People:
- Lars H from salesforce
- Chris Trezzo from SFDC
- Eugene Koontz from Trendmicro
- Francis from Yahoo
- Richard from NExtBio (prev Yahoo)
- Ted from eBay
- Alex from oPower
- Ryan (doesn't work?)

FB:
- Karthik from FB
- Jerry C from FB
- Nicolas
- Mikhail

- Roman from Cloudera
- Abhijit (? did I hear right?) from Rocketfuel
- Thomas, Swati, Carson from eBay
- Jimmy, Todd, Jon H from Cloudera
- Stack
- JGray himself
- JD from SU

Suggested topics:

T**esting**

- Stack brings up that it's hanging some builds on the Apache server. Lars suggests this might be fixed after we fixed SecureRandom to use urandom

NKeywal's test categorization

- Nicolas Keywal in Paris is breaking up our tests into small/medium/large categories. One last commit to go in, then docs for developers about how to run different categories
- Integrated with surefire - you can ask it to run different sizes. Look at his mails to dev@ for more info.
- "mvn test" would run only small/medium: should do this before you put up a patch
- CI server will run all tests

- Ted questions whether Jesse's work on integration tests is orthogonal or not with regard to Nicolas's work.

- Jesse and Nicolas to work this out
 - jyates: there was a thread on dev@ about how we want to structure tests (pushed out by Ted). Something to discuss in general about the state of testing (wanting levels of unit tests, and then also integration and acceptance tests)
- Some concern about developers knowing which tests to run when, having different methods to run the tests, etc.
- Roman brings up that “integration test” may also mean inter-component - eg Pig or Hive with HBase. Which is different than an HBase large test which integrates different parts of HBase functionality. He points out Pig’s *pig-smoke* artifact as something we could model, which makes it easy for BigTop to hook up to those kind of tests adn run them either against mini cluster or against a real deployed cluster.

- Todd suggests we have an online discussion with Roman, Ted, Jesse, Nicolas, Stack to figure out what we want to do here: (added to Action Items at bottom of this doc)

Roman: bigtop perspective

Bigtop is helpful in some key areas:
- Validate HBase against a particular release of the other components. EG Hadoop 0.22, or particular set of Hive and Pig. (Bigtop provides way to construct a full stack of software to deploy up to a couple hundred node cluster, and a rudimentary environment for integration tests against that stack)
- Good place to put test scenarios that cross projects: eg Pig-HBase integration, or Oozie workflows that submit Hive queries that query HBase. Bigtop’s reason for being is this type of testing.

Requests to HBase community:
- Would be really nice if there were a separation as mentioned above: two categories of tests:
1) Tests that inspect internal state of daemon w/ Minicluster APIs
2) Tests that do NOT inspect state of running cluster - can run against a deployed cluster OR a minicluster

He doesn’t care whether they’re called (large, integration, or what,) but just that it be somehow easy to pull them out and run them within bigtop.

Mikhail: is it possible to test an arbitrary revision?
Roman: yes, you can put at a tarball, or a particular git sha. won’t take much work to make it work with SVN, but everyone hates SVN so doesn’t do it yet. They’re currently doing this to test against Hadoop-0.23 tip of branch against 92 tip of branch.

Mikhail: is deployment config fixed? does it have to go to /usr/local/hbase or something?
Roman: made conscious decision to separate deployment from the tests themselves. Tests never concern themselves with deployment. This is how they can reuse test artifacts like

pig-smoke which doesn't know anything about deployment. They currently use puppet for deployment, but happy to look at patches for other ways of deployment. On jenkins today they deploy the stacks via puppet on ec2.
Mikhail: but is it light-weight enough for developer testing. eg make a quick code change and want to test some version which isn't committed anywhere at all.
Roman: not quite for just substituting a random jar file with some other jarfile. But should be flexible enough for substituting a different HBase tarball

Mikhail: integration test framework

High level ideas: now's a good time to develop a vision on what integration tests should be (on a small/5 node cluster)
- People on the FB team have 5-node clusters where they can run individual load tests.
- Has some slides

