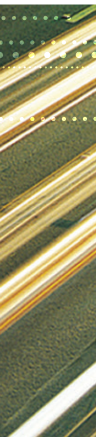
PEPPERDATA ON A HIGHLY-TUNED HADOOP CLUSTER

technical whitepaper

June 2015



pepperdata

PEPPERDATA OVERVIEW

Pepperdata installs in less than 30 minutes on your existing Hadoop cluster without any modifications to the scheduler, workflow, or jobs. It monitors all facets of Hadoop performance, including CPU, memory, disk I/O, and network by user, job, and task, in real time. It dynamically adjusts cluster utilization based on your policies and priorities so that your jobs run faster, more reliably, and more efficiently.

Pepperdata's software provides several key benefits:

- **Spend less time diagnosing and fixing problems.** Pepperdata gives you a both a macro-level and granular view of everything that's happening across the cluster by monitoring the use of CPU, memory, disk I/O, and network for every job and task, by user or group in real time. This detailed telemetry data is captured second by second, and is saved so that you can analyze performance variations and anomalies over time.
- **Implement service-level policies for Hadoop so you can guarantee on-time completion of high-priority jobs.** Pepperdata senses contention for CPU, memory, disk I/O, and network at run time and will automatically slow down low-priority tasks when needed to ensure that high-priority SLAs are maintained.
- **Increase cluster throughput by 30-50%. In many cases, jobs will run much faster.** Pepperdata knows the true hardware resource capacity of your cluster and dynamically allows more tasks to run on servers that have free resources at any given moment.

Pepperdata enables reliable multi-tenancy on your cluster so high-priority production jobs, ad-hoc jobs, and HBase can co-exist without risk of job failures or missed SLAs.

ISN'T HADOOP TUNING ENOUGH?

There is a lot of pride of ownership among the operators of production Hadoop deployments who have spent much time dealing with myriad challenges and continually tuning the Hadoop environment. So their initial reaction to our claim that Pepperdata can increase their cluster's throughput by 30-50% is justifiably skeptical. Many of Pepperdata's customers are very sophisticated operators of Hadoop, with clusters that have been running for many years and are highly tuned using techniques from blogs, message boards, FAQs, and best-practices conference talks. So the question is, how is this increase in capacity achieved? What exactly is Pepperdata doing that makes this possible?

The key idea behind Pepperdata's capacity increase is that the Pepperdata supervisor and agents running on each node are constantly monitoring and controlling the actual use of (and demand for) each kind of hardware resource by each task in real time. That allows Pepperdata to identify "holes" in the cluster where a node could temporarily do more work, then fill those holes with additional tasks, while ensuring that the cluster continues to operate safely and reliably.

In contrast, Hadoop's configurations only affect up-front resource reservations, so standard Hadoop must assume the worst-case theoretical maximum resource usage by every task in order to avoid over-committing. Also, Hadoop (whether Hadoop v1 or YARN) does not monitor or control the usage of disk I/O and network at all; jobs that saturate disks or network can interfere with other jobs and slow them down significantly.

Hadoop Tuning Best Practices

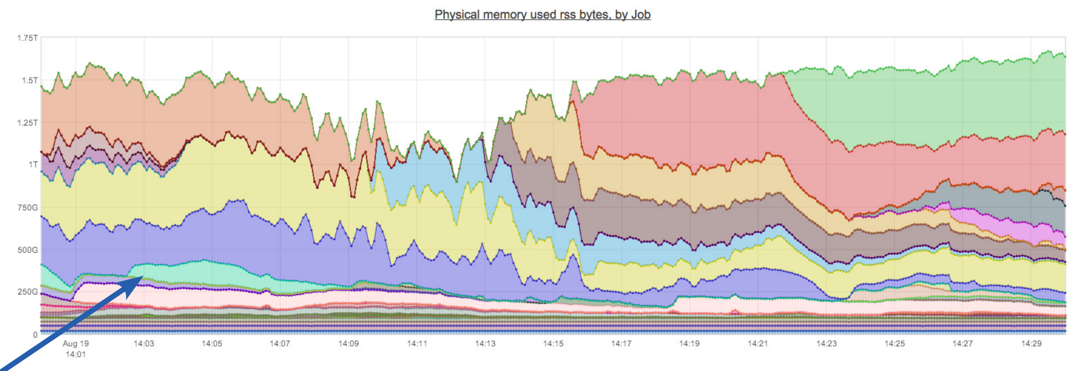
Anyone who has been working with Hadoop for any period of time in production has pulled out all the stops in tuning the environment. There are many recommendations for how to use the tuning options standard Hadoop provides. Here's just a sample.

