YARN Essentials

YARN is the next generation generic resource platform used to manage resources in a typical cluster and is designed to support multitenancy in its core architecture. As optimal resource utilization is central to the design of YARN, learning how to fully utilize the available fine-grained resources (RAM, CPU cycles, and so on) in the cluster becomes vital.

This book is an easy-to-follow, self-learning guide to help you start working with YARN. Beginning with an overview of YARN and Hadoop, you will dive into the pitfalls of Hadoop 1.x and how YARN takes us to the next level. You will learn the concepts, terminology, architecture, core components, and key interactions, and cover the installation and administration of a YARN cluster as well as learning about YARN application development with new and emerging data processing frameworks.

Who this book is written for

If you have a working knowledge of Hadoop 1.x but want to start afresh with YARN, this book is ideal for you. You will be able to install and administer a YARN cluster and also discover the configuration settings to fine-tune your cluster both in terms of performance and scalability. This book will help you develop, deploy, and run multiple applications/ frameworks on the same shared YARN cluster.

What you will learn from this book

- Understand how existing MapReduce applications can run on top of YARN and how they are backward compatible
- Explore the YARN concepts, terminologies, architecture, key components, and interaction between the components
- Set up a standalone and multi-node clustered YARN environment
- Design, develop, and run different frameworks such as MapReduce, Apache Storm, Apache Tez, and Giraffe on top of YARN
- Get to grips with the built-in support for multitenancy in YARN
- Discover the motivation behind YARN's architecture design, implementations, and why YARN was needed
- Learn how failures at each level are gracefully handled by the new framework to achieve fault tolerance and scalability


Amol Fasale
Nirmal Kumar
In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 1 "Need for YARN"
- A synopsis of the book’s content
- More information on **YARN Essentials**

### About the Authors

**Amol Fasale** has more than 4 years of industry experience actively working in the fields of big data and distributed computing; he is also an active blogger in and contributor to the open source community. Amol works as a senior data system engineer at MakeMyTrip.com, a very well-known travel and hospitality portal in India, responsible for real-time personalization of online user experience with Apache Kafka, Apache Storm, Apache Hadoop, and many more. Also, Amol has active hands-on experience in Java/J2EE, Spring Frameworks, Python, machine learning, Hadoop framework components, SQL, NoSQL, and graph databases.

You can follow Amol on Twitter at @amolfasale or on LinkedIn. Amol is very active on social media. You can catch him online for any technical assistance; he would be happy to help.

Amol has completed his bachelor's in engineering (electronics and telecommunication) from Pune University and postgraduate diploma in computers from CDAC.

______________________________

The gift of love is one of the greatest blessings from parents, and I am heartily thankful to my mom, dad, friends, and colleagues who have shown and continue to show their support in different ways. Finally, I owe much to James and Arwa without whose direction and understanding, I would not have completed this work.
Nirmal Kumar is a lead software engineer at iLabs, the R&D team at Impetus Infotech Pvt. Ltd. He has more than 8 years of experience in open source technologies such as Java, JEE, Spring, Hibernate, web services, Hadoop, Hive, Flume, Sqoop, Kafka, Storm, NoSQL databases such as HBase and Cassandra, and MPP databases such as Teradata.

You can follow him on Twitter at @nirmal__kumar. He spends most of his time reading about and playing with different technologies. He has also undertaken many tech talks and training sessions on big data technologies.

He has attained his master's degree in computer applications from Harcourt Butler Technological Institute (HBTI), Kanpur, India and is currently part of the big data R&D team in iLabs at Impetus Infotech Pvt. Ltd.

I would like to thank my organization, especially iLabs, for supporting me in writing this book. Also, a special thanks to the Packt Publishing team; without you guys, this work would not have been possible.
YARN Essentials

In a short span of time, YARN has attained a great deal of momentum and acceptance in the big data world.

YARN essentials is about YARN—the modern operating system for Hadoop. This book contains all that you need to know about YARN, right from its inception to the present and future.

In the first part of the book, you will be introduced to the motivation behind the development of YARN and learn about its core architecture, installation, and administration. This part also talks about the architectural differences that YARN brings to Hadoop 2 with respect to Hadoop 1 and why this redesign was needed.

In the second part, you will learn how to write a YARN application, how to submit an application to YARN, and how to monitor the application. Next, you will learn about the various emerging open source frameworks that are developed to run on top of YARN. You will learn to develop and deploy some use case examples using Apache Samza and Storm YARN.

Finally, we will talk about the failures in YARN, some alternative solutions available on the market, and the future and support for YARN in the big data world.

What This Book Covers

Chapter 1, Need for YARN, discusses the motivation behind the development of YARN. This chapter discusses what YARN is and why it is needed.