- Motivation: testing is time consuming for FB devs
- Agility vs stability

- Workflow for devs at FB:
-- create an issue in JIRA
-- develop on 0.89-fb internal branch
-- run unit tests
-- run 5-node cluster load test with correctness testing
-- commit internally
-- dark launch/shadow cluster, which runs a live FB messages workload. necessary to be confident that changes are good for production
-- create an open source patch for trunk / run unit tests, dig into unit test failures, run small cluster test
-- commit to public 0.89 branch and public trunk

- How much testing is necessary for changes?
-- recent work on test categorization suggests that for some changes you don't need to run all tests, but they think usually it's best to run all tests
-- deal with false positives (flaky tests)
-- level 3 testing: deploy on small cluster w/ read-write workload, verify correctness
-- level 4: deploy and do feature specific testing
-- level 5: automate all of the above

- Agility vs stability?
-- How stable does trnk need to be? Stable enough for other people to work off of. Stable enough for early-stage projects to run off of in dev.

- Release candidate branches: should be under continuous testing w/ a realistic workload
- Production branches should be very stable obviously

Design proposal (no implementation done yet)
- wants to collect feedback on this. See slides for diagram.

“mrunit” (not the same as the ASF project) - runs the unit tests on an MR cluster.
- run all the tests in 10 minutes or so, but dominated by long-tail tests liek TestAdmin that run for 8-10 minutes

Deployment:
- want to use bigtop for this
- basically need to copy files to ~5 node cluster, in a configurable target directory, etc

Load tester itself:
- PerformanceEvaluation tool: doesn’t do any correctness checks
- HBaseTest - an fb internal test tool which has yet to be opensourced (single client read/write workload)
- Christopher Gist’s distr load test tool (soon to be OSS)
- Need to exercise multiple workloads: single row, multiple row, multi column, multi CF, etc
- Need to test through thrift server or other gateways as well

Report:
- need to accumulate all this info into a single report, ideally
- cover the whole range from unit tests through large/integration tests, to real cluster tests. Not talking about multi-product tests - just testing hbase itself.
- other projects might be useful in terms of generating workloads, but the goal isn’t to test that integration
- should also sanity check log to grep for errors, etc, in automated way

Conclude:
- need to reduce false positive unit test failures
- need a benchmark that does correctness testing
- need kill tests for masters
- need software to accumulate into a report

Questions:
Ted: He spends a lot of time working on Jenkins build on Apache machines, but difficult to get access to those machines. Wondered if FB would provide a cluster for this kind of testing.
Mikhail: not sure, would require a lot of approval etc
Stack: suggests EC2 - Amazon might donate credits

Lots of folks: companies willing to run it, but not willing to give out access in their colos, etc.
Mikhail: we should make it easy to run this stuff, then make a central place where we can show results
Roman: bigtop is doing this kind of stuff already - but with full stack. Every time a change hits a repo, the cluster gets re-deployed and these kinds of tests get executed.
Stack: this is priority #1. Everyone says that. But afraid that we'll all go home from meetup and not actually do it. Suggests we have a subgroup huddle on this topic only to figure out action items (Mikhail, Roman, Stack?)

JYates: current state of testing overall (using annotations for tests, integration tests, etc)

What does HBase do (discussion):

Karthik and Stack where chatting, and got into this discussion, wanted to raise: What is HBase? Do you define it as a write-dominated store where writes and reads don't interfere? But increments don't have this property - should we address that? Then we need to start working on bigger heaps, etc? What kind of features should we be working on?

3 years from now, if we're doing everything efficiently, what would HBase look like? From that we can decide what to work on now.

Other question: it's tempting to put custom business logic inside region servers.. especially with coprocessors. But no one would do that with MySQL... (people contest this, bringing up stored procs as an example... some debate how much people do this in oracle, etc.) Distinction is that PLSQL/etc is well sandboxed.

Karthik suggests we might be over-reaching by allowing people to write arbitrary code inside the server. Going to be difficult to understand performance, bugs, scaling, etc, when we have arbitrary user code on the server. Acknowledges this is philosophical.
jesse: We probably should support a core set of CPs that are useful in common practice (aggregation CP is a great example). Use them if you want, not if you don't need them

Jgray: his takes: 1) see his hbase roadmap discussion from hadoop world: it's need-driven. 2) hbase is part of hadoop ecosystem - the things hbase needs to do are the things that other parts of hadoop don't do. eg increments are impossible in other parts, but HBase has to do that.

