Stretching WSP

With some elastic….

Noel Power
SUSE/Samba team
noel.power@suse.com

Agenda

- **WSP recap**
  - What is WSP
  - Introduction to the protocol
  - The WSP server, client & tools

Agenda

- **ElasticSearch**
  - What is it
  - Some investigation into using elasticsearch
    - what I tried
    - what I found out
  - What’s promising
  - What’s not promising

WSP recap

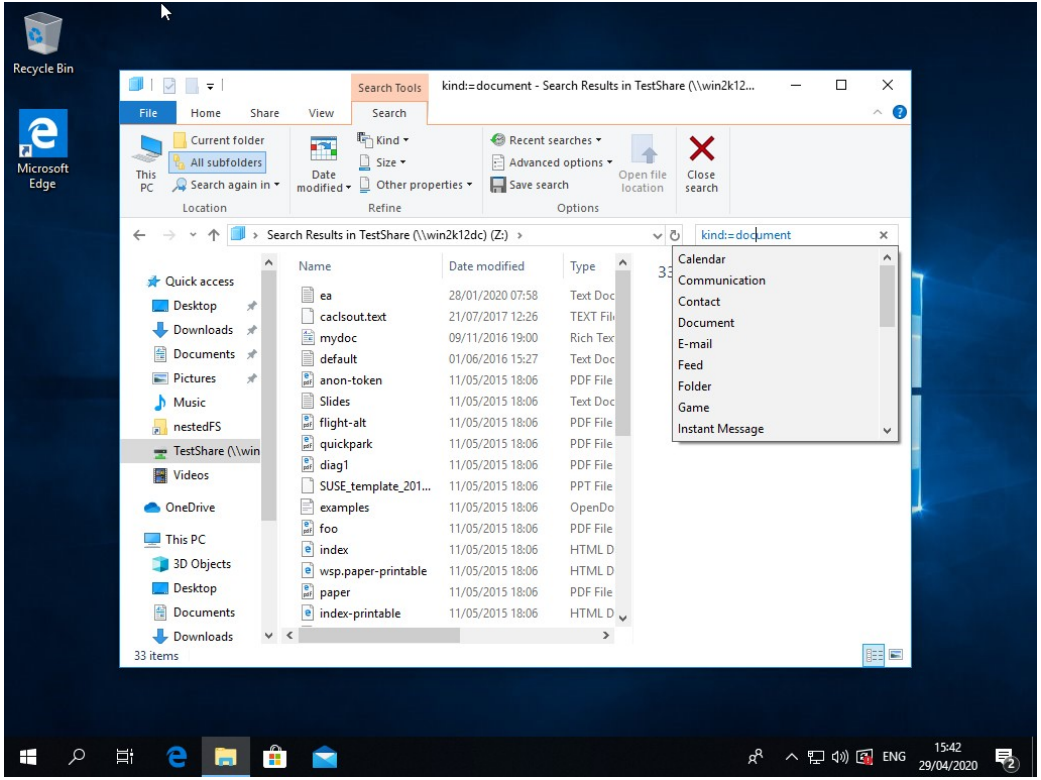
History

- **Started playing with implementing Windows Search protocol as a hackweek project at Suse. This was when I first started working on samba and my intention was to use the experience to get to grips with some of the internals of the samba code e.g. talloc, tevent etc.**
- **Additionally I wanted to get familiar with SMB so I randomly picked a protocol specification that seemed interesting to see if I could implement ‘something’. Thus my interest in implementing WSP was born.**
  - Note: This is still a hobby project for me, to be honest I spend very little time on it. Mostly lots of time passes before I attempt to return to it but inevitably I end up dealing with the fallout from trying to get it to run again on master.

Windows Search

- **What is Windows Search**
  - Basically Windows Search is a desktop platform that indexes common content and allows users to find, manage and organize that data.
  - Windows search protocol is the mechanism that allows client to perform search queries against a remote server hosting the Windows Search Service.
  - Windows users will be familiar with the ability to search for example different types of files from the search UI.

