Gmail **Ishan Chattopadhyaya <ichattopadhyaya@gmail.com>**

First class support for node roles

84 messages

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Wed, Oct 27, 2021 at 12:46 PM
To: dev@solr.apache.org

Hi,

Here's an SIP for introducing the concept of node roles:
https://issues.apache.org/jira/browse/SOLR-15694
https://cwiki.apache.org/confluence/display/SOLR/SIP-15+Node+roles

We also wish to add first class support for Query nodes that are used to process user queries by forwarding to data nodes, merging/aggregating them and presenting to users. This concept exists as first class citizens in most other search engines. This is a chance for Solr to catch up.
https://issues.apache.org/jira/browse/SOLR-15715

Regards,
Ishan / Noble / Hitesh

Gus Heck <gus.heck@gmail.com> Wed, Oct 27, 2021 at 6:31 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

The SIP looks like a good start, and I was already thinking of something very similar to this as a follow on to my attempts to split the uber filter (SolrDispatchFilter) into servlets such that roles determine what servlets are deployed, but I would like to recommend that the roles be all positive ("Can do this") and nodes with no role at all are ineligible for all activities. (just like standard role permissioning systems). This will make it much more familiar and easy to think about. Therefore there would be no need for a role such as !data which I presume was meant to mean "no data on this node"... rather just don't give the "data" role to the node.

Additional node roles I think should exist:

I think that we should expand/edit your list of roles to be

- QUERY - accepts and analyzes queries up to the point of actually consulting the lucene index (useful if you have a very heavy analysis phase)
- UPDATE - accepts update requests, and performs update functionality prior to and including DistributedUpdateProcessorFactory (useful if you have a very heavy analysis phase)
- ADMIN - accepts admin/management commands
- UI - hosts an admin ui
- ZOOKEEPER - hosts embedded zookeeper
- OVERSEER - performs overseer related functionality (though IIRC there's a proposal to eliminate overseer that might eliminate this)
- DATA - nodes where there is a lucene index and matching against the analyzed results of a query may be conducted to generate a response, also performs update steps that come after DistributedUpdateProcesserFactory

I also suggest that these roles each have a node in zookeeper listing the current member nodes (as child nodes) so that code that wants to find a node with an appropriate role does not need to scan the list of all nodes parsing something to discover which nodes apply and also does not have to parse json to do it. I think this will be particularly key for zookeeper nodes which might be 3 out of 100 or more nodes. Similar to how we track live nodes. I think we should have a nodes.json too that tracks what roles a node is ALLOWED to take (as opposed to which roles it currently servicing)

So running code consults the zookeeper role list of nodes, and any code seeking to transition a node (an admin operation with much lower performance requirements) consults the json data in the nodes.json node, parses it, finds the node in question and checks what it's eligible for (this will correspond to which servlets/apps have been loaded).

I know of a case that would benefit from having separate Query/Update nodes that handle a heavy analysis process which would be deployed to a number of CPU heavy boxes (which might add more in prep for bulk indexing, and remove them when bulk was done), data could then be hosted on cheaper nodes....

Also maybe think about how this relates to NRT/TLOG/PULL which are also maybe role like

WDYT?

-Gus

[Quoted text hidden]
--
http://www.needhamsoftware.com (work)
http://www.the111shift.com (play)

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Wed, Oct 27, 2021 at 7:24 PM
To: dev@solr.apache.org

Hi Gus,

> I think that we should expand/edit your list of roles to be

The list can be expanded as and when more isolation and features are needed. I only listed those roles that we already have a functionality for or is under development.

> I would like to recommend that the roles be all positive ("Can do this") and nodes with no role at all are ineligible for all activities.

It comes down to the defaults and backcompat. If we want all Solr nodes to be able to host data replicas by default (without user explicitly specifying role=data), then we need a way to unset this role. The most reasonable way sounded like a "!data". We can do away with !data if we mandate each and every data node have the role "data" explicitly defined for it, which breaks backcompat and also is cumbersome to use for those who don't want to use these special roles.

> I also suggest that these roles each have a node in zookeeper listing the current member nodes (as child nodes) so that code that wants to find a node with an appropriate role does not need to scan the list of all nodes parsing something to discover which nodes apply and also does not have to parse json to do it.

/roles.json exists today, it has role as key and list of nodes as value. In the next major version, we can change the format of that file and use key as node, value as list of roles. Or, maybe we can go for adding the roles to the data for each item in the list of live_nodes.

> any code seeking to transition a node

We considered this situation and realized that it is very risky to have nodes change roles while they are up and running. Better to assign fixed roles upon startup.

> I know of a case that would benefit from having separate Query/Update nodes that handle a heavy analysis process which would be deployed to a number of CPU heavy boxes (which might add more in prep for bulk indexing, and remove them when bulk was done), data could then be hosted on cheaper nodes....

This is the main motivation behind this work. SOLR-15715 needs this, and hence it would be good to get this in as soon as possible.

Regards,
Ishan
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Wed, Oct 27, 2021 at 9:17 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Wed, Oct 27, 2021 at 9:55 AM Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> wrote:

> Hi Gus,
>
> > I think that we should expand/edit your list of roles to be
>
> The list can be expanded as and when more isolation and features are needed. I only listed those roles that we already have a functionality for or is under development.

Well all of those roles (except zookeeper) are things nodes do today. As it stands they are all doing all of them. What we add support for as we move forward is starting without a role, and add the zookeeper role when that feature is ready.

> > I would like to recommend that the roles be all positive ("Can do this") and nodes with no role at all are ineligible for all activities.
>
> It comes down to the defaults and backcompat. If we want all Solr nodes to be able to host data replicas by default (without user explicitly specifying role=data), then we need a way to unset this role. The most reasonable way sounded like a "!data". We can do away with !data if we mandate each and every data node have the role "data" explicitly defined for it, which breaks backcompat and also is cumbersome to use for those who don't want to use these special roles.

Not sure I understand, which of the roles I mentioned (other than zookeeper, which I expect is intended as different from our current embedded zk) is NOT currently supported by a single cloud node brought up as shown in our tutorials/docs? I'm certainly not proposing that the default change to nothing. The default is all roles, unless you specify roles at startup.

> > I also suggest that these roles each have a node in zookeeper listing the current member nodes (as child nodes) so that code that wants to find a node with an appropriate role does not need to scan the list of all nodes parsing something to discover which nodes apply and also does not have to parse json to do it.
>
> /roles.json exists today, it has role as key and list of nodes as value. In the next major version, we can change the format of that file and use key as node, value as list of roles. Or, maybe we can go for adding the roles to the data for each item in the list of live_nodes.

I'm not finding anything in our documentation about roles.json so I think it's an internal implementation detail, which reduces back compat concerns. ADDROLE/REMOVEROLE don't accept json or anything like that and could be made to work with zk nodes too.

The fact that some precursor work was done without a SIP (or before SIPs existed) should not hamstring our design once a SIP that clearly covers the same topic is under consideration. By their nature SIP's are non-trivial and often will include compatibility breaks. Good news is I don't think I see one here, just a code change to transition to a different zk backend. I think that it's probably a mistake to consider our zookeeper data a public API and we should be moving away from that or at the very least segregating clearly what in zk is long term reliable. Ideally our v1/v2 api's should be the public api through which information about the cluster is obtained. Programming directly against zk is kind of like a custom build of solr. Sometimes useful and appropriate, but maintenance is your concern. For code plugging into solr, it should in theory be against an internal information java api, and zookeeper should not be touched directly. (I know this is not in a good state or at least wasn't last time I looked closely, but it should be where we are heading).

> > any code seeking to transition a node
>
> We considered this situation and realized that it is very risky to have nodes change roles while they are up and running. Better to assign fixed roles upon startup.

I agree that concurrency is hard. I definitely think startup time assignments should be involved here. I'm not thinking that every transition must be supported. As a starting point it would be fine if none were. Having something suddenly become zookeeper is probably tricky to support (see discussion in that thread regarding nodes not actually participating until they have a partner to join with them to avoid even numbered clusters), but I think the design should not preclude the possibility of nodes becoming eligible for some roles or withdrawing from some roles, and treatment of roles should be consistent. In some cases someone may decide it's worth the work of handling the concurrency concerns, best if they don't have to break back compat or hack their code around the assumption it wouldn't happen to do it.

Taking the zookeeper case as an example, it very much might be desirable to have the possibility to heal the zk cluster by promoting another node (configured as eligible for zk) to active zk duty if one of the current zk nodes has been down long enough (say on prem hardware, motherboard pops a capacitor, server gone for a week while new hardware is purchased, built and configured). Especially if the down node didn't hold data or other nodes had sufficient replicas and the cluster is still answering queries just fine.

> > I know of a case that would benefit from having separate Query/Update nodes that handle a heavy analysis process which would be deployed to a number of CPU heavy boxes (which might add more in prep for bulk indexing, and remove them when bulk was done), data could then be hosted on cheaper nodes....
>
> This is the main motivation behind this work. SOLR-15715 needs this, and hence it would be good to get this in as soon as possible.

I think we can incrementally work towards configurability for all of these roles. The current default state is that a node has all roles and the incremental progress is to enable removing a role from a node. This I think is why it might be good to to

A) Determine the set of roles our current solr nodes are performing (that might be removed in some scenario) and document this via assigning these roles as default on as this SIP goes live.
B) Figure out what the process of adding something entirely new that we haven't yet thought of with its own role would look like.

I think it would be great if we not only satisfied the current need but determined how we expect this to change over time.
[Quoted text hidden]

Ilan Ginzburg <ilansolr@gmail.com> Wed, Oct 27, 2021 at 9:44 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

How do we expect the roles to be used?
One way I see is a node refusing to do anything related to a role it doesn't have.
For example if a node does not have role "data", any attempt to create a core on it would fail.
A node not having the role "query", will refuse to have anything to do with handling a query etc.
Then it would be up to other code to make sure only the appropriate nodes are requested to do any type of action.
So for example any replica placement code plugin would have to restrict the set of candidate nodes for a new replica placement to those having "data". Otherwise the call would fail, and there should be nothing the replica placement code can do about it.

Similarly, the "overseer" role would limit the nodes that participate in the Overseer election. The Overseer election code would have to remove (or not add) all non qualifying nodes from the election, and we should expect a node without role "overseer" to refuse to start the Overseer machinery if asked to...

Trying to make the use case clear regarding how roles are used.
Ilan
[Quoted text hidden]

Ilan Ginzburg <ilansolr@gmail.com> Wed, Oct 27, 2021 at 9:47 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

In other words, roles are all "positive", but their consequences are only negative (rejecting when the matching positive role is not present).

We can also consider no role defined = all roles allowed. Will make things simpler.
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Wed, Oct 27, 2021 at 9:53 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

> In other words, roles are all "positive", but their consequences are only negative (rejecting when the matching positive role is not present).

Yeah right. to do something the machine needs the role

> We can also consider no role defined = all roles allowed. Will make things simpler.

in terms of startup command yes. Internally we should have all explicitly assigned when no roles are specified at startup so that the code doesn't have a million if checks for the empty case
[Quoted text hidden]
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Wed, Oct 27, 2021 at 11:03 PM
To: dev@solr.apache.org

bq. In other words, roles are all "positive", but their consequences are only negative (rejecting when the matching positive role is not present).

Essentially, yes. A node that doesn't specify any role should be able to do everything.

Let me just take a brief detour and mention our thoughts on the "query" role. While all data nodes can also be used for querying, our idea was to create a layer of nodes that have some special mechanism to be able to proxy/forward queries to data nodes (lets call it "pseudo cores" or "synthetic cores" or "proxy cores". Our thought was that any node that has "query,!data" role would enable this special mode on startup (whereby requests are served by these special pseudo cores). We'll discuss about this in detail in that issue.

Back to the main subject here.

Lets take a practical scenario:
* Layer1: Organization has about 100 nodes, each node has many data replicas
* Layer2: To manage such a large cluster reliably, they keep aside 4-5 dedicated overseer nodes.
* Layer3: Since query aggregations/coordination can potentially be expensive, they keep aside 5-10 query nodes.

My preference would be as follows:
* I'd like to refer to Layer1 nodes as the "data nodes" and hence get either no role defined for them or -Dnode.roles=data.
* I'd like to refer to Layer2 nodes as "overseer nodes" (even though I understand, only one of them can be an overseer at a time). I'd like to have -Dnode.roles=!data,overseer
* I'd like to refer to Layer3 nodes as "query nodes", with -Dnode.roles=!data,query

^ This seems very practical from operational standpoint.