Karthik: imagine that mysql is more efficient than hbase for increments? should we try to make hbase better than mysql for it?

What are we saying "no" to from user requests? Where do we define our scope?

Someone suggests: Is HBase a solution or a platform? Databases give you triggers - you can be stupid in databases doing heavy duty work in triggers. Doesn't mean its a bad feature - HBase will let you be stupid too.

Jgray agrees: allow people to shoot yourself in the foot if they do something stupid

Alex N: rather than tell people "no" on stuff, figure out what we're good at, and focus on that

Jgray: people will work on what they need to have for their app

Alex N: although people work on what they need, they'll also listen to what the community vision is.

Todd: suggests that coprocessors should be seen as an internal API, not a user feature -- it's just a new way to extend hbase for dev features -- instead of subclassiong HRegionServer. we just need to market it right.

Karthik: maybe we've reached a point of maturity where we need to pick and choose which things we attack -- up to this point it's just been feature parity with BigTable. But now there is more open road ahead. (coprocessors was one example but not his general point)

Stack: Marketing-wise: likes the idea that we're 1) big data store, 2) does simple ops, 3) fast, 4) stable. Yea you can get your fingers in there and add stuff if you're jgray, but that's not the main point. Should we tighten up the website front page?

Doug: I agree that the website front page should be tightened up, and I agree with the above message (effectively, same thing as what Karthik says below). That's what we use it for as well.

Karthik: let's the idea that we have a single goal for the project: super-fast reliable big data store (eg).

Jon H: "What is HBase's Mission?"

People seem to like that definition above.

- **RPC**

- 0.94 (combined w/ discussion on releases)

When is it?
Lars H suggests we branch soon, or as soon as we make the 92 rc. But other 89fb patches are

still coming into trunk - maybe we should wait for that before we branch 94.
Nicolas suggests we might want to wait for snapshot?
Ted: we need clearer distinction between branches? how different are 92 and 94?
People bring up there's a bunch of good perf stuff in 94, for example

Ted: his feeling is that no one has put 92 or 94 in production. 94 isn't so different than 92... if we can't distinguish 94 from 92, then it's too early to branch 94? or just release a 94 rc0 at this point? The trunk build actually seems more stable than the 92 build.

Todd: time based or feature based?

JGray: branching and feature freezing are not the same thing - we can still put features in from 89fb. So long as there aren't massively destabilizing features, we can still put them in.

LarsH: if we cut 94 branch now, we have to put all the new stuff into more branches - more work applying patches.

Ted: we now have 4 builds: (insecure + secure) * (92 + trunk). Insecure build hung for more than 5 days. If we branch now we'd have 6 builds - out of control. One goal: combine insecure and secure builds into 1 before 94 to reduce this.

Stack: suggests we branch now, since it'll come out 3-5 months after 92. We branched 92 6 months ago and it took this long to get to an rc..

Jgray: are we waiting on some big trigger to branch? Wait until we have something big that we _don't_ want in the next release. Something scary we need to push out for later.

Stack: we need to release more often - more than once a year. Stability and testing - a few months of work on that will give us good headway.. asks FB people to shout when all the 89fb stuff has been ported, then maybe a good time for 94? Plus Mikhail's testing stuff?

Ted: time-based release is difficult.. initially planned 92rc0 for 4 weeks ago, then we lost weeks to hadoop-world, then security, then holiday, etc. We never make releases on time.

Todd: if we want to do time-based, we just need to be hard-asses about it.

LarsH: most important thing is more than one a year.

JGray: we don't need time-based, but we do need more than 1 a year

Roman: experience with other projects: people cut branches early, and then start treating trunk as a dumping ground because no one is really working on it -- and then trunk is not releasable. advises branching late.

Stack: to change topic: is 0.94 == 1.0?
JGray: let's just release 94, and if it "becomes" the 1.0, let's call it that when we decide it

JonH: we should wait - some of the features are relatively new, APIs not solid yet. if 94 is basically 92, then it's not 1.0 yet.. another 1 or 2 after that.