Chapter 2, YARN Architecture, is a deep dive into YARN's architecture. All the major components and their inner workings are explained in this chapter.

Chapter 3, YARN Installation, describes the steps required to set up a single-node and fully-distributed YARN cluster. It also talks about the important configurations/properties that you should be aware of while installing the YARN cluster.

Chapter 4, YARN and Hadoop Ecosystems, talks about Hadoop with respect to YARN. It gives a short introduction to the Hadoop 1.x version, the architectural differences between Hadoop 1.x and Hadoop 2.x, and where exactly YARN fits into Hadoop 2.x.

Chapter 5, YARN Administration, covers information on the administration of YARN clusters. It explains the administrative tools that are available in YARN, what they mean, and how to use them. This chapter covers various topics from YARN container allocation and configuration to various scheduling policies/configurations and in-built support for multitenancy.
Chapter 6, *Developing and Running a Simple YARN Application*, focuses on some real applications with YARN, with some hands-on examples. It explains how to write a YARN application, how to submit an application to YARN, and finally, how to monitor the application.

Chapter 7, *YARN Frameworks*, discusses the various emerging open source frameworks that are developed to run on top of YARN. The chapter then talks in detail about Apache Samza and Storm on YARN, where we will develop and run some sample applications using these frameworks.

Chapter 8, *Failures in YARN*, discusses the fault-tolerance aspect of YARN. This chapter focuses on various failures that can occur in the YARN framework, their causes, and how YARN gracefully handles those failures.

Chapter 9, *YARN – Alternative Solutions*, discusses other alternative solutions that are available on the market today. These systems, like YARN, share common inspiration/requirements and the high-level goal of improving scalability, latency, fault-tolerance, and programming model flexibility. This chapter highlights the key differences in the way these alternative solutions address the same features provided by YARN.

Chapter 10, *YARN Future and Support*, talks about YARN’s journey and its present and future in the world of distributed computing.
YARN stands for Yet Another Resource Negotiator. YARN is a generic resource platform to manage resources in a typical cluster. YARN was introduced with Hadoop 2.0, which is an open source distributed processing framework from the Apache Software Foundation.

In 2012, YARN became one of the subprojects of the larger Apache Hadoop project. YARN is also coined by the name of MapReduce 2.0. This is since Apache Hadoop MapReduce has been re-architected from the ground up to Apache Hadoop YARN.

Think of YARN as a generic computing fabric to support MapReduce and other application paradigms within the same Hadoop cluster; earlier, this was limited to batch processing using MapReduce. This really changed the game to recast Apache Hadoop as a much more powerful data processing system. With the advent of YARN, Hadoop now looks very different compared to the way it was only a year ago.

YARN enables multiple applications to run simultaneously on the same shared cluster and allows applications to negotiate resources based on need. Therefore, resource allocation/management is central to YARN.

YARN has been thoroughly tested at Yahoo! since September 2012. It has been in production across 30,000 nodes and 325 PB of data since January 2013.

Recently, Apache Hadoop YARN won the Best Paper Award at ACM Symposium on Cloud Computing (SoCC) in 2013!
The redesign idea
Initially, Hadoop was written solely as a MapReduce engine. Since it runs on a cluster, its cluster management components were also tightly coupled with the MapReduce programming paradigm.

The concepts of MapReduce and its programming paradigm were so deeply ingrained in Hadoop that one could not use it for anything else except MapReduce. MapReduce therefore became the base for Hadoop, and as a result, the only thing that could be run on Hadoop was a MapReduce job, batch processing. In Hadoop 1.x, there was a single JobTracker service that was overloaded with many things such as cluster resource management, scheduling jobs, managing computational resources, restarting failed tasks, monitoring TaskTrackers, and so on.

There was definitely a need to separate the MapReduce (specific programming model) part and the resource management infrastructure in Hadoop. YARN was the first attempt to perform this separation.

Limitations of the classical MapReduce or Hadoop 1.x
The main limitations of Hadoop 1.x can be categorized into the following areas:

- Limited scalability:
  - Large Hadoop clusters reported some serious limitations on scalability. This is caused mainly by a single JobTracker service, which ultimately results in a serious deterioration of the overall cluster performance because of attempts to re-replicate data and overload live nodes, thus causing a network flood.
  - According to Yahoo!, the practical limits of such a design are reached with a cluster of ~5,000 nodes and 40,000 tasks running concurrently. Therefore, it is recommended that you create smaller and less powerful clusters for such a design.

