DeveloperOffsite20090612

On the Friday after the Hadoop Summit 2009 a group of Hadoop committers and developers met at Cloudera's office in Burlingame to talk about Hadoop development challenges. Here are some notes and pictures (see the attachments) from the discussions that we had.

- Attendees
- Five things
- Project challenges
- Wish Lists
    - MapReduce
    - MapReduce and HDFS
    - HDFS
    - Build And Test
    - Avro
    - Common
    - Pig
- Subproject discussions
    - MapReduce
    - HDFS
    - Pig/Hive
    - Avro/Common
        - Configuration
        - Avro
    - Build And Test

Attendees

Eric Baldeschwieler, Dhruba Borthakur, Doug Cutting, Nigel Daley, Alan Gates, Jeff Hammerbacher, Russell Jurney, Jim Kellerman, Aaron Kimball, Mahadev Konar, Todd Lipcon, Alex Loddengaard, Matt Massie, Arun Murthy, Owen O'Malley, Johan Oskarsson, Dmitriy Ryaboy, Joydeep Sen Sarma, Dan Templeton, Ashish Thusoo, Craig Weisenfluh, Tom White, Matei Zaharia, Philip Zeiliger

Five things

Things we don't like about Hadoop

- config hell
- unit tests need improvement
    - too few unit tests vs functional tests
- Hudson, JIRA, patch process woes
    - too slow (hudson)
    - too much mail (JIRA)
- Hard to debug / profile
- Confusing / arcane APIs
    - JobInProgress
    - Dcache
- Docs could use improvement
    - new functionality lacks specs, test plan
    - NN SPOF

Things we like about Hadoop

- community
- open source
- ecosystem
- good components: HDFS, ZK

Project challenges

Nigel brings up:
Patch Development Process:

- pretty convoluted
- attach patch to JIRA, Hudson, committer takes it
- Releaseaudit and other output unreadable
- Running ant hudson-test-patch takes five hours, usually fails
- Compiling is slow too

... for testing we need to know that the tests are sufficient – this means that docs are required

Test plan template: What are you testing? what are the risks? How do you test this?

Phil: What are examples of *good* JIRAs?

Doug: Should we add a test plan field to JIRA?

Nigel: People need to take ownership of features and consider scalability, etc.

The patch queue is too long...

Doug: committers need to be better here

Dhruba: doing reviews right is hard

Phil: What do people do here?

- read patch by itself
- apply and use a diff viewer
- Can we use a review tool – e.g., FishEye, Reviewboard? Phil will take point on this

Doug: We need a "wall of shame" to incentivise people to review other peoples' code rather than just write new patches of their own

Pushing Warnings to Zero:

- Running test-patch is slow
- "rat" has verbose output, needs an overrides system
    - nobody really volunteering to fix this
- checkstyle has 1200+ warnings =-- this is absurd
- Need ECliupse settings file that reflects reality
- Patch system disincentivizes "janatorial work" to lines near what you're really working on

Typo Fixes, etc:

- Can we commit things without a JIRA?
- Can we commit things without a review?

Most things require both

Exceptions: www site, rolling a release

How do you continuously refactor?

Phil : Can we agree that it's ok to fix comments/typos/etc in the same file you're working in?

Doug Yes, but people need to say that this is what they're doing in the JIRA comments

Tom: This should be clarified on the HowToCommit wiki page

Checkstyle:

- Eclipse style file (Phil)
- NetBeans style file (Paul)
    - both of these should be in top level and included via svn-extenrals
- Tune checkstyle config (Tom)
- Wholesale codebase sweep
    - Need to clear patch queue first?
- Per-project Core (Owen) / MR / HDFS – who will take care of the last two here?
- Enable in patch testing (Nigel)

Matt: Could we think about an auto-reformatting patch service? (lukewarm reaction here)

Testing:

- We have 10 new machines which can be used soon
- Need "ten minute test" so that individuals can run this
    - Some MR folks are removing MiniMR from tests where it's not needed
- Phil: We need a build sheriff to nag on build/test breaks
- Nigel: need continuous build on all active branches on commit
    - not just patch-isolated tests
    - need code coverage (and diffs of this)
- Functional tests are not first-class citizens
    - need a harness for running shell scripts, etc.
    - maybe run on EC2? Alex will do some work here
- Phil will work on improving docs related to test utils.
- We should start using some mock framework
    - JMock, EasyMock, etc
- LocalJobRunner needs improvement
- TestNG allows for shared MiniMRCluster which will help
    - This should do things like *not* start Jetty, etc.

