PEPPERDATA IN MULTI-TENANT ENVIRONMENTS

technical whitepaper

June 2015

pepperdata

SUMMARY OF WHAT'S WRITTEN IN THIS DOCUMENT

If you are short on time and don't want to read the whole document, here's what you need to know:

- Pepperdata facilitates multi-tenancy and eliminates the need to physically separate workloads onto different clusters via our SLA enforcement which is configured via policies.
- Pepperdata policies work in combination with YARN to ensure that jobs behave as you intended. YARN forgets about jobs once they are running on the cluster, but Pepperdata's is watching in real-time and will slow down low-priority jobs on the contended resource to ensure high-priority jobs continue unimpeded.
- Pepperdata's SLA enforcement is different from cgroups in that our policies are expressed in terms of minimums, rather than maximums with cgroups, allowing you to guarantee high-priority jobs a minimum of the cluster's resources, but only when needed. If there is no contention on a resource, then the low-priority job is free to use as much of that resource it needs. With cgroups, the maximum is enforced at all times, unnecessarily clamping down on resources even when the cluster is not busy.
- Pepperdata's fine-grained visibility into how a cluster's resources are being utilized allows operators and users to pinpoint ways to improve jobs so that over time, things run faster and more smoothly.

Now read on for more detail...

INTRODUCTION

Multi-tenancy refers to multiple business users and processes sharing a common set of resources, such as a Hadoop cluster. In these kinds of environments, organizations get a central system to store and work with all data of all types and formats, with the flexibility to run a variety of workloads including batch processing, analytic SQL, stream processing, enterprise search, and machine learning. The overarching challenge with all of these mixed workloads is the impact they have when contending for the four major resources: CPU, RAM, disk i/o, and network bandwidth. This document was written to explain how Pepperdata provides reliable multi-tenancy on Hadoop clusters.

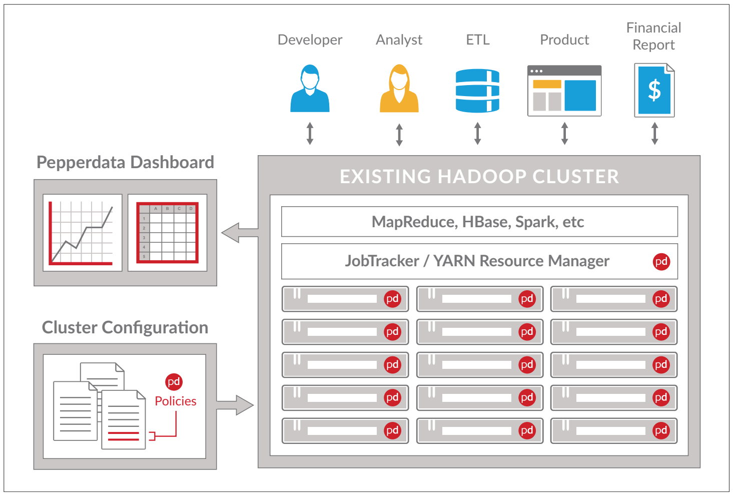
PEPPERDATA OVERVIEW

Pepperdata is an enterprise software product that installs in about 20 minutes on your existing Hadoop cluster without any modifications to scheduler, workflow, or job submission processes. Once installed, Pepperdata provides you with three immediate benefits:

VISIBILITY

Captures an unprecedented level of detail on cluster resource usage. Pepperdata collects 200+ metrics in real-time for the four resources CPU, RAM, disk I/O, and network for any given job or task, by user or group or queue. This allows operators to quickly identify what job is causing a problem and which user submitted it. And it allows users to see what and how their jobs are doing on the cluster while they are running.

PEPPERDATA REAL-TIME ARCHITECTURE



CONTROL

Enables you to implement service-level policies that guarantee on-time completion of high-priority jobs.
Pepperdata senses contention among the four resources in real-time and will slow down low-priority jobs just enough to ensure the high-priority SLAs are always maintained.

CAPACITY

Increases cluster throughput by 30-50%.
Pepperdata knows the actual true hardware resource capacity of your cluster and allows more tasks to run on nodes that have free resources at any given moment. In many instances jobs will run much faster because Pepperdata will dynamically allow them to use more of the true resource on the cluster when it is available.