Windows Search UI



WSP protocol

- **Allows a client to issue queries to a server hosting the Windows Search Service**
- **Protocol is primarily intended to be used for full-text queries**
- **Sits on top of the SMB pipe protocol**
- **Heart of the protocol is the query request which includes**
  - Rowset properties like the catalog name and configuration information
  - Restrictions to specify which document are to be included and/or excluded from the search results
  - Order in which the search results are returned

WSP protocol

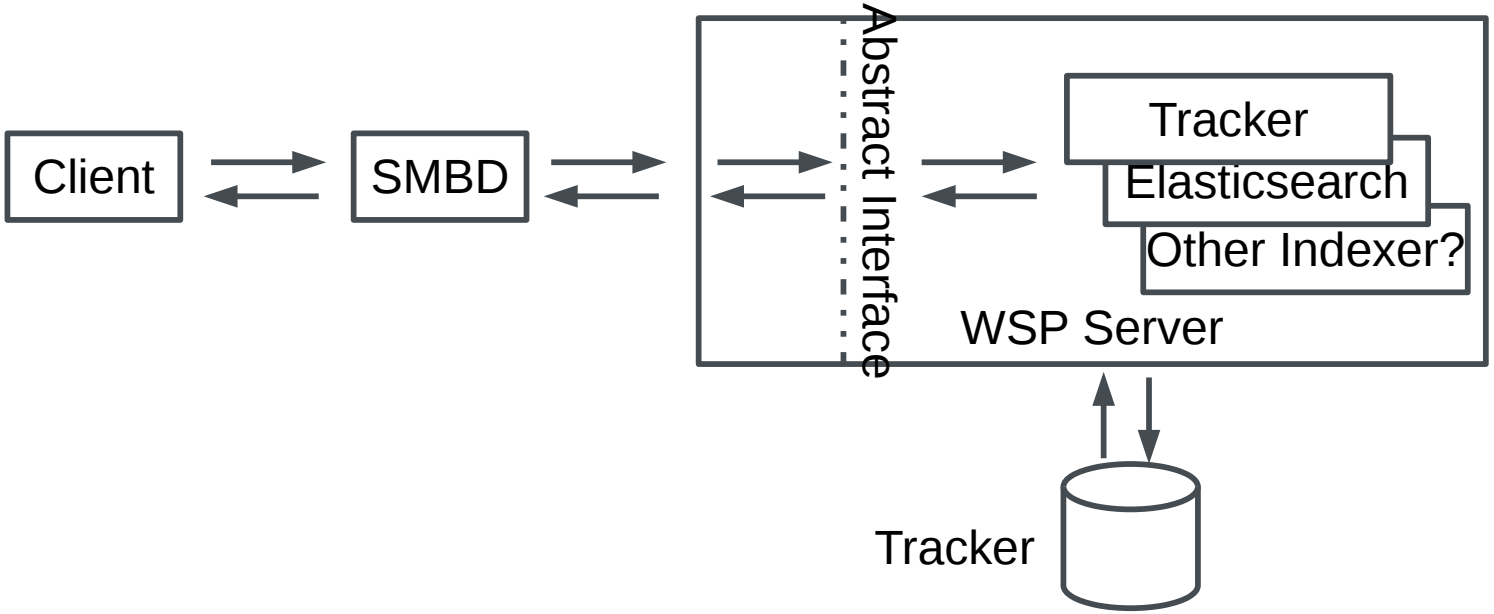
- After the query is initiated the client typically requests status information about the query (is it complete?, how many results etc.)
- Then client specifies the columns to be returned for the query
- The client can then request results (columns) to be returned
- Results that are too big can be requested later (we don’t support that) More about this later
- Protocol is mostly simple request/response.

The implementation

WSP implementation

- **Not upstream yet**
- **Consists of**
  - Server (part of smbd)
  - Client (wspsearch)
  - Translator (wsp-to-sparql)

WSP Implementation



WSP Server implementation

- **Currently supports gnome tracker**
- **Uses tevent glib integration to communicate with tracker using native glib api(s)**
- **Supports basic queries you would get expect from the windows file explorer search UI e.g.**
  - Supported ‘Kind’(s) are Contact, Document, Email, Feed, Folder, Music, Picture, Program
  - Not Supported are Calendar, Communication, Game, InstantMessage, Journal, Link, Note, Movie, RecordedTV, SearchFolder, Task, WebHistory