So, for the proposal, lets say "data" is a special role which is assumed by default, and is enabled on all nodes unless there's a !data. It is presumed that data nodes can also serve queries directly, so adding a "query" to those nodes is meaningless (also because there's no practical benefit to stopping a data node from receiving a query for "!query" role to be useful).

"query" role on nodes that don't host data really refers to a special capability for lightweight, stateless nodes. I don't want to add a "!query" on dedicated overseer nodes, and hence I don't want to assume that "query" is implicitly avaiable on any node even if the role isn't specified.

"overseer" role is complicated, since it is already defined and we don't have the opportunity to define it the right way. I'd hate having to put a "!overseer" on every data node on startup in order to have a few dedicated overseers.

In short, in this SIP, I just wish to implement the concept of nodes and its handling. How individual roles are leveraged can be up to every new role's implementation.

[Quoted text hidden]

Houston Putman <houstonputman@gmail.com> Thu, Oct 28, 2021 at 12:13 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

As for the "query" role, let's name it something better like "compute", since data nodes are always going to be "querying". "compute" is for something like the first node for a distributed query, or a StreamingExpressions query.

But I agree with the idea that roles should only be "positive", you shouldn't be able to specify -Dnodes.roles=!data. Also if no nodes have a given role specified, then all nodes should be considered for that role. E.g. if no live nodes have roles=overseer (or roles=all), then we should just select any node to be overseer. This should be the same for compute, data, etc.

> So, for the proposal, lets say "data" is a special role which is assumed by default, and is enabled on all nodes unless there's a !data.

Instead of this, maybe we have role groups. Such as admin~=overseer,zk or worker~=compute,data, updateProcessing

As for the suggested Roles, I'm not sure ADMIN or UI really fit, since there is another option to disable the UI for a solr node, and various ADMIN commands have to be accepted across other node roles. (Data nodes require the Collections API, same with the overseer.)

- Houston
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Thu, Oct 28, 2021 at 12:22 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Excellent to see what you are thinking there, some of it seems pretty specific to your case but setting aside the specialized aggregation functionality and casting it in my proposed refinement seems to only require:

> My preference would be as follows:
> * I'd like to refer to Layer1 nodes as the "data nodes" and hence get either no role defined for them or -Dnode.roles=DATA.
> * I'd like to refer to Layer2 nodes as "overseer nodes" (even though I understand, only one of them can be an overseer at a time). I'd like to have -Dnode.roles=OVERSEER
> * I'd like to refer to Layer3 nodes as "query nodes", with -Dnode.roles=QUERY

Possibly you want one of those to also have UPDATE and maybe overseer has ADMIN too ... maybe some or none with UI

Since data isn't specified it's not available and no need to have a negative role to disable anything. It's like good security systems, deny by default, enable by specification.

For back compat the startup script would pass -Dnode.roles=QUERY,UPDATE,DATA,OVERSEER,ADMIN,UI by default if no option were specified (ie. /bin/solr start --roles=OVERSEER

Your specialized query functionality sounds like it's trying to get to something like elastic's transforms? That could be its another role. And ability to plug in roles is an interesting notion, possibly such a thing would exist via being it's own servlet (much as I think query, update, ui, admin_v1 and admin_v2 should all be separate servlets... and zookeeper embedded probably should be a context listener that starts before the listener I've proposed in https://issues.apache.org/jira/browse/SOLR-15590)...

In my case the heavy work would be right in the analysis chain as opposed to aggregation related (and those boxes need a jni lib for such analysis, others would not). Presently this is addressed by having external code invoke solr analysis and pre-analyze the field on ingestion boxes, but we still need the query side capability of course to produce the same set of tokens for a query. Might simplify things to have ephemeral/temporary update nodes that didn't handle/affect the queries but were otherwise identically configured...)

As a side note query routing in general has also been bouncing around the back of my brain for TRA purposes... queries filtering on the date field don't need to go to all collections in the TRA, this relates to the routing of queries to query nodes perhaps... perhaps not. Need more time than I have right now to think about that one :)

On Wed, Oct 27, 2021 at 1:34 PM Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> wrote:
[Quoted text hidden]
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Thu, Oct 28, 2021 at 12:33 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Wed, Oct 27, 2021 at 2:44 PM Houston Putman <houstonputman@gmail.com> wrote:

> As for the "query" role, let's name it something better like "compute", since data nodes are always going to be "querying".

I don't think it's unreasonable to want to have nodes that don't accept queries. This is just ishan's use case.

> if no live nodes have roles=overseer (or roles=all), then we should just select any node to be overseer. This should be the same for compute, data, etc.

Definitely not what I would like. If I'm going to try to segregate data nodes out to certain nodes, I don't want them appearing elsewhere just cause something went down or filled up. Nor would I want updates to suddenly start bogging down my query nodes....

> > So, for the proposal, lets say "data" is a special role which is assumed by default, and is enabled on all nodes unless there's a !data.
>
> Instead of this, maybe we have role groups. Such as admin~=overseer,zk or worker~=compute,data, updateProcessing

Roll groups sounds like a next level feature to be built on top once we figure out how roles work independently.

> As for the suggested Roles, I'm not sure ADMIN or UI really fit, since there is another option to disable the UI for a solr node, and various ADMIN commands have to be accepted across other node roles. (Data nodes require the Collections API, same with the overseer.)

I admit I'm angling towards a world in which enabling and disabling broad features is done in one way in one place... As for admin there might be a distinction between commands issued between nodes and from the outside world... I have this other idea about inter-node communication that's even less baked that I wont distract with here ;)
[Quoted text hidden]
[Quoted text hidden]

Houston Putman <houstonputman@gmail.com> Thu, Oct 28, 2021 at 1:04 AM

Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

> I don't think it's unreasonable to want to have nodes that don't accept queries. This is just ishan's use case.

Maybe I am misunderstanding, and it does deal with your last point about inter-node communication, but Peer-sync uses queries when doing replication between replicas. If a node doesn't have queries enabled, then it's possible to break peer sync. There are other options to make sure certain replicas aren't queried (shards.preference).
For the separation of update/query traffic, I understand having compute nodes that deal with pre-replica commands, such as managing distributed queries or pre-processing documents in the URP chain. But for actual non-distrib queries and final update requests, the only way to actually separate this traffic is using PULL/TLOG replicas, because otherwise (with NRT) all update requests are still going to the query nodes, just the same as non-query nodes that are hosting those replicas. (and shard leadership could go to any "data" node, since I imagine we wouldn't filter out the "query" nodes...) The shards.preference option makes it easy to send queries to only PULL replicas in this scenario.
That's why I saw this more as a "compute" role rather than "query".

> Definitely not what I would like. If I'm going to try to segregate data nodes out to certain nodes, I don't want them appearing elsewhere just cause something went down or filled up. Nor would I want updates to suddenly start bogging down my query nodes....

I guess it depends on what you are optimizing for. Maybe there can be an option for this. like
-DnonLenientRoles=data,update or something like that.
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Thu, Oct 28, 2021 at 11:29 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Wed, Oct 27, 2021 at 3:34 PM Houston Putman <houstonputman@gmail.com> wrote:
> > I don't think it's unreasonable to want to have nodes that don't accept queries. This is just ishan's use case.
>
> Maybe I am misunderstanding, and it does deal with your last point about inter-node communication, but Peer-sync uses queries when doing replication between replicas. If a node doesn't have queries enabled, then it's possible to break peer sync. There are other options to make sure certain replicas aren't queried (shards.preference).
> For the separation of update/query traffic, I understand having compute nodes that deal with pre-replica commands, such as managing distributed queries or pre-processing documents in the URP chain. But for actual non-distrib queries and final update requests, the only way to actually separate this traffic is using PULL/TLOG replicas, because otherwise (with NRT) all update requests are still going to the query nodes, just the same as non-query nodes that are hosting those replicas. (and shard leadership could go to any "data" node, since I imagine we wouldn't filter out the "query" nodes...) The shards.preference option makes it easy to send queries to only PULL replicas in this scenario.
> That's why I saw this more as a "compute" role rather than "query".

Yeah for internal stuff we still need the ability to query so we might need to accommodate that that, but you may not have noticed the bit where I mentioned Query nodes doing the parsing/analysis of the query and then sending a fully parsed (possibly serialized lucene objects) query to the data node. So data nodes would only speak a single lucene level query language and not parse queries or analyze text. In theory, with that, one could even have something like elastic reduce a request to lucene objects and then get an answer from a solr data node (for simple cases without need to find shards via zookeeper etc) certainly many details and corner cases to think about there.

> > Definitely not what I would like. If I'm going to try to segregate data nodes out to certain nodes, I don't want them appearing elsewhere just cause something went down or filled up. Nor would I want updates to suddenly start bogging down my query nodes....
>
> I guess it depends on what you are optimizing for. Maybe there can be an option for this. like
> -DnonLenientRoles=data,update or something like that.

A possibility

[Quoted text hidden]
[Quoted text hidden]

Ilan Ginzburg <ilansolr@gmail.com> Fri, Oct 29, 2021 at 1:54 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

If we make collections role-aware for example (replicas of that collection can only be placed on nodes with a specific role, in addition to the other role based constraints), the set of roles should be user extensible and not fixed.

If collections are not role aware, the constraints introduced by roles apply to all collections equally which might be insufficient if a user needs for example a heavily used collection to only be placed on more powerful nodes.

Ilan
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Fri, Oct 29, 2021 at 2:52 PM
To: dev@solr.apache.org

It seems to me, after going through this thread, that the role "query" is misleading for what we're planning to introduce in SOLR-15715. We're essentially introducing a functionality for performing, what we initially called, "query aggregations". The actual queries run on data nodes anyway, just that the first point of entry for such distributed queries will be a separate node with this extra functionality. Towards that, I feel we should call a node with such capability as a "coordinator" node (instead of "query node"). It fits nicely with the mental model of ElasticSearch as well: https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html#coordinating-node.

Proposing that if a node has a role "coordinator", then that node is already assumed to have no data replicas on it. If a node starts with roles "coordinator,data" both, then the startup should fail with a message saying a coordinator node should not host data on it. A coordinator node, though, can have other roles like "zookeeper" or "overseer" etc.

I'll introduce a change to the SIP document, unless there are objections/questions/concerns. WDYT?

[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Fri, Oct 29, 2021 at 3:18 PM
To: dev@solr.apache.org

> I'll introduce a change to the SIP document, unless there are objections/questions/concerns. WDYT?

I've made the change to the document. Feedback on this welcome.
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Fri, Oct 29, 2021 at 7:10 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

I've heard a number of folks agree that we should not have negative (role removal) values for roles (!data in the sip).

I also don't like the idea of the "coordinator" creating assumptions about other roles. I think the point of avoiding "!data" is to make it programmatically and logically easy to tell what role a node has, if we have to have a method called figureOutImpliedRoles() with a lot of logic in it that's bad. It should just be getRoles().contains(role), trivially returning the roles that are already declared in config/zk/whatever.

We don't have to support every possible role all at once. We can have "basic functionality" that all nodes provide regardless of roles (right now that's everything), and then lop off chunks of basic functionality and assign them to roles. That should be easy and backward compatible if we then give the new role to every node by default on upgrade.

However we should carefully think about what should and shouldn't be part of any role, because moving functionality out of a role back to basic functionality or between roles will create backwards compatibility issues. This is why I think we should have a concept of what roles we will have in the future, so we don't inadvertently move functionality into a role that later needs to go in some other role (mistakes/bugs may happen of course, but best effort).

So boiling it down I've seen suggestion for the following additions/edits to the SIP:

1. (+Gus, +Houston,+Ilan) Positive roles, the existence of which implies functionality such that if a node can provide functionality. i.e. it always has the role if it can and if it doesn't have the role it can't provide the functionality.
2. (+Houston,+Ishan,+Gus - below) Rename query role
3. (+Gus) We should include a plan for the overall set of roles to work towards and then build them out as time allows us to.
4. (+Gus) We have a distinction between "capable" and "currently providing"
5. (+Gus) Capable be evidenced by a config/startup designation that adds a list of roles to a json file in zk where the nodes are all listed
6. (+Gus) Providing be evidenced by the node adding an list of ephemeral nodes (similar to live_nodes) for each role
7. (+Ilan, +Gus) Making collections role aware

Ilan suggested that we make collections role-aware which would make some sense since the collection might want to have a minimum of 2 query-aggregator nodes available, might want to avoid zk nodes, etc. I think that this is a good next feature and the intention should be added to the SIP, but need not be in the initial implementation since by default everything can have all roles (roles implemented to date) and initially removing roles from nodes will be an advanced/manual feature mostly applicable to static clusters that don't add collections regularly, then support for role aware collections can be added to make the feature useful for a wider audience (should be its own ticket anyway, and it interacts with replica placement).