PEPPERDATA MULTI-TENANCY

HOW PEPPERDATA ELIMINATES HAVING TO PHYSICALLY ISOLATE WORKLOADS

It only takes one process to interfere with a production high-priority job on your Hadoop cluster. Once this starts happening, organizations are forced into building new additional clusters to separate the workloads. Over time it is not uncommon for more and more clusters to be added. First production jobs get their own cluster. Ad-hoc jobs get another. Then another cluster is built for the job or set of jobs that are so critical, they need to be isolated. HBase and MapReduce jobs contend for the same resources, so HBase gets its own cluster. Data loading SLAs start getting compromised so ETL gets its own cluster. Of course this is a very inefficient and expensive way of dealing with mixed workloads and defeats the goal of achieving a true multi-tenant Hadoop environment.

Pepperdata eliminates the need to physically isolate mixed workloads because it is able to watch everything running on the cluster in real-time and enforce SLAs that you set up via simple policies. Administrators can set policies for a user, job group, queue, a job, or even a particular application. These policies guarantee that high-priority jobs and applications get a minimum amount of resources. Some of these may use a lot of memory. If they use too much memory, it can cause swapping on worker nodes, slowing down other jobs and applications running on the cluster. Pepperdata can be used to limit the amount of memory used in order to protect nodes in the cluster from swapping. Here are a few examples of policies you can set with Pepperdata.

EXAMPLE 1 - PROD AND DEV RUNNING ON THE SAME CLUSTER

```
<configuration>
  <queue>
    <name>prod</name>
    <minPercent>60</minPercent>
  </queue>
  <queue>
    <name>dev</name>
    <minPercent>10</minPercent>
  </queue>
</configuration>
```

Example 1 configures two queue-level job groups. The two job groups are named `prod` and `dev`. These names should match the associated job queue names that exist on the cluster.

Using the `minPercent` key, the example specifies minimum resource percentages to each of the two job groups, 60% and 10%, respectively.

EXAMPLE 2 - HBASE AND MAPREDUCE RUNNING ON THE SAME CLUSTER

Example 2 configures two job groups with the first group using the program-level group with the reserved name `HBase`, which will internally map to the Java class name of the HBase region server programs. This lets all region server instances running on the cluster be recognized as members of the program group with access to a minimum of 60% of the cluster resources. With the `protected` flag, the `HBase` group's resource use beyond the minimum will be protected from other groups' interference when the resource in question is contended. The second job group uses the `default` job group. In this example, all MapReduce jobs will belong to this default group since there is no other job group configured.

```
<configuration>
  <program>
    <name>HBase</name>
    <minPercent>60</minPercent>
    <protected>true</protected>
  </program>
  <default>
    <minPercent>30</minPercent>
  </default>
</configuration>
```

EXAMPLE 3 - MEMORY PROTECTION

Example 3 shows how you can set memory limits. The memory limit can be set on the resident set size (RSS) or on the total virtual memory (VM) size of individual tasks or users. Here we are setting Maximum RSS size and Maximum VM size to 8 GB only for tasks submitted by user `prod`.

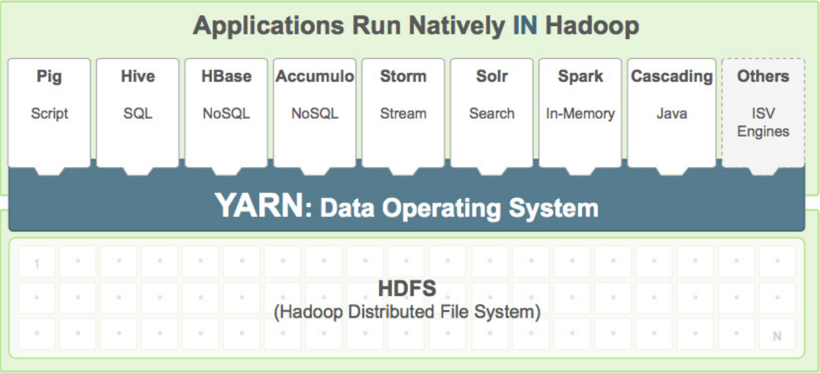
```
<configuration>
<user>
<name>prod</name>
<taskMemory>
  <rssLimitBytes>8000000000</rssLimitBytes>
  <vmLimitBytes>8000000000</vmLimitBytes>
</taskMemory>
</user>
</configuration>
```

