ADMIN

Network & Security

Digital Special

Practical Hadoop

Hadoop 2

Big Data isn't just for Big Business

Is Hadoop the New HPC?

rackspace®

Fanatical Support® Made Us the #1 managed cloud company

Managed cloud means we design, build, and run the best cloud for your needs. We handle the headaches and hidden costs, so you can run your business.

www.rackspace.com

YARN

Run MPI applications on a Hadoop cluster

Big Data in the Cloud

Meet Rackspace's chief data technologist Kenny Gorman

WWW.ADMIN-MAGAZINE.COM

US$ 7.95

rackspace®

the #1 ***managed cloud*** *company*

Hadoop is Hard.

Rackspace Cloud Big Data Makes it Easy.

- Deploy Hadoop and Apache Spark in three clicks
- Get bare metal performance with cloud elasticity
- Iterate with data in real time

Sign up for a free trial today.

www.**baremetalbigdata**.com

ADMIN Network & Security

Practical Hadoop

Dear Readers:

The data revolution is all about opportunity. A new generation of systems collect and store data from online order forms, cash register transactions, help desk calls, and more. What happens then? Big Data experts look inside that data to search for trends and discover important information.
If your organization competes in the global market today, and you want to keep up with the competition, you'd better get started with exploring and deploying the tools for Big Data.
One of the best and most promising tools of the Big Data age is Apache Hadoop, an open source implementation of the MapReduce framework designed for managing massive amounts of data on high-performance computer cluster.
This special edition helps you get started with Hadoop and offers some insights on how to integrate the promise of Big Data into your own computing environment.

Joe Casad
Editor in Chief
ADMIN Magazine

Table of Contents

Contact Info

Editor in Chief
Joe Casad, jcasad@admin-magazine.com

Managing Editor
Rita L Sooby, rsooby@admin-magazine.com

Localization & Proofreading
Amber Ankerholz, Ian Travis

Layout and Graphic Design
Dena Friesen, Lori White

Advertising
www.admin-magazine.com/Advertise
Ann Jesse, ajesse@admin-magazine.com
Phone: +1-785-841-8834

Corporate Management (Vorstand)
Hermann Plank, hplank@linuxnewmedia.de
Brian Osborn, bosborn@linuxnewmedia.com

Publisher
Brian Osborn, bosborn@linuxnewmedia.com

Marketing Communications
Darrah Buren, dburen@linuxnewmedia.com

Customer Service / Subscription
For USA and Canada:
Email: cs@admin-magazine.com
Phone: 1-866-247-2802
(toll-free from the US and Canada)

www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it.

Copyright and Trademarks © 2014 Linux New Media Ltd.

Cover Illustration © , 123RF.com

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Germany

ADMIN ISSN 2045-0702

ADMIN is published by Linux New Media USA, LLC, 616 Kentucky St, Lawrence, KS 66044, USA.

Big data tools for midcaps and others

Arithmagician

Hadoop 2.x and its associated tools promise to deliver big data solutions, not just to the IT-heavy big players, but to anyone with unstructured data and the need for multidimensional data analysis.

By Anna Kobylinska and Filipe Martins

Every company, from a small online store to a multinational group, collects detailed data about daily events. The events range from purchase transactions, through the effect of marketing initiatives, to the company's social media activities. This huge volume of unstructured data – big data – promises to deliver valuable insights and abundant material for decision making. If you want to benefit from this information, you have to face the challenge of big data.

SQL, NoSQL, Hadoop

Conventional big data solutions still lug around the legacy of an ecosystem built around a database – whether SQL or NoSQL. Astronomical licensing costs take them virtually out of the reach of medium-sized enterprises, especially if high-availability features are desired.

The biggest bottleneck in these solutions is often the database, because it typically will only scale beyond the boundaries of individual servers with considerable administrative overhead. Conventional data analysis methods and relational databases can reach their limits. Even some cloud solutions do not scale without mandatory downtime. One possible way out of this dilemma is Hadoop [1].

Apache Hadoop is a framework for distributed processing of, in particular, unstructured data on computer clusters. Hadoop makes it possible to run computational processes at low cost, whether on-premises on commodity hardware, at a data center, or in the virtualized environment of a cloud service provider.

Hadoop Features

Access to an ordinary relational database relies on queries in one of the many dialects of SQL (Structured Query Language). If you need to access non-relational databases, other languages be-

Lead Image © Roman Gorielov, 123RF.com

sides SQL are possible (hence the term NoSQL). Hadoop does not fall into these two categories, because it simply does not use a database; it is this distinction that lends Hadoop its flexibility and robustness.

Hadoop consists of two basic components: the Hadoop Distributed Filesystem (HDFS) and a distributed, modularized data processing framework: Hadoop uses MapReduce 1.x, whereas Hadoop 2.x relies on either MapReduce or its successor YARN [2] (discussed later).

The role of HDFS is to provide the most efficient, fastest possible fail-safe storage of data. HDFS is not a "clusterized" filesystem but a distributed filesystem: It runs on multiple nodes in a network – but without an expensive SAN solution. HDFS is therefore very cost efficient.

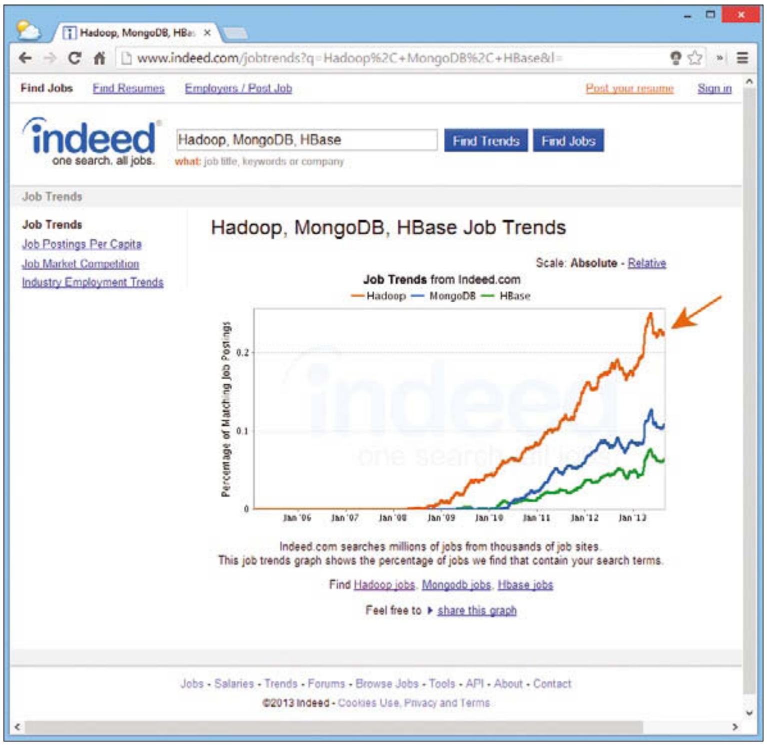
The data processing framework talks to the filesystem, which manages the resources and monitors the execution of the commands sent by a Hadoop-compatible application on the framework. These commands form jobs, and jobs are implemented as individual, tiny Java applications.

Thanks to this architecture, workloads can be spread across multiple nodes of a computer cluster, and the cluster itself can be reconfigured even while performing jobs. This approach results in several important advantages. For example, Hadoop scores points with its ability to scale as needed, without downtime. This elasticity is useful not only if the amount of data sharply increases or decreases, but also if deadlines make temporary provisioning of additional computational resources desirable.

As the load on the active nodes increases, additional nodes can be started automatically, for example, Amazon API AWS autoscaling. In this case, the cloud uses CloudWatch to monitor the load on the individual instances. Once the conditions set by the administrator are met, AWS automatically launches new server instances that are integrated into the Hadoop cluster, registered with the resource manager, and then finally able to take on jobs.

Hadoop also works in a very resource-friendly way; instead of copying large amounts of data back and forth across the network between different nodes, as is the case with many relational database management systems, it sends comparatively small instructions exactly to where the required data already exists. In a database running on multiple servers, the data typically is stored separately from the software logic and on different instances. In contrast, in a Hadoop cluster, both data and the data processing logic exist on each machine. This setup means that the framework can run individual jobs very efficiently: on the instance holding the required data in each case. The redundancy resulting from distributed data processing also improves the robustness of the cluster.

A variety of free and commercial tools exist to enhance Hadoop with additional capabilities. For example, with the Apache Hive open source application, you can convert SQL queries into MapReduce jobs and address Hadoop as a (distributed!) SQL database.


Hadoop, MongoDB, HBa ×
www.indeed.com/jobtrends?q=Hadoop%2C+MongoDB%2C+HBase&l=
Find Jobs
Find Resumes
Employers / Post Job
Post your resume
Sign in
indeed
one search. all jobs.
Hadoop, MongoDB, HBase
what: job title, keywords or company
Find Trends
Find Jobs
Job Trends
Job Trends
Job Postings Per Capita
Job Market Competition
Industry Employment Trends
Hadoop, MongoDB, HBase Job Trends
Scale: Absolute - Relative
Job Trends from Indeed.com
Hadoop
MongoDB
HBase
Percentage of Matching Job Postings
0.2
0.1
0
Jan '06
Jan '07
Jan '08
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Find Hadoop jobs, Mongodb jobs, Hbase jobs
Feel free to ▸ share this graph
Jobs - Salaries - Trends - Forums - Browse Jobs - Tools - API - About - Contact
©2013 Indeed - Cookies Use, Privacy and Terms

Indeed.com searches millions of jobs from thousands of job sites.
This job trends graph shows the percentage of jobs we find that contain your search terms.