I've heard several agree with #1, and it seems 3-6 were either not yet clear or folks are still deliberating as I haven't noticed positive or negative opinions there, just some discussion of the definition of candidate roles. I'm fond of 3-5 because it allows for things like knowing what the capabilities of a down node are, and finding a provider without having to cross-coordinate with live_nodes. (keeps code simple, avoids racing between the check for liveness and the check for the capability) Also, a node joining as live and able to serve queries can be decoupled from when it's ready to provide a service (thinking at least zk here, waiting for a 2nd node capable of zk before expanding the zk cluster to avoid even numbered clusters).

Ishan, specifics on how your coordinator node would work would be interesting to know if it really is distinct from my concept of a "query" node. I agree that that term is probably confusing, I used it to mean "query parsing" you meant it as "query aggregator".

As a side note, with positive only roles and all roles added unless specified otherwise, Ishan's use case might be as simple as just removing the DATA role from a few nodes and restricting the aggregation queries concerned to those nodes. To get solr to enforce the restriction for you, then a "query/compute/coordinator" role must be removed from the remainder of the nodes.

-Gus
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Fri, Oct 29, 2021 at 7:20 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

edit:
6. (+Gus) Providing be evidenced by a the node **adding itself to a list** of ephemeral nodes (similar to live_nodes) for each role
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 5:22 PM
To: dev@solr.apache.org

Hi Gus,

Thanks for the summary.

> (+Gus, +Houston,+Ilan) Positive roles, the existence of which implies functionality such that if a node can provide functionality. i.e. it always has the role if it can and if it doesn't have the role it can't provide the functionality.

I've removed the concept of "!data" from the SIP proposal. A node that doesn't have -Dnode.roles parameter will be assumed to have -Dnode.roles=data. If a node is started with a node.roles param, it must include "data" for all nodes hosting data.

> (+Houston,+Ishan,+Gus - below) Rename query role

Coordinator role should be better now.

> (+Gus) We should include a plan for the overall set of roles to work towards and then build them out as time allows us to.
> (+Gus) We have a distinction between "capable" and "currently providing"
> (+Gus) Capable be evidenced by a config/startup designation that adds a list of roles to a json file in zk where the nodes are all listed
> (+Gus) Providing be evidenced by the node adding an list of ephemeral nodes (similar to live_nodes) for each role

From an overall conceptual point of view, there doesn't need to be any specialization for a role. When a new role is introduced, such details on behaviour and implementation can be documented and defined then. As for OVERSEER role today, it can be documented as a role that marks a node to be a "preferred" overseer (or eligible/capable etc.), and "currently providing" can be determined by the OVERSEERSTATUS api call or the overseer leader election queue.

> (+Ilan, +Gus) Making collections role aware

Seems to me that this is something that can be introduced as a follow up, and we don't want to complicate the proposed design early on.

> Ishan, specifics on how your coordinator node would work would be interesting to know if it really is distinct from my concept of a "query" node. I agree that that term is probably confusing, I used it to mean "query parsing" you meant it as "query aggregator".

As of now, the coordinator node would be capable to servicing query (or indexing at a later point in time) requests by handling the queries on the coordinator nodes itself, and making shard-requests to data nodes. If we want to have the coordinator nodes do even more work, i.e. do query parsing on behalf of the shards, the capability can be further enhanced.

Regards,
Ishan
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 5:31 PM
To: dev@solr.apache.org

If the changes and the scope seem acceptable, should we proceed for a vote?
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 5:37 PM
To: dev@solr.apache.org

> Similarly, the "overseer" role would limit the nodes that participate in the Overseer election. The Overseer election
> code would have to remove (or not add) all non qualifying nodes from the election, and we should expect a node
> without role "overseer" to refuse to start the Overseer machinery if asked to...

In this issue, I'd rather that we just define the broad concept of roles and implement the mechanism for users to assign

the roles to the nodes. For the overseer role, that already exists, I propose that we leave it as is for now (and document the current behaviour) and improve later as necessary. At that time, we can consider making changes to election code to not add nodes that aren't designated with "overseer" role.
[Quoted text hidden]

Ilan Ginzburg <ilansolr@gmail.com> Mon, Nov 1, 2021 at 7:16 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Mon, Nov 1, 2021 at 12:53 PM Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> wrote:

> I've removed the concept of "!data" from the SIP proposal. A node that doesn't have -Dnode.roles parameter will be assumed to have -Dnode.roles=data. If a node is started with a node.roles param, it must include "data" for all nodes hosting data.

A node not having node.roles defined should be assumed to have all roles. Not only data. I don't see a reason to special case this one or any role.

> As for OVERSEER role today, it can be documented as a role that marks a node to be a "preferred" overseer (or eligible/capable etc.), and "currently providing" can be determined by the OVERSEERSTATUS api call or the overseer leader election queue.

If node roles are used on all nodes, nodes having OVERSEER role should not be "preferred" overseer, but should be **the only nodes** where Overseer can run. Otherwise things are not clear. Nodes not defining roles for themselves are assumed to have all roles (as per comment above) and can run Overseer.
If we want the notion of "preferred overseer", then let's call a role PREFERRED_OVERSEER, and understand that overseer can run anywhere. Same goes with other roles such as ZK etc.
If we want a seamless transition from the existing Overseer role (that is a preferred overseer) we should consider an ad hoc transition or a rename.

> \> (+Ilan, +Gus) Making collections role aware
>
> Seems to me that this is something that can be introduced as a follow up, and we don't want to complicate the proposed design early on.

It does impact the design from early on: the set of roles need to be expandable by a user by creating a collection with new roles for example (consumed by placement plugins) and be able to start nodes with new (arbitrary) roles. Should such roles follow some naming syntax to differentiate them from built in roles? To be able to fail on typos on roles - that otherwise can be crippling and hard to debug.
This implies in any case that the current design can't assume all roles are known at compile time or define them in a Java enum.

Ilan

Gus Heck <gus.heck@gmail.com> Mon, Nov 1, 2021 at 8:11 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Mon, Nov 1, 2021 at 7:53 AM Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> wrote:

> Hi Gus,
>
> Thanks for the summary.
>
> \> (+Gus, +Houston,+Ilan) Positive roles, the existence of which implies functionality such that if a node can provide functionality. i.e. it always has the role if it can and if it doesn't have the role it can't provide the functionality.
>
> I've removed the concept of "!data" from the SIP proposal. A node that doesn't have -Dnode.roles parameter will be assumed to have -Dnode.roles=data. If a node is started with a node.roles param, it must include "data" for all

> nodes hosting data.

That's a good start, but I think you are not quite hearing me. There should be no "assumptions" Nothing to figure out. A node has a role or not. For back compatibility reasons, all roles would be assumed on startup if none specified. Also, behavior of roles must be consistent. It's not good to have some strictly specify the absence of functionality whereas others imply a preference, this leaves the programmer (and the person looking at roles in a UI or json response) having to do a bunch of calculations, or check other information to know what nodes provide what functionality. Things like preference order would be internal to the role functionality. So if there's preference, that's preference among nodes that have the role.

Let's clarify that I think roles ought to be both positive, and absolute.

Restating something I said above, there should be

- Basic functionality - all nodes have it regardless of role
- Role functionality - If the node has the role it can provide the functionality, if it does not it will not until restarted with the role.

> \> (+Houston,+Ishan,+Gus - below) Rename query role
>
> Coordinator role should be better now.
>
> - \> (+Gus) We should include a plan for the overall set of roles to work towards and then build them out as time allows us to.
> - \> (+Gus) We have a distinction between "capable" and "currently providing"
> - \> (+Gus) Capable be evidenced by a config/startup designation that adds a list of roles to a json file in zk where the nodes are all listed
> - \> (+Gus) Providing be evidenced by the node adding an list of ephemeral nodes (similar to live_nodes) for each role
>
> From an overall conceptual point of view, there doesn't need to be any specialization for a role. When a new role is introduced, such details on behaviour and implementation can be documented and defined then. As for OVERSEER role today, it can be documented as a role that marks a node to be a "preferred" overseer (or eligible/capable etc.), and "currently providing" can be determined by the OVERSEERSTATUS api call or the overseer leader election queue.

What you quote above is not about specialization? It's about specific implementation, and about providing

- a facility for distinguishing nodes capable of role duties,
- a facility for distinguishing the subset of those nodes actively performing those duties

For overseer this likely means only one node at a time is a "live" overseer even if 4 of 10 nodes have the role.Wouldn't it be nice to be able to find the current overseer as a single child found at /<chroot>/roles/overseer/ solr73.example.net_8983?

Imagine how simple the code for finding the active overseer would be...
Imagine administrators knowing at a glance simply by expanding the zk tree in the UI... (you don't have to add/edit a UI page! That can come later :))
Replica placement would simply start with the list at <chroot>/roles/data...

Let's do a little work now to make our code simple and our lives easier later.

> \> (+Ilan, +Gus) Making collections role aware
>
> Seems to me that this is something that can be introduced as a follow up, and we don't want to complicate the proposed design early on.

I am 100% on board with an incremental implementation within the constraints of a clear design. An example incremental path I can easily agree with is:

1. Declare all current functionality "basic functionality" (no role required)
2. Define a set of "rules" that roles must follow (in the SIP)
3. Create tickets to implement individual roles (conforming to the SIP derived specification of requirements for a role, i.e. positive, absolute,)
4. Create follow on tickets to add things like role awareness to collections

I think the core need (I'm guessing) you have can be satisfied by implementing just DATA and COORDINATOR nodes.

> > Ishan, specifics on how your coordinator node would work would be interesting to know if it really is distinct from my concept of a "query" node. I agree that that term is probably confusing, I used it to mean "query parsing" you meant it as "query aggregator".
>
> As of now, the coordinator node would be capable to servicing query (or indexing at a later point in time) requests by handling the queries on the coordinator nodes itself, and making shard-requests to data nodes. If we want to have the coordinator nodes do even more work, i.e. do query parsing on behalf of the shards, the capability can be further enhanced.

This fits within the basic functionality plus roles paradigm

I actually don't want a coordinator to do more work, I would prefer small focused roles with names that accurately describe their function. In that light, COORDINATOR might be too nebulous. How about AGREGATOR role? (what I was thinking of would better be called a QUERY_ANALYSIS role)

One thing that's missing from your description is whether or not you intend to provide query routing based on roles, or whether clients will be responsible for finding out the nodes and only sending requests to them (a reason I want it to be dead simple to find the list of nodes). Query routing is something we probably should do carefully so I'd be inclined not to add it into this SIP unless you find it critical for your use case. You can expect me to want to know how the same infrastructure will serve other routing use cases like routing to specific sub-collections of a TRA etc...

I suggest we add a Non-Goal to the sip for query-routing.
[Quoted text hidden]
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Mon, Nov 1, 2021 at 8:21 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Mon, Nov 1, 2021 at 9:47 AM Ilan Ginzburg <ilansolr@gmail.com> wrote:

> On Mon, Nov 1, 2021 at 12:53 PM Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> wrote:
>
> > I've removed the concept of "!data" from the SIP proposal. A node that doesn't have -Dnode.roles parameter will be assumed to have -Dnode.roles=data. If a node is started with a node.roles param, it must include "data" for all nodes hosting data.
>
> A node not having node.roles defined should be assumed to have all roles. Not only data. I don't see a reason to special case this one or any role.
>
> > As for OVERSEER role today, it can be documented as a role that marks a node to be a "preferred" overseer (or eligible/capable etc.), and "currently providing" can be determined by the OVERSEERSTATUS api call or the overseer leader election queue.
>
> If node roles are used on all nodes, nodes having OVERSEER role should not be "preferred" overseer, but should be **the only nodes** where Overseer can run. Otherwise things are not clear. Nodes not defining roles for themselves are assumed to have all roles (as per comment above) and can run Overseer.
> If we want the notion of "preferred overseer", then let's call a role PREFERRED_OVERSEER, and understand that overseer can run anywhere. Same goes with other roles such as ZK etc.

See my comment above, but maybe preference is something handled as a feature of the role rather than via role designation?

