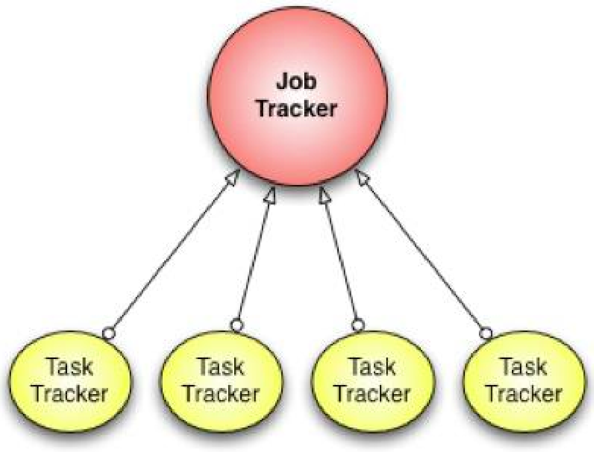
Under the Hood: Scheduling MapReduce jobs more efficiently with Corona

November 8, 2012 at 11:59am

Nearly every team at Facebook depends on our custom-built data infrastructure for warehousing and analytics, with roughly 1,000 people across the company – technical and non-technical – using these technologies every day. Over half a petabyte of new data arrives in the warehouse every 24 hours, and ad-hoc queries, data pipelines, and custom MapReduce jobs process this raw data around the clock to generate more meaningful features and aggregations.

Given our unique scalability challenges (our largest cluster has more 100 PB of data) and processing needs (we crunch more than 60,000 Hive queries a day), our data infrastructure team has to ensure that our systems are prepared to handle not just today's challenges, but tomorrow's as well. Our data warehouse has grown by 2500x in the past four years, and we expect it to continue growing with Facebook's expanding user base and the ongoing addition of new features to the site.

Limitations in the Hadoop MapReduce scheduling framework



We initially employed the MapReduce implementation from Apache Hadoop as the foundation of this infrastructure, and that served us well for several years. But by early 2011, we started reaching the limits of that system.

Specifically, we began to see issues with the scheduling framework, which consists of a job tracker and many task trackers (one for each worker machine). Task trackers are responsible for running the tasks that the job tracker assigns them. The job tracker has two primary responsibilities: 1) managing the cluster resources and 2) scheduling all user jobs. As the cluster size and the number of jobs at Facebook grew, the scalability limitations of this design became clear. The job tracker could not handle its dual responsibilities adequately. At peak load, cluster utilization would drop precipitously due to scheduling overhead.

Another limitation of the Hadoop MapReduce framework was its pull-based scheduling model. Task trackers provide a heartbeat status to the job tracker in order to get tasks to run. Since the heartbeat is periodic, there is always a pre-defined delay when scheduling tasks for any job. For small jobs this delay was problematic.

Hadoop MapReduce is also constrained by its static slot-based resource management model. Rather than using a true resource management system, a MapReduce cluster is divided into a fixed number of map and reduce slots based on a static configuration – so slots are wasted anytime the cluster workload does not fit the static configuration.

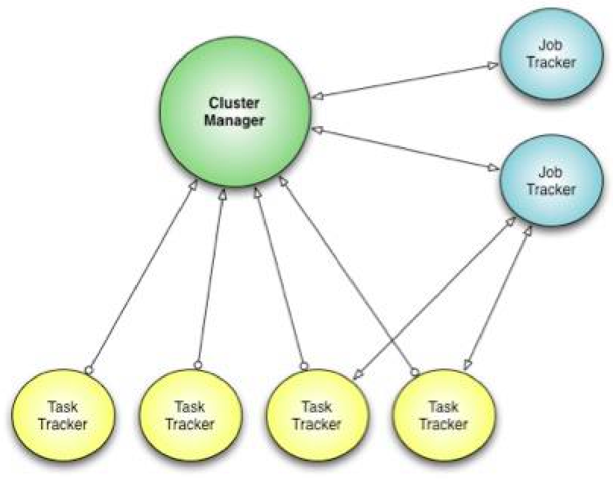
Furthermore, the slot-based model makes it hard for non-MapReduce applications to be scheduled appropriately.

Finally, the original job tracker design required hard downtime (all running jobs are killed) during a software upgrade, which meant that every software upgrade resulted in significant wasted computation.

