Hadoop implements the ability for pluggable schedulers that assign resources to jobs. However, as we know from traditional scheduling, not all algorithms are the same, and efficiency is workload and cluster dependent. Get to know Hadoop scheduling, and explore two of the algorithms available today: fair scheduling and capacity scheduling. Also, learn how these algorithms are tuned and in what scenarios they're relevant.

Hadoop is a general-purpose system that enables high-performance processing of data over a set of distributed nodes. But within this definition is the fact that Hadoop is a multi-tasking system that can process multiple data sets for multiple jobs for multiple users at the same time. This capability of multi-processing means that Hadoop has the opportunity to more optimally map jobs to resources in a way that optimizes their use.

Up until 2008, Hadoop supported a single scheduler that was intermixed with the JobTracker logic. Although this implementation was perfect for the traditional batch jobs of Hadoop (such as log mining and Web indexing), the implementation was inflexible and could not be tailored. Further, Hadoop operated in a batch mode, where jobs were submitted to a queue, and the Hadoop infrastructure simply executed them in the order of receipt.

Luckily, a bug report (HADOOP-3412) was submitted for an implementation of a scheduler that was independent of the JobTracker. More importantly, the new scheduler is pluggable, which allows use of new scheduling algorithms to help optimize jobs that have specific characteristics. A further advantage to this change is the increased readability of the scheduler, which has opened it up to greater experimentation and the potential for a growing list of schedulers to specialize in Hadoop's ever-increasing list of applications.

With this change, Hadoop is now a multi-user data warehouse that supports a variety of different types of processing jobs, with a pluggable scheduler framework providing greater control. This framework allows optimal use of a Hadoop cluster over a varied set of workloads (from small jobs to large jobs and everything in between). Moving away from FIFO scheduling (which treats a
job's importance relative to when it was submitted) allows a Hadoop cluster to support a variety of workloads with varying priority and performance constraints.

Note: This article assumes some knowledge of Hadoop. See Resources for links to an introduction to the Hadoop architecture and the practical Hadoop series for installing, configuring, and writing Hadoop applications.

The core Hadoop architecture

A Hadoop cluster consists of a relatively simple architecture of masters and slaves (see Figure 1). The NameNode is the overall master of a Hadoop cluster and is responsible for the file system namespace and access control for clients. There also exists a JobTracker, whose job is to distribute jobs to waiting nodes. These two entities (NameNode and JobTracker) are the masters of the Hadoop architecture. The slaves consist of the TaskTracker, which manages the job execution (including starting and monitoring jobs, capturing their output, and notifying the JobTracker of job completion). The DataNode is the storage node in a Hadoop cluster and represents the distributed file system (or at least a portion of it for multiple DataNodes). The TaskTracker and the DataNode are the slaves within the Hadoop cluster.

Figure 1. Elements of a Hadoop cluster

Note that Hadoop is flexible, supporting a single node cluster (where all entities exist on a single node) or a multi-node cluster (where JobTracker and NameNodes are distributed across thousands of nodes). Although little information exists on the larger production environments that exist, the largest known Hadoop cluster is Facebook's, which consists of 4000 nodes. These nodes are split among several sizes (half include 8- and 16-core CPUs). The Facebook cluster also supports 21PB of storage distributed across the many DataNodes. Given the large number of resources and the potential for many jobs from numerous users, scheduling is an important optimization going forward.

Hadoop schedulers

Since the pluggable scheduler was implemented, several scheduler algorithms have been developed for it. The sections that follow explore the various algorithms available and when it makes sense to use them.
FIFO scheduler

The original scheduling algorithm that was integrated within the JobTracker was called *FIFO*. In FIFO scheduling, a JobTracker pulled jobs from a work queue, oldest job first. This schedule had no concept of the priority or size of the job, but the approach was simple to implement and efficient.

Fair scheduler

The core idea behind the fair share scheduler was to assign resources to jobs such that on average over time, each job gets an equal share of the available resources. The result is that jobs that require less time are able to access the CPU and finish intermixed with the execution of jobs that require more time to execute. This behavior allows for some interactivity among Hadoop jobs and permits greater responsiveness of the Hadoop cluster to the variety of job types submitted. The fair scheduler was developed by Facebook.

The Hadoop implementation creates a set of pools into which jobs are placed for selection by the scheduler. Each pool can be assigned a set of shares to balance resources across jobs in pools (more shares equals greater resources from which jobs are executed). By default, all pools have equal shares, but configuration is possible to provide more or fewer shares depending upon the job type. The number of jobs active at one time can also be constrained, if desired, to minimize congestion and allow work to finish in a timely manner.