> If we want a seamless transition from the existing Overseer role (that is a preferred overseer) we should consider an ad hoc transition or a rename.
>
> > > (+Ilan, +Gus) Making collections role aware
> >
> > Seems to me that this is something that can be introduced as a follow up, and we don't want to complicate the proposed design early on.
>
> It does impact the design from early on: the set of roles need to be expandable by a user by creating a collection with new roles for example (consumed by placement plugins) and be able to start nodes with new (arbitrary) roles. Should such roles follow some naming syntax to differentiate them from built in roles? To be able to fail on typos on roles - that otherwise can be crippling and hard to debug.
> This implies in any case that the current design can't assume all roles are known at compile time or define them in a Java enum.

We need to be careful not to speak of collection roles. That would be a separate thing from "Node Roles" I think.
I don't think we should allow arbitrary role names for end-users to hinge client functionality on. Then when we want to add a new role with a name we break some unknown set of users that used that name for something else. If we want fre-rorm selectors we should have "tags" or "selectors" as a separate feature.

I agree however we need to code to assume an unbounded set of future roles (features that we add in the future)

> Ilan

[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Mon, Nov 1, 2021 at 8:31 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

TLDR summary of what I'm advocating. I want node roles to be:

- Positive - They denote the existence of a capability
- Absolute - Absence/Presence binary identification of a capability; no implications, no assumptions
- Focused - Do one thing per role
- Accessible - It should be dead simple to determine the members of a role, avoid parsing blobs of json, avoid calculating implications, avoid consulting other resources after listing nodes with the role
- Independent - One role should not require other roles to be present
- Persistent - roles should not be lost across reboot
- Immutable - roles should not change while the node is running
- Lively - A node with a capability may not be presently providing that capability.

-Gus
[Quoted text hidden]

Jan Høydahl <jan.asf@cominvent.com> Mon, Nov 1, 2021 at 8:42 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

> 1. nov. 2021 kl. 14:46 skrev Ilan Ginzburg <ilansolr@gmail.com>:
> A node not having node.roles defined should be assumed to have all roles. Not only data. I don't see a reason to special case this one or any role.

+1, make it simple and transparent. No role == all roles. Explicit list of roles = exactly those roles.

> (Gus) See my comment above, but maybe preference is something handled as a feature of the role rather than via

role designation?

Yea, we always need an overseer, so that feature can decide to use its list of nodes as a preference if it so chooses.

Aside: I think it makes it easier if we always prefix Solr env.vars and sys.props with "SOLR_" or "solr.", i.e. -Dsolr.node.roles=foo. That way we can get away from having to have explicit code in bin/solr, bin/solr.cmd and SolrCLI to manage every single property. Instead we can parse all ENVs and Props with the solr prefix in our bootstrap code. And we can by convention allow e.g. docker run -e SOLR_NODE_ROLES=foo solr:9 and it would be the same ting...

Jan

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 9:01 PM
To: dev@solr.apache.org

> If we make collections role-aware for example (replicas of that collection can only be
> placed on nodes with a specific role, in addition to the other role based constraints),
> the set of roles should be user extensible and not fixed.
> If collections are not role aware, the constraints introduced by roles apply to all collections
> equally which might be insufficient if a user needs for example a heavily used collection to
> only be placed on more powerful nodes.

I feel node roles and role-aware collections are orthogonal topics. What you describe above can be achieved by the autoscaling+replica placement framework where the placement plugins take the node roles as one of the inputs.

> It does impact the design from early on: the set of roles need to be expandable by a user
> by creating a collection with new roles for example (consumed by placement plugins) and be
> able to start nodes with new (arbitrary) roles. Should such roles follow some naming syntax to
> differentiate them from built in roles? To be able to fail on typos on roles - that otherwise can be
> crippling and hard to debug. This implies in any case that the current design can't assume all
> roles are known at compile time or define them in a Java enum.

I think this should be achieved by something different from roles. Something like node **labels** (user defined) which can then be used in a replica placement plugin to assign replicas. I see roles as more closely associated with kinds of functionality a node is designated for. Therefore, I feel that replica placements and user defined node labels is out of scope for this SIP. It can be added later in a separate SIP, without being at odds with this proposal.

[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Mon, Nov 1, 2021 at 9:01 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Mon, Nov 1, 2021 at 11:12 AM Jan Høydahl <jan.asf@cominvent.com> wrote:

> > 1. nov. 2021 kl. 14:46 skrev Ilan Ginzburg <ilansolr@gmail.com>:
> > A node not having node.roles defined should be assumed to have all roles. Not only data. I don't see a reason to special case this one or any role.

> +1, make it simple and transparent. No role == all roles. Explicit list of roles = exactly those roles.

I want this to be the user's experience, but I'm hoping for it to be implemented such that on startup, the node assumes (declares) all roles if none are specified. It then displays exactly the same configuration as if one manually explicitly configured all roles. Failing to declare roles when starting the node should be indistinguishable at runtime from having declared every role. This is so that client code isn't special casing every check.

> > (Gus) See my comment above, but maybe preference is something handled as a feature of the role rather than via role designation?
>
> Yea, we always need an overseer, so that feature can decide to use its list of nodes as a preference if it so chooses.
>
> Aside: I think it makes it easier if we always prefix Solr env.vars and sys.props with "SOLR_" or "solr.", i.e. -Dsolr.node.roles=foo. That way we can get away from having to have explicit code in bin/solr, bin/solr.cmd and SolrCLI to manage every single property. Instead we can parse all ENVs and Props with the solr prefix in our bootstrap code. And we can by convention allow e.g. docker run -e SOLR_NODE_ROLES=foo solr:9 and it would be the same ting...
>
> Jan
> ---
> To unsubscribe, e-mail: dev-unsubscribe@solr.apache.org
> For additional commands, e-mail: dev-help@solr.apache.org

[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 9:30 PM
To: dev@solr.apache.org

> Ilan: A node not having node.roles defined should be assumed to have all roles. Not only data. I don't see a reason to special case this one or any role.
> Gus: There should be no "assumptions" Nothing to figure out. A node has a role or not. For back compatibility reasons, all roles would be assumed on startup if none specified.
> Jan: No role == all roles. Explicit list of roles = exactly those roles.

Problem with this approach is mainly to do with backcompat.

1. Overseer backcompat:
If we don't make any modifications to how overseer works and adopt this approach (as quoted), then imagine this situation:

Solr1-100: No roles param (assumed to be "data,overseer").
Solr101: -Dnode.roles=overseer (intention: dedicated overseer)

User wants this node Solr101 to be a dedicated overseer, but for that to happen, he/she would need to restart all the data nodes with -Dnode.roles=data. This will cause unnecessary disruption to running clusters where a dedicated overseer is needed. Keep in mind, if a user needs a dedicated overseer, he's likely in an emergency situation and restarting the whole cluster might not be viable for him/her.

2. Future roles might not be compatible with this "assumed to have all roles" idea:
Take the proposed "zookeeper" role for example. Today, regular nodes are not supposed to have embedded ZK running on them. By introducing this artificial limitation ("assumed to have all roles"), we constrain adoption of all future roles to necessarily require a full cluster restart.

Keep in mind newer Solr versions can introduce new capabilities and roles. Imagine we have a role that is defined in a new Solr version (and there's functionality to go with that role), and user upgrades to that version. However, his/her nodes all were started with no node.roles param. Hence, if those nodes are "assumed to have all roles", then just by virtue of upgrading to this new version, new capabilities will be turned on for the entire cluster, whether or not the user opted for such a capability. This is totally undesirable.

> Gus: I actually don't want a coordinator to do more work, I would prefer small focused roles with names that accurately describe their function. In that light, COORDINATOR might be too nebulous. How about AGREGATOR role? (what I was thinking of would better be called a QUERY_ANALYSIS role)

If you want to do specific things like query analysis or query aggregation or bulk indexing etc, all of those can be done on COORDINATOR nodes (as is the case in ElasticSearch). Having tens of of " small focused roles" defined as first class concepts would be confusing to the user. As a remedy to your situation where you want the coordinator role to also do query-analysis for shards, one possible solution is to send such a query to a coordinator node with a parameter like "coordinator.query_analysis=true", and then the coordinator, instead of blindly hitting remote shards, also does some extra work on behalf of the shards.

[Quoted text hidden]

Jan Høydahl <jan.asf@cominvent.com> Mon, Nov 1, 2021 at 9:49 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

I think it is safe to assume that small clusters, say 1-5 nodes will most often want to have all features on all nodes as the cluster is too small to specialize, and then the default is perfect.
For large clusters we should recommend explicitly specifying roles during the 9.0 upgrade. So if you have 100 nodes, you would likely have assigned the overseer role to a handful nodes when upgrading to 9.0.
And for every new feature in 9.x you will explicitly decide whether to use it and what nodes should have the role.

But I assume that a new feature in 9.x that introduces a new role can also decide for some alternative back-compat logic to support rolling restart if it is needed.

Jan
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 9:52 PM
To: dev@solr.apache.org

> Positive - They denote the existence of a capability

Agree, the SIP already reflects this.

> Absolute - Absence/Presence binary identification of a capability; no implications, no assumptions

Disagree, we need backcompat handling on nodes running without any roles. There has to be an implicit assumption as to what roles are those nodes assumed to have. My proposal is that only the "data" role be assumed, but not the "overseer" role. For any future roles ("coordinator", "zookeeper" etc.), this decision as to what absence of any role implies should be left to the implementation of that future role. Documentation should reflect clearly about these implicit assumptions.

> Focused - Do one thing per role

Agree. However, I disagree with ideas where "query analysis" has a role of its own. Where would that lead us to? Separate roles for nodes that do "faceting" or "spell correction" etc.? But anyway, that is for discussion when we add future roles. This is beyond this SIP.

> Accessible - It should be dead simple to determine the members of a role, avoid parsing blobs of json, avoid calculating implications, avoid consulting other resources after listing nodes with the role

Agree. I'm open to any implementation details that make it easy. There should be a reasonable API to return these node roles, with ability to filter by role or filter by node.

> Independent - One role should not require other roles to be present

Do we need to have this hard and fast requirement upfront? There might be situations where this is desirable. I feel we can discuss on a case by case basis whenever a future role is added.

> Persistent - roles should not be lost across reboot

Agree.

> Immutable - roles should not change while the node is running

Agree

> Lively - A node with a capability may not be presently providing that capability.

I don't understand, can you please elaborate?
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Mon, Nov 1, 2021 at 9:53 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Mon, Nov 1, 2021 at 12:00 PM Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> wrote:

> > Ilan: A node not having node.roles defined should be assumed to have all roles. Not only data. I don't see a reason to special case this one or any role.
> > Gus: There should be no "assumptions" Nothing to figure out. A node has a role or not. For back compatibility reasons, all roles would be assumed on startup if none specified.
> > Jan: No role == all roles. Explicit list of roles = exactly those roles.
>
> Problem with this approach is mainly to do with backcompat.
>
> **1. Overseer backcompat:**
> If we don't make any modifications to how overseer works and adopt this approach (as quoted), then imagine this situation:
>
> Solr1-100: No roles param (assumed to be "data,overseer").
> Solr101: -Dnode.roles=overseer (intention: dedicated overseer)

This person has not read the documentation we certainly will provide. Docs would clearly state that the correct way to achieve this is

Solr1-100: -Dnode.roles=data
Solr101: -Dnode.roles=overseer

> User wants this node Solr101 to be a dedicated overseer, but for that to happen, he/she would need to restart all the data nodes with -Dnode.roles=data. This will cause unnecessary disruption to running clusters where a dedicated overseer is needed. Keep in mind, if a user needs a dedicated overseer, he's likely in an emergency situation and restarting the whole cluster might not be viable for him/her.

You say it's unnecessary, I disagree. I think it is necessary. Roles should never change without restart. 1-100 should not suddenly become ineligible when you start 101. Certainly that leaves you with a situation where at least briefly the overseer is on an invalid node. Keep in mind none of this takes effect until they upgrade in the first place.

> **2. Future roles might not be compatible with this "assumed to have all roles" idea:**
> Take the proposed "zookeeper" role for example. Today, regular nodes are not supposed to have embedded ZK running on them. By introducing this artificial limitation ("assumed to have all roles"), we constrain adoption of all future roles to necessarily require a full cluster restart.
>
> Keep in mind newer Solr versions can introduce new capabilities and roles. Imagine we have a role that is defined in a new Solr version (and there's functionality to go with that role), and user upgrades to that version. However, his/her nodes all were started with no node.roles param. Hence, if those nodes are "assumed to have all roles",

> then just by virtue of upgrading to this new version, new capabilities will be turned on for the entire cluster, whether or not the user opted for such a capability. This is totally undesirable.