It was pretty clear that we would ultimately need a better scheduling framework that would improve this situation in the following ways:

- Better scalability and cluster utilization
- Lower latency for small jobs
- Ability to upgrade without disruption
- Scheduling based on actual task resource requirements rather than a count of map and reduce tasks

Facebook's solution: Corona



To address these issues, we developed Corona, a new scheduling framework that separates cluster resource management from job coordination.[1] Corona introduces a cluster manager whose only purpose is to track the nodes in the cluster and the amount of free resources. A dedicated job tracker is created for each job, and can run either in the same process as the client (for small jobs) or as a separate process in the cluster (for large jobs). One major difference from our previous Hadoop MapReduce implementation is that Corona uses push-based, rather than pull-based, scheduling. After the cluster manager receives resource requests from the job tracker, it pushes the resource grants back to the job tracker. Also, once the job tracker gets resource grants, it creates tasks and then pushes these tasks to the task trackers for running. There is no periodic heartbeat involved in this scheduling, so the scheduling latency is minimized.

The cluster manager does not perform any tracking or monitoring of the job's progress, as that responsibility is left to the individual job trackers. This separation of duties allows the cluster manager to focus on making fast scheduling decisions. Job trackers now track only one job each and have less code complexity. This simple design decision allows Corona to manage a lot more jobs and achieve better cluster utilization.

The cluster manager also has a new implementation of fair-share scheduling. This scheduler is able to provide better fairness guarantees because it has access to the full snapshot of the cluster and jobs when making scheduling decisions. It also provides better support for multi-tenant usage by providing the ability to group the scheduler pools into pool groups. A pool group can be assigned to a team that can then in turn manage the pools within its pool group. The pool group concept gives every team fine-grained control over their assigned resource allocation.

Deployment challenges
Since Corona employs a completely different scheduling mechanism, rolling out the new system in production was a significant challenge. We needed to make it work at Facebook scale, iteratively test it, and roll it out without affecting any of the users of the system.

The main driver in our deployment strategy was the fact that the bulk of the workload consisted of Hive queries, and those queries could be attributed to different teams at Facebook. Given this, we split the nodes in the cluster into two systems - one running the old MapReduce job tracker, and one running Corona.

The plan was to move the workload on a team-by-team basis from the old MapReduce cluster into the Corona cluster. We created a Hive hook to send queries to one of the two clusters based on which team was issuing the query.

We staged the deployment in three phases:

1) Rollout to 500 nodes
Our first step was to deploy Corona on 500 of the machines in the cluster. This let us get feedback from early adopters.

2) Handle all non-critical workloads
Next, we started moving the non-critical workloads for each team to the Corona cluster, along with their compute capacity. This let us monitor the system performance with increasing load. When the cluster had 1,000 nodes, we saw our first Facebook-scale problem - the cluster manager scheduler had a bug that slowed it down. We were able to make the fix without much disruption because of the staged deployment.

3) Corona takes over all MapReduce jobs
The final step was to move the mission-critical workloads to Corona as well; eventually the old MapReduce cluster was reduced to 60 nodes. At this point we removed the Hive hook and made the new cluster the default for all workloads.

By mid-2012, we had successfully deployed Corona across all our production systems. The entire process took about three months.

Benefits of Corona
Corona has allowed us to achieve our initial goals of greater scalability, lower latency, no-downtime upgrades, and better resource management. It has also helped us achieve better scheduling fairness, faster job restartability, a cleaner codebase, and the ability to integrate with other systems for scheduling.

We measured some of these improvements with several key metrics:

Average time to refill slot
This metric gauges the amount of time a map or reduce slot remains idle on a task tracker. Given that the scalability of the scheduler is a bottleneck, we saw great improvement with Corona when compared with a similar period for MapReduce. During the given period, MapReduce took around 66 seconds to refill a slot, while Corona took around 55 seconds (an improvement of approximately 17%). In a benchmark run in our simulation cluster, the slot refill times dropped from 10 seconds with MapReduce to 600 milliseconds with Corona.

Cluster utilization
Cluster utilization has improved along with the refill-slot metric. With lower latencies for task scheduling, we were expecting to get a better utilization of the slots on the task trackers, and this has proven to be the case. This metric was difficult to compare fairly in production, so we ran some experiments on a simulation cluster. In heavy workloads during our testing, the utilization in the Hadoop MapReduce system topped out at 70%. Corona was able to reach more than 95%.

