BristolHadoopWorkshop

Bristol Hadoop Workshop, University of Bristol, August 10, 2009

This was a little local workshop put together by Simon Metson of Bristol University, and Steve Loughran of HP, to get some of the local Hadoop users in a room and talk about our ongoing work.

Acknowledgements

1. Bristol Centre for Nanoscience and Quantum Information NSQI for the room and other facilities
2. University of Bristol Particle Physics group for hosting the workshop
3. HP Laboratories for the food and coffee
4. Cloudera for supplying beer at the Highbury Vaults.

Presentation

These presentations were intended to start discussion and thought

- Hadoop Futures (Tom White, Cloudera)
- Hadoop and High-Energy Physics (Simon Metson, Bristol University)
- HDFS (Johan Oskarsson, Last.fm)
- Graphs Paolo Castagna, HP
- Long Haul Hadoop (Steve Loughran, HP)

Hadoop Futures

- Hadoop Futures (Tom White, Cloudera)

Tom's goals for Hadoop

- make it modular
- support more languages than just Java
- better integration with management tools

Schedulers

CapacityScheduler. Yahoo!'s -designed for very large clusters with different people working on it. Can take RAM requirements into account and place work machines with free RAM space, rather than just a free "slot"

FairScheduler -Facebook's. For a datacentre running production work with latency requirements, some people also running Hive jobs which are lower priority.

Languages

- streaming: stdin and stdout, text or typed binaries; slow
- pipes: C++ interface
- HDFS is pure Java. FUSE is slow because of this

Security

This is going to take lots of work. Its really hard to get security right.

Scaling Down

- standalone doesn't have >1 reducer.
- MiniMR will run multicore, but you have the overhead of the full RPC protocol, even though everything is running in a single process.
- Ideal: a multicore-ready single client.

Someone needs to make the local job runner better. It's been neglected because all big projects don't use it. To make Hadoop useful for small amounts of data, single machine work, the standalone runner needs work.

Pig in local mode doesnt use local job runner => need to take what they've done.

Project split

New list structure

- ${project}-dev: every issue when created, other discussion
- ${project}-issues: every JIRA update
- ${project}-user: user discussions. This is a bit confused now, there are so many of these.

- hadoop-general - worth getting on this list too

0.21 release

- Any 0.21 features must go in in this month!
- MAPREDUCE-207 Computing splits on the cluster: reduces effort on the client
- HADOOP-6165 Avro and Thrift
- Context Objects - new API, finished for 0.21
- new shuffle : read the Y! paper on sortbenchmark.org

Hadoop 1.0 goals

The goal for 1.0 is to have some things stable for a year: API, wire format (Avro). Some things will be marked "unstable, developer only" to avoid guaranteeing fixing things.

- HADOOP-5073 -interface classification
- HADOOP-5071 -wire protocol

Paolo wants a hadoop-client POM that pulls in only the dependencies for the client. Similarly, a hadoop-local that only pulls in stuff for local things.

Eclipse Plugin

The Eclipse plugin is not in sync with eclipse. Nobody is supporting/using it right now. With a stable API/wire format it would work better. (of course, with a stable long-haul API, the plugin could always work with a remote cluster, even if you had to be careful that the jobs you ran were compiled against a compatible version.)

Benchmarking Hadoop

- Benchmarking Hadoop (Steve Loughran & Julio Guijarro, HP)

Terasort, while a good way of regression testing performance across Hadoop versions, isn't ideal for assessing which hardware is best for other algorithms than sort, because things that are more iterative and CPU/memory hungry may not behave as expected on a cluster which has good IO, but not enough RAM for their algorithm.

In the discussion, though, it became clear that a common need people have that isn't that well address right now ~~and for which terasort is the best that people have to date~~ is QA-ing a new cluster.