Let's explain a little more about how minimum resource percentages work with Pepperdata. Jobs running in a job group may not actually use the guaranteed minimums at a given moment. For instance, the production jobs could be in a phase that only consumes CPU, but does not use disk or network. The Pepperdata software monitors actual bandwidth usage and shifts the unused amount to other job groups with more demand. So in Example 1, if the jobs in the `dev` group actually demand more than 10% while the `prod` group is not using their share, the `dev` group can receive more than the 10% bandwidth you specified in the configuration. Then if `prod` begins to demand their previously underutilized resources, then they will get back the guaranteed bandwidth amounts.

HOW PEPPERDATA WORKS WITH YARN TO ENFORCE MULTI-TENANCY

YARN stands for "Yet Another Resource Negotiator" and was introduced as part of Hadoop 2.0. YARN takes the resource management capabilities that were in MapReduce and packages them so they can be used by new engines. With YARN, you can now run multiple applications in Hadoop, all sharing a common resource management.

YARN coordinates consumption and usage reservations to ensure resources are fairly allocated and used.



This approach is sometimes referred to as dynamic partitioning. With the YARN and your Hadoop Scheduler, you can assign minimum guaranteed capacities to groups of users or applications. For instance, in an enterprise with three business units that share a cluster, each business unit can be assigned to a Scheduler queue and guaranteed a minimum capacity. The Scheduler was designed to allow significantly higher cluster utilization while still providing predictability for Hadoop workloads, while sharing resources in a predictable and simple manner, using the common notion of job queues.

But once a container is running on the cluster, YARN and the Scheduler forgets about it. YARN is not watching the jobs or applications in real-time once they are running. As a result, you could have a job start thrashing your disks or consuming too much network and YARN isn't able to do anything about it. With Pepperdata you are able to set policies that will absolutely enforce SLAs on the cluster. Because Pepperdata is actually watching in real-time, if it senses contention for a given resource, Pepperdata is able to reach in and, based on the policies that you have set, will micro-pause the low-priority job in real-time, which is something YARN is unable to do.

HOW PEPPERDATA IS DIFFERENT FROM CGROUPS

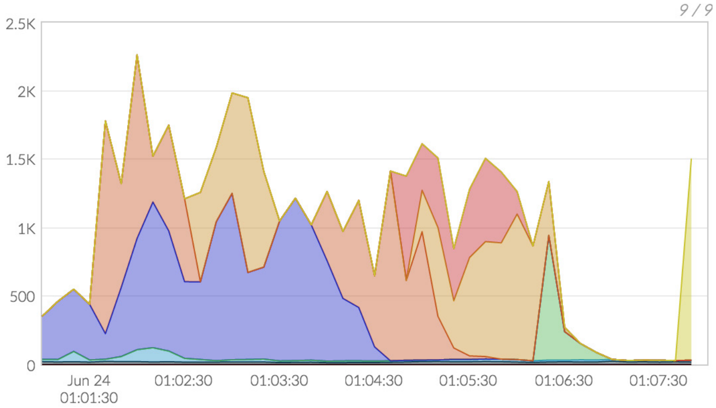
As opposed to dynamic partitioning as described above for YARN, there is also an approach that's a static partitioning model, which leverages a technology available on modern Linux operating systems called container groups, or cgroups. Cgroups provide a mechanism for managing and monitoring system resources by partitioning things like CPU time, system memory, disk i/o and network bandwidth into groups, and then assigning tasks to those groups. Cgroups limit the maximum amount CPU/memory/disk, and network. Pepperdata deals with minimums, which is much more useful. Let's say you run two jobs and they use the same resources. In order to pick a winner with cgroups, you have to limit the low-priority job (by re-running it and changing its config) instead of talking about the high-priority job and saying it gets a minimum guaranteed amount of system resource. That's the key difference between minimums and maximums. In order to use maximums, it's up to the Hadoop operator to correctly limit every potentially offending low-priority job as opposed to setting minimum guarantees for the high-priority jobs. And then there is the issue of knowing what to set the low-priority jobs' values to with cgroups. The operator basically needs to guess, and then via trial and error, try to get the values right across the different resources to try to minimize impact to the high-priority job. And currently, cgroups' disk block I/O subsystem does not work with buffered write operations. It is primarily targeted at direct I/O, although it works for buffered read operations. Hadoop only works with buffered I/O, so you can't rely on cgroups to address Hadoop write operations. In addition, cgroups only addresses local I/O operations, and with Hadoop these are not necessarily the most important ones.