Yes, good point I didn't express that I was thinking of current functionality. Net new functionally provided by a newly added role, (which zookeeper will be) should be off by default. This however is controlled by the startup routine, In code we may need to list back-compat roles separately from new roles, but that distinction should be isolated to the startup routines however, and nodes should clearly express the final result of startup in a manner indistinguishable from manually assigned roles. Also if we want to associate a currently disabled by default functionality with a role that needs to default to off.

> > Gus: I actually don't want a coordinator to do more work, I would prefer small focused roles with names that accurately describe their function. In that light, COORDINATOR might be too nebulous. How about AGREGATOR role? (what I was thinking of would better be called a QUERY_ANALYSIS role)
>
> If you want to do specific things like query analysis or query aggregation or bulk indexing etc, all of those can be done on COORDINATOR nodes (as is the case in ElasticSearch). Having tens of of " small focused roles" defined as first class concepts would be confusing to the user. As a remedy to your situation where you want the coordinator role to also do query-analysis for shards, one possible solution is to send such a query to a coordinator node with a parameter like "coordinator.query_analysis=true", and then the coordinator, instead of blindly hitting remote shards, also does some extra work on behalf of the shards.

But what if I DON'T want any special treatment of aggregation? Simplifying things to a level simpler than reality just moves the complexity around. In this case it would move complexit from setup to end-user custom code required to to avoid baked-in functionality.

[Quoted text hidden]

[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 9:58 PM
To: dev@solr.apache.org

> This person has not read the documentation we certainly will provide. Docs would clearly state that the correct way to achieve this is
> Solr1-100: -Dnode.roles=data
> Solr101: -Dnode.roles=overseer

Yes, and that would need the user to restart all 100 nodes in the cluster. Totally avoidable, esp. considering such operations become necessary during production outages.

[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Mon, Nov 1, 2021 at 10:05 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Mon, Nov 1, 2021 at 12:22 PM Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> wrote:

> >   Positive - They denote the existence of a capability
>
> Agree, the SIP already reflects this.
>
> >   Absolute - Absence/Presence binary identification of a capability; no implications, no assumptions
>
> Disagree, we need backcompat handling on nodes running without any roles. There has to be an implicit assumption as to what roles are those nodes assumed to have. My proposal is that only the "data" role be assumed, but not the "overseer" role. For any future roles ("coordinator", "zookeeper" etc.), this decision as to what absence of any role implies should be left to the implementation of that future role. Documentation should reflect clearly about these implicit assumptions.

If you read more closely, my way can provide full back compatibility. To say or imply it doesn't isn't helping. Perhaps you need to re-read?

> > Focused - Do one thing per role
>
> Agree. However, I disagree with ideas where "query analysis" has a role of its own. Where would that lead us to? Separate roles for nodes that do "faceting" or "spell correction" etc.? But anyway, that is for discussion when we add future roles. This is beyond this SIP.

I am not asking you to implement every possible role of course :). As a note I know a company that is running an entire separate cluster to offload and better serve highlighting on a subset of large docs, so YES I think there are people who may want such fine grained control.

> > Accessible - It should be dead simple to determine the members of a role, avoid parsing blobs of json, avoid calculating implications, avoid consulting other resources after listing nodes with the role
>
> Agree. I'm open to any implementation details that make it easy. There should be a reasonable API to return these node roles, with ability to filter by role or filter by node.
>
> > Independent - One role should not require other roles to be present
>
> Do we need to have this hard and fast requirement upfront? There might be situations where this is desirable. I feel we can discuss on a case by case basis whenever a future role is added.
>
> > Persistent - roles should not be lost across reboot
>
> Agree.
>
> > Immutable - roles should not change while the node is running
>
> Agree
>
> > Lively - A node with a capability may not be presently providing that capability.
>
> I don't understand, can you please elaborate?

Specifically imagine the case where there are 100 nodes:
1-100 ==> DATA
101-103 ==> OVERSEER
104-106 ==> ZOOKEEPER

But you won't have 3 overseers... you'll want only one of those to be **providing** overseer functionality and the other two to be **capable**, but not providing (so that if the current overseer goes down a new one can be assigned).

Then you decide you'd ike 5 Zookeepers. You start nodes 107-108 with that role, but you probably want to ensure that zookeepers require some sort of command for them to actually join the zookeeper cluster (i.e. /admin?action=ZKADD&nodes=node107,node18) ... to do that the nodes need to be up. But oh look I typoed 108... we want that to fail... how? because 18 does not have the **capability** to become a zookeeper.
[Quoted text hidden]

[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 10:06 PM
To: dev@solr.apache.org

In this SIP, I'm mainly trying to introduce the broader concept of first class roles and mechanism to assign roles to nodes. I'm then advocating that the implementation of each role be left to decide how to use those roles.

> Roles should never change without restart. 1-100 should not suddenly become ineligible when you start 101.
> Certainly that leaves you with a situation where at least briefly the overseer is on an invalid node. Keep in mind
> none of this takes effect until they upgrade in the first place.

Here, we already have a role called "overseer" which basically designates some nodes to be "preferred overseers". I understand that this could have been implemented differently, but that is a discussion of the past. And in future, we can change the implementation of the overseer role. However, the specific handling of the overseer role falls under the scope of the overseer implementation, and I don't think our broader discussion of this SIP needs to resolve this right here and right now.

> Net new functionally provided by a newly added role, (which zookeeper will be) should be off by default. This however is controlled by the startup routine, In code we may need to list back-compat roles separately from new roles,

This is why I was advocating that 1) we assume the "data" as a default, 2) not assume overseer to be implicitly defined (because of the way overseer role is written today), 3) not assume any future roles to be true by default.

[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Mon, Nov 1, 2021 at 10:08 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Anything can be avoided. The question is what is the long term cost for all users, and all contributors in the future. I'd rather make the system clean and easy to understand and easy to maintain vs this one special case of re-designing your cluster on the fly during an outage.
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 10:09 PM
To: dev@solr.apache.org

> If you read more closely, my way can provide full back compatibility. To say or imply it doesn't isn't helping. Perhaps you need to re-read?

I understand e-mails are frustrating, and I'm trying my best. Please don't be offended, and kindly point me to the exact part you want me to re-read.
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 10:18 PM
To: dev@solr.apache.org

> But I assume that a new feature in 9.x that introduces a new role can also decide for some alternative back-compat logic to support rolling restart if it is needed.

IMHO, having per feature enable/disable flag would be ugly user experience.

Imagine, telling users that for the newly introduced "zookeeper" role, you need to start nodes with:

-Dnodes.role=zookeeper and -Dembedded.zk=true

instead of

-Dnodes.role=zookeeper
(itself enables the functionality needed for that role).
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Mon, Nov 1, 2021 at 10:18 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Very sorry don't mean to sound offended, Frustrated yes offended no :)... the most difficult thing about communication is the illusion it has occurred :)

If you read back just a few emails you'll see where I talk about roles being applied on startup. Boiling it down the idea I'm proposing is that roles required for back compatibility get explicitly added on startup, if not by the user then by the code. This is more flexible than assuming that no role means every role, because then every new feature that has a role will end up on legacy clusters which are also not back compatible.

There are points where I said all roles rather than back compatibility roles because I was thinking about back compatibility specifically, but you can't know that if I don't say that can you :).
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 10:20 PM
To: dev@solr.apache.org

Jan, +1 to using -Dsolr.node.roles instead of -Dnode.roles.
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 10:25 PM
To: dev@solr.apache.org

> Boiling it down the idea I'm proposing is that roles required for back compatibility get explicitly added on startup, if not by the user then by the code. This is more flexible than assuming that no role means every role, because then every new feature that has a role will end up on legacy clusters which are also not back compatible.

+1, I totally agree. I even said so, when I said: "This is why I was advocating that 1) we assume the "data" as a default, 2) not assume overseer to be implicitly defined (because of the way overseer role is written today), 3) not assume any future roles to be true by default."

So, basically, I'm proposing that the "roles required for back compatibility" (that should be explicitly added on startup) be just the ["data"] role, and not the "overseer" role (due to the way overseer role is currently defined, i.e. it is "preferred overseer").
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Mon, Nov 1, 2021 at 10:30 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

So we need to balance first time startup vs while running concerns for sure.

If they haven't read anything and just start up the way they always did, they provide an external zk... no problem 100% back compat
If they want to start a set of nodes for the first time using embedded zk, they should come up but in a down state until an api command enables the zk nodes explicitly

A feature of the ZOOKEEPER role (not of the role system in general) is that a zookeeper node remembers which nodes were providing and automatically allows them to return to providing the service when they start a second time, this will be safe because zookeeper clusters have configuration identifying the machines belonging to the cluster and a maliciously configured node won't be added because the command to put it in the zk cluster configuration hasn't been run.

Why? because we need security on the addition of a zk so that an attacker can't just start their own node and become part of the zk cluster. Also, issues with inadvertent joins due to misconfiguration could be horrendous (I think this pushed elastic away from a bunch of their automated stuff).

Anyhow, I'm now going to take a break and do work :) I hope more people become involved in the discussion.
[Quoted text hidden]
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 1, 2021 at 10:31 PM
To: dev@solr.apache.org

> >> Agree. However, I disagree with ideas where "query analysis" has a role of its own. Where would that lead us to? Separate roles for
> >> nodes that do "faceting" or "spell correction" etc.? But anyway, that is for discussion when we add future roles. This is beyond this SIP.

> I am not asking you to implement every possible role of course :). As a note I know a company that is running an entire separate
> cluster to offload and better serve highlighting on a subset of large docs, so YES I think there are people who may want such fine grained control.

Cool, I think we can discuss adding any additional roles (for highlighting?) on a case by case basis at a later point.

[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Tue, Nov 2, 2021 at 8:51 PM
To: dev@solr.apache.org

Are there any unaddressed outstanding concerns that we should hold up the SIP for?
[Quoted text hidden]

Jason Gerlowski <gerlowskija@gmail.com> Tue, Nov 2, 2021 at 8:57 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Hey Ishan,

I appreciate you writing up the SIP! Here's some notes/questions I
had as I was reading through your writeup and this mail thread.
("----" separators between thoughts, hopefully that helps.)

I'll add my vote to what Jan, Gus, Ilan, and Houston already
suggested: roles should default to "all-on". I see the downsides
you're worried about with that approach (esp. around 'overseer'), but
they may be mitigatable, at least in part.

> [mail thread] User wants this node Solr101 to be a dedicated overseer, but for that to happen, he/she would need to restart all the data nodes with -Dnode.roles=data

Sure, if roles can only be specified at startup. But that may be a
self-imposed constraint.

An API to change a node's roles would remove the need for a restart
and make it easy for users to affect the semantics they want. You
decided you want a dedicated overseer N nodes into your cluster
deployment? Deploy node 'N' with the 'overseer', and toggle the
overseer role off on the remainder.

Now, I understand that you don't want roles to change at runtime, but
I haven't seen you get much into "why", beyond saying "it is very
risky to have nodes change roles while they are up and running." Can
you expand a bit on the risks you're worried about? If you're
explicit about them here maybe someone can think of a clever way to
address them?

> Hence, if those nodes are "assumed to have all roles", then just by virtue of upgrading to this new version, new capabilities will be turned on for the entire cluster, whether or not the user opted for such a capability. This is totally

undesirable.

Obviously "roles" refer to much bigger chunks of functionality than
usual, so in a sense defaulting roles on is scarier. But in a sense
you're describing something that's an inherent part of software
releases. Releases expose new features that are typically on by
default. A new default-on role in 9.1 might hurt a user, but there's
no fundamental difference between that and a change to backups or
replication or whatever in the same release.

I don't mean to belittle the difference in scope - I get your concern.
But IMO this is something to address with good release notes and
documentation. Designing for admins who don't do even cursory
research before an upgrade ties both our hands behind our back as a
project.

> [SIP] Internal representation in ZK ... Implementation details like these can be fleshed out in the PR

IMO this is important enough to flush out as part of the SIP, at least
in broad strokes. It affects backcompat, SolrJ client design, etc.

> [SIP] GET /api/cluster/roles?node=node1

Woohoo - way to include a v2 API definition!