WSP to Tracker query conversion

- **A WSP query is consists of a command tree of restrictions and sort orders that specify which documents are to be included (or excluded) from the search results.**
- **While WSP defines many restriction types that can be used in the command tree we are only able to convert a small subset.**
  - Luckily it seems that this is enough to satisfy the basic queries from the windows explorer search ui
- **The binary Query message is converted into tracker sparql (sparql is the query language that tracker uses)**

Conversion to Tracker sparql

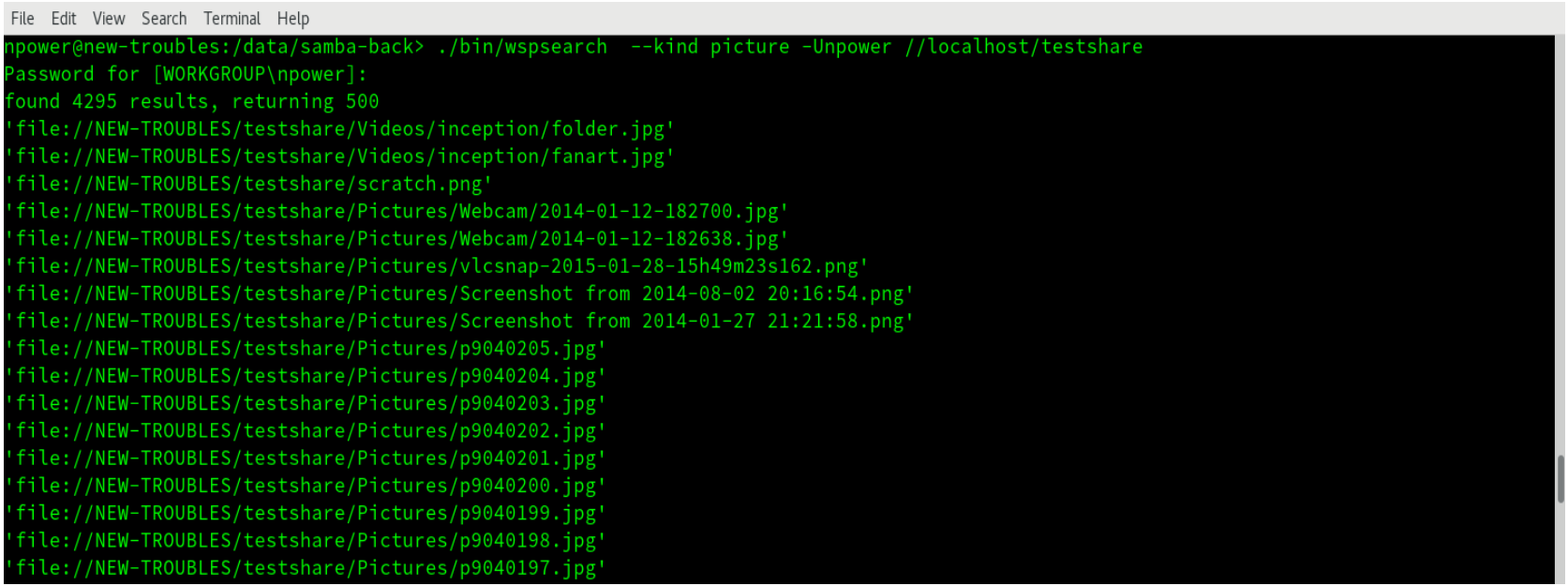
- **We start with a template**
  - Select nie:url(?u) $columns WHERE {?u nie:url ?url FILTER($restrictions)} ORDER BY $sort)
- **$columns, $restrictions & optionally $sort are generated from the information in the query message**
  - Any restriction in the binary query command tree that cannot be converted is ignored

WSP Client

- **There is a simple cli search client called wspsearch**
- **It support two modes**
  - Simple
    - Which can search for the supported ‘kind’(s) (e.g. Picture, Document etc.) in addition to an optional ‘phrase’ which is used as in a full-text-search which just returns the url of matching documents
  - Complex
    - Which can accept an AQS-like (advanced query syntax) query. This allows dynamic and expressive queries to be built (great for testing the server/protocol and useful for actual searches too!