- Choose the optimal number of mappers and reducers per node and oversubscribed the CPU by 20-30% (8 cores can generally handle 10 slots).
- Adjust memory allocation using the formula Total Memory = Map Slots + Reduce Slots + TT + DN + Other Services + OS.
- Increase buffers for sorting and shuffling, and compressed intermediate data.
- For each job, set up an appropriate number of reducers by checking the job stats to figure out how many are needed, and refactor your mappers to output as little data as possible.
- Adjust the configuration settings for the use of memory and parameters to minimize disk spilling.
- Further tune reducer buffer sizes, mapper sort record percent, and various other settings.

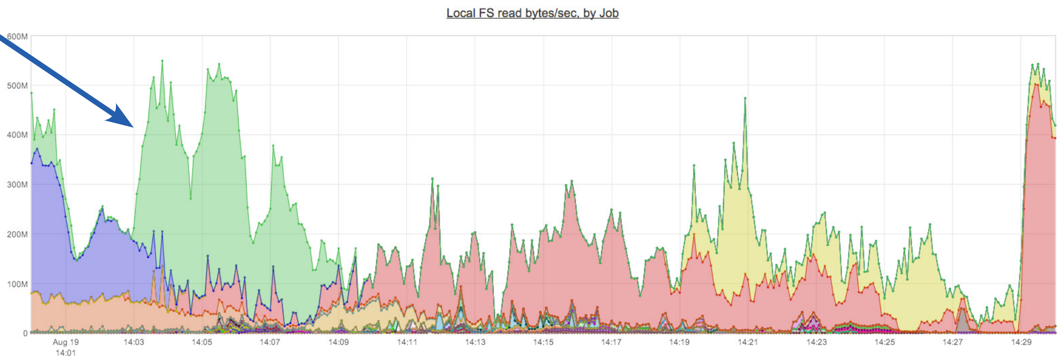
What does that mean? No matter how much you optimize individual jobs and tune your cluster using Hadoop parameters, Hadoop just won't let you achieve the kind of throughput you can get by using Pepperdata's real-time cluster optimizer.

HARDWARE USAGE OVER TIME

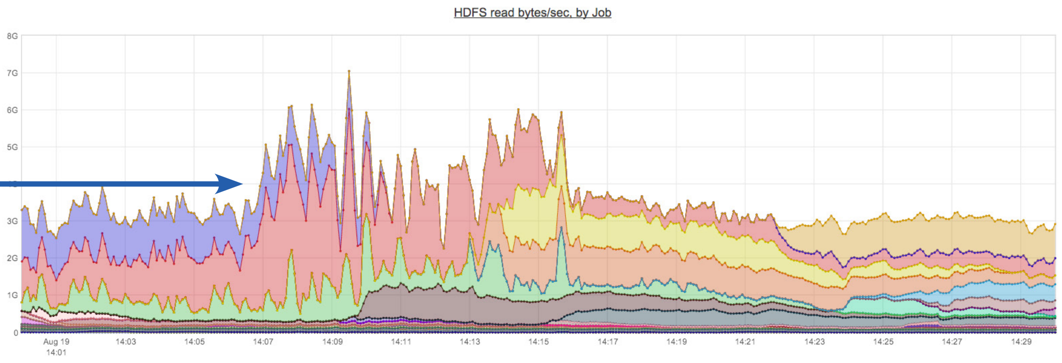
To illustrate why up-front resource allocation and static tuning cannot maximize throughput while ensuring predictable job run times, these graphs show a sample production cluster's usage of a few resources for a recent period. Note the spikes in local file system and HDFS usage, which do not correspond to the number of tasks running (slots used) or memory. These graphs show data that is only available using Pepperdata's dashboard; no other Hadoop management or monitoring tools show metrics like these with this level of granularity in time.



This job (in green) that uses a relatively small amount of memory is the largest user of the local file system, with changing usage over time.



These spikes in HDFS usage last just a few seconds, within the liftime of a long-running job.



Even when the cluster might appear fully utilized, there are always some "holes" in the resource usage that Pepperdata's real-time optimization can fill with extra work in order to increase the effective throughput of the cluster.