AFAIR, the v2 API has a /nodes path defined - I wonder whether "GET
/nodes/someNode/roles" wouldn't be a more intuitive endpoint for the
"get the roles this node has" functionality. Though I leave that for
your consideration.

Looking forward to your responses and seeing the SIP progress! It's a
really cool, promising idea IMO.

Best,

Jason
[Quoted text hidden]

Noble Paul <noble.paul@gmail.com> Wed, Nov 3, 2021 at 1:25 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Hi Jason,
my responses inline

On Wed, Nov 3, 2021 at 2:28 AM Jason Gerlowski <gerlowskija@gmail.com> wrote:
>
> Hey Ishan,
>
> I appreciate you writing up the SIP! Here's some notes/questions I
> had as I was reading through your writeup and this mail thread.
> ("----" separators between thoughts, hopefully that helps.)
>
> ----

>
> I'll add my vote to what Jan, Gus, Ilan, and Houston already
> suggested: roles should default to "all-on". I see the downsides
> you're worried about with that approach (esp. around 'overseer'), but
> they may be mitigatable, at least in part.
>
> > [mail thread] User wants this node Solr101 to be a dedicated overseer, but for that to happen, he/she would need to restart all the data nodes with -Dnode.roles=data

I don't think that is what the SIP says. If you wish Solr101 to be a
dedicated overseer, you just restart Solr101 with
-Dnode.roles=overseer
>
> Sure, if roles can only be specified at startup. But that may be a
> self-imposed constraint.
>
> An API to change a node's roles would remove the need for a restart
> and make it easy for users to affect the semantics they want. You
> decided you want a dedicated overseer N nodes into your cluster
> deployment? Deploy node 'N' with the 'overseer', and toggle the
> overseer role off on the remainder.

Not really, the role "overseer" actually means "preferred overseer" as
it is implemented today

The fact that node 'N' is "overseer" just means that it is always
first in the queue to become an overseer

>
> Now, I understand that you don't want roles to change at runtime, but
> I haven't seen you get much into "why", beyond saying "it is very
> risky to have nodes change roles while they are up and running." Can
> you expand a bit on the risks you're worried about? If you're
> explicit about them here maybe someone can think of a clever way to
> address them?

There is. potential for confusion. what if users give 2 conflicting
roles on API and startup param

Sometimes a role is better decided at startup of the system instead of
deciding at runtime. Imagine a "role"
that is supposed to not have any data, and it already has data, it's
much easier to handle at startup

>
> > Hence, if those nodes are "assumed to have all roles", then just by virtue of upgrading to this new version, new capabilities will be turned on for the entire cluster, whether or not the user opted for such a capability. This is totally undesirable.

Today we have a role called "overseer" (which actually means a
preferred overseer) We can't make every node a preferred overseer.
 I'm sure future roles will have some such constraints.
>
> Obviously "roles" refer to much bigger chunks of functionality than
> usual, so in a sense defaulting roles on is scarier. But in a sense
> you're describing something that's an inherent part of software
> releases. Releases expose new features that are typically on by
> default. A new default-on role in 9.1 might hurt a user, but there's
> no fundamental difference between that and a change to backups or
> replication or whatever in the same release.
>

> I don't mean to belittle the difference in scope - I get your concern.
> But IMO this is something to address with good release notes and
> documentation. Designing for admins who don't do even cursory
> research before an upgrade ties both our hands behind our back as a
> project.
>
> ----
>
> > [SIP] Internal representation in ZK ... Implementation details like these can be fleshed out in the PR
>
> IMO this is important enough to flush out as part of the SIP, at least
> in broad strokes. It affects backcompat, SolrJ client design, etc.
>
> ----
>
> > [SIP] GET /api/cluster/roles?node=node1
>
> Woohoo - way to include a v2 API definition!
>
> AFAIR, the v2 API has a /nodes path defined - I wonder whether "GET
> /nodes/someNode/roles" wouldn't be a more intuitive endpoint for the
> "get the roles this node has" functionality. Though I leave that for
> your consideration.

The /api/node prefix is meant to be an operation specific to a node .
Even though the role is specified for a node, it is something that
affects the entire cluster.

>
> ----
>
> Looking forward to your responses and seeing the SIP progress! It's a
> really cool, promising idea IMO.

Thanks for the review Jason
[Quoted text hidden]
--

Noble Paul
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Wed, Nov 3, 2021 at 2:50 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

I think there are things not yet accounted for. Time I spent yesterday is biting me today. Pls give a couple days.
[Quoted text hidden]

Timothy Potter <thelabdude@gmail.com> Wed, Nov 3, 2021 at 3:22 AM
Reply-To: dev@solr.apache.org
To: Solr Dev <dev@solr.apache.org>

I'm just not convinced this feature is even needed and the SIP is not
convincing that "There is no proper alternative today."

1) Just b/c Elastic and Vespa have a concept of node roles, doesn't
mean Solr needs this. Also, some of Elastic's roles overlap with
concepts Solr already has in a different form, i.e data_hot sounds
like NRT and data_warm sounds a lot like our Pull Replica Type

2) You can achieve the "coordinator" role with auto-scaling rules
pre-9.x and with the AffinityPlacementPlugin (heck, it even has a node
type built in: .requestNodeSystemProperty(AffinityPlacementConfig.NODE_TYPE_SYSPROP).
Simply build your replica placement rules such that no replicas land
on "coordinator" nodes. And you can route queries using node.sysprop
already using shards.preference.

3) Dedicated overseer role? I thought we were removing the overseer?!?
Also, we already have the ability to run the overseer on specific
nodes w/o a new framework, so this doesn't really convince me we need
a new framework.

4) We will indeed need to decide which nodes host embedded Zookeeper's
but I'd argue that solution hasn't been designed entirely and we
probably don't need a formal node role framework to determine which
nodes host embedded ZKs. Moreover, embedded ZK seems more like a small
cluster thing and anyone running a large cluster will probably have a
dedicated ZK ensemble as they do today. The node role thing seems like
it's intended for large clusters and my gut says few will use embedded
ZK for large clusters.

5) You can also achieve a lot of "node role" functionality in query
routing using the shards.preference parameter.

At the very least, the SIP needs to list specific use cases that
require this feature that are not achievable with the current features
before getting bogged down in the impl. details.

Tim
[Quoted text hidden]

Noble Paul <noble.paul@gmail.com> Wed, Nov 3, 2021 at 3:38 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Wed, Nov 3, 2021 at 8:52 AM Timothy Potter <thelabdude@gmail.com> wrote:
>
> I'm just not convinced this feature is even needed and the SIP is not
> convincing that "There is no proper alternative today."
>
> 1) Just b/c Elastic and Vespa have a concept of node roles, doesn't
> mean Solr needs this. Also, some of Elastic's roles overlap with
> concepts Solr already has in a different form, i.e data_hot sounds
> like NRT and data_warm sounds a lot like our Pull Replica Type

This feature was not built because ES has it. It is built for a specific purpose
>
> 2) You can achieve the "coordinator" role with auto-scaling rules
> pre-9.x and with the AffinityPlacementPlugin (heck, it even has a node
> type built in: .requestNodeSystemProperty(AffinityPlacementConfig.NODE_TYPE_SYSPROP).
> Simply build your replica placement rules such that no replicas land
> on "coordinator" nodes. And you can route queries using node.sysprop
> already using shards.preference.

The objective is somewhat different.

Replica placement is just one small part of it.

When there are 1000's of shards for a collection, and there are 100's
of collections, the distributed query becomes very resource intensive
operation . Our data nodes go out of memory often. We want to ensure

that certain nodes have special capabilities to process requests to
any
collection/shard without hosting those collections or shards
>
> 3) Dedicated overseer role? I thought we were removing the overseer?!?
> Also, we already have the ability to run the overseer on specific
> nodes w/o a new framework, so this doesn't really convince me we need
> a new framework.
>
> 4) We will indeed need to decide which nodes host embedded Zookeeper's
> but I'd argue that solution hasn't been designed entirely and we
> probably don't need a formal node role framework to determine which
> nodes host embedded ZKs. Moreover, embedded ZK seems more like a small
> cluster thing and anyone running a large cluster will probably have a
> dedicated ZK ensemble as they do today. The node role thing seems like
> it's intended for large clusters and my gut says few will use embedded
> ZK for large clusters.
>
> 5) You can also achieve a lot of "node role" functionality in query
> routing using the shards.preference parameter.

Routing is not a problem for us
>
> At the very least, the SIP needs to list specific use cases that
> require this feature that are not achievable with the current features
> before getting bogged down in the impl. details.

Sure TIm, We can document our concrete use cases
[Quoted text hidden]
--

Noble Paul
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Wed, Nov 3, 2021 at 4:42 AM
To: dev@solr.apache.org

Hi Tim,
Here are my responses inline.

On Wed, Nov 3, 2021 at 3:22 AM Timothy Potter <thelabdude@gmail.com> wrote:
> I'm just not convinced this feature is even needed and the SIP is not
> convincing that "There is no proper alternative today."

There are no proper alternatives today, just hacks. On 8x, we have two different deprecated frameworks to stop nodes from being placed on a node (1. rule based replica placement, 2. autoscaling framework). On 9x, we have a new autoscaling framework, which I don't even think is fully implemented. And, there's definitely no way to have a node act as a query coordinator without having data on it.

> 1) Just b/c Elastic and Vespa have a concept of node roles, doesn't
> mean Solr needs this.

Solr needs this. Elastic has such concepts is a coincidence, and also means we have an opportunity to catch up with them; they have these concepts for a reason.

> Also, some of Elastic's roles overlap with
> concepts Solr already has in a different form, i.e data_hot sounds
> like NRT and data_warm sounds a lot like our Pull Replica Type

I think that is beyond the scope of this SIP.

> 2) You can achieve the "coordinator" role with auto-scaling rules
> pre-9.x and with the AffinityPlacementPlugin (heck, it even has a node
> type built in: .requestNodeSystemProperty(AffinityPlacementConfig.NODE_TYPE_SYSPROP).
> Simply build your replica placement rules such that no replicas land
> on "coordinator" nodes. And you can route queries using node.sysprop
> already using shards.preference.

I think you missed the whole point of the query coordinator. Please refer to this https://issues.apache.org/jira/browse/SOLR-15715.
Let me summarize the main difference between what (I think) you refer to and what is proposed in SOLR-15715.

With your suggestion, we'll have a node that doesn't host any replicas. And you suggest queries landing on such nodes be routed using shards.preference? Well, in such a case, these queries will be forwarded/proxied to a random node hosting a replica of the collection and that node then acts as the coordinator. This situation is no better than sending the query directly to that particular node.

What is proposed in SOLR-15715 is a query aggregation functionality. There will be pseudo replicas (aware of the configset) on this coordinator node that handle the request themselves, sends shard requests to data hosting replicas, collects responses and merges them, and sends back to the user. This merge step is usually extremely memory intensive, and it would be good to serve these off stateless nodes (that host no data).

> 3) Dedicated overseer role? I thought we were removing the overseer?!?
> Also, we already have the ability to run the overseer on specific
> nodes w/o a new framework, so this doesn't really convince me we need
> a new framework.

There's absolutely no change proposed to the "overseer" role. What users need on production clusters are nodes dedicated for overseer operations, and for that the current "overseer" role suffices, together with some functionality to not place replicas on such nodes.

> 4) We will indeed need to decide which nodes host embedded Zookeeper's
> but I'd argue that solution hasn't been designed entirely and we
> probably don't need a formal node role framework to determine which
> nodes host embedded ZKs. Moreover, embedded ZK seems more like a small
> cluster thing and anyone running a large cluster will probably have a
> dedicated ZK ensemble as they do today. The node role thing seems like
> it's intended for large clusters and my gut says few will use embedded
> ZK for large clusters.

This SIP is not the right place for this discussion. There's a separate SIP for this.

> 5) You can also achieve a lot of "node role" functionality in query
> routing using the shards.preference parameter.

That doesn't solve the purpose behind https://issues.apache.org/jira/browse/SOLR-15715.

> At the very least, the SIP needs to list specific use cases that
> require this feature that are not achievable with the current features
> before getting bogged down in the impl. details.