Figure 1: Trends in the US labor market promise Hadoop-savvy IT professionals a bright future.

Highly Sought Skills

If you follow developments in the US IT job market as a benchmark for future trends, you will find that the need for Hadoop skills is skyrocketing (Figure 1). This is little wonder, because the framework has many practical applications. In Germany, too, job offers relating to big data with Hadoop are on the increase. For example, recruiter Hays AG is looking for Java developers, software architects, and system administrators with Apache Hadoop skills for different locations throughout Germany. JobLeads GmbH tried to recruit nearly a hundred Hadoop-savvy IT professionals in the same period on behalf of unspecified customers.

According to a study by IDC, the worldwide market for Hadoop has an annual growth rate of 60 percent. However, the same study also stated that this rate of growth is limited almost exclusively to the use of Hadoop as elastic and cheap distributed mass storage. Although Hadoop is a world leader in the field of data analysis, it is precisely this functionality that often seems to lie fallow in practical applications. Hadoop users with lower development capacities understandably tend to keep their growing datasets on the low-cost HDFS and use external, but less powerful solutions to handle data analysis. The reason for this is obvious: MapReduce in Hadoop 1.x is perceived by many as too complicated and not sufficiently flexible. Hadoop 2.x seeks to remedy the situation and help achieve wider dissemination of the powerful big data framework.

Applications and Examples

Practical applications of Hadoop are very diverse. They include: analyzing web clickstream data to optimize conversion rates, evaluating sensor data of machinery for the optimization of production processes, analyzing server logfiles for improved security, making forecasts on the basis of geolocation data, data mining of social media activities for a better understanding of the target audience, statistical processing of search indexes and RFID data – the list never ends.

Banks and insurance companies use applications based on Hadoop to rate clients on the basis of their financial history using pattern recognition algorithms. This approach helps financial institutions put a stop to credit card abuse, among other things, and better assess the creditworthiness of their customers in the scope of risk management.

In e-commerce and online and mobile advertising, Hadoop is used to compute product recommendations. The behavior of a visitor in your own webshop and on social media serves as the basis for exploring the visitor's preferences. Data centers, telcos, and web hosting providers use Hadoop-based solutions to identify bottlenecks or errors in networks at an early stage by evaluating network traffic statistics. Another example is algorithms for analyzing the meaning of text messages. Some e-commerce providers and telecommunications service providers use this technique to evaluate customer requests.

Solutions based on Hadoop also let you combine several different data sources and thus create multidimensional analyses.

One of the pioneers in the use of Hadoop for multidimensional data analysis is the gaming industry. After all, casinos are particularly vulnerable: Fraud occurs, and they can lose a lot of money in a few minutes. Analytics solutions for big data have proven their value in fraud detection and in exploring the target audience. Thanks to big data, casino operators can create granular customer profiles. The Flamingo Hotel of Caesars Entertainment in Las Vegas alone employs 200 big data analysts. Specialized analytics solutions for the gaming industry, such as Kognitio [3], are based on Hadoop. Game chips, customer loyalty cards, and even liquor bottles in bars like Aria Hotel and Casino are equipped with RFID tags. This technology makes it possible to track activities in real time. Casinos consistently analyze all these acquired values as finely granular data. “Casinos employ the most talented cryptographers, computer security experts and game theorists” says John Pironti, the chief information risk strategist with data protection specialists Archer Technologies [4].

Security technologies such as video surveillance or RFID tracking produce huge amounts of data. Relevant data are never discarded because they are just too valuable, and this is where Hadoop enters the game – as a distributed filesystem. While casinos are experimenting with these innovations, companies in other industries are trying to apply their experiences to their own business.

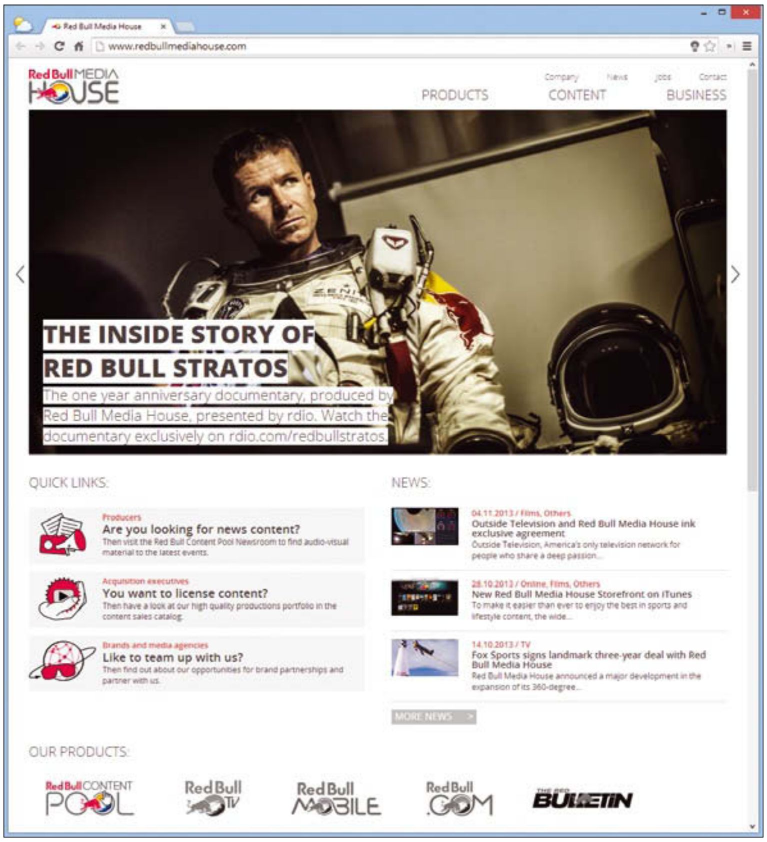
On the basis of Hadoop, tailor-made solutions can be programmed to manage data assets more efficiently. An example of this is Red Bull Media House GmbH's Media Asset Management (Figure 2). Germany's ADACOR Hosting GmbH from Essen tested multiple solutions for media asset management on behalf of the Austrian company. The task was to create a central repository of content, such as video clips, photos and audio files, in various formats and at various quality levels so that customers could quickly and easily access this data at any time from anywhere.

The requirements included elastic scalability without maintenance windows, minimal downtime for hardware glitches, data replication, fast data delivery, ease of management, and a better cost-benefit ratio than standard solutions such as EMC storage could offer. The shortlist included, among other options, NFS, GlusterFS, Lustre, Openfiler, CloudStore, and finally HDFS.

As an initial approach, NFS was investigated as a very popular, stable, and proven filesystem in the Unix world. NFS disappointed ADACOR with its poor performance and lack of features, such as data replication. The resulting application would have had to handle distributed storage management itself. The overhead would have been just too great.

The second candidate under the microscope was GlusterFS. ADACOR Hosting GmbH already had some good experience with this filesystem in a different context, especially in terms of performance. As the number of nodes in a GlusterFS cluster increases, so does the maximum data throughput. What disqualified GlusterFS in the eyes of the testers was its very high administrative overhead and impractical scalability with mandatory downtime.

Lustre impressed in terms of performance, scalability, and convenient administration, but this solution also lacked robust replication at the time of implementation. Openfiler dropped out of the shortlist, because the system took several days to complete a rebuild of only 3TB of data. CloudStore was rejected by ADACOR because of a lack of stability. Only HDFS impressed across the board and most closely met the customer's requirements.



THE INSIDE STORY OF RED BULL STRATOS

The one year anniversary documentary, produced by Red Bull Media House, presented by rdio. Watch the documentary exclusively on rdio.com/redbullstratos.

Producers
Are you looking for news content?
Then visit the Red Bull Content Pool Newsroom to find audio-visual material to the latest events.

Acquisition executives
You want to license content?
Then have a look at our high quality productions portfolio in the content sales catalog.

Brands and media agencies
Like to team up with us?
Then find out about our opportunities for brand partnerships and partner with us.

04.11.2013 / Films, Others
Outside Television and Red Bull Media House ink exclusive agreement
Outside Television, America's only television network for people who share a deep passion...

28.10.2013 / Online, Films, Others
New Red Bull Media House Storefront on iTunes
To make it easier than ever to enjoy the best in sports and lifestyle content, the wide...

14.10.2013 / TV
Fox Sports signs landmark three-year deal with Red Bull Media House
Red Bull Media House announced a major development in the expansion of its 360-degree...

Figure 2: The media asset management platform of Austria's Red Bull Media House GmbH is based on HDFS.

Big Data: A Question of Flexibility

A myth that stubbornly persists in the IT industry is that big data is only usable or affordable for large enterprises. Midcaps almost all self-evidently and firmly believe you cannot even think about big data unless you have petabytes of data. Nothing could be further from the truth. Even if you "only" have datasets of 10 or 50TB, Hadoop is still a useful solution. Big data is not about a certain amount of data but about the lack of data structure.