Pepperdata has patented algorithms that look at all the resources second-by-second, in real-time, and dynamically make adjustments. With cgroups, it is set it and forget it. To even slightly approximate what Pepperdata does using cgroups, you'd have to manually change the cgroup setting constantly and it would have to be for all the low-priority jobs (which there are a lot more of those than there are high-priority jobs on any given cluster) which of course is an impossible task. And with Pepperdata, if there is no contention for a given resource, low-priority jobs are able to use as much of the resources as the need, unlike cgroups where the low-priority job is pegged for a maximum regardless of what's actually happening on the cluster.

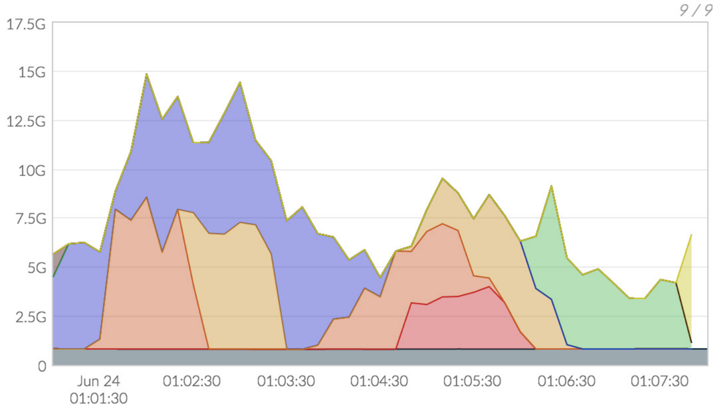
HOW PEPPERDATA IMPROVES MULTI-TENANCY VIA MONITORING

Pepperdata provides visibility into a particular job or application and what it is doing while it is running on the cluster. Operators and users are able to quickly identify unforeseen situations such as an errant job or process that is overwhelming an allotted resource. This allows investigation and possible correction to something like a poorly constructed ad-hoc query. The effect is smoother cluster operations over time, which is of course very important in a multi-tenant environment. Pepperdata collects 200+ metrics in real-time, every 3 to 5 seconds depending on the metric, that all boil down to different views on CPU, RAM, disk I/O, and network usage. You can break these down by user, group or queue to pinpoint exactly where a problem is occurring. As an example, the image below shows two charts produced by the Pepperdata Dashboard for CPU and RAM with time-series data color-coded by job.