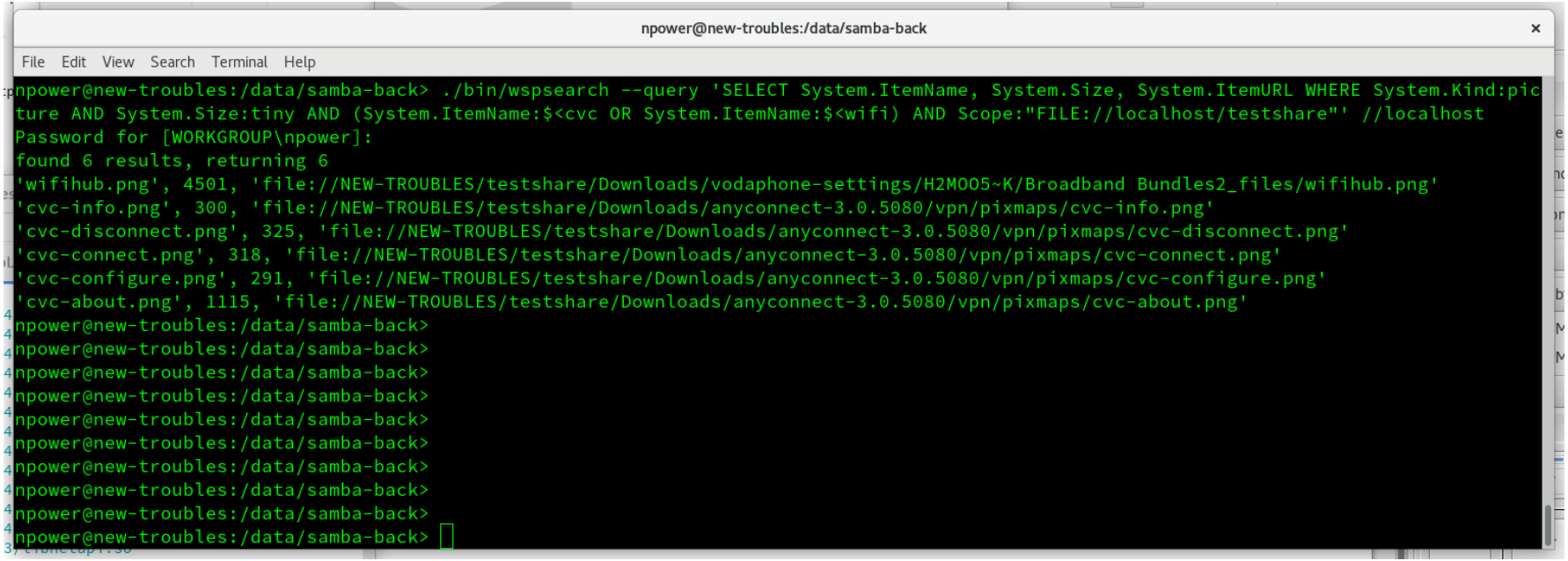
WSP Client

Simple Search

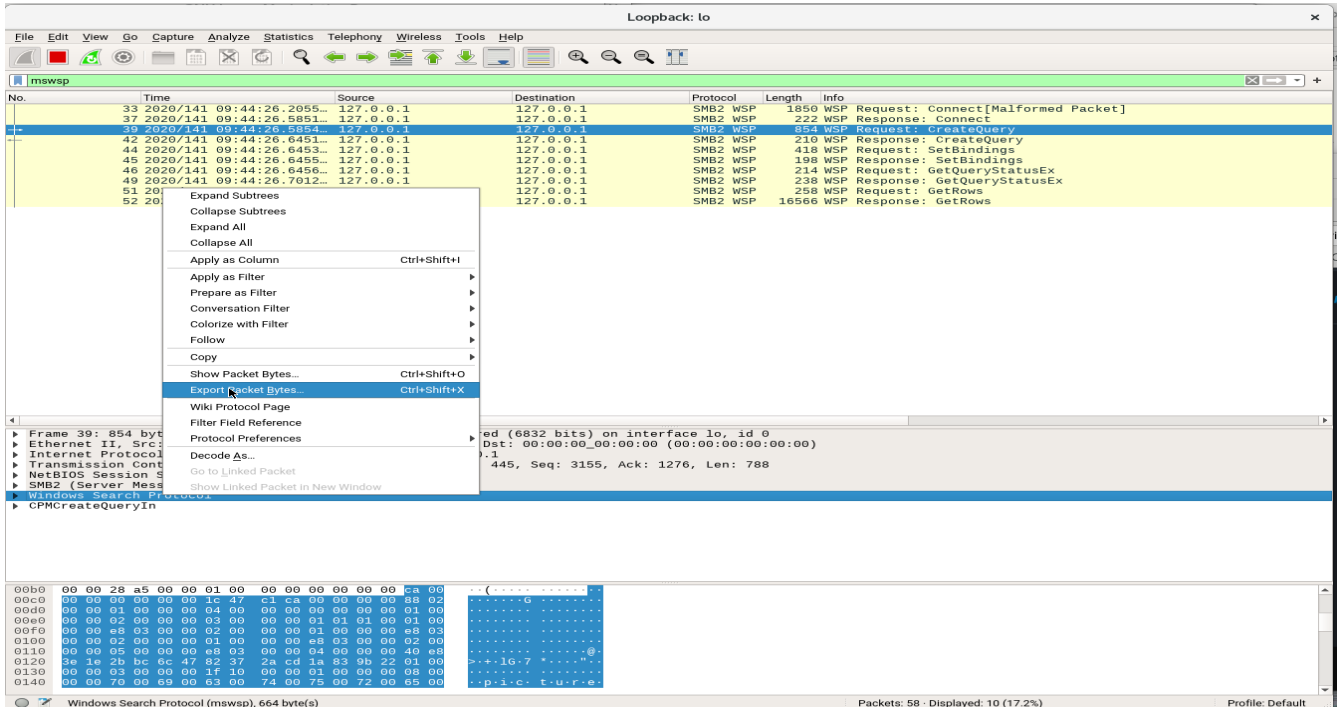


WSP Client

Complex search



Translator


Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
mswsp
No. Time Source Destination Protocol Length Info
33 2020/141 09:44:26.2055… 127.0.0.1 127.0.0.1 SMB2 WSP 1850 WSP Request: Connect[Malformed Packet]
37 2020/141 09:44:26.5851… 127.0.0.1 127.0.0.1 SMB2 WSP 222 WSP Response: Connect
39 2020/141 09:44:26.5854… 127.0.0.1 127.0.0.1 SMB2 WSP 854 WSP Request: CreateQuery
42 2020/141 09:44:26.6451… 127.0.0.1 127.0.0.1 SMB2 WSP 210 WSP Response: CreateQuery
44 2020/141 09:44:26.6453… 127.0.0.1 127.0.0.1 SMB2 WSP 418 WSP Request: SetBindings
45 2020/141 09:44:26.6455… 127.0.0.1 127.0.0.1 SMB2 WSP 198 WSP Response: SetBindings
46 2020/141 09:44:26.6456… 127.0.0.1 127.0.0.1 SMB2 WSP 214 WSP Request: GetQueryStatusEx
49 2020/141 09:44:26.7012… 127.0.0.1 127.0.0.1 SMB2 WSP 238 WSP Response: GetQueryStatusEx
51 20 127.0.0.1 SMB2 WSP 258 WSP Request: GetRows
52 20 127.0.0.1 SMB2 WSP 16566 WSP Response: GetRows
Expand Subtrees
Collapse Subtrees
Expand All
Collapse All
Apply as Column Ctrl+Shift+I