Todd: 1.0 == BigTable. what big features would come after?

Ted: Secondary indexes would be a 2.0.?

Conclusions: not going to branch as soon as we thought, until we have some trigger to branch 0.94. Equate 1.0 with something after 94.. but we're pretty much done with new features for time being.

Todd: suggests that even if we released many times a year, people like FB and Cloudera would park themselves on particular versions anyway. Let's just assume that any x.0 release is unstable and bill them as alpha.

Stack: building the testing tools up so we're more confident
Stack: feature branches: to keep trunk more stable

People in general: we have to push destabilizing stuff to branches, and make them politic to get it in if they don't have their own resources to test

Nicolas: we should do min(6 months, feature) -- not just time or feature. We get away from time-based when our project is done enough that no one can think of features to add.

Conlusions: push people into feature branches, push for testing before commit

Operability

Stack: online config/schema change is pretty critical. hbck - keep filling it out for more corner cases. 92 has improvements like slow query log, process list, regions-in-transition
Todd: two sides of a coin between stability and operability
Stack: we'll always have some bug,etc, so we need repair tools and monitoring tools to describe how the thing got hosed.
JGray: we're getting better here,

Nicolas: docs -> need to point people to the HBase book. Once upon a time we had an RSS aggregator. FB has a blog but no one knows about it.
Stack: book is getting better w/ time (thanks to Doug)
JGray: HBase planet? He plans to bring back his hbase blog site.

Nicolas: repair utilities - they have some common repair cases on the 89 master. Would like to look into this on the new master. hbck -fix works 95% of the time for them.

Jon: hbck mostly deals with cases where cluster state is inconsistent. But it doesn't deal with some tricky cases like failed splits - need a mechanism to allow users to make decisions that might lose data? He's working on getting these in.

Nicolas: by 1.0, you should have decent strategy for helping naive users to fix their stuff w/o making them learn how to use the code base. JGray: or look at logs

Todd: suggests a crash report thing that would email anonymized report data to the hbase committers when they run hbck and it isn't happy

- **Hiring/training** (developers, sysadmin)

How do we train operators and developers?

Jon: We need an operators guide? War stories?
Chris: Playbook?

Nicolas: since most of us have core devs on staff, we don't develop tools that are good for non-core-dev operators. We need to think from sysadmin perspective, get other people building tools like that.

Jon: has a slide deck he's prepping for Cloudera supporters. a little deeper and more operational than Lars's book or HBase's book. Will look into sharing it more widely -- covers what's in filesystem, zookeeper, etc, from black-box point of view.

Karthik: has some sysadmins at FB who have contributed some code (Paul, Ryan)

Jerry: "HBase DBA" is the right term.
JGray: MySQL DBAs are focused on making db efficient/optimizing. Hadoop/hbase ops guys do a lot more networking, racking, server ops
Richard: there are different people who do site ops and people who do software part.
JGray: when mysql breaks, it's not because of some network/line card issue with ZK quorum -> client, etc. Not the same kind of problems. But Hadoop ops people deal with this. Different class of problems they deal with on a regular basis.

JGray: ops guys at FB are vertically integrated - bridging gaps from site ops to engineering
Richard: issue is figuring out where the problem is, for his ops people
JonH: we need to improve error messages throughout the system -- errors should always be actionable.
JD: had a guy with an INFO level exception on the local DN - which was a DFS client issue, but someone who didn't know the stack had no idea what was going on. DFS Client errors - we should almost be translating them. They look scary, and we print the whole stack trace.

Stack: skill set is very different from the MySQL dba skillset - different mix of stuff.
Andreas: there are two kinds of devs: app devs and core devs.
Stack: growing pains are on this frontier. No one *wants* to hire core devs, they just *have* to right now.
Stack: wants to put Cloudera out of business. Or make money by charging to run "hbck".
JonH: cloudera people agree.
Doug: I don't want to put Cloudera out of business, but meta corruption issues scare the crap out of me. The better we can ensure table metadata stays consistent, the better off everybody will be.

