Expert Developer Recommendation Using Very Large Datasets

Project Presentation

Saim Mehmood
Tilemachos Pechlivanoglou

Project goals

We want to create a tool that:

- can be used to search for expert developers
- provides search criteria for developers’ skills and experience
- bases results on actual contributions
- uses a very large dataset of possible experts (20+ million)
- search queries are reasonably fast
- can be extended to include more criteria
  - *(code quality, bugs introduced, etc.)*

Example search

- Looking for a developer who:
  - *Has 5 years expertise in C++*
  - *Has contributed to an Apache project*

- Or:
  - *Worked recently with Python*
  - *Is familiar with (has used) the Flask framework*
  - *Commits often*

Pre-Processing

BigQuery and big troubles

The dataset available

- The dataset:
  - ***20+ million** developers*
  - ***215 million** commits*
  - ***2.3 billion** files*
  - ***3+ TB** of data*

- Sample dataset available
  - *used during development*
  - *500 times smaller (0.2%)*

Some dataset issues

- Some columns available in sample missing in full
  - ***Workarounds needed***

- Available online only
  - *Can't directly download data above a few MB*
  - *Too big to process locally anyway*
  - ***Preprocessing needed to reduce size***

BigQuery

Google’s BigQuery is perfect for preprocessing

- Allows running SQL queries on dataset
  - *Traditional and extended SQL supported*

- Distributed processing, optimizations for big data
  - *Mostly transparent to the user*

Data reduction (1)

Step 1: Throw away useless data

- We focus on developer commit data only
- Other information is discarded:
  - *file contents may be used in the future*
  - *hashes etc. not useful to the project*
- Data reduction: 3TB $\longrightarrow$ 100GB !!

Data reduction (2)

Step 2: Associate file extensions with programming languages/frameworks

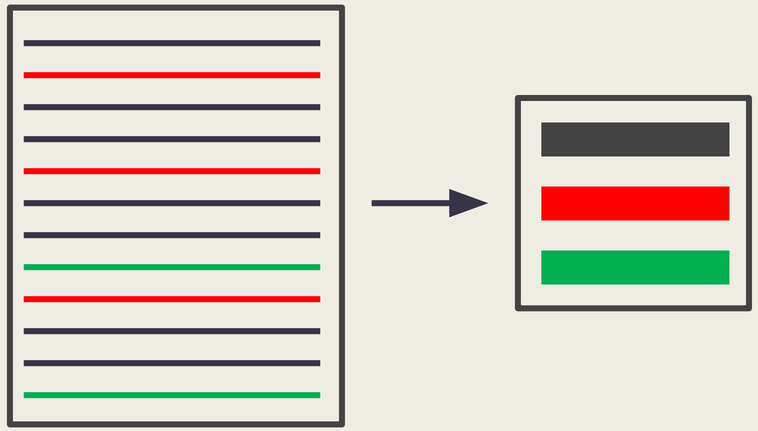
- Find files altered in every commit
- Match file extensions to languages
  - *Using external list*
  - *Throw away file path data*
- Data reduction: 100GB → 51GB

Extension	**Language/ Framework**
.cpp →	C++
.java →	Java
... →	...

Data reduction (3)

Step 3: Condense time data

- For every developer skill:
  - *Aggregate commits*
  - *Store first, last commit*
  - *Store nr. of commits, files*



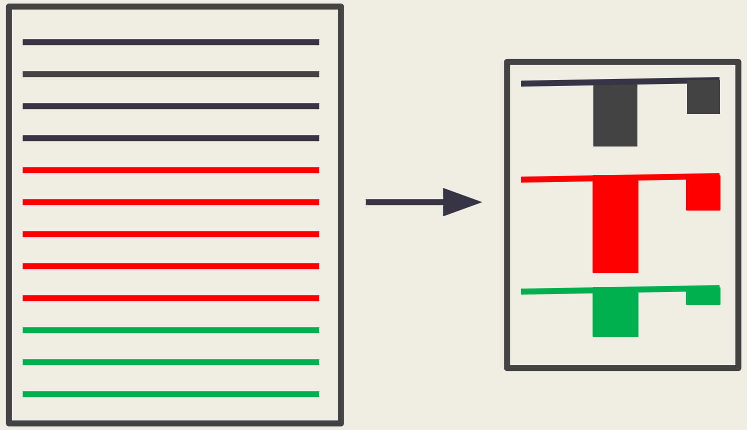
- Data reduction: 51GB → 1.1GB !!