Actually, the question is not whether a company qualifies for big data. Many companies face a very different challenge: a data management problem. A proprietary data warehouse will quickly reach the capacity limits of a single machine, which is how isolated data silos initially emerged. If you want to gain actionable insights from data, you need a distributed cluster filesystem like HDFS, which grows with the requirements, and a framework like Hadoop.

For midcaps, no factual or financial reasons exist for not using big data. Admittedly, the most vocal users of Hadoop include some of the biggest names in IT, social media, and the entertainment industry, including Amazon Web Services, AOL, Apple, eBay, Facebook, Netflix, and HP. However, Hadoop 2.2.x is especially appealing for smaller companies with tight budgets: It is easy to program, free, platform independent, and open source.

The biggest challenge to using Hadoop is by no means finan-

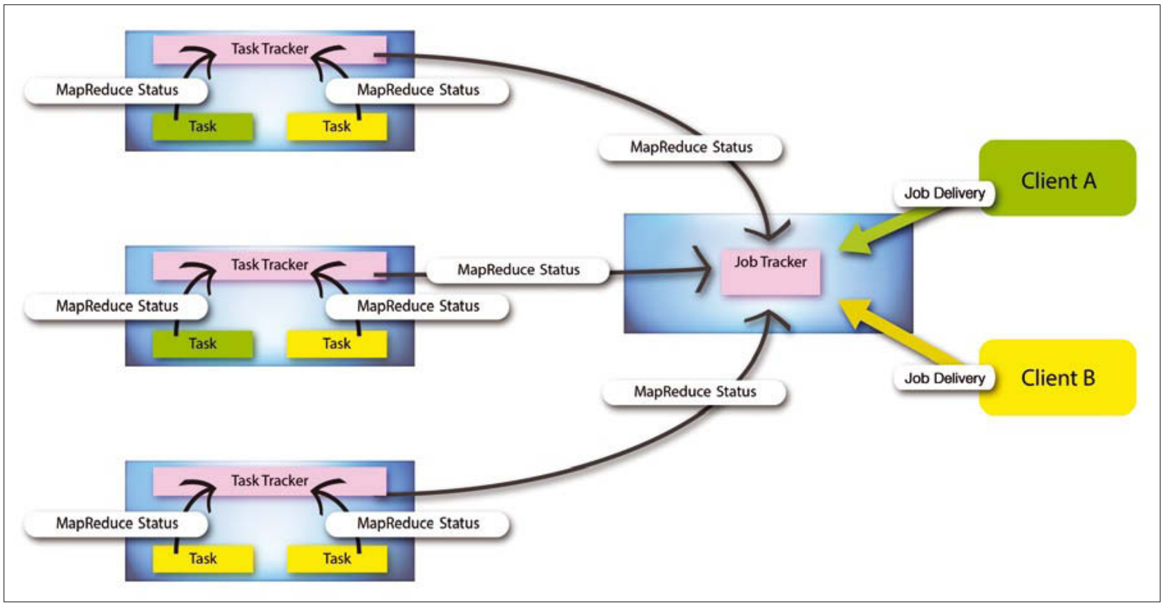


Figure 3: Hadoop 1.x relies on hard partitioning of the existing resources in the cluster, which in turn results in less than optimal use of the available capacity. Jobs that could not be distributed by `Map()` and `Reduce()` ran accordingly slower.

cial; rather, it stems from a lack of know-how. The first step is to leverage the low-budget data processing capabilities and rugged data protection with Hadoop. After the company has begun to reap the benefits of these cost reductions, it can then expand its own data analysis activities. Only at this stage does it make sense to employ data scientists to pursue more challenging questions using data analysis solutions based on Hadoop.

Next-Gen MapReduce

The changes in Hadoop 2.2.0 are profound and thoughtful. The innovations are based on modularization of the engine. This bold step is designed to enrich the Hadoop ecosystem, adding plugins and other enhancements. It also promises additional flexibility for the Hadoop administrator. For example, the admin can already replace some built-in algorithms with external modules to benefit from extended functionality. This is true of, for example, *shuffle* and *sort*. These modules can be used in parallel and even combined with the built-in algorithms.

The key new features in version 2.2.0 include the introduction of YARN (Yet Another Resource Negotiator) as an optional replacement for MapReduce. MapReduce in Hadoop 1.x (Figure 3) is not optimal for all workloads. It achieves its best performance where duties can be clearly distributed and parallelized. Many shortcomings of MapReduce are things of the past, however, following the introduction of YARN. YARN is a development of MapReduce version 2 (MRv2). YARN is based directly on HDFS and assumes the role of a distributed operating system for resource management for big data applications (Figure 4). Thanks to YARN, you can use Hadoop 2.2.x to interweave interactive workloads, real-time workloads, and automated workloads (see the box “Big Data Applications with Support for YARN”) .

Additionally, YARN is backward-compatible with MapReduce at the API level (hadoop-0.20.205); at the same time, it improves Hadoop’s compatibility with other projects by the Apache Software Foundation. If you insist on using the legacy version of MapReduce, you can now load it as a module. This should not be necessary, however, because MapReduce applications are binary-compatible between the two generations of Hadoop.

The most important change in YARN compared with the classic MapReduce is allocation of the two job tracker functions – resource management and time management/workload monitoring – to two separate daemons: the global ResourceManager (RM) and the job-specific ApplicationMaster (AM).

The ResourceManager comprises two components: the scheduler and the application manager. The scheduler is responsible for allocating resources to different active applications but is not responsible

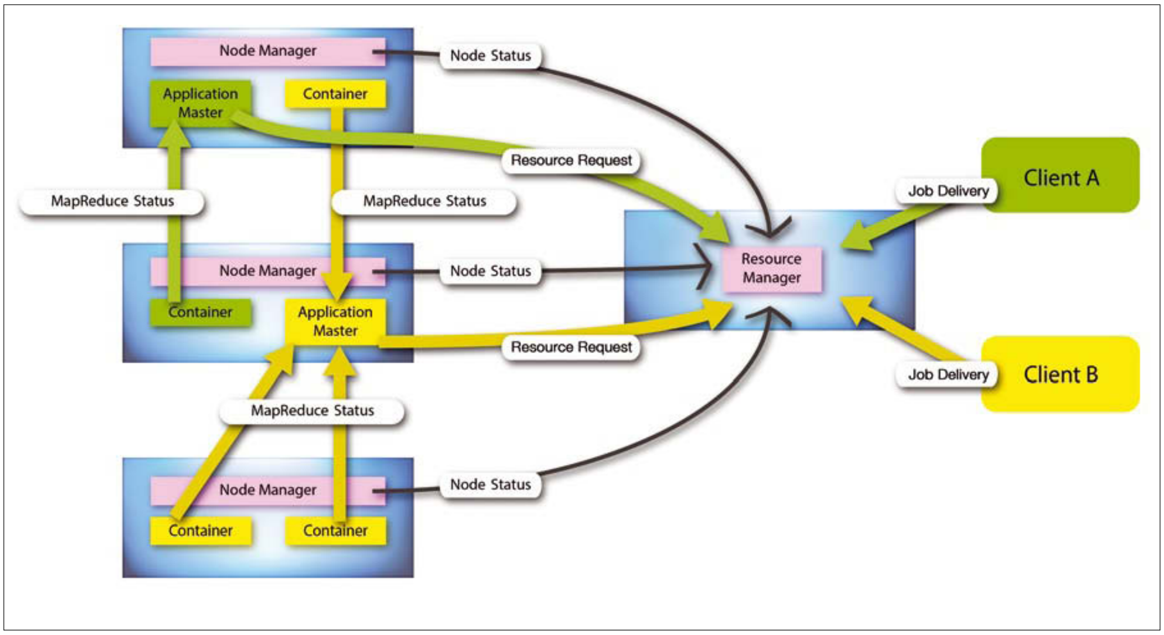


Figure 4: **The architecture of Hadoop 2.x: Resource management by YARN is based on logical units of resource containers; requesting resources is now separated from the application logic.**

for monitoring the workloads. The scheduler takes into account both the resource requirements of individual applications as well as the capacity limitations of the cluster. In the current version, the scheduler, unfortunately, can only manage one resource: memory. In future versions of YARN, it will be possible to allocate CPU cycles, storage, and network bandwidth to the cluster's individual applications.

Resources are allocated by partitioning resource containers. These are virtual compute entities in a cluster node, and a node can possess several such containers. The application manager (the second basic component in the ResourceManager besides the scheduler) accepts workload orders. To do this, the application manager initiates the process of setting up the first resource container for the ApplicationMaster and launches it (or reboots it after a crash).

The application-specific ApplicationMaster requests the neces-

Big Data Applications with Support for YARN

A variety of applications can benefit from efficient resource management by YARN (Figure 5).

The list of YARN-optimized big data applications includes: Apache Giraph (visualization); Apache Hama (BSP); Apache Hadoop MapReduce (batch processing); Apache Tez (batch and interactive jobs in RAM); Apache S4/Samza/Storm (real-time processing of data streams); Apache Spark (iterative and interactive applications); Elastic Search; Cloudera Llama (a YARN implementation of Impala, a hybrid ad hoc query engine with support for the SQL dialect HiveQL); Data Torrent (data analysis); HOYA (HBase on YARN); and RedPoint (data management).

