2019-11 Meeting on SolrCloud and project health

Date: November 18th, 11am PST

Agenda / Discussion Topics

The agenda is largely based around themes that Mark Miller raised recently. Mark Miller said a lot of things and was not very coherent either. He gave some WIP code to his colleagues and did some virtual meetings with them and with me (David). In this meeting, we will attempt to distill the major ideas, both technical and community/process.

- Technical
  - Big picture: “Solr Cloud 2” isn’t a rewrite; it’s fixing a 100 little things that are greater than the sum of their parts. Exception to this is pervasive use of Curator and large work in the Overseer.
  - Surprising insight: Seek performance and you may achieve stability – a more important goal
    - Our instability is difficult to diagnose because hard to reproduce problems
    - Focusing on performance tightens the timings and exposes real/existing problems that used to occur rarely
    - Me: Implies we have benchmarks regularly running, similar to Lucene nightly benchmarks
    - Another outcome may be code clarity / reasoning?
  - More use of Apache Curator
    - Not using Curator is tech debt. Less code for us to maintain; simpler code remains. Curator is faster & safer than our feeble attempts, generally.
    - Lets get familiar with it
  - Overseer
    - TBD if Mark left notes on what becomes of it.
  - Logging
    - Need continuous attention to cleanup. Sometimes we don't log enough, sometimes too much.
    - Separate log configs depending on what work you are focusing on.
    - Colored logs.
  - Tech-Debt
    - We don't finish efforts. Finishing means completely removing the old stuff.
    - Hurts the community; fewer contributors due to our complexity

- Community
  - Code reviews. Do we change policy?
  - Should we adopt a formal decision process for proposing major changes/APIs to Solr, aka “Solr Implementation Proposal (SIP)”?
    - See proposal from Jörn Franke in dev@ referring to KIP and FLIP
  - Documentation
    - Code level, e.g. Javadocs
    - High level. new Dev guide somewhere (different from the Solr Ref Guide!)
    - See [Experimental] Getting Started with Solr Development

Meeting Notes

Attended by: Andrzej Bialecki, Anshum Gupta, Cassandra Targett, Chris Hostetter, David Smiley, Erick Erickson, Gus Heck, Ishan Chattopadhyaya, Jan Hoydahl, Jason Gerlofiski, Mike D. (Apple), Noble Paul, Shawn Heisey, Scott Blum, Tomas F. Lobbe, Yonik Seeley

Duration: 95 minutes.

Mark’s WIP Code

We heard from Thomas, Anshum, and Mike D. (Mark Miller’s colleagues.) who spoke with Mark at length and have his work in progress.

- Mark shared a large code dump with his colleagues.
- He’s not comfortable sharing it in this state as it’s too work-in-progress. His colleagues are going to respect this and not simply share it as-is.
- His colleagues are actively working together on teasing out separate improvements that will each get their own JIRA issue and code. This is hard work and it will occur rather slowly over time (probably more than a month). And when each issue is filed with code, it will usually be WIP. Rarely will it be something immediately committable.

Apache Curator

Migrating to use Curator is a great thing for many reasons (see agenda) including performance, though it’s not a singular solution for any/most SolrCloud problems. Probably no draw-backs but it’s work. Changing this (or many SolrCloud internals for that matter) cause tests to break (Mark said), and it’ll take time to fix such tests.

Overseer

- It “clings to leadership” much more than it should.
- SolrCloud over-uses the Overseer for too many functions that could be done without it. We’ll probably always want an Overseer though.
  - Noble’s work on Smart State Caching (SOLR-13951) will help.
  - Sometimes writing directly to ZooKeeper (helped with Curator recipes) is sufficient.

Service Protection
Solr doesn’t have much service protection. If you create thousands of collections, it’ll lock up and become inoperable. Scott reported that if you boot up a 100+ node cluster, SolrCloud won’t get to a happy state; currently you need to start them gradually. A well-written service won’t lock-up; it will make the client wait and/or give the client an error. The autoscaling framework is supposed to help; it’s a start and AB is working on that somewhat. It’s probably not the only answer here.

**Benchmarks**

Ishan is working on addressing the need for continuously running benchmarks [SOLR-13933]. Having such benchmarks is rather foundational for the theme of performance improvements. And that, perhaps surprisingly, helps us achieve stability.

**Tests**

Mark believes in tightly limiting test times that shouldn’t take long. He used this while working on his improvements. Smiley suspects this approach may only be useful in local dev but not in C.I. where virtual overloaded machines could be quite slow. Furthermore, he believes the objective there can be addressed better via benchmarks.

**Logging**

Much was said but unclear what action to take here; it’s a bike-shed topic. Separate concerns depending on the audience -- production users or us developers? Hoss reminded us of the LogLevel annotation and suggested it’d be neat if the level could be automatically set to debug based on the package of the test.

**Tech Debt**

The overarching theme of what Mark raised is perhaps tech debt. Some miscellaneous things to add here: We should spend more effort removing old things (Smiley cares about this). And for lots of functionality to continue to maintain, we hope the plugin system would lead to a future where Solr needn’t absorb everything or needn’t be official contribs.

**Code Reviews**

We want to get reviews, even extremely superficial reviews that might not look at the code but do look at the description and comments said about the state of the code. Apparently ASF “RTC” policy suggests 3 binding votes are required, which is of course an extremely high bar and not palatable. Even without formally changing, we’d like to try out a 6 month period of behaving this way for all but the most trivial of changes. Smiley takes an action item to make a specific proposal soon.

**Major Change Proposals**

See Kafka “KIP” as an example. This interests us but it’s very unfamiliar to us. We want to try it out. Perhaps SOLR-13951 might be worth experimenting with this. Perhaps a Confluence page is the right place to put the text? Issue argued Google Docs is more collaborative, e.g. inline commenting. Hoss argued Confluence has this to, but might need to be enabled. Today, without a major change proposal mechanism, some JIRA issues are onerous to decipher. Irrespective of this, Hoss advocated we continuously update our JIRA issue descriptions to be useful during the course of the issue, especially at conclusion.

**Documentation**

We agreed we need several layers of docs: Javadocs, Developer Guide, User Guide. Javadocs is clearly in the code and we want more of them! Developer Guide is currently unknown whether we prefer Confluence or use asciidoc/markdown in a dedicated directory in our code repo.

**Closing Remarks**

We really liked meeting to discuss these matters. Gus and Jan proposed doing this quarterly timed to occur near when the ASF board reports are due so that we can discuss anything to add.

**Action items**

- Mark's colleagues to introduce Mark's code piece by piece into new JIRA issues over time
- Ishan to introduce a periodic benchmark system
- Noble to try out a “Solr Improvement Proposal” or some-such in a new initiative pertaining to ZK/clusterstate matters.
- David to propose a code review proposal to discuss on the dev list
- David to organize the next meeting near March 1st (before ASF board report being due that month)