To ensure fairness, each user is assigned to a pool. In this way, if one user submits many jobs, he or she can receive the same share of cluster resources as all other users (independent of the work they have submitted). Regardless of the shares assigned to pools, if the system is not loaded, jobs receive the shares that would otherwise go unused (split among the available jobs).

The scheduler implementation keeps track of the compute time for each job in the system. Periodically, the scheduler inspects jobs to compute the difference between the compute time the job received and the time it should have received in an ideal scheduler. The result determines the deficit for the task. The job of the scheduler is then to ensure that the task with the highest deficit is scheduled next.

You configure fair share in the mapred-site.xml file. This file defines the properties that collectively govern the behavior of the fair share scheduler. An XML file—referred to with the property `mapred.fairscheduler.allocation.file`—defines the allocation of shares to each pool. To optimize for job size, you can set the `mapred.fairscheduler.sizebasedweight` to assign shares to jobs as a function of their size. A similar property allows smaller jobs to finish faster by adjusting the weight of the job after 5 minutes (`mapred.fairscheduler.weightadjuster`). Numerous other properties exist that you can use to tune loads over the nodes (such as the number of maps and reduces that a given TaskTracker can manage) and define whether preemption should be performed. See Resources for a link to a full list of configurable properties.

Capacity scheduler

The capacity scheduler shares some of the principles of the fair scheduler but has distinct differences, too. First, capacity scheduling was defined for large clusters, which may have multiple,
independent consumers and target applications. For this reason, capacity scheduling provides greater control as well as the ability to provide a minimum capacity guarantee and share excess capacity among users. The capacity scheduler was developed by Yahoo!.

In capacity scheduling, instead of pools, several queues are created, each with a configurable number of map and reduce slots. Each queue is also assigned a guaranteed capacity (where the overall capacity of the cluster is the sum of each queue’s capacity).

Queues are monitored; if a queue is not consuming its allocated capacity, this excess capacity can be temporarily allocated to other queues. Given that queues can represent a person or larger organization, any available capacity is redistributed for use by other users.

Another difference of fair scheduling is the ability to prioritize jobs within a queue. Generally, jobs with a higher priority have access to resources sooner than lower-priority jobs. The Hadoop road map includes a desire to support preemption (where a low-priority job could be temporarily swapped out to allow a higher-priority job to execute), but this functionality has not yet been implemented.

Another difference is the presence of strict access controls on queues (given that queues are tied to a person or organization). These access controls are defined on a per-queue basis. They restrict the ability to submit jobs to queues and the ability to view and modify jobs in queues.

You configure the capacity scheduler within multiple Hadoop configuration files. The queues are defined within hadoop-site.xml, and the queue configurations are set in capacity-scheduler.xml. You can configure ACLs within mapred-queue-acls.xml. Individual queue properties include capacity percentage (where the capacity of all queues in the cluster is less than or equal to 100), the maximum capacity (limit for a queue's use of excess capacity), and whether the queue supports priorities. Most importantly, these queue properties can be manipulated at run time, allowing them to change and avoid disruptions in cluster use.

**Other approaches**

Although not a scheduler per se, Hadoop also supports the idea of provisioning virtual clusters from within larger physical clusters, called Hadoop On Demand (HOD). The HOD approach uses the Torque resource manager for node allocation based on the needs of the virtual cluster. With allocated nodes, the HOD system automatically prepares configuration files, and then initializes the system based on the nodes within the virtual cluster. Once initialized, the HOD virtual cluster can be used in a relatively independent way.

HOD is also adaptive in that it can shrink when the workload changes. HOD automatically de-allocates nodes from the virtual cluster after it detects no running jobs for a given time period. This behavior permits the most efficient use of the overall physical cluster assets.

HOD is an interesting model for deployments of Hadoop clusters within a cloud infrastructure. It offers an advantage in that with less sharing of the nodes, there is greater security and, in some cases, improved performance because of a lack of contention within the nodes for multiple users' jobs.
When to use each scheduler

From the discussion above, you can see where these scheduling algorithms are targeted. If you're running a large Hadoop cluster, with multiple clients and different types and priorities of jobs, then the capacity scheduler is the right choice to ensure guaranteed access with the potential to reuse unused capacity and prioritize jobs within queues.

Although less complex, the fair scheduler works well when both small and large clusters are used by the same organization with a limited number of workloads. Fair scheduling still provides the means to non-uniformly distribute capacity to pools (of jobs) but in a simpler and less configurable way. The fair scheduler is useful in the presence of diverse jobs, because it can provide fast response times for small jobs mixed with larger jobs (supporting more interactive use models).