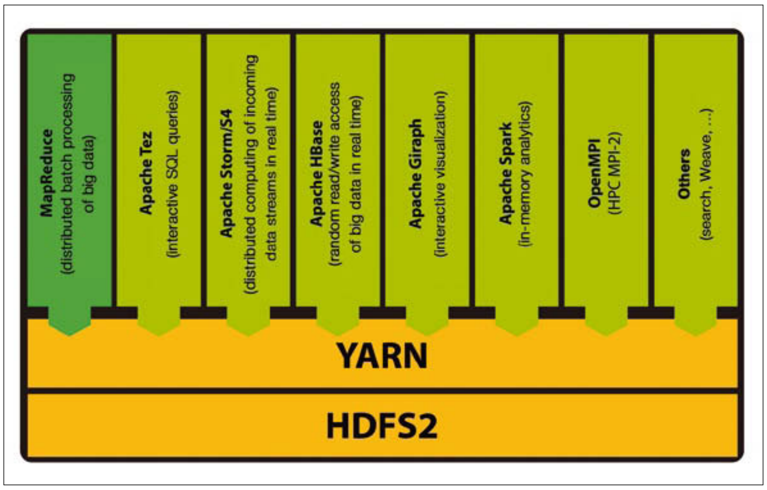

MapReduce
(distributed batch processing of big data)
Apache Tez
(interactive SQL queries)
Apache Storm/S4
(distributed computing of incoming data streams in real time)
Apache HBase
(random read/write access of big data in real time)
Apache Giraph
(interactive visualization)
Apache Spark
(in-memory analytics)
OpenMPI
(HPC MPI-2)
Others
(search, Weave, ...)
YARN
HDFS2

Figure 5: **Applications with support for YARN in Hadoop 2.x: MapReduce is now a module in userspace and is binary-compatible with legacy applications from Hadoop 1.x.**

sary resource containers from the scheduler (the scheduler is part of the ResourceManager) and begins to monitor them.

HDFS has two types of server or cluster nodes: NameNodes and DataNodes. NameNodes manage the metadata, and the actual data blocks are kept on the DataNodes. A separate NodeManager is responsible for each node in the cluster (i.e., each single machine). It monitors the use of container resources and reports the current activities of the respective nodes to the ResourceManager/Scheduler.

The new architecture allows for significant cost savings (Figure 5). Yahoo estimates the improvements achieved in node utilization to be 60 to 150 percent per day. Yahoo tested YARN with 365PB of data with 400,000 jobs on 40,000 cluster nodes and a total computation time of 10 million hours. A high-availability implementation of the YARN ResourceManager is planned for a future release.

HDFS2

HDFS has always been considered reliable. During use at Yahoo on 20,000 nodes in 10 clusters, HDFS errors were only responsible for the loss of 650 data blocks out of a total of 329 million in 2009. Since then, the Apache Foundation has worked intensively on improving the reliability of HDFS.

Despite its reliability, HDFS in Hadoop v1 had a clear single point of failure: the NameNode, which is the control center for managing access to data via metadata. Although NameNodes were redundant, they could only be operated in an active/passive node architecture. If an active NameNode failed, the administrator had to adjust the configuration manually. Thus, the failure of a single NameNode could potentially take down the whole HDFS; all active write processes and jobs in the queue were canceled with an error message.

The implementation of mission-critical workloads that need to run interactively in real time was thus very problematic.

The Hadoop developers identified the problem and came up with a solution: the high-availability NameNode (HA NameNode). The HA NameNode waits on the bench and can step in when needed for the active NameNode. In Hadoop 2.0, this failover can still only be triggered through manual intervention by the Hadoop administrator.

HDFS Federation

To scale the name service horizontally, Hadoop uses 2.2.0 Federation with several completely independent NameNodes and namespaces. The NameNodes remain independent by not coordinating their work. All NameNodes independently access a common pool of DataNodes. Each of these DataNodes registers with all the NameNodes in the cluster, periodically sending a heartbeat signal and block reports and accepting commands. An implementation of symlinks was planned for this release but canceled at the last minute.

HDFS Snapshots

Snapshots of the HDFS filesystem also make their debut in Hadoop 2.x. These are non-writable copies of the filesystem that capture its state at a defined time (point-in-time copies).

No DataNodes are copied for an HDFS snapshot. The snapshot only captures the list of data blocks and the size of the files. The process has no negative effect on other I/O operations. Changes are recorded in reverse chronological order, so the current data can be accessed directly. HDFS2 computes the data status for the snapshot by subtracting changes from the current state of the filesystem. The operation usually does not require any additional memory (unless writes occur in parallel). To allow snapshots, the administrator uses the following command with superuser privileges:

```
hdfs dfsadmin -allowSnapshot ↩
  <path-to-snapshot-capable-directory>
```

The directory tree in question can then be grabbed as a snapshot using the owner's user rights, as follows:

```
hdfs dfs -createSnapshot ↩
  <path-to-snapshot-capable-directory>↩
  [<snapshotName>]
```

Or, you can use the Java API.

To mark the path to the snapshots, the HDFS2 developers have created a `.snapshot` object. If this string appears in the HDFS filesystem of your Hadoop installation, you must definitely rename the objects in question before upgrading; otherwise, the upgrade fails.

Distributions

An entire ecosystem of specialized solutions has emerged around Hadoop. Apache's Hadoop distribution primarily addresses providers of big data tools that build their own (commercial) solutions. This category includes, among others, Cloudera, Hortonworks, IBM, SAP, and EMC.

For mission-critical use of Hadoop, a Hadoop distribution with 24/7 support by a service provider such as Cloudera [5] or Hortonworks [6] can be beneficial. However, these providers make you pay quite handsomely for this privilege. If you do not need a service-level agreement, you can instead choose the free versions of these distributions. Additionally, Hadoop distributions, such as Stratosphere by the Technical University of Berlin [7], have been specially created for small to mid-sized companies.

Stratosphere

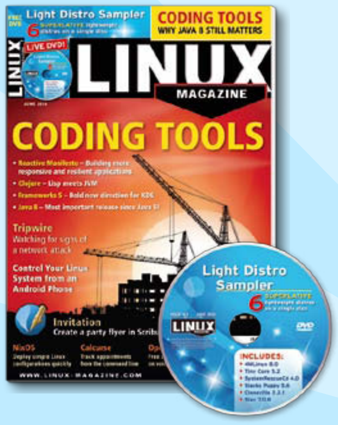
Stratosphere combines easy installation with ease of use and high performance. The platform also scales to large clusters, uses multicore processors, and supports in-memory data processing. It also features advanced analytics functionality and lets users program jobs in Java and Scala.
Stratosphere is developed under the leadership of Professor Volker Markl from TU Berlin's Department of Database Systems and Information Management (DIMA). Stratosphere runs both on-premises and in the cloud (e.g., on Amazon EC2).

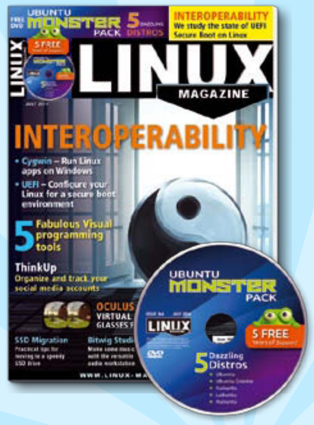
Hadoop Services

Growth-oriented midcaps can choose from a wide range of Hadoop services. Amazon offers Elastic MapReduce (EMR) [8], an implementation of Hadoop with support for Hadoop 2.2 and HBase 0.94.7, as well as the MapR M7, M5, and M3 Hadoop distributions by MapR Technologies [9]. The service targets companies, researchers, data analysts, and developers in the fields of web indexing, data mining, logfile analysis, machine learning, financial analysis, scientific simulation, and bioinformatics research.
For customers who want to implement HBase, Elastic MapReduce with M7 provides seamless splits without compression, immediate error recovery, timing recovery, full HA, mirroring, and consistent low latency. This version does involve some additional costs, however. Google (with Compute Engine) and Microsoft (with Azure) have their own implementations of Hadoop.
Using Hadoop as a service in the cloud means less capital outlay on hardware and avoids delays in the deployment of infrastructure and other expenses. Amazon EMR is a good example because of its clear pricing structure. In EMR, you can only set up a Hadoop cluster temporarily so that it automatically dissolves after analyzing your data, thus avoiding additional charges. Prices start at US$ 0.015/hour per instance for the EMR service, plus the EC2 costs for each instance of the selected type (from US$ 0.06 per instance), which are also billed on an hourly basis.
Thus, you would pay US$ 1.50 for one hour with 100 instances for Hadoop (100 x US$ 0.015) and up to US$ 6.00 for up to 100 instances that run on demand (100 x US$ 0.06). The bottom line is that you are billed for US$ 7.50 per hour for 100 small instances.
To keep costs down even further,

Risk-Free Trial!

GET IT NOW!
SAVE TIME ON DELIVERY WITH OUR PDF EDITIONS
(DVD NOT INCLUDED)





ORDER YOUR TRIAL NOW!
UK £ 3, Europe € 3, USA / Canada US$ 3, Rest of World (by Airmail) US$ 9

shop.linuxnewmedia.com/trial
Terms and conditions: http://goo.gl/SSSQer

you could reserve these instances for up to three years.