Data reduction (4)

Step 4: Small optimizations

- Reducing forked repo names
  - *e.g. torvalds/linux → linux*
- Representing as JSON, not CSV
  - *nested data, less duplication*

- Data reduction: 1.1GB → 600MB

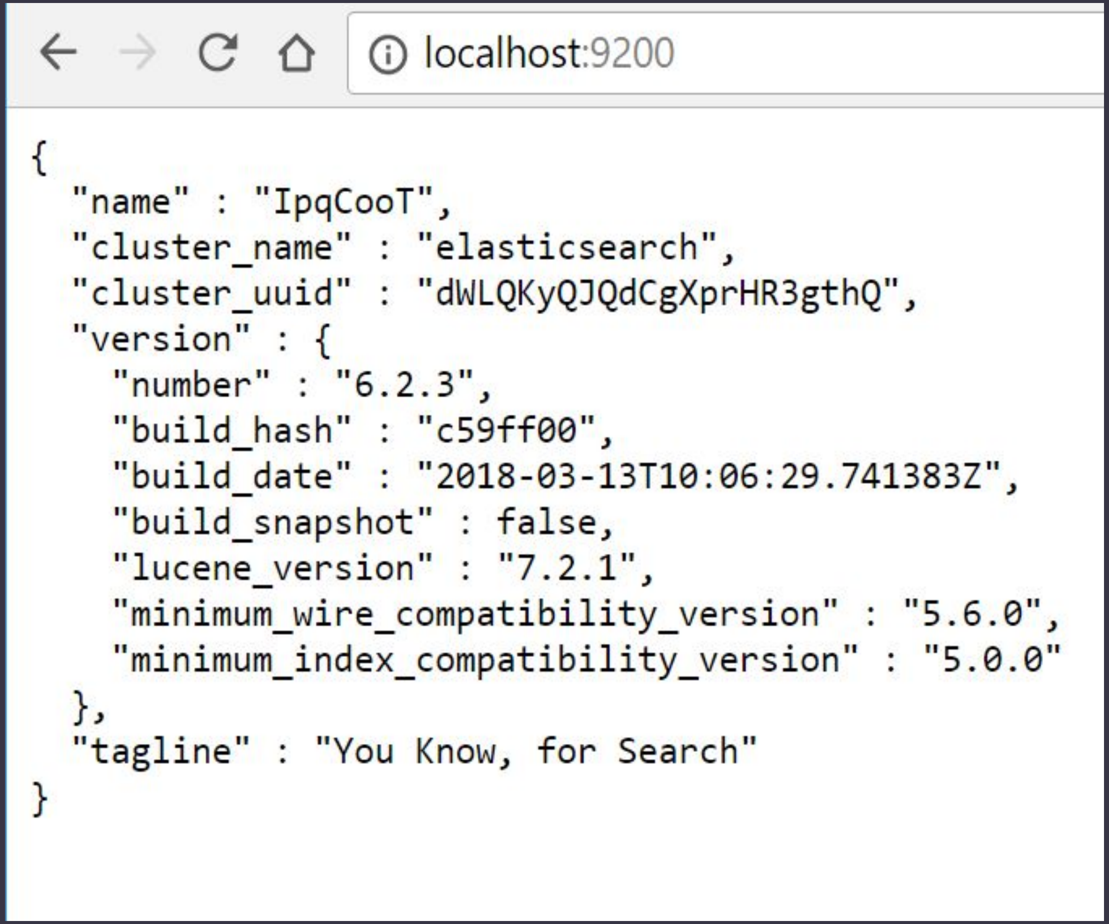


Elasticsearch

Performing Queries

About the tool

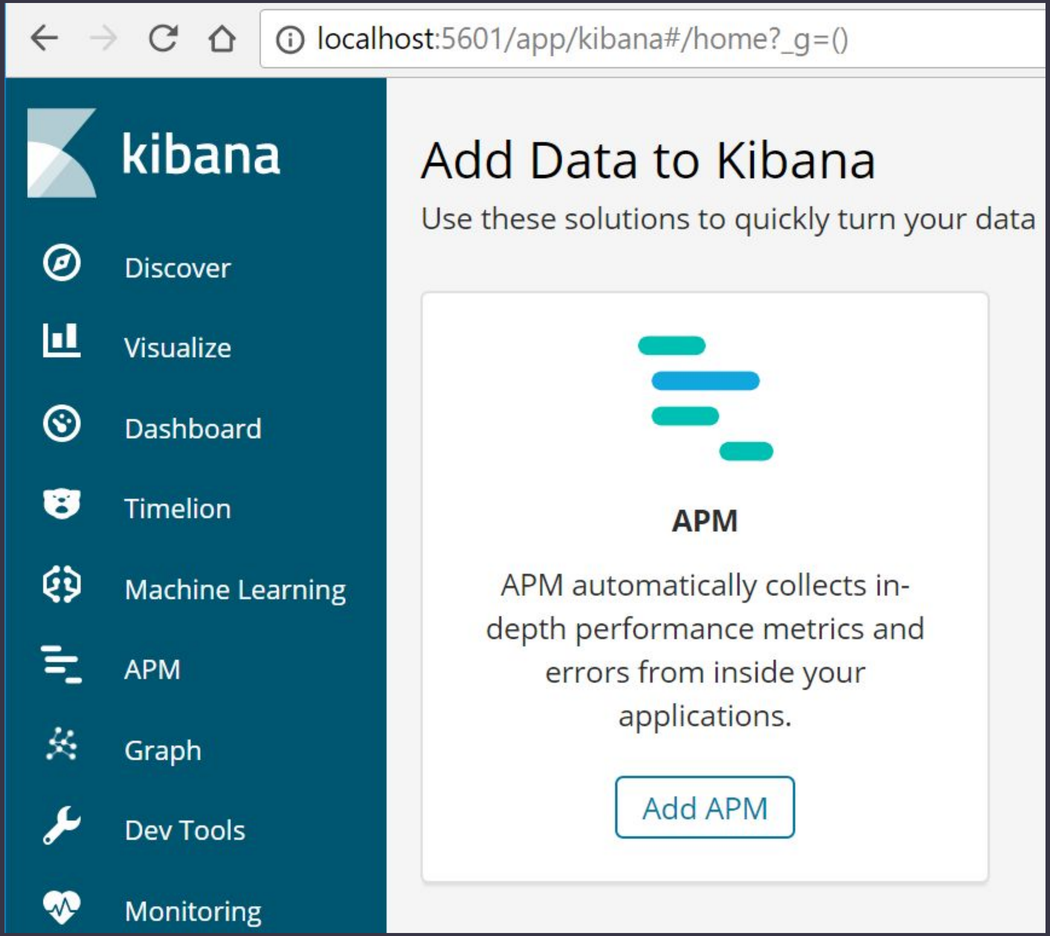
- Elasticsearch is a **highly scalable** open-source full-text search and analytics engine
- Allows you to store, search, and analyze big volumes of data quickly and in near real time
- Used as the underlying engine/technology that powers applications with complex search features



```
localhost:9200

{
  "name" : "IpqCooT",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "dWLQKyQJQdCgXprHR3gthQ",
  "version" : {
    "number" : "6.2.3",
    "build_hash" : "c59ff00",
    "build_date" : "2018-03-13T10:06:29.741383Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Elasticsearch Server



localhost:5601/app/kibana#/home?_g=()

Add Data to Kibana

Use these solutions to quickly turn your data

APM