Here you have new hardware ~~any of which failing is an immediate replacement call to the vendor~~ on a new network ~~which may not be configured right~~ and with a new set of configuration parameters -all of which may be wrong or at least suboptimal. You need something to run on the cluster which tests every node, makes sure it can see every other node's services, and report problems in meaningful summaries. The work should test CPU, FPU and RAM too, just to make sure they are all valid, and at the end of the run, generate some test numbers that can be compared to a spreadsheet-calculated estimate of performance and throughput.

When you bring up a cluster, even if every service has been asked to see if it is healthy, they still have the problem of talking to everything. The best check: push work through the system. Wait for things to fail, try and guess the problem. Having work to push through that is designed to stress the system's interconnected ~~and whose failure can be diagnosed with ease~~ would be nice.

That is, for all those people asking for a HappyHadoop JSP page, it isn't enough. A cluster may cope with some of the workers going down, but it is not actually functional unless every node that is up can talk to every other node that is up, that nothing is coming up listening on IPv6, that the TaskTracker hasn't decided to only run on localhost, etc. etc.

Long-Haul Hadoop

- Long Haul Hadoop (Steve Loughran, HP)

This talk discussed the notion of a long-haul interface to Hadoop.

This is a recurrent theme in various bug reports -anywhere where people want to submit jobs from a distance and keep an eye on them. Often this need surfaces in a request for some kind of remote API to the Job Tracker.

This talk discussed the fact that remote Job Tracker API is not sufficient. Being able to submit one job is nice, but if you are chaining things together, you need to submit a sequence of operations, and the logic to chain it together. You may also want things like notifications of progress/failure. What you have is a workflow.

Workflows could be handled at the low level with something that can run a sequence of MR jobs, and any Java classes which implement the Tool interface. The Tool would be run in the datacentre, in some medium-availability host, so you could switch your laptop off and know that the program was still running.

There is work underway at Yahoo! with Oozie, a workflow system for Hadoop; Cascading and Pig Latin are also languages to describe sequences of operations, so again, you need to run them. In the talk, SmartFrog was used as a constraint-language for ordering work, which shows that the language set is even broader. But there is a key point here: the model of a workflow engine is the same, regardless of the language. You create workflows, you schedule them with parameters and options, you await results.

The long-haul model for workflow can be the same for multiple back ends.

The talk looked at the options for long-haul-ness, of which there are two

1. WS-* : the big, comfortable, safe long-haul option, the Airbus A380. You, the passenger, get looked after by the cabin crew.
2. The floatplane. Agile, can get around fast, but you read the location of the life vest instructions very carefully, make a note of the exit in the roof and hope that you aren't the one who has to get on the float to dock the plane with the boat. It isn't quite as comfy as the big plane, but it is easier to get up and around with it.

Two RESTful world views were discussed

- A pure REST: PUT/DELETE model of workflow objects, in which even their queue state is manipulated using the full REST model. This is clean, ideal for clients such as Restlet, and HTML5 browsers.
- An HTTP Post model, in which work is POSTed to a queue server, URLs returned; operations to the queued workflows via POST or PUT, GET for state updates.

Steve gave a partial demonstration of Mombasa, his prototype "long-haul route to the elephants". This consists of:

- A RESTy interface built from JAX-RS, hosted as the Jersey runtime under Jetty, deployed in-datacentre by SmartFrog
- A Portlet GUI to the same set of operations, this time running in-datacentre in a portlet server. (Which may be liferay-on-tomcat, but does not need to be). It is implicitly implementing the HTTP Post model.

Currently the portlet is not using the long-haul API itself, though there is no reason why it should not, in which case it will not only drive the API, it will test it.

Other Portlets will apparently provide cluster management by talking to the relevant "cloud" APIs: Add/decommission nodes, view logs, etc, and simple HDFS file access.

Long-haul filesystem access is another issue. Ideally, WebDAV would be good, as there are so many clients and it is a pure REST API. But parts of the WebDAV spec are odd (same FS semantics as Win98/FAT), and you can be sure of interop grief. Amazon S3 is simpler, as long as you avoid their daft authentication mechanism.