User CPU percentage, by Job

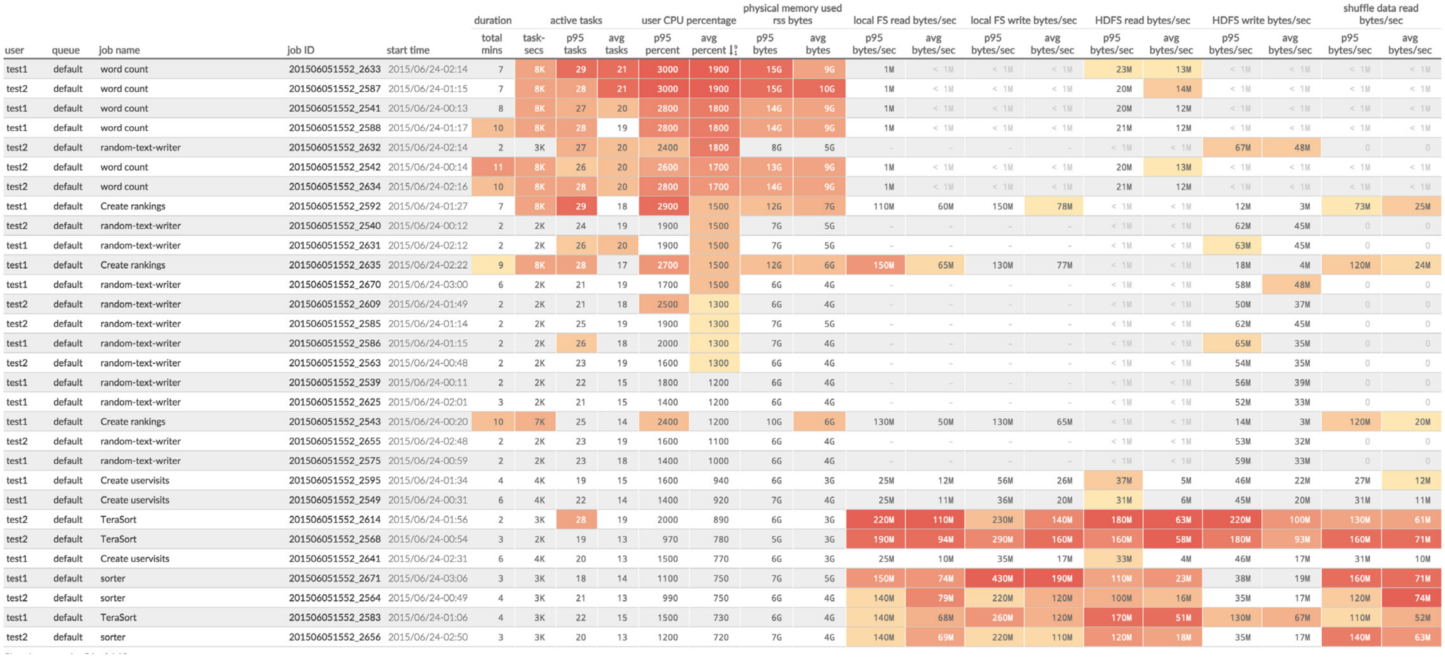


Physical memory used rss bytes, by Job



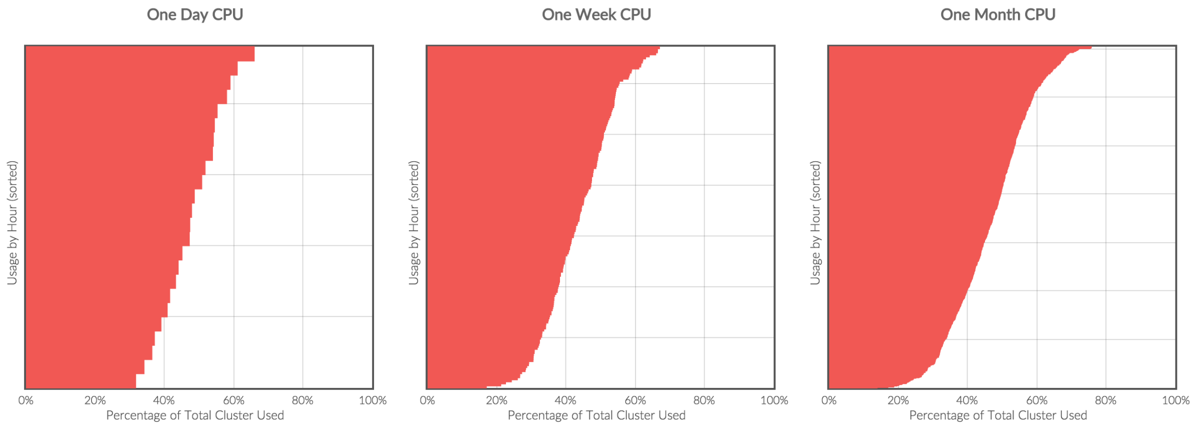
HOW PEPPERDATA HELPS WITH CHARGEBACKS

Another common requirement for multi-tenant environments is the ability to meter the cluster usage of different tenants. One of the key business drivers of multi-tenancy is the aggregation of resources to improve utilization and the multiple participants will build internal budgets to finance this resource pool. In many organizations, IT uses the metered information to drive chargeback models and illustrate compliance. Pepperdata provides historical and trending usage of the entire Hadoop compute layer (CPU, RAM, disk I/O, and network usage). Beyond just and with this information —which can be exported in common formats, such as Microsoft Excel, to financial modeling applications—can provide a strong foundation for an internal chargeback model. These metering capabilities can also facilitate capacity planning and accurate budgeting for growth of the shared platform, thus insuring that IT teams allocate sufficient resources in line with cluster demand. Below is an image showing a table from the Pepperdata Dashboard that shows who are the biggest users of the system.