Nicolas: at FB have two separate roles. (a) actual dist systems problems, (b) multi-tenancy applications. Their DBAs do query analysis - which applications are putting what load on servers? We're still on part A, part B
JGray: FB's hbase eng team has been doing part B.
Nicolas: yea, in 3-5 years we want DBAs doing that, not core developers. We don't want companies to have to hire core devs to run hbase.

Stack: if people are looking to train up new core devs, he's willing to come help people get up to speed.
JGray: cloudera offers a training! now 2 days.

Doug: ops needs detailed statistics on what is happening to which table. I think the CF-metrics that FB added to .92 can help a lot in this regard. The RS-oriented metrics were a good start, but didn't tell you enough about which tables were hot/not.

RPC

Benoit: rolling upgrade is necessary, need wire compatibility especially for minor upgrade.
Stack: also replication across clusters! headache even worse
Replication across major-releases would be nice, so you can do major upgrades by switching DCs

Benoit: problem is writables in RPCs, there's some versioning support but no one really uses it.

Some discussion: how much of a hit are we willing to take for wire compat?
Where's the reasonable middle-ground between perf and wire-compat?

Orthogonal:
- brand new RPC layer?
- other bits, like version numbers not being used everywhere, are just small matters of programming?
- some support for envelopes/metadata - like QoS priorities, dapper-like tracing
- exceptions are kinda borked

Conclusion?
- start with subbing out some methods for protobufs? we can do it method by method, type by type, eventually retire the Writables

Blue sky / discussion

Benoit: everything async?
JGray: push down more stuff to the filesystem?
Benoit: redo all the logging - everything is DEBUG or INFO. Spend more time sorting messages to right levels. People shouldn't run DEBUG, but there's so much useful info there. Think about which ones should really be DEBUG vs INFO.
  JGray: but really we don't want people to *have* to look at logs ever.
  Benoit: but when shit hits the fan, we need more usable information instead of a sea of logging
  Ryan: should basically log major decisions -- know which code path it went through from logs
  Benoit: lots of "WTF" messages at debug or info that should be WARN or ERROR

Benoit: suggests switching to slf4j and use logback which has some nice features like tagging messages and filtering/route messages, keep ring buffers in memory. Thinks this is the de facto standard.
Ryan: thought it wasn't worth it over log4j when he looked at it.

Ryan: deployment automation? service orchestration? chef?
Roman: bigtop wanted to be a central repository for deployment recipes. Uses puppet and has repositories for the recipes. But just does keep-up-service, not orchestration.
Jon: suggests Whirr (Java + shell scripts).
  Ryan: says it doesn't understand errors very well. elaborate shell script

Benoit: availability: waiting 3 seconds is too long, when regions move, etc
Jgray: thinks there's some low-hanging fruit here we just haven't addressed

Doug: (relaying idea from Casey Stella) the ability to mark tables as "read only" (e.g., for generated indexes for slowly changing data). In this mode, there is no need for checking the MemStore during reads because none will be forthcoming.

Doug: I like that idea that JD mentioned at Hadoop World, namely the ability to say which RSs certain tables should live on. That way certain tables could be fenced off from each other on the cluster.

Doug: another blue-sky idea. Support encryption and compression out of the box. One idea that I think people would like combinations (e.g., LZO and encryption, Snappy and encryption), and of course there is a wide variety of encryption options (symmetric? public key?) But having HBase support this would be a nice "Accumulo killer" feature for very sensitive data.

Doug: I apologize if this got added to .92 and I didn't realize it, but what about security per tables? That's another alleged "Accumulo-y" feature (as well as a confirmed RDBMS one).

Working model w/ FB
Jerry: at FB hbase is the only store that has all the private communication at FB. If they lose 10% of their data they all need to find new jobs. It's the permanent stable store. So that's where they come from. But they also would like to stay with the community and upgrade every 18 months or so. How can they work together to be with community but stay stable?

Is Ubuntu a model? A long-term release occasionally?

Ryan: distributors like Cloudera fit this model?
Todd: Linux does this upstream as well - Greg KH (i think?) maintains a long-term-stable in the public

Roman: maybe it's just a goal to get everyone on the same page to pick a branch and treat it that way?