Scheduling fairness
Resource scheduling fairness is important since it ensures that each of the pools actually gets the resources it expects. The old Hadoop MapReduce system was often unfair in its allocation, and we saw a dramatic improvement with the new system. We observed that unfairness dropped from an average of 14.3% in Hadoop MapReduce to only 3.6% with Corona.

Job latency
Average job latencies were expected to go down in Corona. The best metric we have for

measuring this is a latency job that we run every four minutes, and so far we've seen that the latency of this test job has been cut in half (now 25 seconds, down from 50 seconds).

The Future of Corona
While Corona has already been a huge success at Facebook, we're still working to improve it.

We currently have some new features in production testing, including (a) resource-based scheduling that schedules tasks based on CPU, memory, disk, etc. needs rather than the slot-based model; and (b) online upgrades to the cluster manager and task trackers. We're particularly excited about the new scheduling, which we expect will improve our cluster resource utilization and also allow us to run high-memory jobs that were not possible in our previous slot-based system.

Additionally, while Corona can run MapReduce jobs with ease, we are looking to expand our user base by scheduling other types of applications such as Peregrine. Integrating applications with Corona gives them a slew of benefits, including scalability, pool-based scheduling for different classes of jobs, different users, resource guarantees, different types of schedulers within the pool, scheduling statistics for the system, etc.

In summary, Corona has become an integral part of Facebook's data infrastructure and helps power big data analytics for many teams across the company. We are continuing to improve it and are very excited about launching the upcoming features that will enable it to meet the ever-growing needs of our teams for years to come. We have also open-sourced the version we currently have running in production; please check it out on the Facebook hadoop-20 repository on GitHub and let us know what you think!

A number of people on the Facebook data infrastructure team contributed to the development of Corona, including the authors of this post: Avery Ching, Ravi Murthy, Dmytro Molkov, Ramkumar Vadali, and Paul Yang.

[1] It's worth noting that we considered Apache YARN as a possible alternative to Corona. However, after investigating the use of YARN on top of our version of HDFS (a strong requirement due to our many petabytes of archived data) we found numerous incompatibilities that would be time-prohibitive and risky to fix. Also, it is unknown when YARN would be ready to work at Facebook-scale workloads.

Like · Comment · Share

Diogo Rosa, Justin Lee, Sobhan Kahn and 1,135 others like this.

295 shares

View previous comments

Walid Shaari @Ignacio Vivona i would debate the single responsibility principle in the light of overall architecture. why the cluster manager is part of Hadoop MapReduce, why not a single stand alone entity. a resource can be number of core, free memory size, free ... See More
November 10, 2012 at 1:16am · 3

Gaurav Kaul Can you please mention how are you defining slots? Is slot = 1 core, 1 CPU or 1 compute node?
November 10, 2012 at 2:53am · 1

Walid Shaari @Gaurav Kaul a slot is usually one job per node, to simplify scheduling into one resource argument instead of several, so if a 12GB 4cores nodes can take 4 jobs, 4slots. 24 GB 6 core nodes could be 7 slots. that is my saying, not sure how Facebook is doing it, they are saying they are moving away from static labeling to a dynamic one
November 10, 2012 at 7:01am · 2

Premjith Rayaroth There are different implementation for priority queues. For example, let us assume there are 4 priority queues

-Regular Priority... See More
November 10, 2012 at 7:11am · 1

Omar M. Abou Elnil I've got the idea of applying I applied through you
Is the Chat application generally or to facilitate the writing of those who write more than one language
Need your consent... See More
November 10, 2012 at 10:25am · 1

Emylle Farias http://www.facebook.com/pages/One-Direction-1D/514528018566073
November 10, 2012 at 10:47am · 1

Brian Bowling Props to you for making this available to the community. Nice work!
November 10, 2012 at 11:32am · 1

Jay Arrgh I wonder how this compares to YARN in Hadoop-0.23 which breaks up the JobTracker -and- NameNode into multiple daemons that can be distributed in an HA manner.
November 10, 2012 at 7:35pm · 2

Soroush Zohari im civil engineer in iran . in esfahan steel complex 00989133285206
November 11, 2012 at 7:18pm · 2