					duration	active tasks			user CPU percentage		physical memory used rss bytes		local FS read bytes/sec		local FS write bytes/sec		HDFS read bytes/sec		HDFS write bytes/sec		shuffle data read bytes/sec	
user	queue	job name	job ID	start time	total mins	task-secs	p95 tasks	avg tasks	p95 percent	avg percent ↓	p95 bytes	avg bytes	p95 bytes/sec	avg bytes/sec	p95 bytes/sec	avg bytes/sec	p95 bytes/sec	avg bytes/sec	p95 bytes/sec	avg bytes/sec	p95 bytes/sec	avg bytes/sec
test1	default	word count	201506051552_2633	2015/06/24-02:14	7	8K	29	21	3000	1900	15G	9G	1M	< 1M	< 1M	< 1M	23M	13M	< 1M	< 1M	< 1M	< 1M
test2	default	word count	201506051552_2587	2015/06/24-01:15	7	8K	28	21	3000	1900	15G	10G	1M	< 1M	< 1M	< 1M	20M	14M	< 1M	< 1M	< 1M	< 1M
test1	default	word count	201506051552_2541	2015/06/24-00:13	8	8K	27	20	2800	1800	14G	9G	1M	< 1M	< 1M	< 1M	20M	12M	< 1M	< 1M	< 1M	< 1M
test1	default	word count	201506051552_2588	2015/06/24-01:17	10	8K	28	19	2800	1800	14G	9G	1M	< 1M	< 1M	< 1M	21M	12M	< 1M	< 1M	< 1M	< 1M
test2	default	random-text-writer	201506051552_2632	2015/06/24-02:14	2	3K	27	20	2400	1800	8G	5G	-	-	-	-	< 1M	< 1M	67M	48M	0	0
test2	default	word count	201506051552_2542	2015/06/24-00:14	11	8K	26	20	2600	1700	13G	9G	1M	< 1M	< 1M	< 1M	20M	13M	< 1M	< 1M	< 1M	< 1M
test2	default	word count	201506051552_2634	2015/06/24-02:16	10	8K	28	20	2800	1700	14G	9G	1M	< 1M	< 1M	< 1M	21M	12M	< 1M	< 1M	< 1M	< 1M
test1	default	Create rankings	201506051552_2592	2015/06/24-01:27	7	8K	29	18	2900	1500	12G	7G	110M	60M	150M	78M	< 1M	< 1M	12M	3M	73M	25M
test2	default	random-text-writer	201506051552_2540	2015/06/24-00:12	2	2K	24	19	1900	1500	7G	5G	-	-	-	-	< 1M	< 1M	62M	45M	0	0
test1	default	random-text-writer	201506051552_2631	2015/06/24-02:12	2	2K	26	20	1900	1500	7G	5G	-	-	-	-	< 1M	< 1M	63M	45M	0	0
test1	default	Create rankings	201506051552_2635	2015/06/24-02:22	9	8K	28	17	2700	1500	12G	6G	150M	65M	130M	77M	< 1M	< 1M	18M	4M	120M	24M
test1	default	random-text-writer	201506051552_2670	2015/06/24-03:00	6	2K	21	19	1700	1500	6G	4G	-	-	-	-	< 1M	< 1M	58M	48M	0	0
test2	default	random-text-writer	201506051552_2609	2015/06/24-01:49	2	2K	21	18	2500	1300	6G	4G	-	-	-	-	< 1M	< 1M	50M	37M	0	0
test2	default	random-text-writer	201506051552_2585	2015/06/24-01:14	2	2K	25	19	1900	1300	7G	5G	-	-	-	-	< 1M	< 1M	62M	45M	0	0
test1	default	random-text-writer	201506051552_2586	2015/06/24-01:15	2	2K	26	18	2000	1300	7G	4G	-	-	-	-	< 1M	< 1M	65M	35M	0	0
test2	default	random-text-writer	201506051552_2563	2015/06/24-00:48	2	2K	23	19	1600	1300	6G	4G	-	-	-	-	< 1M	< 1M	54M	35M	0	0
test1	default	random-text-writer	201506051552_2539	2015/06/24-00:11	2	2K	22	15	1800	1200	6G	4G	-	-	-	-	< 1M	< 1M	56M	39M	0	0
test1	default	random-text-writer	201506051552_2625	2015/06/24-02:01	3	2K	21	15	1400	1200	6G	4G	-	-	-	-	< 1M	< 1M	52M	33M	0	0
test1	default	Create rankings	201506051552_2543	2015/06/24-00:20	10	7K	25	14	2400	1200	10G	6G	130M	50M	130M	65M	< 1M	< 1M	14M	3M	120M	20M
test2	default	random-text-writer	201506051552_2655	2015/06/24-02:48	2	2K	23	19	1600	1100	6G	4G	-	-	-	-	< 1M	< 1M	53M	32M	0	0
test1	default	random-text-writer	201506051552_2575	2015/06/24-00:59	2	2K	23	18	1400	1000	6G	4G	-	-	-	-	< 1M	< 1M	59M	33M	0	0
test1	default	Create uservisits	201506051552_2595	2015/06/24-01:34	4	4K	19	15	1600	940	6G	3G	25M	12M	56M	26M	37M	5M	46M	22M	27M	12M
test1	default	Create uservisits	201506051552_2549	2015/06/24-00:31	6	4K	22	14	1400	920	7G	4G	25M	11M	36M	20M	31M	6M	45M	20M	31M	11M
test2	default	TeraSort	201506051552_2614	2015/06/24-01:56	2	3K	28	19	2000	890	6G	3G	220M	110M	230M	140M	180M	63M	220M	100M	130M	61M
test2	default	TeraSort	201506051552_2568	2015/06/24-00:54	3	2K	19	13	970	780	5G	3G	190M	94M	290M	160M	160M	58M	180M	93M	160M	71M
test1	default	Create uservisits	201506051552_2641	2015/06/24-02:31	6	4K	20	13	1500	770	6G	3G	25M	10M	35M	17M	33M	4M	46M	17M	31M	10M
test1	default	sorter	201506051552_2671	2015/06/24-03:06	3	3K	18	14	1100	750	7G	5G	150M	74M	430M	190M	110M	23M	38M	19M	160M	71M
test2	default	sorter	201506051552_2564	2015/06/24-00:49	4	3K	21	13	990	750	6G	4G	140M	79M	220M	120M	100M	16M	35M	17M	120M	74M
test1	default	TeraSort	201506051552_2583	2015/06/24-01:06	4	3K	22	15	1500	730	6G	4G	140M	68M	260M	120M	170M	51M	130M	67M	110M	52M
test2	default	sorter	201506051552_2656	2015/06/24-02:50	3	3K	20	13	1200	720	7G	4G	140M	69M	220M	110M	120M	18M	35M	17M	140M	63M