- Low cluster resource utilization:
  - The resources in Hadoop 1.x on each slave node (data node), are divided in terms of a fixed number of map and reduce slots.
  - Consider the scenario where a MapReduce job has already taken up all the available map slots and now wants more new map tasks to run. In this case, it cannot run new map tasks, even though all the reduce slots are still empty. This notion of a fixed number of slots has a serious drawback and results in poor cluster utilization.
• Lack of support for alternative frameworks/paradigms:
  - The main focus of Hadoop right from the beginning was to perform
    computation on large datasets using parallel processing.
  - Therefore, the only programming model it supported was MapReduce.
  - With the current industry needs in terms of new use cases in the world
    of big data, many new and alternative programming models (such
    Apache Giraph, Apache Spark, Storm, Tez, and so on) are coming
    into the picture each day. There is definitely an increasing demand to
    support multiple programming paradigms besides MapReduce, to
    support the varied use cases that the big data world is facing.

**YARN as the modern operating system of Hadoop**

The MapReduce programming model is, no doubt, great for many applications,
but not for everything in the world of computation. There are use cases that are
best suited for MapReduce, but not all.

MapReduce is essentially batch-oriented, but support for real-time and near
real-time processing are the emerging requirements in the field of big data.

YARN took cluster resource management capabilities from the MapReduce
system so that new engines could use these generic cluster resource management
capabilities. This lightened up the MapReduce system to focus on the data
processing part, which it is good at and will ideally continue to be so.

YARN therefore turns into a data operating system for Hadoop 2.0, as it
enables multiple applications to coexist in the same shared cluster. Refer to
the following figure:
What are the design goals for YARN
This section talks about the core design goals of YARN:

- **Scalability:**
  - Scalability is a key requirement for big data. Hadoop was primarily meant to work on a cluster of thousands of nodes with commodity hardware. Also, the cost of hardware is reducing year-on-year.
  - YARN is therefore designed to perform efficiently on this network of a myriad of nodes.

- **High cluster utilization:**
  - In Hadoop 1.x, the cluster resources were divided in terms of fixed size slots for both map and reduce tasks. This means that there could be a scenario where map slots might be full while reduce slots are empty, or vice versa. This was definitely not an optimal utilization of resources, and it needed further optimization.
  - YARN fine-grained resources in terms of RAM, CPU, and disk (containers), leading to an optimal utilization of the available resources.

- **Locality awareness:**
  - This is a key requirement for YARN when dealing with big data; moving computation is cheaper than moving data.
  - This helps to minimize network congestion and increase the overall throughput of the system.

- **Multitenancy:**
  - With the core development of Hadoop at Yahoo, primarily to support large-scale computation, HDFS also acquired a permission model, quotas, and other features to improve its multitenant operation.
  - YARN was therefore designed to support multitenancy in its core architecture. Since cluster resource allocation/management is at the heart of YARN, sharing processing and storage capacity across clusters was central to the design.
  - YARN has the notion of pluggable schedulers and the Capacity Scheduler with YARN has been enhanced to provide a flexible resource model, elastic computing, application limits, and other necessary features that enable multiple tenants to securely share the cluster in an optimized way.
• Support for programming model:
  ° The MapReduce programming model is no doubt great for many applications, but not for everything in the world of computation.
  ° As the world of big data is still in its inception phase, organizations are heavily investing in R&D to develop new and evolving frameworks to solve a variety of problems that big data brings.

• A flexible resource model:
  ° Besides mismatch with the emerging frameworks’ requirements, the fixed number of slots for resources had serious problems. It was straightforward for YARN to come up with a flexible and generic resource management model.

• A secure and auditable operation:
  ° As Hadoop continued to grow to manage more tenants with a myriad of use cases across different industries, the requirements for isolation became more demanding.
  ° Also, the authorization model lacked strong and scalable authentication. This is because Hadoop was designed with parallel processing in mind, with no comprehensive security. Security was an afterthought.
  ° YARN understands this and adds security-related requirements into its design.

• Reliability/availability:
  ° Although fault tolerance is in the core design, in reality maintaining a large Hadoop cluster is a tedious task.
  ° All issues related to high availability, failures, failures on restart, and reliability were therefore a core requirement for YARN.

• Backward compatibility:
  ° Hadoop 1.x has been in the picture for a while, with many successful production deployments across many industries. This massive installation base of MapReduce applications and the ecosystem of related projects, such as Hive, Pig, and so on, would not tolerate a radical redesign. Therefore, the new architecture reused as much code from the existing framework as possible, and no major surgery was conducted on it. This made MRv2 able to ensure satisfactory compatibility with MRv1 applications.
Summary
In this chapter, you learned what YARN is and how it has turned out to be the modern operating system for Hadoop, making it a multiapplication platform.

In Chapter 2, YARN Architecture, we will be talking about the architecture details of YARN.
Where to buy this book

You can buy YARN Essentials from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.