Performance Summary of RADICAL-Pilot YARN

Nikhil Shenoy

May 10, 2016

Abstract

While RADICAL-Pilot and the EnsembleMD Toolkit are self-contained software packages, their extensibility allows developers to utilize the performance benefits for other applications as well. For example, the areas of bio-molecular dynamics and genomics require capabilities for handling compute-intensive and data-intensive tasks, but RADICAL-Pilot can provide only some of this functionality. These fields require a combination of the best techniques from high performance computing and from data processing platforms like Hadoop in order to achieve good results. For this reason, RADICAL-Pilot has been extended to include Hadoop and the associated YARN resource management system to take advantage of RADICAL-Pilot's high performance computing applications and Hadoop's data management capabilities. In this paper, we compare the performance of RADICAL-Pilot with RADICAL-Pilot extended with YARN to demonstrate the usefulness of the additional functionality.

1 Introduction

RADICAL-Pilot is a Python API developed by the RADICAL-Cybertools group that aids developers in submitting and running batches of tasks on high performance machines. The API achieves this through a container called a Pilot; this container is assigned the information associated with each task in the batch, such as the location of input data and what simulation to run, and is then placed in the queue of an HPC machine. Once the scheduler on the HPC machine schedules the Pilot onto a resource, the Pilot will then start its own Agent to begin carrying out the batch of tasks. It aggregates all the necessary resources, and then pulls additional information about each task from MongoDB. The Agent is responsible for carrying out the execution for each task and making sure that the output is sent back to the user.

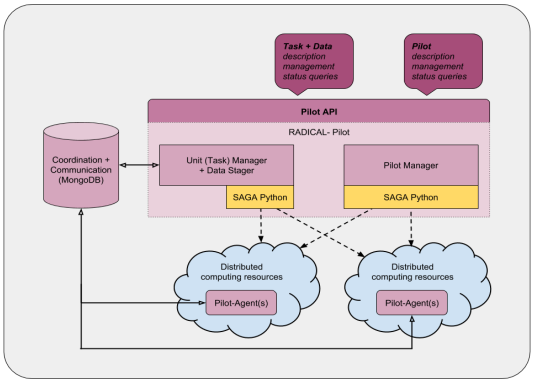

Figure 1: The RADICAL-Pilot Architecture [1]

This method of scheduling tasks provides many advantages, one of which is circumventing the scheduler. In current submission scenarios, each task must occupy a place in the scheduling queue and must individually wait for the required resources to become available before executing. This greatly increases the total time spent in the queue for the collection of tasks, which also increases the time to completion. However, Pilots allow users to avoid this time penalty by taking advantage of late binding and sending only the Pilot through the scheduler, reducing the wait time to that of a single task. Once the Pilot is finally scheduled, the user can then employ the various execution styles provided by the API based on the nature of their application. For example, if the batch is split into two types of tasks where one must occur before the other, the API provides functionality to schedule

and execute such chained tasks. This, and the other options provided by RADICAL-Pilot, presents the user with a simple but powerful interface for task execution that outstrips current methods. By permitting any executable to be associated with a task in a Pilot, the RADICAL-Pilot API is flexible enough to handle a variety of HPC tasks, making it ideal for users who regularly work on the order of thousands of simulations.

While RADICAL-Pilot offers an improvement in efficiency for HPC tasks, it is not designed to handle data-intensive tasks. However, Apache Hadoop and YARN are able to work on problems involving large amounts of data. YARN in particular plays the roles of resource manager and scheduler in this context, and contributes the main functionality for wrestling with large volumes of data [2]. RADICAL-Pilot has been extended with YARN in order present an iterface which has performs well on HPC and data-intensive tasks. The extension has been implemented at the level of RADICAK-Pilot's Agent, the entity responsible for coordinating execution on the remote machine, within several components. The Local Resource Manager now retrieves new environment variables that detail the number of cores to be used on each node and the assignment of nodes, among other parameters. It then passes this information on to the newly started Hadoop and YARN demons, which examine and record the current state of the cluster. The RADICAL-Pilot scheduler has also been updated to include information about the current state of the cluster, including updates on the total memory available and the total number of cores in use. The scheduler then uses this state information to schedule the next task appropriately. Finally, the Application Manager, which handles the resource allocations, works with the YARN Resource Manager in order to coordinate the execution of tasks. RADICAL-Pilot provisions a Compute Unit with a Description that contains the resource requirements, and then requests that YARN create a container for it. By placing the Compute Unit within the YARN container, the YARN scheduler can then easily assign the container the optimal resources for execution [3].

YARN Image here

These extensions to RADICAL-Pilot give birth to they RADICAL-Pilot-YARN package, which contains functionality to operate on HPC and data-intensive tasks. In this paper, we will examine the performance of RADICAL-Pilot-YARN in comparison to RADICAL-Pilot by itself. We will do this through a simulation with a simple clustering algorithm.

2 Experiments

3 Conclusion

4 Future Work

5 Lessons Learned

References

[1] RADICAL-Cybertools, "Radical-pilot architecture," 2016. [Online]. Available: https://radical-cybertools.github.io/

[2] A. S. Foundation, "Apache hadoop yarn," 2016. [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[3] A. Luckow, I. Paraskevakos, G. Chantzialexiou, and S. Jha, "Hadoop on hpc: Integrating hadoop and pilot-based dynamic resource management," *arXiv*, 2016. [Online]. Available: http://arxiv.org/abs/1602.00345