Below is another image from the Pepperdata Dashboard that shows CPU utilization for a day, a week and a month, sorted from highest to lowest on percent of utilization. This is a handy visualization to determine capacity utilization of the cluster. Pepperdata has similar views for memory and disk I/O utilization.



HOW PEPPERDATA'S DYNAMIC CAPACITY MANAGEMENT BENEFITS MULTI-TENANCY

For multi-tenant environments, over time, more and different parts of the organization will adopt the Hadoop central service. In fact, that is one of the goals of a central service - to make it easier for new groups to use the service. So over time, you may get to the point where the cluster is busy most of the time. One organization, to deal with all of the traffic and avoid contention with production jobs, rather than buy additional hardware to deal with the demand, would only allow ad-hoc queries to be run on weekends! By switching on Pepperdata's Dynamic Capacity Management, clusters see anywhere from 30-50% and in some cases as high as 70% increase in throughput. By turning on this feature, it's as if 30-50% more hardware was added to the cluster. And it means you are getting the maximum value out of your infrastructure because Pepperdata makes your Hadoop cluster run more efficiently.

CONCLUSION

Hadoop has quickly evolved from a single-workload, data processing platform into the foundation of a comprehensive information repository serving a wide variety of user communities and applications, data sets and business processes. Pepperdata is making it easier for enterprises to manage and monitor multi-tenancy Hadoop environments and to realize the full potential of Hadoop infrastructure investment.