Apply as Filter
Prepare as Filter
Conversation Filter
Colorize with Filter
Follow
Copy
Show Packet Bytes… Ctrl+Shift+O
Export Packet Bytes… Ctrl+Shift+X
Wiki Protocol Page
Filter Field Reference
Protocol Preferences
Decode As…
Go to Linked Packet
Show Linked Packet in New Window
Frame 39: 854 byt ed (6832 bits) on interface lo, id 0
Ethernet II, Src: Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol .1
Transmission Cont 445, Seq: 3155, Ack: 1276, Len: 788
NetBIOS Session S
SMB2 (Server Mess
Windows Search Protocol
CPMCreateQueryIn
00b0 00 00 28 a5 00 00 01 00 00 00 00 00 00 00 ca 00
00c0 00 00 00 00 00 00 1c 47 c1 ca 00 00 00 00 88 02
00d0 00 00 01 00 00 00 04 00 00 00 00 00 00 00 01 00
00e0 00 00 02 00 00 00 03 00 00 00 01 01 01 00 01 00
00f0 00 00 e8 03 00 00 02 00 00 00 01 00 00 00 e8 03
0100 00 00 02 00 00 00 01 00 00 00 e8 03 00 00 02 00
0110 00 00 05 00 00 00 e8 03 00 00 04 00 00 00 40 e8
0120 3e 1e 2b bc 6c 47 82 37 2a cd 1a 83 9b 22 01 00
0130 00 00 03 00 00 00 1f 10 00 00 01 00 00 00 08 00
0140 00 00 70 00 69 00 63 00 74 00 75 00 72 00 65 00
Windows Search Protocol (mswsp), 664 byte(s)
Packets: 58 · Displayed: 10 (17.2%)
Profile: Default

