

Scheduling a workflow with precedence constraints is an important problem in scheduling theory and has been shown to be NP-Hard [2]. There are many studies on how to manage a single workflow, specifically when trying to schedule tasks onto heterogeneous domains [3]. There has been an increasing interest in executing several workflows simultaneously. The problem of defining which application from the multiple workflows is going to be executed in a specific node of a cluster, has been described in several works like [4–12].

Workflow-aware storage strategies study data file locations in many levels of the storage and the memory hierarchy as relevant criteria for the application scheduling. These strategies have previously been used to reduce the I/O load of the network of cloud [13] and grid systems. In these systems, when shared files are read several times between different computational nodes, the performance of the system depends on the interconnection network capacity. In these cases, shared files are located on local disks to reduce the amount of I/O operations. In a cluster environment with a high-speed network the disk becomes the demanding resource, which generates I/O waiting times when many applications simultaneously request the access to files. In HPC systems I/O performance has been studied by [14] that proposes the use of Ramdisk as a storage system with data location techniques on a section of the memory system.

When considering the data usage of common bioinformatics workflows for common data analysis cases as variant analysis, read mapping and sequence alignment, we find some common special characteristics:

- Large volume of input data to be processed, starting at 2 GB for bioinformatics data files.
- Large volume of data are sequentially processed in their entirety, like input files for read mapping and data format transformation.
- Important amount of data being shared by similar applications: typically files like human genome indexes.
- Amount and volume of temporal files generated by some applications. Input files of 2 GB can generate between 4 and 6 GB of temporal files.

Due to the peculiarity of bioinformatics workflows that are composed of many applications that share data files, we can take advantage of keeping a cached version of input and temporal files. Then, these files are kept in a highly accessible storage such as ramdisk built in the main memory of the system or in a solid state disk. From here, we propose an extension of the classic model to a shared input file policy of execution and mapping of workflow applications on this kind of system architectures.

In summary, we proposed at [15] a scheduling algorithm for multiworkflows in a cluster environment called CPFL (Critical Path File Location) that is a continuation of the work presented at [16]. Our objective is to improve the effective use of High Performance Computing (HPC) platforms for the execution of Data Intensive Applications (DIC) by extending the multiworkflow model on to store-aware scheduling. Multiworkflow such as bioinformatic and Epigenomics use shared input and temporal data files. Typical bioinformatic workflow has input files starting at 2 GB that generates temporal files of 6 GB. We realize that other workflows such Montage [17] generate several shared temporal data files. For a small Montage, 200 input files of 2 GB generate over 1000 shared temporal data files of up to 500 MB each. Due to that, store-aware scheduling approach on cluster environment helps to improve overall makespan. Keeping shared files on a storage hierarchy system we reduce the time access to disk on regards to network access.

We evaluate the effect of moving the execution of certain tasks to nodes where needed data items are previously located. For bioinformatics applications with input files in common, we move those files to a fast memory storage level. As a result, we increase locality and reduce the number of disk accesses. We also want to support the execution of different workflows considering their main resource limitations like Input/Output (I/O), memory or CPU. To consider new technologies and different kind of workflows we introduce a simulator and the extension needed to deploy the scheduler on it.

This approach has been evaluated with an initial set of experimental environments: a batch of workflows statically merged into a meta-workflow and then applied a classic List Scheduling like Heterogeneous Earliest Finish Time (HEFT). This heuristic is typically used to schedule a set of dependent tasks onto a network of heterogeneous workers taking computation time into account. In our case, we have introduced the use of a Network File System (NFS) as the storage system for all the data files. This is compared against a new List Scheduling heuristic for data-aware multiworkflows with a critical path using a local disk and local Ramdisk as storage hierarchy. In our experiments, we use synthetic bioinformatics workflows as a benchmark to test our proposals as well as Montage and Epigenomics benchmarks because they typically produce plenty of temporal files. The results show performance improvements up to 70% against HEFT modified for multiworkflow with better usage of storage hierarchy such as local disk and ramdisk.

The rest of the paper is organized as follows. State of the art is discussed in Section 2. Then, we give an overview of the scheduler architecture in Section 3. Section 4 describes WorkFlowSim simulation environment and introduces its use to validate schedulers scalability. Section 5 elaborates the experiment design and evaluates the performance of proposed algorithm in the experimental platforms presented. Finally, in Section 6 we summarize the results obtained and lay out the future work.

2. Related work

The scheduling problem, understood as the task of allocating computational jobs to processors to define their order of execution without restrictions, is NP-complete [2]. As a relevant problem, many heuristics have been proposed for its resolution [18]. Among the most important we can find: clustering heuristics [19], duplication based heuristics [20], meta heuristics (Genetic Algorithms, Simulated Annealing, tabu search) and list scheduling heuristics [21] based in assigning priorities depending on the critical path length associated to each node [3].

In our work, we are considering a generalization of the problem taking scientific workflow tasks as jobs to manage. This case of workflow scheduling has been widely studied and we can find many algorithms based on DAG list scheduling heuristics. Some