Stoney Vintson I read an article that says you tried using Yarn, but did you try using the Apache Mesos scheduler? You get better cluster utilization. You would be able to use other compute frameworks than MR, such as UC Berkeley Spark 0.6, Shark 0.2 (Spark 0.6 + Hive 0.9), and possibly CMU GraphLab 2.1 via the existing MPI scheduler. http://incubator.apache.org/mesos/ Twitter is running a 1500 node cluster on Mesos.
November 11, 2012 at 9:35pm · 3

Luis Miguel Silva Great post, thanks for sharing your experiences!
November 12, 2012 at 10:36am

James Geraghty great work fb-engs
November 12, 2012 at 3:22pm

Dror Mizrachi Great post, great work FB
November 12, 2012 at 9:49pm

Sunny Drake Despite i half understand logic in this text and have no exp with clusters, not looked into code i don't see actual work on follows : Did you optimize Data access layouts dynamic/static for your tasks? What type of priority rules/algoritms did you use/... See More
November 13, 2012 at 10:03am

Ervin Peretz Here's a super-simple way to learn MapReduce, and scale later (with none of Hadoop's strange reducer behavior) : http://jsmapreduce.com/
November 13, 2012 at 10:52am

Ervin Peretz ... and great work FB. Anything to improve Hadoop is much appreciated, especially startup latency.
November 13, 2012 at 11:27am

Fatos da Vida ◄~~Curta minha pagina ae galera ~~►https://www.facebook.com/.../Fatos-da.../272849252836568◄~~ Obrigado a toodos que curtiiu.
See Translation
November 13, 2012 at 11:28am

Andy Augustine Great article, thanks for sharing.
November 14, 2012 at 10:35am

Leo Kimminau Seems inconsistent: " During the given period, MapReduce took around 66 seconds to refill a slot, while Corona took around 55 seconds (an improvement of approximately 17%). In a benchmark run in our simulation cluster, the slot refill times dropped from 10 seconds with MapReduce to 600 milliseconds with Corona."
November 15, 2012 at 6:20am · 2

Steven Bernard Perete this kind of stuff excitesme
November 15, 2012 at 10:17am

Aravind S Murthy gr8 article
November 15, 2012 at 9:11pm

Michael Tripi Awesome article...kept my interest from top to bottom. Facebook sounds like the train to be riding. Caliber of technical staff must be amazing! Would love to be there with those creative minds...
November 16, 2012 at 7:57am

Arvind Arya Good to know about future directions of map-reduce by chasing YARN or Corona..
November 16, 2012 at 10:01am

Sherif Hamdy Very good site for engineers
- Technology
- Community... See More
November 17, 2012 at 10:10am · 1

Paritosh Sharma A welcome contribution in Big data Analytics by FB
November 19, 2012 at 12:07am

Pallavi Reddy http://www.facebook.com/.../andhraloverscom/223190681038458
Hi friends Click this link, like me and share..
November 19, 2012 at 12:16am

Radu Murzea The font of these articles is annoyingly small. Fix that... Anyway, sounds like Corona is a success, congratulations. Hopefully, some of the improvements will be included in Hadoop .... if you let them
November 19, 2012 at 1:14am

松崎功保 いいね!
See Translation
November 19, 2012 at 7:03pm

John Schllegell Link for my page check it out and hit like
https://www.facebook.com/EverythingMadeOfStone?ref=hl
December 2, 2012 at 8:03am

William Payne Hmm .. task scheduling, resource utilization - this sounds almost like an OS for compute clusters. I know I sound like an idiot suggesting this ... but do you think, one day, this will all live in the Linux kernel?
December 10, 2012 at 2:13am · 1

Guillermo Iguaran Suarez Just curious, have you evaluated Storm?
February 16, 2013 at 5:14pm

Paul Forester Interesting approach to the limitations, great share!
April 2, 2013 at 11:12am

Vitaly Grinberg The integration of MROrchestrator with NGM might be better
June 5, 2013 at 5:36am

Radoff Gasfinti Xavier "it is unknown when YARN would be ready to work at Facebook-scale workloads." <- damning
April 11, 2014 at 12:54am

Atul Patil anyone give me algorithms of schedulers in map reduce...
September 8, 2014 at 10:52pm

Navid Daaria Golnaz
February 26 at 4:47pm