The coordinator role is the biggest motivation for introducing the concept of roles. However, in addition to what is proposed in SOLR-15715, a coordinator node can later on also be used as a node for users to run streaming expressions on, do bulk indexing on (impl details for this to come later, don't want distraction here).
[Quoted text hidden]

Timothy Potter <thelabdude@gmail.com> Wed, Nov 3, 2021 at 5:16 AM
Reply-To: dev@solr.apache.org
To: Solr Dev <dev@solr.apache.org>

I'm not missing the point of the query coordinator, but I actually didn't realize that an empty Solr node would forward the top-level request onward instead of just being the query controller itself? That actually seems like a bug vs. a feature, IMO any node that receives the top-level query should just be the coordinator, what stops it?

Anyway, it sounds to me like you guys have your minds made up regardless of feedback.

Btw ~ I only mentioned the Zookeeper part b/c it's in your SIP as a specific role, not sure why you took that as me wanting to discuss the embedded ZK in your SIP?

On Tue, Nov 2, 2021 at 5:13 PM Ishan Chattopadhyaya
[Quoted text hidden]

Noble Paul <noble.paul@gmail.com> Wed, Nov 3, 2021 at 5:19 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On Wed, Nov 3, 2021, 10:46 AM Timothy Potter <thelabdude@gmail.com> wrote:
> I'm not missing the point of the query coordinator, but I actually didn't realize that an empty Solr node would forward the top-level request onward instead of just being the query controller itself? That actually seems like a bug vs. a feature, IMO any node that receives the top-level query should just be the coordinator, what stops it?

To process a request there should be a core that uses the same configset as the requested collection.
[Quoted text hidden]

Timothy Potter <thelabdude@gmail.com> Wed, Nov 3, 2021 at 5:29 AM
Reply-To: dev@solr.apache.org
To: Solr Dev <dev@solr.apache.org>

As opposed to what? Looking up the configset for the addressed collection and pulling whatever information it needs from cached data. I'm sure there are some nuances but I hardly think you need a node role framework to deal with determine the unique key field to do scatter gather on an empty node when you have easy access to collection metadata.

Doesn't seem like a hard thing to overcome to me.
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Wed, Nov 3, 2021 at 5:30 AM
To: dev@solr.apache.org

> Btw ~ I only mentioned the Zookeeper part b/c it's in your SIP as a specific role

Sorry for the confusion, I mentioned it in the SIP because Jan wrote in the JIRA that it can be a potential role. However, specific details about that feel out of scope for this discussion here, which is only about Solr having the broad concept of node roles (and the mechanism for users to assign them).
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Wed, Nov 3, 2021 at 5:39 AM
To: dev@solr.apache.org

If we were to tell users how to do "scatter gather on an empty node", *how exactly* would you recommend users have an empty node to begin with? Wouldn't you say something like "for 8x you can do this (rule based replica placement) or do that (autoscaling), but for 9x you do this new thing". Having a node that doesn't have a data role seems like a consistent and an elegant way for users to invoke such a functionality and also easily relate to a broad concept, without having to deal with autoscaling frameworks of the ancient past, medieval past or the future.
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Wed, Nov 3, 2021 at 5:43 AM
To: dev@solr.apache.org

Also, in a cluster where new collections/shards/replicas are continuously added all the time, it would be pretty awkward to start a node (in regular mode), briefly have it become eligible for replica assignment, then invoking a replica placement rule/autoscaling policy for that node to not place replicas on it. Instead, starting a node with a defined role (as a startup param) precludes that brief period of eligibility for replica placement on such a node.
[Quoted text hidden]

Timothy Potter <thelabdude@gmail.com> Wed, Nov 3, 2021 at 5:56 AM
Reply-To: dev@solr.apache.org
To: Solr Dev <dev@solr.apache.org>

One last thought on this for me ... I think it would be beneficial for
the SIP to address how this new feature will work with the existing
shards.preference solution and affinity based placement plugin.
Moreover, your pseudo-replica solution sounds like a new replica type
vs. a node level thing. The placement strategy can place replicas
based on replica type and node type (just a system property), so
please address why you can't achieve a query coordinator behavior with
a new replica type + improvements to the Affinity placement plugin?

Cheers,
Tim

On Tue, Nov 2, 2021 at 6:14 PM Ishan Chattopadhyaya
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Wed, Nov 3, 2021 at 7:26 AM
To: dev@solr.apache.org

Answers inline below.

On Wed, Nov 3, 2021 at 5:56 AM Timothy Potter <thelabdude@gmail.com> wrote:
> One last thought on this for me ... I think it would be beneficial for
> the SIP to address how this new feature will work with the existing
> shards.preference solution and affinity based placement plugin.

I was more inclined to keep this SIP focused on broad concept of roles, and any upcoming roles (coordinator role, along with that pseudo-core functionality) to be described in their own issue (e.g. SOLR-15715).

> Moreover, your pseudo-replica solution sounds like a new replica type
> vs. a node level thing.

I misspoke when I called it "pseudo replica", it is actually a "pseudo core". Replicas are shard level concepts, but such a pseudo core that we plan to introduce will pertain to one or more collections. Imagine collection1 has shard1 and shard2, there will be a single pseudo core for collection1 (we haven't decided on the prefix of this pseudo core yet, but a candidate can be ".collection1_coordinator"). Replica type won't fit this mental model here. We can discuss this more in the SOLR-15715 issue.

> The placement strategy can place replicas
> based on replica type and node type (just a system property), so
> please address why you can't achieve a query coordinator behavior with
> a new replica type + improvements to the Affinity placement plugin?

To put down my thoughts on why Affinity placement plugin won't work for the purpose of ensuring that we have nodes that host no data on it:
1. We want the ability to have nodes with no data on it as a first class concept for users. Hence, if the Affinity placement plugin is used for that purpose, users won't be able to switch out that plugin and use anything of their own. Currently, IIUC, there's not way for users to use multiple placement plugins.
2. Nodes that shouldn't host any replica on it are generally ephemeral in nature; many of them may join the cluster, they may go away. If such a node joins the cluster, they immediately become eligible for replica placement, before even the sysadmin is able to assign an affinity placement configuration for that node. This is a problem.

> Cheers,
> Tim

Thanks for your thoughts and feedback, I think it will help us put together the document with more insights into our design choices.

Regards,
Ishan
[Quoted text hidden]

Ilan Ginzburg <ilansolr@gmail.com> Wed, Nov 3, 2021 at 7:30 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

I think if we have the new "pseudo core" abstraction (I like it! Will it really be a core with an index on disk or some new abstraction only tracked in ZK and in memory?) to play the role of coordinator, then we have all we need with the affinity placement plugin framework for a data free coordinator node implementation.
It is easy to use system properties to exclude nodes from receiving replicas using the placement plugins, a minor change in the Affinity Placement Plugin. Such nodes will not receive any replicas by the placement plugin not even at startup (the system property will be assigned at startup so no manual intervention needed).

It will not work if switching to another placement plugin, unless that other plugin reimplements that (simple) aspect. Is that an issue?

Ilan

[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Wed, Nov 3, 2021 at 10:18 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Still not having enough time to go back and comment on all of this, but scanning along I'm wondering if we can avoid using more than one word for an abstraction layer. In cloud mode at least we have clusters/nodes/collections/shards/replicas.... throwing the word core in there (even with pseudo in front of it) seems like it just creates confusion, especially since in code a solrcore is instantiated per replica, and you are talking of a pseudo-core relating to a collection and shard1/shard2. I think pseudo-collection would cause less confusion?

[Quoted text hidden]

Michael Gibney <michael@michaelgibney.net> Wed, Nov 3, 2021 at 10:20 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

>I actually didn't realize that an empty Solr node would forward the top-level
>request onward instead of just being the query controller itself? That
>actually seems like a bug vs. a feature, IMO any node that receives
>the top-level query should just be the coordinator, what stops it?

+1 to Tim's statement quoted above; unless I'm missing something, this feels like an issue that should be addressed regardless of this SIP. (perhaps it would be addressed incidentally by this SIP? -- in any event the current situation seems to not make sense. As Tim points out, the relevant configs should in principle be accessible from ZK whether or not there's a core for a given collection on a given node).

Considering the above, and especially given Ishan that you say "The coordinator role is the biggest motivation for introducing the concept of roles", while reading the SIP I found myself wishing for a fuller enumeration of use cases, and a more sympathetic characterization of alternatives (existing alternatives, and perhaps, as with the above "proxy request" issue, simpler-but-not-yet-implemented alternatives).

Combining questions about use cases with questions about alternatives: assuming that 9.x autoscaling can indeed be reliably used to stop replicas from being placed on nodes, how close would addressing the orthogonal "proxy request" issue come to addressing potential use cases?

Michael

On Wed, Nov 3, 2021 at 10:00 AM Ilan Ginzburg <ilansolr@gmail.com> wrote:
[Quoted text hidden]

Noble Paul <noble.paul@gmail.com> Thu, Nov 4, 2021 at 1:02 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Michael

We explored all options to before arriving at this solution. Ishan has already explained why Tim's suggestions have their shortcomings when it comes to user experience.

Thanks
[Quoted text hidden]

Noble Paul <noble.paul@gmail.com> Thu, Nov 4, 2021 at 8:06 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Thanks everyone for participating in the discussion. I have gone through all your valuable inputs and these are my suggestions

Requirements?

1. Users should be able to designate a node with some role by starting (say `-Dnode.roles=coordinator`)
2. This node should be able to perform a certain behavior

3. Replica placement should be aware of this and may choose to place or not place a replica in this node
4. Any client should be able to query any node in the cluster to get a list of nodes with a specified role or get the roles of a given node

Implementation?

Here is how we could implement each of the requirements:

1. We could theoretically use a well known system property and
2. The actual behavior will have to be implemented in both 8.x or 9.x
3. Placement of replicas
    a. It's not possible to do this in 8.x
    b. In 9.x, replica placement plugin can be internally used to ensure proper placement of replicas in the roles feature.

        i. It can't be done with the current design as users cannot chain multiple placement plugins or user has to build a custom placement plugin of his own
        ii. There is no standard UX to achieve this. It will be a recipe (start nodes with this property and create these rules etc, etc). This is awkward & error prone, as compared to saying "start a node with `coordinator` role" and Solr will take care of it.
4. There will be a new API endpoint to publish this information in 8.x and 9.x. This end point is important to make this feature usable

Conclusion

1. With a roles feature, we can achieve the objectives in a user friendly and intuitive way
2. The user interface can be consistent across 8.x and 9.x even though 9.x can use the placement plugin internally
3. The actual roles definition will be same across 8.x and 9.x

[Quoted text hidden]
--

Noble Paul

Jan Høydahl <jan.asf@cominvent.com> Thu, Nov 4, 2021 at 2:58 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Let's do ourself a service and target 9.0 for roles. It's too late to plan new features into 8.x.

I don't understand the urgency either. I can get that certain Solr users would wish for such a feature "yesterday" but that cannot drive our decisions on what version to target for features. When targeting 9.0, all upgrade or back-compat worries will need to be baked into the feature itself, so that there is either code support or good documentation for how to start using roles after upgrading a cluster to 9.0. Perhaps there must be a temporary cluster-property in 9.0 "enableRoles=false" that can be set, even if all 9.0 nodes are given roles on startup. Then, initially after the upgrade, the cluster behaves as it did in 8.x. Then once you are ready to enforce roles, you can flip the cluster property, and placement and routing starts using roles. In 10.0 that property can then go away.

When it comes to placement plugins, we can document in that they MUST respect certain node roles (at least the data role), and treat it as a bug if they don't.

Jan
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Thu, Nov 4, 2021 at 3:46 PM
To: dev@solr.apache.org

We are targeting this for 9x. However, it can't hurt anyone to also backport to 8x (if there's enough time and interest). For large clusters, the separation of data and querying can potentially be a game changer.

Compatability worries are not just for those who upgrade, but also about defaults for those who haven't opted in to roles and want to do so.

On Thu, 4 Nov, 2021, 2:58 pm Jan Høydahl, <jan.asf@cominvent.com> wrote:

> Let's do ourself a service and target 9.0 for roles. It's too late to plan new features into 8.x.
>
> I don't understand the urgency either. I can get that certain Solr users would wish for such a feature "yesterday" but that cannot drive our decisions on what version to target for features.

There's no urgency, except that the performance numbers we have seen are so impressive that we're excited to upstream this as soon as possible for the sake of our users.

> When targeting 9.0, all upgrade or back-compat worries will need to be baked into the feature itself, so that there is either code support or good documentation for how to start using roles after upgrading a cluster to 9.0. Perhaps there must be a temporary cluster-property in 9.0 "enableRoles=false" that can be set, even if all 9.0 nodes are given roles on startup. Then, initially after the upgrade, the cluster behaves as it did in 8.x.