Future developments in Hadoop scheduling

Now that the Hadoop scheduler is pluggable, you should see new schedulers developed for unique cluster deployments. Two in-process schedulers (from the Hadoop issues list) include the adaptive scheduler and the learning scheduler. The learning scheduler (MAPREDUCE-1349) is designed to maintain a level of utilization when presented with a diverse set of workloads. Currently, this scheduler implementation focuses on CPU load averages, but utilization of network and disk I/O is planned. The adaptive scheduler (MAPREDUCE-1380) focuses on adaptively adjusting resources for a given job based on its performance and user-defined business goals.

Conclusion

The introduction of the pluggable scheduler was yet another evolution in cluster computing with Hadoop. The pluggable scheduler permits the use (and development) of schedulers optimized for the particular workload and application. The new schedulers have also made it possible to create multi-user data warehouses with Hadoop, given the ability to share the overall Hadoop infrastructure with multiple users and organizations.

Hadoop is evolving as its use models evolve and now supports new types of workloads and usage scenarios (such as multi-user or multi-organization big data warehouses). The new flexibility that Hadoop provides is a great step toward more optimized use of cluster resources in big data analytics.
Resources

Learn

- The Apache Hadoop website is the best source for documentation, mailing lists, and where to learn more about Hadoop, including its installation and configuration.
- Each of the schedulers offers a large range of configurable properties. You can find these properties through Apache for the fair scheduler and the capacity scheduler.
- Distributed computing with Linux and Hadoop (Ken Mann and M. Tim Jones, developerWorks, December 2008) provides an introduction to the architecture of Hadoop, including the basics of the MapReduce paradigm for distributed processing of bulk data.
- HADOOP-3412 defines the issue for refactoring the scheduler out of the JobTracker. The original implementation of Hadoop mixed the scheduler algorithm within the JobTracker node. This issue, submitted in 2008, resulted in the removal of the fixed scheduler and the addition of a pluggable scheduler supporting greater flexibility for the varied workloads that Hadoop supports.
- You can find a practical introduction to Hadoop in the developerWorks "Distributed data processing with Hadoop" series by M. Tim Jones. Part 1, Getting started (May 2010) provides an introduction to setting up and using a single-node Hadoop cluster. In part 2, Going further (June 2010), you learn how to set up and use a multi-node cluster. Finally, in part 3, Application development (July 2010), you learn how to develop map and reduce applications within the Hadoop environment.
- The topic of scheduling is an interesting one in Linux®, and you can find similarities among scheduling jobs in Hadoop and scheduling threads within Linux. You can learn more about Linux scheduling in Inside the Linux Scheduler (M. Tim Jones, developerWorks, June 2006), Inside the Linux 2.6 Completely Fair Scheduler (M. Tim Jones, developerWorks, December 2009), and Linux Scheduler simulation (M. Tim Jones, developerWorks, February 2011).
- IBM InfoSphere BigInsights Basic Edition offers a highly scalable and powerful analytics platform that can handle incredibly high data throughput rates that can range to millions of events or messages per second.
- The Open Source developerWorks zone provides a wealth of information on open source tools and using open source technologies.
- Follow developerWorks on Twitter. You can also follow this author on Twitter at M. Tim Jones.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners to advanced functionality for experienced developers.
- IBM InfoSphere BigInsights Basic Edition -- IBM's Hadoop distribution -- is an integrated, tested and pre-configured, no-charge download for anyone who wants to experiment with and learn about Hadoop.
- Find free courses on Hadoop fundamentals, stream computing, text analytics, and more at Big Data University.

Get products and technologies

- IBM InfoSphere BigInsights Basic Edition -- IBM's Hadoop distribution -- is an integrated, tested and pre-configured, no-charge download for anyone who wants to experiment with and learn about Hadoop.
• **Download IBM InfoSphere BigInsights Basic Edition** at no charge and build a solution that turns large, complex volumes of data into insight by combining Apache Hadoop with unique technologies and capabilities from IBM.

• **Evaluate IBM products** in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the **SOA Sandbox** learning how to implement service-oriented architecture efficiently.

**Discuss**

• Get involved in the **My developerWorks community**. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.
About the author

M. Tim Jones

M. Tim Jones is an embedded firmware architect and the author of Artificial Intelligence: A Systems Approach, GNU/Linux Application Programming (now in its second edition), AI Application Programming (in its second edition), and BSD Sockets Programming from a Multilanguage Perspective. His engineering background ranges from the development of kernels for geosynchronous spacecraft to embedded systems architecture and networking protocols development. Tim is a platform architect with Intel and author in Longmont, Colo.