REEF

PRATIK JAIN

School of Informatics and Computing, Bloomington, IN 47408, U.S.A.

Corresponding authors: jainps@iu.edu

Paper-2, March 22, 2017

Apache REEF is a Big Data system that makes it easy to implement scalable, fault-tolerant runtime environments for a range of data processing models on top of resource managers such as Apache YARN and Mesos. The key features and abstractions of REEF are discussed. Two libraries of independent value are introduced. Wake is an event-based-programming framework and Tang is a dependency injection framework designed specifically for configuring distributed systems.

© 2017 https://creativecommons.org/licenses/. The authors verify that the text is not plagiarized.

Keywords: REEF, Tang, Wake

https://github.com/pratik11jain/sp17-i524/blob/master/paper2/S17-IR-2012/report.pdf

INTRODUCTION

With the continuous growth of Hadoop the range of computational primitives expected by its users has also broadened. A number of performance and workload studies have shown that Hadoop MapReduce is a poor fit for iterative computations, such as machine learning and graph processing. Also, for extremely small computations like ad-hoc queries that compose the vast majority of jobs on production clusters, Hadoop MapReduce is not a good fit. Hadoop 2 addresses this problem by factoring MapReduce into two components: an application master that schedules computations for a single job at a time, and YARN, a cluster resource manager that coordinates between multiple jobs and tenants. In spite of the fact that this resource manager, YARN, allows a wide range of computational frameworks to coexist in one cluster, many challenges remain [1]. Apache REEF (Retainable Evaluator Execution Framework), a library for developing portable applications for cluster resource managers such as Apache Hadoop YARN or Apache Mesos, addresses these challenges.

FEATURES

Due to the following critical features, Apache REEF drastically simplifies development of resource managers [2].

Centralized Control Flow

Apache REEF turns the chaos of a distributed application into various events in a single machine. These events include container allocation, Task launch, completion, and failure.

Task runtime

Apache REEF provides a Task runtime which is instantiated in every container of a REEF application and can keep data in memory in between Tasks. This enables efficient pipelines on REEF.

Support for multiple resource managers

Apache REEF applications are portable to any supported resource manager with minimal effort. In addition to this, new resource managers are easy to support in REEF.

.NET and Java API

Apache REEF is the only API to write YARN or Mesos applications in .NET. Additionally, a single REEF application is free to mix and match tasks written for .NET or Java.

Plugins

Apache REEF allows for plugins to augment its feature set without hindering the core. REEF includes many plugins, such as a name-based communications between Tasks, MPI-inspired group communications, and data ingress.

As a result of such features Apache REEF shows properties like retainability of hardware resources across tasks and jobs, composability of operators written for multiple computational frameworks and storage backends, cost modeling for data movement and single machine parallelism, fault handling [3] and elasticity.

KEY ABSTRACTIONS

REEF is structured around the following key abstractions [4]:

Driver: This is a user-supplied control logic that implements the resource allocation and Task scheduling logic. There is exactly one Driver for each Job. The duration and characteristics of the Job are determined by this module.

Task: This encapsulates the task-level client code to be executed in an Evaluator.

Evaluator: This is a runtime environment on a container that can retain state within Contexts and execute Tasks (one at a time). A single evaluator may run many activities throughout its lifetime. This enables sharing among Activities and reduces scheduling costs.

Context: It is a state management environment within an Evaluator that is accessible to any Task hosted on that Evaluator.

Services: Objects and daemon threads that are retained across Tasks that run within an Evaluator [1]. Examples include caches of parsed data, intermediate state, and network connection pools.

WAKE AND TANG

The lower levels of REEF can be decoupled from the data models and semantics of systems built atop it. This results in two standalone systems, Tang and Wake which are both language independent and allow REEF to bridge the JVM and .NET.

Tang is a configuration management and checking framework [5]. It emphasizes explicit documentation and automatic checkability of configurations and applications instead of ad-hoc, application-specific configuration and bootstrapping logic. It not only supports distributed, multi-language applications but also gracefully handles simpler use cases. It makes use of dependency injection to automatically instantiate applications. Given a request for some type of object, and information that explains how dependencies between objects should be resolved, dependency injectors automatically instantiate the requested object and all of the objects it depends upon. Tang makes use of a few simple wire formats to support remote and even cross-language dependency injection.

Wake is an event-driven framework based on ideas from SEDA, Click, Akka and Rx [6]. It is general purpose in the sense that it is designed to support computationally intensive applications as well as high-performance networking, storage, and legacy I/O systems. Wake is implemented to support high-performance, scalable analytical processing systems i.e. big data applications. It can be used to achieve high fanout and low latency as well as high-throughput processing and it can thus aid to implement control plane logic and the data plane.

REFERENCES

[1] Byung-Gon Chun, Chris Douglas, Shravan Narayanamurthy, Josh Rosen, Tyson Condie, Sergiy Matusevych, Raghu Ramakrishnan, Russell Sears, Carlo Curino, Brandon Myers, Sriram Rao, Markus Weimer, "Reef: Retainable evaluator execution framework," in *VLDB Endowment, Vol. 6, No. 12*, Aug. 2013. [Online]. Available: http://db.disi.unitn.eu/pages/VLDBProgram/pdf/demo/p841-sears.pdf

[2] The Apache Software Foundation, "Apache reef™ - a stdlib for big data," Web Page, Nov. 2016, accessed 2017-03-15. [Online]. Available: http://reef.apache.org/

[3] Techopedia Inc., "Retainable evaluator execution framework (reef)," Web Page, Jan. 2017, accessed 2017-03-17. [Online]. Available: https://www.techopedia.com/definition/29891/retainable-evaluator-execution-framework-reef

[4] Markus Weimer, Yingda Chen, Byung-Gon Chun, Tyson Condie, Carlo Curino, Chris Douglas, Yunseong Lee, Tony Majestro, Dahlia Malkhi, Sergiy Matusevych, Brandon Myers, Shravan Narayanamurthy, Raghu Ramakrishnan, Sriram Rao, Russell Sears, Beysim Sezgin, Julia Wang, "Reef: Retainable evaluator execution framework," in *Proc ACM SIGMOD Int Conf Manag Data. Author manuscript*, Jan. 2016, pp. 1343–1355. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4724804/

[5] The Apache Software Foundation, "Tang," Web Page, Nov. 2016, accessed 2017-03-15. [Online]. Available: http://reef.apache.org/tang.html

[6] ——, "Wake," Web Page, Nov. 2016, accessed 2017-03-15. [Online]. Available: http://reef.apache.org/wake.html