Numbering

Hadoop developers have not done themselves any favors with their choice of version numbers. If you identify remarkable development steps with a version number change in the second decimal place (e.g., from version 0.20 to version 0.23), do not be surprised if users take little note of your progress.

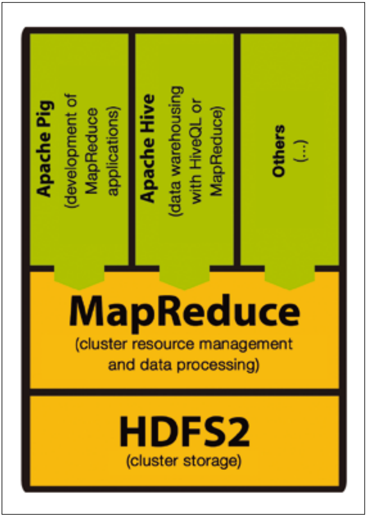

Apache Pig
(development of MapReduce applications)
Apache Hive
(data warehousing with HiveQL or MapReduce)
Others
(...)
MapReduce
(cluster resource management and data processing)
HDFS2
(cluster storage)

Figure 6: **Just for comparison's sake: In Hadoop 1.x, all applications were dependent on the use of MapReduce.**

Version 0.20 designates the first generation Hadoop (v1.0, Figure 6). Whenever you hear people refer to the 0.23 branch (Figure 7), they are talking about Hadoop 2.2.x. The version number 2.2.0 refers to the first release of the second generation with general availability.
An enterprise-class product that has proven itself in the tough big data daily grind for years should have earned a higher version number. The seemingly minimal generation leap to version 2.2.0 does little justice to the significantly advanced maturity of Hadoop; however, the hesitant numbering does not detract from the quality of Hadoop.
The growing importance of Hadoop is evidenced by the many prominent providers that are adapting their commercial solutions for Hadoop. SAP sells the Intel distribution and the Hortonworks Data Platform for Hadoop. SAP Hana, a data analysis platform for big data, seamlessly integrates with Hadoop. DataStax provides a distribution of Hadoop and Solr with its own NoSQL solution, DataStax Enterprise. Users of DataStax Enterprise use Hadoop for data processing, Apache Cassandra as a database for transactional data, and the Solr search engine for distributed searching. Incidentally, Cassandra lets you run Hadoop MapReduce jobs on a Cassandra cluster.

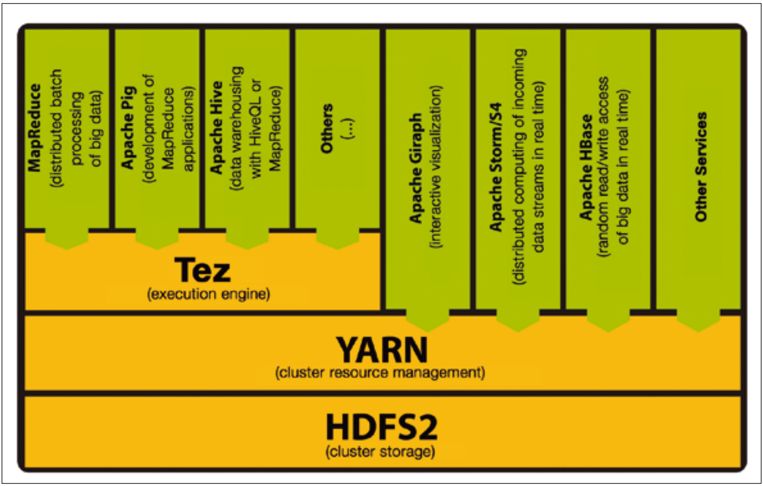

MapReduce
(distributed batch processing of big data)
Apache Pig
(development of MapReduce applications)
Apache Hive
(data warehousing with HiveQL or MapReduce)
Others
(...)
Apache Giraph
(interactive visualization)
Apache Storm/S4
(distributed computing of incoming data streams in real time)
Apache HBase
(random read/write access of big data in real time)
Other Services
Tez
(execution engine)
YARN
(cluster resource management)
HDFS2
(cluster storage)

Figure 7: **Hadoop 2.x stack with Apache Tez: performance improvements with data processing in cluster memory.**

Conclusions

After four years of development, Hadoop 2.2.0 surprises its users with some groundbreaking innovations. Thanks to its modularity and HA HDFS, the Apache Foundation has succeeded in keeping ahead of the field and even extended its lead. Because of its significantly improved management of workloads, Hadoop has become much more attractive for midcaps. Large companies have always been able to tailor additional needed functions, whereas painstaking development has always overtasked the midcaps. This situation has now finally changed with Hadoop 2.2.0. ■

Info

[1] Apache Hadoop: [http://hadoop.apache.org/]
[2] YARN: [http://hadoop.apache.org/docs/r2.3.0/hadoop-yarn/hadoop-yarn-site/YARN.html]
[3] Kognitio: [kognitio.com]
[4] Archer Technologies: [http://www.archerits.com/]
[5] Cloudera Hadoop distribution: [http://www.cloudera.com/content/cloudera/en/products-and-services/cdh.html]
[6] Hortonworks Hadoop distribution: [http://hortonworks.com/products/hadoop-support]
[7] Stratosphere Hadoop distribution by the Technical University of Berlin: [https://github.com/stratosphere/stratosphere]
[8] Elastic MapReduce: [http://aws.amazon.com/elasticmapreduce/]
[9] Comparison of Hadoop distributions by MapR Technologies: [http://www.mapr.com/products/mapr-editions]

Is Hadoop the new HPC?

Where Worlds Collide

Hadoop has been growing clusters in data centers at a rapid pace. Is Hadoop the new corporate high-performance computing? By Douglas Eadline

Apache Hadoop [1] has been generating a lot of headlines lately. For those who are not aware, Hadoop is an open source project that provides a distributed filesystem and MapReduce framework for massive amounts of data. The primary hardware used for Hadoop comprises clusters of commodity servers. File sizes can easily be in the petabyte range and use hundreds or thousands of compute servers.

Hadoop also has many components that live on top of the core Hadoop filesystem (HDFS) and MapReduce mechanism. Interestingly, high-performance computing (HPC) and Hadoop clusters share some features, but how much crossover you will see between the two disciplines depends on the application. Hadoop's strengths lie in the sheer size of data it can process and its high redundancy and

©Dzmitrky Sukhavarau, 123RF

toleration of node failures without halting user jobs.
Many organizations use Hadoop on a daily basis, including Yahoo!, Facebook, American Airlines, eBay, and others. Hadoop is designed to allow users to manipulate large *unstructured* or *unrelated* data sets. It is not intended to be a replacement for a relational database management system (RDMS). For example, Hadoop can be used to scan weblogs, online transaction data, or web content, all of which are growing each year.

MapReduce

To many HPC users, MapReduce is a methodology used by Google to process large amounts of web data. Indeed, the now famous Google MapReduce paper [2] was the inspiration for Hadoop.
The MapReduce idea is quite simple and, when used in parallel, can provide extremely powerful search and compute capabilities. Two major steps constitute the MapReduce process. If you have not figured it out, they are the "Map" step followed by a "Reduce" step. Some people are surprised to learn that mapping is done all the time in the *nix world. For example, consider:

```
grep "the" file.txt
```

In this simple example, I am "mapping" all the occurrences, by line, of the word "the" in a text file. Although the task seems somewhat trivial, suppose the file was 1TB. How could I speed up the mapping step? The answer is also simple: Break the file into chunks and put a different chunk on a separate computer. The results can be combined when the job is finished because the map step has no dependencies. The popular mpiBLAST tool takes the same approach by breaking the human genome file into chunks and performing "BLAST" mapping on separate cluster nodes.
Suppose you want to calculate the total number of lines containing "the"; a simple approach is to pipe the results into `wc` (word count):

```
grep "the" file.txt | wc -l
```

You have just introduced a "Reduce" step. For the large-file parallel mode, each computer would perform this step (`grep` and `wc`) and send the count to the master node. That, in a nutshell, is how MapReduce works, with, of course, a few more details, like key-value pairs and "the shuffle." But, for the purposes of this discussion, MapReduce can be that simple.
With Hadoop, large files are placed in HDFS, which automatically breaks the file into chunks and spreads them across the cluster (usually in a redundant fashion). In this way, parallelizing the Map process is trivial; all that needs to happen is to place a separate Map process on each node with the file chunk. The results are then sent to Reduce processes, which also run on the cluster. As you can imagine, large files produce large amounts of intermediate data; thus, multiple reducers help keep things moving. Several aspects to the MapReduce process are worth noting:

- MapReduce can be transparently scalable. The user does not need to manage data placement or the number of nodes used for their job. The underlying hardware has no dependencies.
- Data flow is highly defined and in one direction from the Map to the Reduce, with no communication between independent mapper or reducer processes.
- Because processing is independent, failover is trivial. A failed process can be restarted, provided that the underlying filesystem is redundant – like HDFS.

MapReduce, while powerful, does not fit all problem types. To understand the difference between Hadoop and a typical HPC cluster, I'll compare several aspects of both systems.