We advocate for a consistent way to assign roles that works across major versions not because of upgrade compatability etc, but because of ease of use and reducing scope for user to learn new ways. Internally, under the hood, placement plugin can potentially be used on 9x; but the support for roles should be first class citizen, not dependent on the correct functioning of a plugged in component (into the new autoscaling framework).

> Then once you are ready to enforce roles, you can flip the cluster property, and placement and routing starts using roles. In 10.0 that property can then go away.

> When it comes to placement plugins, we can document in that they MUST respect certain node roles (at least the data role), and treat it as a bug if they don't.

+1
[Quoted text hidden]

Noble Paul <noble.paul@gmail.com> Thu, Nov 4, 2021 at 3:49 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

None of the design is dictated by the version in which we implement this. The SIP is mostly about the "what", "why" and the UX

I don't have any affinity to any particular version. This is definitely going to happen in 9.x. Even if it is built in 9.x we will have to build and support all versions of solr we use internally. When we eventually upgrade from our current version to a 9.x version , it has to be backward compatible.The choice of whether this is available for public consumption as a branch/release is up for debate
[Quoted text hidden]

Ilan Ginzburg <ilansolr@gmail.com> Thu, Nov 4, 2021 at 3:56 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

A lot of the value of this SIP relies on the pseudo-core thing (because placing on specific nodes is achievable today, Overseer role already exists). Roles as described without the coordinator concept are just another way to do things already possible today (with a very minor update on the Affinity placement plugin - it might even support it right away actually, didn't check).
Maybe "pseudo core" should go in first and condition the rest of the work? It feels like a bigger chunk with more challenging integration issues (routing, new concept in the collection/shard/replica hierarchy).

Ilan
[Quoted text hidden]

Noble Paul <noble.paul@gmail.com> Thu, Nov 4, 2021 at 4:32 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

The placement part of roles feature may use placement plugin API .

The implementation is not what we're discussing here. We need a consistent story for the user when it comes to roles. This discussion is about the UX rather than the impl.

Most of our discussions are about how we should implement it

[Quoted text hidden]

Ilan Ginzburg <ilansolr@gmail.com> Thu, Nov 4, 2021 at 8:30 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

I was noting that the real value of the proposal (real value = being able to do things that are currently impossible with Solr) was due to an independent concept of a coordinator "core", and that if we had this (currently does not exist in Solr but apparently you do have it on a fork), we can achieve most/all of what the SIP proposes with existing means, i.e. without roles. Maybe in a less flexible/user friendly way, maybe not (given the details of rolling out roles are still fuzzy).
And if we don't have the concept of coordinator core, then the roles by themselves do not allow much more than what is already achievable by other means.

Ilan
[Quoted text hidden]

Timothy Potter <thelabdude@gmail.com> Thu, Nov 4, 2021 at 8:46 PM
Reply-To: dev@solr.apache.org
To: Solr Dev <dev@solr.apache.org>

+1 to what Ilan said, that was my main point all along as well ;-)
There is merit in using what's already there vs. introducing some new
concept that might be useful for future use cases, but for now is just
useful for the query coordinator concept.

Also, I don't like the pseudo-core / -collection nomenclature ...
let's call it what it is: stateless query controller (the fact that it
uses a core in the impl to achieve that functionality is meaningless
to users ~ don't leak the impl into the naming / interface ;-)

Tim
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Thu, Nov 4, 2021 at 11:17 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

So one of the reasons my response to this proposal was that I like the idea of roles, but not some of the details of the sip, is that while a lot of things "are possible today" I see this feature as imparting a coherent organization to the availability of those features. Seems like it would be more user friendly.

+1 to Stateless Query Controller vs pseudo anything :)

-Gus
[Quoted text hidden]

Noble Paul <noble.paul@gmail.com> Fri, Nov 5, 2021 at 1:30 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Yes Ilan
The coordinator is the first compelling usecase. The roles is the UX and it's a very simple piece. The real work is coming as a separate PR.

Roles can be achieved in a clumsy way today. It's unintuitive and we don't want to make the user to jump through the hoops.

I'll open a PR and you be the judge on the simplicity of this SIP. It's not going to have any major impact on any component of Solr.

[Quoted text hidden]

Noble Paul <noble.paul@gmail.com> Fri, Nov 5, 2021 at 2:21 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

The SIP can be boiled down to the following

* **Tag a node with a label (role) using a system property**
* **Use the placement plugin to whitelist/block list certain nodes**
* **Publish the roles through an API**

That's it

If you wish to add a new role, use the same concept.

Period
[Quoted text hidden]

Shawn Heisey <elyograg@elyograg.org> Fri, Nov 5, 2021 at 3:33 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

On 11/4/21 2:51 PM, Noble Paul wrote:
> The SIP can be boiled down to the following
>
> * *Tag a node with a label (role) using a system property*
> ** Use the placement plugin to whitelist/block list certain nodes*
> ** Publish the roles through an API*

In general, for Solr, do we like the idea of having things controlled by system properties?

I would think solr.xml would be the right place to configure this, except that people can and probably do put solr.xml in zookeeper, which would mean every system would have the SAME solr.xml, and we're back to system properties as a way to customize solr.xml on each system.

I have never used system properties to configure Solr. When I customize the config, I will often remove property substitutions from it and go with explicit settings. My general opinion about system properties is that if they're going to be used, they should DIRECTLY configure the application, not be sent in via property substitution in a config file. I've never liked the way our default configs use that paradigm. It means you cannot look at the config and know exactly how things are configured, without finding out whether system properties have been set.

What color do others think that bikeshed should be painted?

Thanks,
Shawn
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Fri, Nov 5, 2021 at 7:34 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Agree better to something other than sysprops. an arg in the start script would be friendlier than -D props which generally are irritatingly verbose and expose too much implementation.

We lack a config file per level. solr.xml does double duty as global and per-node depending on how it's used (zk or filesystem).

Config file names are confusing too. Our file names are legacy of non-cloud mode I think, and we really should at some point (10.x?) rework configs to be cluster.xml, node.xml, collection.xml (formerly solrconfig.xml) and schema.xml (and maybe support something other than xml, but that's not nearly as important as clarity in naming, and having features)

But this is all straying way off topic and should have its own SIP if someone seems to have time for it :)
[Quoted text hidden]
[Quoted text hidden]

Jan Høydahl <jan.asf@cominvent.com> Sat, Nov 6, 2021 at 2:49 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Thinking of these roles as labels, I think sysProps and envVars are the two universal methods, and nothing wrong with that.
I keep trying to think cloud native and container, so having excellent 1st class support for env.vars for such configs is a priority to me.
Most tools, CI-environments etc have built-in support for env.vars, and so it makes sense to me.

See https://cwiki.apache.org/confluence/display/SOLR/SIP-11+Uniform+cluster-level+configuration+API for some interesting ideas around cluster/node level config.

See
[Quoted text hidden]

Houston Putman <houstonputman@gmail.com> Thu, Nov 11, 2021 at 9:37 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

I agree with Jan, when thinking about making Solr as cloud friendly as possible EnvVars and (to a lesser extent) sysProps are much preferable than having a setting in the solr.xml.
This is because it's easier to customize EnvVars per-node, while customizing a config file is much harder, as those tend to be static and shared across a whole environment.

Also thanks for linking that SIP Jan, very applicable.

- Houston
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Thu, Nov 11, 2021 at 10:13 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

I guess all I mean is that it shouldn't be only sysprops. Enabling sysprops, Env vars etc seems fine but we need to clearly document precedence among any/all options. What is convenient varies from case to case and in a perfect world what I'd like to see is full support across each style (files, zk, props, env vars) with consistent and obvious naming and well documented resolution order.

What I don't like is a little bit of env vars for some stuff, props for others, files for yet more stuff and some unclear aggregation of that showing up in zk... (or maybe some of it not showing up anywhere code could check it...)
[Quoted text hidden]

Jan Høydahl <jan.asf@cominvent.com> Fri, Nov 12, 2021 at 12:39 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

+1 to a roundup of env and props across the board. I think SIP 11 is on the track of something. But can be done independent of this.

Jan Høydahl

> 11. nov. 2021 kl. 17:44 skrev Gus Heck <gus.heck@gmail.com>:

[Quoted text hidden]

Eric Pugh <epugh@opensourceconnections.com> Fri, Nov 12, 2021 at 12:41 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Agreed!

I've noticed that in the Play Framework, you can configure everything via a property based configuration file, however it makes it easy to override the property file via another one, or via an ENV variables:

```
db.default.username="smui"
db.default.username=${?SMUI_DB_USER}
```

Which turns out to be very liberating!
[Quoted text hidden]

____

Eric Pugh | Founder & CEO | OpenSource Connections, LLC | 434.466.1467 | http://www.opensourceconnections.com | My Free/Busy
Co-Author: Apache Solr Enterprise Search Server, 3rd Ed
This e-mail and all contents, including attachments, is considered to be Company Confidential unless explicitly stated otherwise, regardless of whether attachments are marked as such.

Ilan Ginzburg <ilansolr@gmail.com> Fri, Nov 12, 2021 at 2:19 AM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Houston made a very valid comment back then on the placement plugin support of environment variables (dropped as a consequence).

https://issues.apache.org/jira/browse/SOLR-15019?page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel&focusedCommentId=17286680#comment-17286680

It could be possible to unintentionally leak node data that should be kept secret if Solr is allowed to freely access (random?) environment variables as part of configuration.

Something to keep in mind.

Ilan
[Quoted text hidden]

Gus Heck <gus.heck@gmail.com> Fri, Nov 12, 2021 at 11:37 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Yeah we should only be looking for and only be reporting (if we choose to report to the user) a specific set of env variables. Anything else should be ignored.Should be an enum or constants somewhere listing what solr cares about, and we should ignore or be blind to anything else.

Perhaps we'd like to have a ConfigParams (or whatever) enum that has methods returning the env, sysprop, bin/solr arg, configFile and zkLocation that can be used to provide each possible configuration option (for things that are single value or short list, obviously an entire schema probably would not be setable by sysprop :))?

The return type of those methods could be Optional<>() since we neither have all of those for everything any time soon, and not all of them will make sense in all cases.

zkLocation is a bit tricky and nebulous since it's probably a zk path and a JSON path or Xpath combined and relative to the chroot which itself is a potential config param, some stuff to think through there.
[Quoted text hidden]

Ishan Chattopadhyaya <ichattopadhyaya@gmail.com> Mon, Nov 15, 2021 at 9:58 AM
To: dev@solr.apache.org

Thanks to everyone for the feedback.

Here's an attempt to summarize broad topics discussed.

No negative roles
Everyone agree

Roles on/off by default?
Jason+(Ilan,Houston?): All roles to be on by default
Gus,Ishan,Noble: Only those roles to be on by default that are needed for backcompat

Which branch to target?
Jan,Ishan,Noble: New feature to be added to 9x branch

Need for roles?
Tim: new concept of nodes unnecessary since everything that's proposed can be achieved using changes to new autoscaling framework and replica placement plugins.
Ishan,Noble: A first class concept of roles is important so that this functionality is expected to work, irrespective of whatever custom placement plugins users deploy (since placement plugins don't support chaining).

Roles for collections?
Ilan: Role aware collections
Ishan: This can be implemented separately later using node roles and placement plugins.

Configuration
Sysprops vs solr.xml+sysprops vs envvars:
Shawn: Solr.xml and/or envvars
Houston,Ilan: Sysprops and/or envvars
Ishan,Noble: Sysprops
Jan: SIP-11

Outstanding issues
Shawn: Color of the bikeshed ;-)

Please let me know if I missed something here. If there are no further strong objections, we can proceed to the implementation phase. There's already a draft/WIP PR in the works: https://github.com/apache/solr/pull/403

Thanks,
Ishan
[Quoted text hidden]

Jan Høydahl <jan.asf@cominvent.com> Mon, Nov 15, 2021 at 6:53 PM
Reply-To: dev@solr.apache.org
To: dev@solr.apache.org

Thanks for trying to summarize and drive the work Ishan.

I'd like to add

Scope of SIP
Ishan: Role API and config

Jan: Role API, config, and impact of one real role e.g. the "data" role, to examplify and justify the role infrastructure

According to SIP process the next step is not implementation, but rather to iterate the SIP text to something you believe would pass a vote. It's hard to stitch together all these email and mini summaries into a meaningful whole.

Jan
[Quoted text hidden]