Translator

wsp-to-sparql translate binary query

File Edit View Search Terminal Help

```
npower@new-troubles:/data/suse-git-samba> ./bin/wsp-to-sparql ./frame39-query.bin
main: tracker-sparql query is:
"SELECT nie:url(?u) nie:isStoredAs(?u) nfo:fileName(?u) nfo:fileSize(?u) WHERE{?u nie:url ?url . ?u rdf:type ?type
 FILTER(((?type IN (nfo:Image) && (nfo:fileSize(?u) >= 1 && nfo:fileSize(?u) < 10241)) && (regex(nfo:fileName(?u),
'^cvc') || regex(nfo:fileName(?u),'^wifi'))) && tracker:uri-is-descendant ('file:///data/SHARE', nie:url (?u)))} O
RDER BY DESC(nie:isStoredAs(?u))"
main: selected columns:
main: Col[0] System.Search.EntryID is mapped/converted from tracker col[4] (null)
main: Col[1] System.ItemName is mapped/converted from tracker col[2] nfo:fileName
main: Col[2] System.Size is mapped/converted from tracker col[3] nfo:fileSize
main: Col[3] System.ItemUrl is mapped/converted from tracker col[0] nie:url
npower@new-troubles:/data/suse-git-samba>
npower@new-troubles:/data/suse-git-samba>
npower@new-troubles:/data/suse-git-samba>
npower@new-troubles:/data/suse-git-samba>
npower@new-troubles:/data/suse-git-samba>
npower@new-troubles:/data/suse-git-samba>
npower@new-troubles:/data/suse-git-samba>
npower@new-troubles:/data/suse-git-samba>
```

Elasticsearch

Elasticsearch

- Recently Spotlight got support for an elasticsearch backend
- Thanks to another hackweek at SUSE I decided to use the time to have a look into elasticsearch as a potential backend for WSP
- Seems recently there had been a bit of interest again in WSP and this seemed a good opportunity to refresh my own knowledge of what needs to be done with WSP, (re)evaluate where we are with the implementation.

Elasticsearch

- Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elasticsearch is developed in Java. Elasticsearch is the most popular enterprise search engine – (source Wikipedia)
- Elasticsearch is fast, distributed, can be self hosted or cloud based, can be clustered etc. etc.
- It’s easy to communicate with, as mentioned above, http/json
- Rich restful API (that maybe can fill more gaps of the more exotic restrictions)

Elasticsearch Investigation

- **Poked around the spotlight code**
  - Unfortunately I don’t have any ios devices so I couldn’t really run much except some test code but I at least could see lots of snippets of code I could possibly use.
- **Visited https://www.elastic.co/**
  - In particular
    - Intro
      - https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html
    - Full Query DSL (Domain Specific Language)
      - https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html
- **Downloaded the elasticsearch rpm**
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/rpm.html