Hardware

Many modern HPC clusters and Hadoop clusters use commodity hardware, comprising primarily x86-based servers. Hadoop clusters usually include a large amount of local disk space (used for HDFS nodes), whereas many HPC clusters rely on NFS or a parallel filesystem for cluster-wide storage.
HPC uses diskless and diskful nodes, but in terms of data storage, a separate group of hardware is often used for global file storage. HDFS daemons run on all nodes and store data chunks locally. It does not support the POSIX standard. Hadoop is designed to move the computation to the data; thus, HDFS must be distributed throughout the cluster.
In terms of networking, Hadoop clusters almost exclusively use gigabit Ethernet (GigE). As the price continues to fall, newer systems are starting to adopt 10GigE. Although, there are many GigE and 10GigE HPC clusters, InfiniBand is often the preferred network.
Many new HPC clusters are using some form of acceleration

hardware on the nodes. These additions are primarily from NVidia (Kepler) and Intel (Phi). They require additional programming (in some cases) and can provide substantial speed increases for certain applications.

Resource Scheduling

One of the biggest differences between Hadoop and HPC systems is resource management. HPC requires fine-grained control of which resources (cores, accelerators, memory, time, etc.) are given to users. These resources are scheduled with tools like Grid Engine, Moab, LoadLeveler, and the like. Hadoop has an integrated scheduler consisting of a master JobTracker, which communicates with TaskTrackers on the nodes. All MapReduce work is supervised by the JobTracker. No other job types are supported in Hadoop (version 1).

One interesting difference between an HPC resource scheduler and the Hadoop TaskTracker is fault tolerance. HPC schedulers can detect down nodes and reschedule jobs (as an option), but if the job has not been checkpointing, it must start from the beginning. Hadoop, because of the nature of the MapReduce algorithm, can manage failure through the JobTracker.

Because the Task Tracker is aware of job placement and data location, a failed node (or even a rack of nodes) can be managed at run time. Thus, when an HDFS node fails, the JobTracker can reassign a task to a node where a redundant copy of the data exists. Similarly, if a Map or Reduce process fails, the job can be restarted on a new node.

The next-generation scheduler for Hadoop is called YARN (Yet Another Resource Negotiator) and offers better scalability and more fine-grained control over job scheduling. Users can request "containers" for MapReduce and other jobs (possibly MPI), which are managed by individual per-job Application Masters. With YARN, the Hadoop scheduler starts to look like other resource managers; however, it will be backward compatible with many higher level Hadoop tools.

Programming

One of the big differences between Hadoop and HPC involves programming models. Most HPC applications are written in Fortran, C, or C++, with the aid of MPI libraries, as well as CUDA-based applications and those optimized for Intel Phi. The responsibility of the user is actually quite large.

Application authors must manage communications, I/O, synchronization, debugging, and possibly checkpointing/restart operations. These tasks can be difficult to get right and can take significant time to implement correctly and efficiently.

Hadoop, by offering the MapReduce paradigm, only requires the user to create a Map step and a Reduce step (and possibly some others, i.e., a combiner). These tasks are devoid of all the minutiae of HPC programming. Users only need concern themselves with these two tasks, which can be debugged and tested easily using small files on a single system.

Hadoop also presents a single-namespace parallel filesystem (HDFS) to the user. Hadoop was written in Java and has a low-level interface to write and run MapReduce applications, but it also supports an interface (Streams) that allows mappers and reducers to be written in any language. Above these language interfaces sit many high-level tools, such as Apache Pig, a scripting language for Hadoop MapReduce, and Apache Hive, an SQL-like interface to Hadoop MapReduce. Many users operate using these and other higher level tools and might never actually write mappers and reducers. This situation is analogous to application users in HPC that never write MPI code.

Parallel Computing Model

MapReduce can be classified as a single-instruction, multiple-data (SIMD) problem. Indeed, the map step is highly scalable because the same instructions are carried out over all data. Parallelism arises by breaking the data into independent parts with no forward or backward dependencies (side effects) within a Map step; that is, the Map step may not change any data (even its own). The Reduce step is similar, in that it applies the same reduction process to a different set of data (the results of the Map step).

In general, the MapReduce model provides a functional, rather than procedural, programming model. Similar to a functional language, MapReduce cannot change the input data as part of the mapper or reducer process, which is usually a large file. Such restrictions can at first be seen as inefficient; however, the lack of side effects allows for easy scalability and redundancy.

An HPC cluster, on the other hand, can run SIMD and MIMD (multiple-instruction, multiple-data) jobs. The programmer determines how to execute the parallel algorithm. As noted above, this added flexibility comes with addition responsibilities. Users, however, are not restricted when creating their own MapReduce application within the framework of a typical HPC cluster.

Big Data Needs Big Solutions

Without a doubt, Hadoop is useful when analyzing very large data files. HPC has no shortage of "big data" files, and Hadoop has seen crossover into some technical computing areas: BioPig [3] extends Apache Pig with sequence analysis capability, and MR-MSPolygraph [4] is a MapReduce implementation of a hybrid spectral library/database search method for large-scale peptide identification. Results show that, relative to the serial version, MR-MSPolygraph reduces the time to solution from weeks to hours when processing tens of thousands of experimental spectra. Other applications include protein sequencing and linear algebra.

Provided your problem fits into the MapReduce framework, Hadoop is a powerful way to operate on staggeringly large data sets. Because both the Map and Reduce steps are user defined, highly complex operations can be encapsulated in these steps. Indeed, you encounter no hard requirements for a reducer step if all your work can be done in the mapping step.

The growth of Hadoop and the hardware on which it runs has been increasing. Certainly it can be seen as a subset of HPC, offering a single yet powerful algorithm that has been optimized for a large number of commodity servers, with some crossover even into technical computing that could see further growth. Many companies are finding Hadoop to be the new corporate HPC for big data. ■

Info

[1] Apache Hadoop: [http://hadoop.apache.org/]

[2] Dean, J., and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters": [http://research.google.com/archive/mapreduce.html]

[3] BioPig: [http://www.osti.gov/bridge/product.biblio.jsp?osti_id=1050659]

[4] MR-MSPolygraph: [http://compbio.eecs.wsu.edu/MR-MSPolygraph/]

REAL SOLUTIONS FOR REAL NETWORKS

Each issue delivers technical solutions to the real-world problems you face every day.

ADMIN magazine covers Windows, Linux, Solaris, and popular varieties of the Unix platform.

Learn the latest techniques for better network security, system management, troubleshooting, performance tuning, virtualization, cloud computing, and much more!

FREE CD or DVD in Every Issue!

Now 6 issues per year!

ORDER ONLINE AT: shop.linuxnewmedia.com

Run MPI applications on a Hadoop cluster

The YARN Invitation

Hadoop version 2 expands Hadoop beyond MapReduce and opens the door to MPI applications operating on large parallel data stores. By Douglas Eadline

The Big Data world has heard continued news about the introduction of Apache Hadoop YARN [1]. YARN is an acronym for "Yet Another Resource Negotiator" and represents a major change in the design of Apache Hadoop. Although most HPC users might believe they don't need another scheduler or support for big data MapReduce problems, the continued growth of YARN may change that opinion. If anything, YARN could represent a bridge between traditional HPC and the Big Data world.

Hadoop offers an ecosystem of tools and applications based on the data-parallel MapReduce paradigm: MapReduce jobs can operate on very large files using the Hadoop parallel filesystem (HDFS). As opposed to many HPC clusters, in which storage and compute are done on separate servers, Hadoop combines the two processes and attempts to move computation to servers that contain the appropriate data. Although MapReduce has been shown to be very powerful, the ability to operate on data stored within HDFS using other non-MapReduce algorithms has long been a goal of the Hadoop developers. Indeed, YARN now offers new processing frameworks, including MPI, as part of the Hadoop infrastructure.

Please note that existing "well oiled" HPC clusters and software are under no threat from YARN. Later, I'll cover some fundamental design differences that define different use cases for YARN and traditional HPC schedulers (SLURM, Grid Engine, Moab, PBS, etc.) On the other hand, YARN does offer the exciting possibility of bringing HPC methods, ideas, and performance to the vast stores of institutional data sitting in HDFS (and other parallel filesystems).

Hadoop 101

Although Apache Hadoop versioning can be a bit confusing, the

© Khen-Ho-Toh, 123RF.com

most recent version (as of October 18, 2014) of Hadoop YARN is v2.5.1. The last Hadoop version 1 (without YARN), available as release 1.2.1, was released August 1, 2013. Some other Hadoop releases could cause confusion when you're ready to download, but in general, the 0.20.X branch (last released in October 2011) is version 1, and the 0.23.X branch (last release in June 2014) is version 2 [2].

To understand what YARN is and why it is needed, some background on Hadoop could be helpful. A discussion on Hadoop, MapReduce, and YARN from an HPC perspective were discussed in the previous article.

Version 1 of Hadoop has two core components. The first is HDFS, which is part of all Hadoop distributions. Note, however, that HDFS is not explicitly tied to Hadoop and can, in theory, be used by other applications in a serial or parallel fashion. The second component is the monolithic MapReduce process, which contains both the resource management and the data processing elements needed for parallel execution. The Hadoop Version 1 MapReduce process is shown in Figure 1.

