Supporting failure analysis with discoverable, annotated log datasets

Steve Leak (NERSC, presenting)
Annette Greiner (NERSC)
Jim Brandt (SNL)
Ann Gentile (SNL)
“Why was my job 30% slower today than usual?”

- Network congestion caused by whatever else happened to run at that time?
- How do we find out?
  - Browse the logs?
  - Are we collecting HSN counters?
Why is this hard?

- **Volume**
- **Access**
  - It’s on SMW
  - It’s in someone else’s $HOME
- **Variety**
  - Formats, storage
- **Expertise**
  - What do all those messages actually mean?
“We can solve any problem by introducing an extra level of indirection”

• Machine-readable metadata
  – Decouple publication and discovery from storage and access
  – Deal with a tractable volume of data before diving deeper
  – Solve the solvable now, and let local solutions address local constraints
What are the requirements?

- Volume
- Variety
- Access
- Expertise
What are the requirements?

• Format-agnostic, storage-agnostic
  – Work with what we have

• No dependence on a priori knowledge of data
  – “Ann is collecting that” is fine .. If you know Ann, and what she is collecting (and if she’s available today)

• Decentralized
  – If you have everything in one place – great! (But you probably don’t)
What are the requirements?

• Low effort, low risk to publish data
  – “select something non-sensitive to publish” vs. “redact all the sensitive bits”

• Make contributing expertise easy

• Deal in tractable volumes
  – Don’t download the internet

• Understand connected/related components
  – The fault might start somewhere else
A metadata solution

- Format-agnostic, storage-agnostic
- No dependence on a priori knowledge of data
- Decentralized
- Low effort, low risk to publish data
- Make contributing expertise easy
- Deal in tractable volumes
- Understand connected/related components

With metadata we can:
- Decouple publication and discovery from storage and access
- Deal with a tractable volume of data before diving deeper
- Link different data together
- Solve the solvable now, and let local solutions address local constraints
A metadata solution

- RDF vocabulary for describing log data collections and finding relevant logs
  - Machine readable, searchable, decentralized, global graph

- Schema for annotating data within logs and exploring a reduced set of relevant log entries
Linked Data and RDF
Linked Data and RDF

• Triples: subject, predicate, object
  – CUG2018 is a Conference
  – Conference has Research Presentations
  – Stockholm is hosting CUG2018

• We can infer that this talk is happening, here, now
Linked Data and RDF

• Everything* is a URI
• Convention:
  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
  rdfs: type
• Means:
  <http://www.w3.org/2000/01/rdf-schema#type>  
  @prefix nersc: <http://nersc.gov/project/hmdr/nersc#> .
  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
  nersc:nersc rdfs: type foaf: Organization .

* Almost everything (some things are literal strings, etc)
Querying a graph (SPARQL)

SELECT ?name ?interest
WHERE {
  ?type rdfs:subClassOf* foaf:Agent .
  ?uri foaf:name ?name .
  ?uri foaf:interest ?interest .
}
• Returns “Steve”, “RDF”
RDF Vocabulary for log data

- Collections of logs
  - dcat:Distribution
    - logset:LogSet
      - dct:title
      - dct:description
      - dct:publisher
      - dct:conformsTo

- Type of log (e.g., “console”)
  - logset:LogFormatType
    - dct:description
    - dct:mediaType

- What it’s about (e.g., “HSN”)
  - skos:Concept
    - logset:SubjectType
      - skos:prefLabel
      - skos:note

- Specific log files
  - logset:ConcreteLog
    - dct:temporal
    - dct:accessURL
    - dct:downloadURL
    - dct:temporal
    - dct:byteSize
    - logset:recordCount
    - logset:estRecordCount
    - logset:estRecordsPerDay

- Specific subject (e.g., “Cori HSN”)
  - logset:LogSeries
    - logset:infoType
    - logset:logFormatType
    - logset:logFormatInfo
  - logset:Subject
    - dct:description
dct:landingPage
    - logset:affects
    - logset:partOf
Global graph, Catalogs form hubs
What does this get us?

- Format-agnostic, storage-agnostic
- **No dependence on a priori knowledge of data**
- Decentralized
- Low effort, low risk to publish data
- Make contributing expertise easy
- Deal in tractable volumes
- **Understand connected/related components**

**With metadata we can:**

- Decouple publication and discovery from storage and access
- Deal with a tractable volume of data before diving deeper
- Link different data together
- Solve the solvable now, and let local solutions address local constraints
A metadata solution

• RDF vocabulary
  – Discovery **of** data

• Schema for annotating data within logs and exploring a reduced set of relevant log entries
  – Discovery **in** data
Why Annotations?