Build System:

- Ivy vs. Mason
  - Nigel is working with someone to build a POM repo
  - Should we convert to mvn?
  - How bad is Ivy itself vs. just our usage of Ivy?
- Pig and Hive both do something different for depending on Hadoop
- Forrest needs to go.
  - Todd will investigat migrating soimewhere else
  - There are other XML-based documentation systems

Wish Lists

MapReduce

- JT API (e.g., REST)
  - pluggable job submission API
  - InputSplit generation as a task
- Simplify JT/TT interaction
- MR 2.0:
  - Separate persistent rsrc manager from the per-job manager
- Avro input format
  - For Pipes, too
- Preemption and better priorities
- JobInProgress refactoring
- JSON Job History
- Pipeline MR (M*R*)
- Streaming in mapred, not contrib

MapReduce and HDFS

- Separate client/server (interface/implementation) jars, packages

HDFS

- Append and 4379 (flush/sync)
- NN metadata inspection tool (already exists in some form in 20)
- Multi-NN for same DNs (federated NN)
- Multi-DC Hadoop
- HDFS mounts + symlinks
- pluggable block placement
- DFSClient requery NN if it can't find a block

Build And Test

- One framework for system testing (loca,l and in the cloud)
- Standard build and dep managemtn framework
- Failure injection
  - Based on Aspect/J
- Back compatibility and verificatino framework
  - Need ability to run big sets of real jobs without apache necessarily owning those jobs
  - A messaging issue to the community is that non-committers can and should independently test and then vote +1/-1 on release candidates
- Symlink test suite

Avro

- language-agnostic RPC (C, C++, Ruby, Python...)
- column store format in Avro
  - or put in pig/contrib?
- Standard text format (currently only has a binary fmt)
  - JSON based
- HTTP RPC
- RPC proxy for testing

Common

- config variable names, uniform schema

- better errors for bad config
- range testing for config values
- LDAP-based config
- Config should support lists of elements, and not just a big string "value" wiuth commas
- remove non-public things from public API

Pig

- decent error messages
- column store format in Avro

Subproject discussions

MapReduce

HDFS

- Append/Sync : It is being targeted for 0.21 release. Project status "yellow".... meaning that it could be "at-risk" for this release.
- HADOOP-4379 : seems to satisfy the append-tests that the Hbase team is doing. The Hbase team is requesting that this jira be included in some sort of a Hadoop release.
- HDFS mounts and symlinks : HADOOP-4044. This is very much essential for federating a flock of hadoop clusters. This is targeted for 0.21 release.
- NameNode scaling to large number of files: One idea that was discussed was to pagein/pageout metedata from the namenode memory on demand. This introduces aditional complexity in the NN code, especially needs fine-grain locking of data structures. This is not going to be attempted in the short term. Federating the hadoop namespace across multiple hadoop namenode (using symlinks) would be a way to solve the problem of a "large number of files".
- DFSClient read latency performance: It was discovered that the performance bottleneck was because of a single thread doing sequential reads from one data block followed by the next. An alternative approach would be to open multiple DFSClient instances, and make each instance read different blocks in parallel. This can improve the latency of a single file-read tremendously. Todd Lipcon will try this out.

Pig/Hive

Approaches to column storage were discussed. The Hive and Pig implementations are fundamentally different – Hive uses a PAX-like block-columnar approach, wherein a block is organized in such a way that all values of the same column are stored next to each other and compressed, with a header that indicates where the different columns are stored inside the block. This file format, RCStore, can be used by Pig by virtue of writing a custom slicer/loader. The Pig representatives indicated that they were interested in experimenting with RCStore as an option for their workloads.