A master process called the *JobTracker* is the central scheduler for all MapReduce jobs in the cluster and is similar to the master daemon of an HPC scheduler (e.g., `sge_qmaster` in Grid Engine or `pbs_server` in Torque). Nodes have a *TaskTracker* process that manages tasks on the individual nodes. The TaskTrackers, which are similar to node daemons in HPC clusters (e.g., `sge_execd` in Grid Engine or `pbs_mom` in Torque) communicate with and are controlled by the JobTracker. Similar to most resource managers, the JobTracker has two pluggable scheduler modules: *Capacity* and *Fair*.

The JobTracker is responsible for managing the TaskTrackers on worker server nodes, tracking resource consumption and availability, scheduling individual job tasks, tracking progress, and providing fault tolerance for tasks. Similar to many HPC nodes, the TaskTracker is directed by the JobTracker and is responsible for launch and teardown of jobs and provides task status information to the JobTracker. The TaskTrackers also communicate through heartbeats to the JobTracker: If the JobTracker does not receive a heartbeat from a TaskTracker, it assumes it has failed and takes appropriate action (e.g., restarts jobs).

One key feature of Hadoop is the use of a simple redundancy model. Most Hadoop clusters are constructed from commodity hardware (x86 servers, Ethernet, and hard drives). Hardware is assumed to fail and thus processing and storage redundancy are part of the top-level Hadoop design. Because the MapReduce process is "functional" in nature, data can only move in one direction. For instance, input files cannot be altered as part of the MapReduce process. This restriction allows for a very simple computation redundancy model in which dead processes on failed nodes can be restarted on other servers with no loss of results (execution time may be extended, however). Moreover, HDFS can be configured easily with enough redundancy so that losing a node or rack of nodes will not result in job failure or data loss. This "non-stop design" has been a hallmark of Hadoop clusters and is very different from most HPC systems that accept the occasional hardware failure and subsequent job termination.

The monolithic MapReduce design is not without issues, however. As Hadoop clusters have grown in size, pressure on the JobTracker increases with regard to scalability, cluster utilization, system upgrades, and support for workloads other than MapReduce. These issues have been the rationale for Yet Another Resource Negotiator for Hadoop clusters.

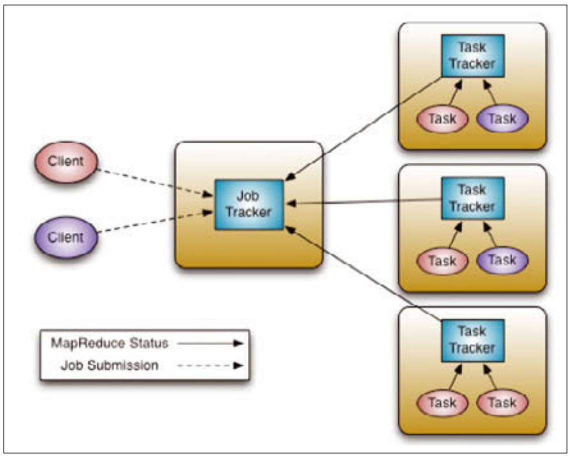


Figure 1: Hadoop version 1 MapReduce architecture (courtesy Apache Hadoop).

YARN Framework Architecture

When designing YARN, the obvious change was to separate resource management from data processing. That is, MapReduce had to be decoupled from workload management. As part of the decoupling, it would also be possible to create other data processing frameworks that could run on Hadoop clusters. These changes, from Hadoop version 1 to Hadoop

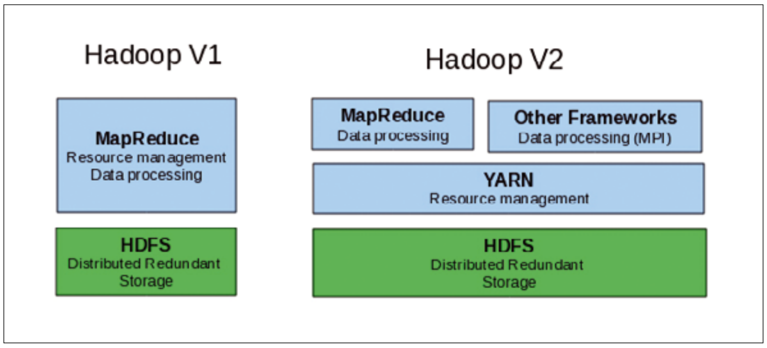


Figure 2: Hadoop version 2 splits the scheduling and data processing tasks.

version 2, are shown in Figure 2. MapReduce has now become "just another framework" that is managed by YARN. Great effort, however, has gone into keeping the YARN MapReduce functionality backward compatible with existing Hadoop version 1 MapReduce jobs. Existing MapReduce programs and applications, such as Pig and Hive, will continue to run under YARN without any code changes. Also note that HDFS remains an "independent" core component and can be used by either Hadoop version 1 or version 2. HDFS 2.0 was released in October 2013, after this article was originally written, with the introduction of "a slew of major features" [3].

The job of YARN is scheduling jobs on a Hadoop cluster. Breaking out scheduling from data processing required a more complex relationship between jobs and servers. As such, YARN introduces a number of new components: a ResourceManager, an ApplicationMaster, application Containers, and NodeManagers. The *ResourceManager* is a pure scheduler. Its sole purpose is to manage available resources among multiple applications on the cluster. As with version 1, both Fair and Capacity scheduling options are available.

The *ApplicationMaster* is responsible for accepting job submissions, negotiating resource Containers from the ResourceManager, and tracking progress. ApplicationMasters are specific to and written for each type of application. For example, YARN includes a distributed shell framework that runs a shell script on multiple nodes on the cluster. The ApplicationMaster also provides the service for restarting the ApplicationMaster Container on failure.

ApplicationMasters request and manage *Containers*, which grant rights to an application to use a specific amount of resources (memory, CPU, etc.) on a specific host. The idea is similar to a resource slot in an HPC scheduler. The ApplicationMaster, once given resources by the ResourceManager, contacts the NodeManager to start individual tasks. For example, using the MapReduce framework, these tasks would be mapper and reducer processes. In other frameworks, the tasks are different.

The *NodeManager* is the per-machine framework agent that is responsible for Containers, monitoring their resource usage (CPU, memory, disk, network), and reporting back to the ResourceManager.

Figure 3 shows the various components of the new YARN design, with two ApplicationMasters running within the cluster, one of which has three Containers (the red client) and one that has one Container (the purple client). Note that the ApplicationMasters run on cluster nodes and not as part of the ResourceManager, thus reducing the pressure on a central scheduler. Also, because ApplicationMasters have dynamic control of Containers, cluster utilization can be improved.

YARN supports a very general resource model. User applications, through the ApplicationMaster, can request resources with specific requirements, such as the amount of memory, number of cores (and type), and specific hostnames, racknames, and possibly more complex network topologies. Eventually support for disk/network I/O, GPUs, and other resources will be provided as well.

In addition to better scalability, utilization, and user agility, the design of YARN has introduced the possibility of multiple frameworks that go beyond MapReduce.

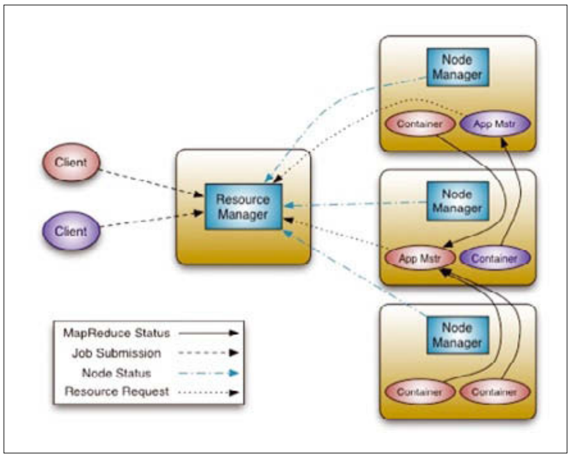


Figure 3: Hadoop version 2, YARN architecture (courtesy Apache Hadoop).

As mentioned, MPI is a logical choice for such a framework; others include Apache Giraph [4], Spark [5], Tez [6], Hadoop Distributed Shell, and others that are under development.

MPI and YARN

The prospect of running MPI jobs under YARN is an enticing proposition. As mentioned, the vast stores of data now living in HDFS can be accessed by non-MapReduce applications like those written in MPI. (Please note, however, that the intimate interface of MapReduce to HDFS is part of the MapReduce framework, is not automatic in other frameworks, and must be managed by the user's application.)

The prospect of running Open MPI under YARN [7] has been investigated by Ralph H. Castain of the Open MPI team. YARN was compared with the SLURM HPC scheduler as a platform for running Open MPI applications. The results heavily favored SLURM because of some of the fundamental design differences between the two approaches to cluster resource utilization.

Hadoop is designed around heartbeat communication between nodes. Unlike most HPC schedulers, there is no global status information on nodes. All status and task assignment is done via the heartbeat connections, which occur 200 times a second. This design also forces *N* linear launch scaling, whereas SLURM has log *N* launch scalability, resulting in much faster start-up times (*N* being the number of nodes). YARN has no internode communication on start-up, which precludes collective communication for wire-up exchange. Additionally, Hadoop clusters are exclusively Ethernet based with no support for native InfiniBand communication.