Discussion: Simon mentioned that they had a REST API to some of the CERN job submission services, and later sent out a link. There was general agreement that you need to push out more than just MR jobs

Hadoop and High-Energy Physics

- Hadoop and High-Energy Physics (Simon Metson, Bristol University)

The CMS experiment is on the Large Hadron Collider; it will run for 20-30 years colliding heavy ions, such as lead ions. Every collision is an event; 1MB of data. Over a year, you are looking at 10+PB of data. Right now, as the LHC isn't live, everything is simulation data, which helps debug the dataflow and the code, but reduces the stress. Most events are unexciting, you may need to run through a few hundred million events to find a handful that are relevant.

Jobs get sent to specific cluster round the world where the data exists. It is the "move work to data" across datacentres, but once in place, there isn't so much locality. The Grid protocols are used to place work, but a lot of the underlying grid stuff isn't appropriate; written with a vision that doesn't match the needs. Specifically, while the schedulers are great at work placement on specific machine types, meeting hardware and software requirements (Hadoop doesn't do any of that), you can't ask for time on the MPI-enabled bit of the infrastructure, as the grid placement treats every machine as standalone; doesn't care about interconnectivity.

New concept: "dark data" - data kept on somebody's laptop. This makes up the secret heavy weight of the data sets. When you think that its laptop data that is the enemy of corporate security and AV teams, its apt everyone. In CMS, people like to have their own sample data on their site, they are possessive about it. This is probably because the LHC isn't running, and the data rate isn't overloading everyone. When the beam goes live, you will be grateful for storage and processing anywhere.

A lot of the physicists who worked on the LEP predecessor are used to storing everything on a hard disk. The data rates render this viewpoint obsolete.

In the LHC-era clusters, there is a big problem of disk, tape, CPU balance. For example, multicore doesn't help as the memory footprint is such that multicore doesn't benefit that much unless you have 32/64 GB. It also means that job setup/teardown costs are steep. You don't want to work an event at a time, you want to run through a few thousand. The events end up being stored in 2GB files for this reason.

The code is all FORTRAN coded in C++.

This was a really interesting talk that simon should give at apachecon. Physicists may be used to discussing the event rate of a high-flux-density hadron beam, but for the rest of us, it makes a change from web server logs.

Data flow

- LHC -> Tier 0, in Geneva
- Tier 0 records everything to tape, pushes it out to the tier 1 sites round the world. In the UK, Rutherford Appleton Labs is the tier 1 site.
- Tier 1 do some computation as well as storage -you can "skim" the data on on tier one, quick reject of dull stuff -and can share the results. This is effectively a reduce.
- Tier 2 sites do most of the computation; they have their own storage and are scattered round the world in various institutions. In the US, the designs are fairly homogeneous, in EU, less so.

The architecture of the LHC pipeline is done more for national/organisation politics than for efficient processing. The physicists don't get billed for network traffic.

Staff issues: lots of spare time cluster managers. People are the SPOFs of the CERN tooling. In the long term, they may consolidate onto one or two UK sites.

File Systems

- Castor: CERN, includes tapes, team turnover/staffing bad
- Dcache: fermilab. works there
- DPM: doesn't scale beyond 10TB of RAID.
- GPFS -Bristol. Taking a while to work as promised.

Hadoop Integration

HDFS has proven v. successful at Tier-2 sites; popularity may increase as centres expand. Appreciated features: checksumming, admin tools. Validate that the data is OK.

Could you run CMSSW under Hadoop? Probably not. Very slow startup/teardown cost, so you don't want to just run it for one/two events.

Issue: How to convince physicists to embrace MR? Need to see the benefits, as physicists don't see/care about the the costs.

Graphs

- Graphs Paolo Castagna, HP

This was a talk by Paolo Castagna on graph work under MR, of which PageRank is classic application

- graph topology does not change every iteration, so why ship it around every MR?
- the graph defines the other jobs you need to communicate with.