The Pig model splits columns into separate files, and inxeses into them; there is a storage layer that needs to coordinate the separate files. The layer is (naturally) able to do projection pushdown, looking into also pushing down filters. Pig integration is in progress. No technical impediments to implementing a Hive SerDe for Zebra.

Sharing code and ideas would be easier if there was a common place to look for this sort of stuff. For example, RCStore can be pulled out of hive – but where? Commons? Avro?

Neither project has plans for storage-layer vector operations (a-la vertica).

Metadata discussion pushed off to next week.

Join support is roughly equivalent for both systems (map-side aka FRJoin, and regular hash join). Neither supporting or planning on bloom joins (no real demand – be it because the users aren't familiar with it, or because the workloads don't need it, is unknown). Either way, "we accept patches 😉"

Blue-sky stuff:
Oozie integration – a way to define a hive node? Some other custom processing node? Having Hive and Pig clients automatically push queries off as Oozie jobs so that they are restartable, etc?

Pig is looking to move off JavaCC. Currently leaning in the direction of cup+jflex ; hive uses Antlr, which is now a lot less memory-intensive as they've started using lookaheads. This information might be putting Antlr back on the table for Pig.

Avro/Common

Configuration

Registry class for configuration that allows you to "register" configuration

- Allows for easier documentation and more control
- Allows for warning for unknown keys to help with typos
- Give support for description of units and range testings
- Do before 1.0
- Push the value validation into the registered class
  (e.g. range tests, units, allows for <elem></elem>, umask in octal (HADOOP-3847)
- Allows for explicit tagging/filtering of "stable", "experimental" and "deprecated"

Hadoop configs should be read from a distributed filesystem (HADOOP-5670)

- Read from LDAP, ZK, HTTP

Avro

- C/C++ Support
- Single RPC Mechanism across all Hadoop processes/daemons
- Language Agnostic RPC w/ versioning and security
- RPC Proxy Object (e.g. fault-injection, light-weight embedding)
- To/From Import/Export JSON

Build And Test

Top Level components Nigel proposed:

- Backwards Compatibility Testing
- System Testing Framework
- Patch Testing Improvements
- Test Plan Template (for JIRA)
- Mock Objects

Backwards Compatibility Testing:

- API (syntactic)
  - Static Analysis
- API (semantic)
  - We need scrutiny during code review when compatibility breaks; ways to determine if compatibility is broken during CR:
    - Be strict with the JIRA incompatible change flag
    - jdiff changes
    - Javadoc changes
    - Test changes (e.g., a JUnit test changes)
  - Other ideas:
    - Community-driven testing
      - Either allow users to submit a workflow and test their workflow for them, or
      - Just have users submit to a website if compatibility is good for them
        - Think of a long checklist, where each item is a volunteer running Hadoop that says if compatibility is good or not
    - Write API tests around 1 or 2 key APIs
      - Get well-defined specs for these APIs
      - Test all assertions
    - Sub-project test suites can help a lot
      - Solr, Pig, Hive, Mahout, HBase, Chukwa
      - Sub-projects can be part of the community idea mentioned above
- Protocol
  - Not 0.21, because of Avro
- Config
  - Syntax-static analysis (diff), also semantic tests apply
- Data

System Test Framework:

- Shell framework (perhaps a derivative of Y!'s)
- Run on cluster (possibly AWS) or in local debug mode
- Can run system and performance tests

Patch Testing:

- RAT - need more control
- Code coverage diff (really interesting)
- Better test detection
  - +0 if they have a test/ folder
  - +1 if they have test/**/Test*.java
  - +1 if they have a new @test or "test*(" in a test/ file
  - -1 otherwise
- Other tools (JDepend, Classycle)
- Sonar (would run alongside Hudson)

Test Plan Template:

- New features have the template in the JIRA
- Nigel has done some work on this

Mock Objects:

- (nothing discussed)