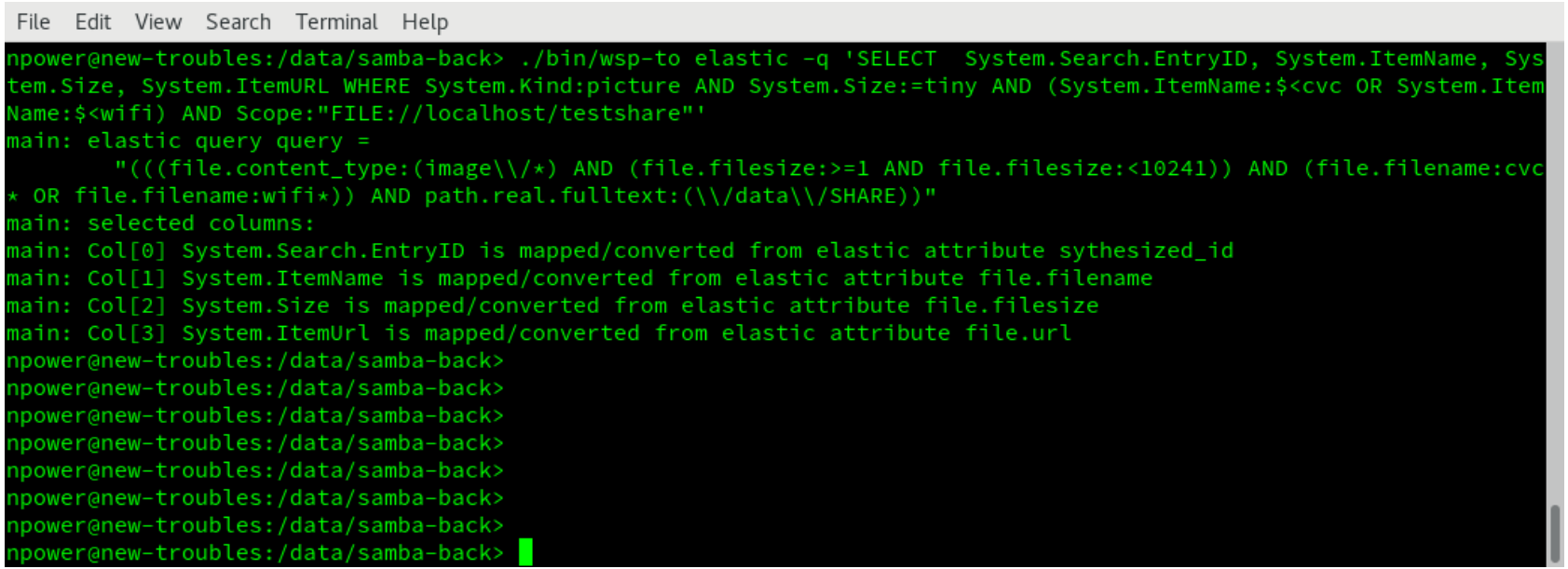
Elasticsearch investigation

- **While the elasticsearch documentation shows examples of populating and querying an index, these examples aren’t very useful for experimenting with something that is supposed to represent an index of say a filesystem.**
- **To that end you can use fscrawler which will**
  - Index local file system (or a mounted drive) crawling and index new files, update existing ones and removes old ones
    - https://fscrawler.readthedocs.io/en/latest/
    - https://fscrawler.readthedocs.io/en/latest/user/getting_started.html

Elasticsearch simple query conversion

- It’s easy to test queries against elasticsearch on the command line just using curl so as a first step I modified the existing translator tool wsp-to-sparql to accept a query string in addition to a binary message.
- Secondly I modified wsp-to-sparql to take an extra parameter to specify the backend in order specify the ‘flavour’ of query statement to translate. E.g be able to translate to elasticsearch Query DSL (specifically a ‘query_string’ query) or translate to tracker/sparql

Elastic query translate



Elastic query translate

Elasticsearch Query

Full Query Template

```
{
    "from": "$from",
    "size": "$size",
    "_source": [$cols],
    "query": {
        "query_string": {
            "query": "$restrictions"
        }
    }
}
```

$from	Index at which to returns row resuls from
$size	Number of rows to return
$cols	List of properties (or column values) to return
$restictions	Query string generated from binary command tree

Elasticsearch what I tried

- Started with duplicated abstract implementation for tracker
- Elastic query conversion code is shared with translator tool (the unfortunately named wsp-to-sparql)
- Modified the duplicated tracker code to communicate with elasticsearch overt http/json (thanks to pilfering similar spotlight code)
- Got very basic support for simple queries to work