People seem to think 92 will be this branch, but we won't know until after it's released.

Contribs
Stack makes a case to kick out REST/Thrift/Avro.

Phabricator:

FB's internal version of reviewboard which they've open sourced
Available at reviews.facebook.net

Nicer JIRA integration - eg when you upload a patch there it puts it on JIRA.

Command line tool: *arc* - command line automation of the normal tasks. integrates with git - you commit the revision, then "arc diff" and supply a JIRA and it'll upload the patch, etc, autofill. When you do revisions, "arc amend" to upload new diff. Person who commits it to svn can do "arc patch"

HBASE-4896 has the documentation for how to set it up. Wouldn't recommend it for common occasional contributors, but if you do it a lot, worth setting it up, much faster.

Action items:

- testing: have an onlline discussion with Roman, Ted, Jesse, Nicolas, Stack: difference between large test + integration test
- ask Mikhail for copy of his slides to attach here
- Mikhail, Roman, Stack, maybe some others to have a smaller group discussion on testing. Will publicize on list.

Testing mini-summit:

Present: Mikhail, Roman, Stack

The following issues were discussed:

1. Grouping of HBase tests into small, medium and large.
2. Using maven failsafe plugin (mvn verify) for integration/large tests
3. Facebook work on integration tests
4. Facebook's desire to have a simplified workflow for running integration tests against a real cluster.

I'll summarize what I feel was a common ground in those discussions. This, of course, will be mostly from Bigtop perspective (although I do care a great deal about helping HBase community).

First of all, it was agreed that Facebook (and other HBase devs) who don't concerned themselves with anything but HBase/HDFS have a much simpler workflow compared to somebody who's involved

in Bigtop and thus has to care about the entire stack.

To the Bigtop person deploying from packages in EXACTLY the same
way the customers would do is of a paramount importance and we
can't cut corners there. For an HBase dev deploying from jar files
and copying configs around is just fine. This is not to say that HBase
devs wouldn't benefit from using puppet as a common deployment
tools, but rather that for most of them learning puppet, etc. is an
additional investment of time that can't be easily justified.

To that end, for an HBase dev it is very much desirable to have
a workflow where (based on setting properties)

mvn verify

would be flexible to a point of actually deploying a small, fully-distributed
cluster. The best way to implement that is to invest in a plugin that
would use something like Apache Whirr to do the actual deployment
and have Maven failsafe plugin use the deployment plugin for setting
up the env. It is unlikely that Bigtop devs will have time/motiviation
to implement something like that, but it does sound like a useful
project. I also remarked that, given how generic this deployment
plugin sounds, HBase is probably not the right place to host the
implementation, but Bigtop could be. Our repo would welcome a
patch that has that implementation and we can start publishing
this deployment plugin as a Manen artifact for all the downstream
projects to use.

On that same note, using maven falisafe plugin for its environment
setup capabilities would be, in my opinion, THE ONLY thing that
justifies its use over maven surefire plugin. On the other hand,
if somebody wants to spend time hooking it up to HBase -- there's
not harm. It is just that I don't see much value either.

The only three things that Bigtop really cares about are:

1. that tests are easily grouped into a category that does NOT reach
   into the guts of MiniDFS/MiniMR/HBase implementation and the
   rest of them. Basically it is a difference between tests using
   mostly public APIs to talk to the cluster vs. using private ones.

2. tests have knobs that can make it NOT spin off a mini cluster and
   use a real cluster instead (see pigsmoke with its -Dpigunit.exectype.cluster=true)
   as a good example.

3. tests have knobs that can control the size of datasets being operated on (after all a Minicluster can process way more data per sec).

It would be nice if tests started to honor the manifest protocol that we're developing in Bigtop but it is not critical.

Finally, we need tests NOW. In any shape, size or form. Something is better than nothing. If Facebook guys can give me all the jar files that they use for testing internally (better yet a source code for them). I can start running those tests against HBase in Bigtop ASAP.
We shouldn't be waiting for a perfect implementation of

mvn verify

We should be testing as much of HBase functionality in any given moment as possible.