- That’s a lot to search through!

Boot session: ~16M lines

One day: ~5.5M lines

One logfile: ~2.6M lines
Annotations

• Human-provided commentary
  – “I swapped a DIMM in node nid00123”
  – “These messages are due to a fault injection experiment”

• Machine–generated annotations
  – Eg subset of entries matching significant Baler patterns, with timestamps, components called out
An annotation schema

```sql
CREATE TABLE 'annotations' (  id integer,  authorid char(3) NOT NULL,  description text NOT NULL,  -- timespan of the action or event:  starttime datetime NOT NULL,  endtime datetime NOT NULL,  -- impact of the action or event:  startstate text,  endstate text,  systemdow boolean,  system text,  components text,  -- was the event manually induced?  manual boolean,  -- subject type and annotation context:  LDCategory text,  LDbag text,  balepatternid integer,  -- event source:  logfiles text,  PRIMARY KEY('id','authorid') )
```

Summary, who to ask

When?

What?

Where?

Why?

Categorization

Pointer back to full, raw data
What does this get us?

- Format-agnostic, storage-agnostic
- No dependence on a priori knowledge of data
- Decentralized
- Low effort, low risk to publish data
- Make contributing expertise easy
- Deal in tractable volumes
- Understand connected/related components

With metadata we can:
- Decouple publication and discovery from storage and access
- Deal with a tractable volume of data before diving deeper
- Link different data together
- Solve the solvable now, and let local solutions address local constraints
Physical architecture: parent:child

Link architecture: peer:peer

Router architecture: parent:child, peer:peer
“Why was my job 30% slower today than usual?”

• Network congestion caused by whatever else happened to run at that time?

• How do we find out?
  – Browse the logs? Search a set of annotations
Testing it out

• Some prototype tools:
  – Construct a graph from RDF files on web (or locally)
  – Scan a directory of logfiles and generate RDF to describe them
    • Asks some questions, basic inspection of file characteristics, infers most metadata from the answers and graph
  – Populate an annotation database from a subset of log entries identified via Baler, and some admin notes
    • Baler: finds patterns in log files, weights by presence of listed keywords, filter by highest-weighted patterns
  – Search the annotation database for things of interest
Are applications interfering with each other via HSN congestion?

<table>
<thead>
<tr>
<th>description</th>
<th>logfiles</th>
<th>LDc</th>
</tr>
</thead>
<tbody>
<tr>
<td>System computing and listing congestion candidate applications</td>
<td>nlrd</td>
<td>NE</td>
</tr>
<tr>
<td>System computing and listing congestion candidate nodes</td>
<td>nlrd</td>
<td>NE</td>
</tr>
</tbody>
</table>

"What applications did it find?" (fetch that part of nlrd file)

- ... none at all!
- What else happened? Search annotations for the half-hour leading up to this one
Are applications interfering with each other via HSN congestion?

• The last half hour:
  – 300 annotated events, 7 distinct
  – 192 were:
    c0-0c1s8a0n0 Correctable memory error. This may result in degraded performance.
  – 47 were:
    c0-0c1s8a0n0 Component failed

• Let’s look at that component more closely...
Are applications interfering with each other via HSN congestion?

- c0-0c1s8a0n0 “Component failed” and “Correctable memory error”
- Issues started a few weeks earlier and stopped a few days later (.. Why did it stop?)
  - Start coincided with deliberately induced faults for system testing – difficult to ascertain
  - Why did it stop? Search a bit wider, over a couple of levels of physical architecture
    - Found at a couple of levels up that the blade was reseated on that day. Constraining the search to around the time the errors stopped, can see entries documenting a warm swap, after which the errors stopped
What did we learn?

• Our intuition was wrong – we expected to find a communication-heavy application but instead found a component issue

• Searching a database of annotated log entries reduced the search space from 150000+ lines to a few hundred
Where are we now?

• RDF vocabulary defined
• Annotation schema defined
• Prototype tools
  – (further development in progress)
• Finding: this can make exploration more tractable, and lead to interesting insights
Making log data discoverable and tractable – machine readable metadata

• Format-agnostic, storage-agnostic
• No dependence on a priori knowledge of data
• Decentralized
• Low effort, low risk to publish data
• Make contributing expertise easy
• Deal in tractable volumes
• Understand connected/related components

With metadata we can:
  – Decouple publication and discovery from storage and access
  – Deal with a tractable volume of data before diving deeper
  – Link different data together
  – Solve the solvable now, and let local solutions address local constraints
Q&A