Because of these issues, the MR+ (MapReduce Plus) project was started as a way to bring the Hadoop MapReduce classes to general-purpose HPC clusters. In essence, MR+ is designed to eliminate YARN altogether and use existing HPC resource managers to launch and run parallel MapReduce jobs using an MPI framework. MR+, although still under development, provides an effective way for existing HPC clusters to integrate Hadoop MapReduce with other traditional HPC applications.

The YARN heartbeat design, while inefficient for large-scale start-up, has a distinct advantage in terms of hardware failure and job recovery. Recall that in the Hadoop MapReduce design, node failure is expected and managed in real time – jobs do not stop. Recovery in Open MPI and SLURM environments is not as easy because global state information and coordinated action are needed to respond to failures. Thus, a trade-off between easy failure recovery and performance separates the two approaches. Recall that Hadoop clusters have combined nodes that are used for both the filesystem (i.e., HDFS) and processing (e.g., YARN frameworks); thus, it requires a cleaner failure model than most HPC clusters in which filesystem hardware is separate from computation hardware.

An Invitation

YARN brings exciting capabilities to Big Data clusters. In particular, the prospect of running MPI applications on a Hadoop cluster could open up a whole new mode of Big Data processing. To be clear, YARN is not intended as a replacement for existing HPC clusters, where entrenched, or where mature resource managers meet the needs of current technical computing users. YARN can be viewed as an invitation to bring many of the skills and lessons learned in the MPI HPC world to the ever-growing world of Big Data.

If you are interested in testing YARN or learning more, you can visit the official Apache Hadoop YARN site [8] or download a free eBook, *Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2* [9]. If you look closely, you may recognize one of the authors. ■

Info

[1] Apache Hadoop YARN: [http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html]

[2] Apache Hadoop releases: [http://hadoop.apache.org/releases.html#News]

[3] HDFS upgrade: [http://hortonworks.com/blog/hdfs-2-0-next-generation-architecture/]

[4] Apache Giraph: [http://giraph.apache.org/]

[5] Spark: [http://spark-project.org/docs/0.6.0/index.html]

[6] Tez: [http://hortonworks.com/blog/introducing-tez-faster-hadoop-processing/]

[7] MR+: A Technical Overview: [http://www.open-mpi.org/video/mrplus/Greenplum_RalphCastain-2up.pdf]

[8] YARN: [http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html]

[9] Murthy, Arun C., Vinod Kumar Vavilapalli, Doug Eadline, Joseph Niemiec, and Jeff Markham. *Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2* Addison-Wesley. 2014, [http://it-ebooks.info/book/3676/]

Rackspace's Kenny Gorman talks about the new era of data in the cloud

Data Time

Rackspace founded the OpenStack project in 2010, and the company is still a leading voice in the hosting and cloud community. We talked to Rackspace Chief Technologist Kenny Gorman about what Rackspace is doing to integrate database services and the Big Data revolution with the cloud environment.

ADMIN Magazine: You were recently name Chief Technologist of Data at Rackspace. Databases have traditionally been a thorny issue for cloud companies. What is Rackspace's approach to databases?

Kenny Gorman: I am very humbled and excited in this new role for sure. Rackspace has great data offerings today, but the data landscape is changing fast. We must help customers succeed in this next era of data-oriented applications. Data is at the heart of applications these days. If a customer isn't using data to have an edge, they are behind. Data is a core component in ensuring companies are more competitive, apps are more engaging, or customers are responded to better.

The data landscape is such that companies often need help to get running and stay running. In addition to conventional hosting, we offer a range of support services, architecture design, and even help with simple tasks like writing automation scripts. We support both public and private clouds and provide everything from raw infrastructure to fully managed cloud environments. And we have tailored our offerings to support both small and large businesses.

AM: A big announcement this week was Rackspace Cloud Big Data OnMetal. Could you tell us a little about what Big Data onMetal is and how you think it will be used?

KG: The new Big Data OnMetal service will allow users to deploy dedicated on-metal instances of Hadoop or Spark in three clicks. I think customers who have high-performance requirements for Spark will love this offering. The combination of Spark 1.1 and onMetal servers is a great match. The value of the onMetal offering is quite good: lots of performance for a good price. Ultimately, the value statement for Spark is to help customers gain insight into their data to please customers, be more competitive, or save costs.

AM: What are the advantages of Rackspace Cloud Big Data OnMetal, as opposed to a multi-tenant cloud environment?

KG: The primary benefit of an onMetal configuration is performance. The customer gets a larger memory footprint and a massive amount of I/O. The I/O subsystem is all PCIE flash (LSI Warp Drives). The server is provisioned via OpenStack, just like any other cloud server. But in this case the customer gets a bare metal OpenCompute-based Linux server packed with PCIE flash storage. Because it's single tenant, there isn't any additional software layer required, and thus 100 percent of the bare metal performance is exposed to the calling application: in this case Spark. I have spent quite some time benchmarking these servers for various database workloads, including PostgreSQL and MongoDB/TokuMX, and I have to say the Rackspace onMetal team has done a great job; these things are blazing fast.

AM: You're one of the founders of ObjectRocket. What is ObjectRocket, and how does it fit in with the big picture of Rackspace and the cloud?

KG: ObjectRocket is a premium NoSQL database as a service platform. We started ObjectRocket because we felt there was a whole class of MongoDB users not getting premium performance and premium support. We felt both these components were key for customers to be successful with MongoDB. We ended up building out our own hardware stack, all based on Solid State Disk.
We built in AWS Direct Connect so customers with apps on AWS could access our back-end database with low latency and without charges. Today ObjectRocket still has this same philosophy, but with updates across the board and support for other NoSQL databases like Redis. Plus, we now have a staff of some of the most talented folks I have ever met. These philosophies of best in breed components with top talent for support is perfectly in line with Rackspace.

AM: Why MongoDB?

KG: MongoDB is a widely accepted and popular document-based data store. It's great for developers to use because they can write and read data records in a way that closely matches how they model data in modern applications (JSON). MongoDB has tons of native drivers for almost any language, and it makes it very easy to get started and helps time to market when building applications. One elegant design pattern in MongoDB is that not only is the data being returned in JSON format, but so are the arguments to the queries being run. The drivers serialize the results right into native data structures. For instance, in Python, data is returned right into native dictionaries. It's a slick and wonderful way to interact with data.

AM: How does ObjectRocket fit with Hadoop?

KG: ObjectRocket is more than just MongoDB these days, it's Redis, MySQL, and Hadoop. MongoDB itself and Hadoop are a nice pair. They work nicely together when, say, an application needs to perform a large analytic query in Hadoop to build a dataset of purchase recommendations based on historical trends in a demographic; then, that dataset can be stored in MongoDB beside the rest of the online application and display those recommendations within the regular application workflow. Those recommendations can be periodically updated using the Hadoop cluster. The MongoDB database can focus towards the online app and not be burdened with the tough analytic query crunching. They work great together.

AM: Could you describe a real-world scenario showing how this technology would be used? What is an example of a problem or situation where a company would want to use ObjectRocket and the tools that surround it?

KG: Sure. We see it all the time. Customers want to focus on compelling applications, they don't want to have to be a DBA or know every little nuance of a database system. They just want the database to work, and if there is a problem or they have design questions, they want to be able to get some help. Some of the most interesting applications are iOS or Andriod apps where the engineering team wants to focus on an amazing customer experience and let ObjectRocket/Rackspace handle the details of running the database. MapMyFitness (fitness) and Untappd (beer) are two customers who use our services for just this reason.

AM: Where do you see all this going? Where does Rackspace want to be in five years? Which technologies will be important, and what will your customers be doing then that they can't do now?

KG: Data is growing, everyone says that, but not everyone is ready for it. To remain competitive – heck, to remain in business – customers must utilize all the data at their disposal. They must collect the right data, smartly clean and transform the data, and ensure they ultimately gain insights from that data. Otherwise they are behind. The database offerings at Rackspace are uniquely positioned to help customers make the most out of this data. Rackspace is the only vendor who not only has cutting edge technologies, but also backs it up with Fanatical Support. This stuff is complicated, and Rackspace has the unique culture that ensures customers are successful. I am very proud of that fact.
If you are a data geek like me, the next few years are likely to be the most exciting of our lives. Technologies are rapidly changing to meet new data needs. Data itself is more recognized as a key business asset than previously. It's all adding up to a space where Rackspace is going to lead, not follow. I am excited to be a small part of that. ■

rackspace®

the #1 ***managed cloud*** *company*

You Manage Your Business. We'll Manage Your Cloud.

- Over 60% of the Fortune® 100 trust us to host their workloads
- 200,000+ customers worldwide depend on our **Fanatical Support®**
- A leader in the Gartner Magic Quadrant® for Cloud-Enabled Managed Hosting, North America & Europe, 2014

Let us help make your job easier.

www.**rackspace**.com

rackspace®

the #1 **managed cloud** company

Focus on Your Application.
Not Your Database.

- Scalable, fast, reliable, automated
- Managed MongoDB & Redis
- All backed by **Fanatical Support**®

Start your free 30-day trial today.

ObjectRocket

by Rackspace.

www.**objectrocket**.com