Natjam: Prioritizing Production Jobs in the Cloud (Extended Abstract)

Brian Cho*, *Muntasir Rahman*†, Indranil Gupta†, *Cristina Abad*†**, *Tej Chajed*†, Nathan Roberts**, *Philbert Lin*†
†University of Illinois (Urbana-Champaign), *Samsung Inc., **Yahoo! Inc.
{bcho2, mrahman2, indy, cabad, tchajed, prlin2}@illinois.edu, nroberts@yahoo-inc.com
(Note: Italicized author names are students.)

Abstract
We present Natjam, a system allowing high priority production jobs and low priority research jobs to co-exist on an Apache Hadoop cluster. Natjam's contributions include cheap task checkpointing and eviction policies for jobs and tasks. Trace-driven experiments reveal Natjam outperforms Hadoop and is very close to the ideal performance.

Extended Abstract
Today, computation engines like Apache Hadoop are used to process both batch jobs ("research jobs") as well as time-sensitive jobs ("production jobs"). For instance, logs of click data are continuously sent to a Hadoop cluster. A production job runs every five minutes to count the number of clicks per advertisement – if the results are dated it can lead to revenue loss. On the other hand, a research job might run on the same dataset overnight, e.g., for discovering better ways of placing ads.

Unfortunately, many organizations today provision physically separate production and research clusters, and tightly restrict the workloads that are allowed on the production clusters. This has the disadvantages of both inefficient resource utilization and long job run times at times of overload.

The Natjam project provides strict priority between queues in a single cluster, allowing production and research jobs to co-exist, and for production jobs to be able to preempt research jobs and tasks. Natjam is an extension of the Hadoop YARN scheduler [1]. Today, in YARN, if the cluster is under 100% utilization when a production job arrives, the latter cannot be scheduled right away.

A combined cluster with an effective strict queue priority scheme is more efficient: (a) research jobs can fill in areas of under-utilization, running as low-priority jobs that yield resources to incoming high-priority production jobs, and (b) combining the capital investment of separate clusters into a single cluster can raise the capacity, increasing the total available resources for research jobs, and raising available resources for production jobs.

At the heart of Natjam is a low-overhead suspend/resume mechanism for Hadoop tasks. Hadoop tasks belonging to research jobs (especially Reduces) can be checkpointed cheaply midway through, and swapped out so that resources are freed up for production jobs. Natjam involves changes to the Hadoop Capacity Scheduler for preemption, and to the Application Master to save and use suspended task states. The Node Manager is unchanged.

Natjam also innovates in policies for job eviction and task eviction. Concretely, when a production job arrives, one or more of the research jobs' tasks may need to be preempted if the cluster is full. First, a victim job is selected, and then within it, one or more victim tasks for preemption. Eviction policies need to be sensitive to: i) time remaining, and ii) resources required, for the job and/or the task. These are important respectively since the job tail (longest task) determines the job completion time, and since an early finishing task early frees up resources earlier.

We incorporated Natjam into Hadoop 0.23. Our experiments were performed on the Cloud Computing Testbed (CCT) at Illinois. The workloads were both synthetically generated as well as based on real Yahoo! Inc Hadoop workload traces.

As shown in Figure 1, Natjam outperforms both mechanisms for preemption that exist in Hadoop (the Capacity scheduler), as well as the strategy of killing tasks without checkpoints. Compared to the ideal (optimal) outcome, Natjam is only 2% slower for research jobs and 7% slower for production jobs.

We plan to demo Natjam at NSDI 2013.

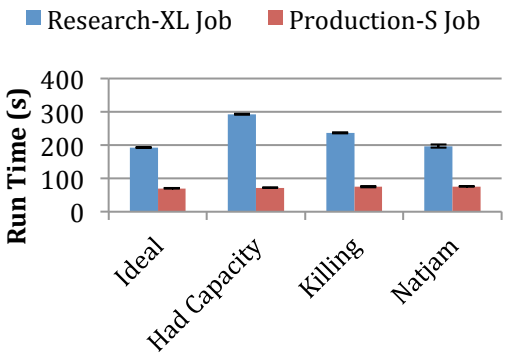

Fig 1: Hadoop Preemption vs. Natjam on a 7-node cluster.

References
[1] Apache Hadoop NextGen MapReduce (YARN). http://hadoop.apache.org/
This work is funded by AFRL/AFOSR grant FA8750-11-2-0084.