The graph is a massive data structure which, if you are doing inference work, only grows in relationships. Steve thinks: You may need some graph model which is shared across servers, which they can all add to. There is a small problem here: keeping the information current for 4000 servers, but what if you don't have to, what if you treat updates to the graph as lazy facts to propagate round?

Google: pregel. what do you need from a language to describe PageRank in 15 lines?

MS: dryad does not do graphs.

Projects

- Apache Hamburg -proposed by Edward Yoon, of Hamas, is it making progress? We need code.
- Thread based prototype code of Hamburg
- Apache Common Graph. Dead, pre-MapReduce code.

Graph Algorithms

These are what a graph project should start with

- Graph Search
- Directed/acyclic graphs
- Minimum Spanning Tree
- Shortest Path
- Network Flow

Paolo claimed that Depth First Search, DFS, doesn't work in this space. if Depth First Search doesn't work, what to do?

The current best printed work on Graph over MapReduce is the recent paper Graph Twiddling in a MapReduce World, by Jonathan Cohen.

- You can't do transitive closure in SQL.
- What would an efficient transitive closure algorithm over MR be?

Discussion

There was discussion on handing big graphs in the system, ones where the graph itself is very large. Someone need's to take Paolo's PageRank-over-MapReduce code and test it on bigger data sets.

There was a good point on what is "efficient" in this world.

Yes, something done as a chain of MR jobs on a Hadoop cluster may seem an inefficient approach, but if there is no other way to store that much data, or run through it, then graph people will be happy.

Yahoo! MS search deal

This was a discussion topic run by Julio

- 400 Y! staff are moving to MS. How many are search specialists, versus Hadoop hackers.
- Y! is driving large scale tests, facebook is #2.
- Y! are making Hadoop the core of the company; it is their LOB of datacentre.

What are the risks of the Merger, and warning signs of trouble:

1. silence: Y! developers do their own fork, it goes closed source. We have seen this happen in other OSS projects (Axis), where a single company suddenly disappears. There is no defence from this other than making sure development knowledge is widespread. The JIRA-based discussion /documentation is good here, as it preserves all knowledge, and makes decisions in the open.
2. staff departure. Key staff in the Hadoop team could leave, which would set things back. Moving into MS could be bad, but moving to Google would set back development the worst.
3. slower development/rate of feature addition
4. reduced release rate. This can compensate for reduced testing resources.
5. reduced rate of bug fixes. We can assume that Y!s own problems will be addressed, then everything else is other people's problems.
6. Less testing, reduced quality

Apparently under contributors] - number of messages/JIRA and infer activity, such as [http://community.cloudera.com/reports/47/contributors/ and popular issues

With Yahoo! outsourcing searching to MS, it means that MS can take on a big project that -even if it isn't profitable to MS, can be subsidised by other parts of their business. It ensures Yahoo! continuing survival as an independent company, which is the best state for Hadoop development. It also frees up some Yahoo! datacentres for other projects. Those big datacentres are large, slowly-depreciating assets, and by offloading the indexing to someone else, there is now spare datacentre capacity for Yahoo! to use for other uses, uses that are highly likely to use Hadoop at the back end -because what else would they build a datacentre-scale application on?

At the same time, there are opportunities for people outside Yahoo!

- more agile deployments
- more open to contributions from other people, universities etc.

Of course, this could impact release schedule/quality; needs to be managed well. Clearly for Cloudera, this gives them a greater opportunity to position themselves as "the owners of Hadoop", especially if they get more of the core Hadoop people on board. However, Apache do try to add their own management layer to stop handing off full ownership of a project to outside companies. But the reality is whoever provides the engineering effort owns the project, so any organisation that can provide FTEs can dominate.

What are the increased responsibilities for everyone else involved with Hadoop?

- Everyone has to test on larger cluster. EC2 may get tested, but it's not enough as it is virtual, and only represents one single site/network config.
- Everyone should pull down and play with the pre-releases, on test clusters. Check the FS upgrades work, etc.