Elasticsearch impressions

- **Elasticsearch is pretty simple (especially compared to tracker) to integrate with**
  - Communication is simple (http|https) & json
- **The Restful API and query support seems much richer and more suitable than tracker**
  - Most likely someone with more elasticsearch/Full text search knowledge could really improve the conversions here
- **Processing the results seems a little slower than tracker (most likely due to all the string processing required)**

Elasticsearch impressions

- **You have to be careful, although it seems although with elasticsearch http pipelining is supported, that wasn’t my experience.**
  - Multiple client messages sent without waiting for reply sometimes responded with out of order responses (which led to weird hard to reproduce problems)
- **The WSP communication model however fits having multiple connections where within each connection the message order is effectively serialized so this is not going to be a problem.**

Implications for Elasticsearch implementation

- **I suppose not surprisingly the implementation of the abstract interface for elasticsearch is mostly very very similar (but not the same) as the implementation code for tracker. Mostly to do with**
  - persisting state between calls
  - tracking queries (and associated data)
  - Converting requested result columns (where possible) into either appropriate retrievable properties for the indexed document or use a suitable property to ‘synthesize’ the required column value
  - Converting values of properties used in WSP query into appropriate values for use in elastic query
    - For example paths in WSP typically are of the form ‘file://netbios_name/share_path/…/something.ext’
    - Such paths need to be converted when incoming as part of the query conversion and additionally a ‘normal’ file path will need to be converted when returning the value to the WSP client.

Implications for Elasticsearch implementation

- **rework needed**
  - avoid code duplication
  - Possibly provide a reusable internal api for elasticsearch & tracker and possibly other indexers
  - add support to be able to configure wsp to use any of the supported indexers
- **Multi-index support**
  - It is not unusual different types of data to be stored in separate elasticsearch indices. FSCrawler for example by default stores folders in one index and files in another, this setup is of course configurable. A ‘Folder’ is one of the ‘kind’(s) the windows UI already supports searching for, it is not possible to query for a ‘Folder’ as there is no distinguishing property to disambiguate a folder from a file in the index. The only possibility is specifically searching the ‘folder’ index.

Implications for Elasticsearch implementation

- **Properties**
  - Given the schema-free nature of elasticsearch unlike tracker/sparql we have no idea what properties might be used with a given implementation using elasticsearch. This aspect (and the associated conversions etc.) probably will need to be somewhat configurable and/or pluggable

Elasticsearch – does it help

- **WSP implementation**
  - Mapping of restriction types ?
    - Seems like there are more similar concepts to WSP than tracker in the api (haven’t explored in detail)
  - Mapping to & from WSP properties ?
    - Not much if any difference (however lots of flexibility if you wish to customize say what fscrawler populates elastic with)
  - Cursor navigation (e.g. paging through results) ?
    - Large improvement, no need to cache results
  - Nested queries (e.g. queries based on previous open queries) ?
    - No difference

Elasticsearch – does it help

- **WSP Implementation**
  - EntryId mapping ?
    - If anything this seems a little more difficult than tracker, elasticsearch stores documents with ‘_id’ a unique identifier but it is not possible to use this ‘_id’ when using ‘query_string’ as with the template used currently for elasticsearch.
  - A number of other ids such as chapters, bookmarks
    - Not sure
  - Grouping or results, aggregations etc. ?
    - The api caters for grouping and aggregations but not sure how well elasticsearch maps to WSP with this, like EntyId mapping this might have implications for the query template and make the query generation more complicated
  - Scalability
    - Much better than tracker and no need to cache results like we need to do with tracker/sparql

Demo

Conclusions

- **Elasticsearch is a promising alternative indexer to tracker**
- **It’s worth implementing a new backend for elasticsearch**
- **Help would be great (both for elasticsearch & tracker) especially in the area of query translation**
- **Current WIP repo**
  - https://git.samba.org/?p=npower/samba.git;a=shortlog;h=refs/heads/npower_WIP_WSP
  - https://git.samba.org/?p=npower/samba.git;a=shortlog;h=refs/heads/npower_WIP_WSP_elastic_conv
  - https://git.samba.org/?p=npower/samba.git;a=shortlog;h=refs/heads/sambaxp2020-awful-democode


SUSE
We adapt. You succeed.