APM automatically collects in-depth performance metrics and errors from inside your applications.

Kibana

Initial Working

>.\bin\elasticsearch

>.\bin\kibana

Steps

- Elasticsearch requires indexing for every object (i.e. developer) inside the JSON file

- We used text editor for indexing (avoiding the overhead of Kibana)

- Uploading data into Elasticsearch

  - *curl -H "Content-Type: application/json" --user elastic:elastic*

    -XPOST "localhost:9200/developer/email/_bulk?pretty"

    --data-binary @developers.json

Sample Query

- Simple query to fetch developers by name:
  - *curl -H "Content-Type: application/json" --user elastic:elastic*

    -XGET "localhost:9200/_search?pretty"

    -d "{"query" : {"match" : { "developer" : "Aaron" }}}"

Nested Query

- Nested query to refine search criteria:

  - *curl -H “Content-Type: application/json" --user elastic:elastic*

    -XGET "http://localhost:9200/_search?pretty"

    -d" {"query" : { "nested" : {"path" : "languages",

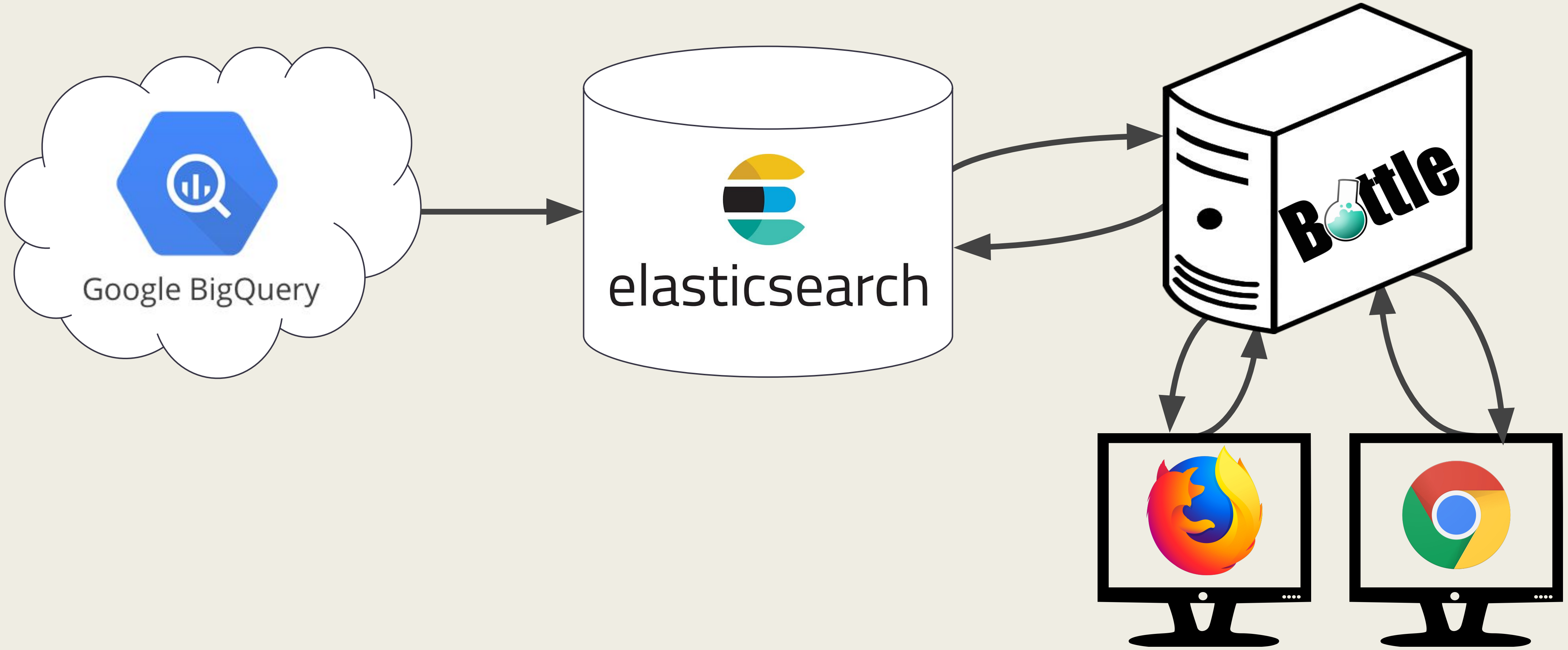
    "query" : { "bool" : { "must" : [{ "match" : { "languages.name" : "Objective-C"} },

    { "range" : { "languages.duration" : {"gt" : 3567688 }} }]}}}}}"

Live Demo

Show time!

The whole system



Conclusion

Things we learned

Future Goals

- Ranking of developers based on Elasticsearch scoring mechanism
- Extending tool to provide search of projects and repositories
- Processing code commits to find Code Quality Metrics specific to a project or developer
- Evaluating tool on diverse data sets

Potential future steps

With this project established, the recommendation algorithm could be extended

- Detailed analysis of code
  - *developer's code wasn't immediately replaced*
  - *code didn't introduce bugs (bugfix comments)*

- More complex requirements
  - *Search for group of experts who cooperated in the past*

Issues

- Financial Constraint: BigQuery 1TB of processing/month for free

- Full data set doesn’t have files changed in commit
  - *Impossible to exactly determine user skills*
  - *Approximation based on repo languages*
  - *Can be mitigated by analyzing file contents*

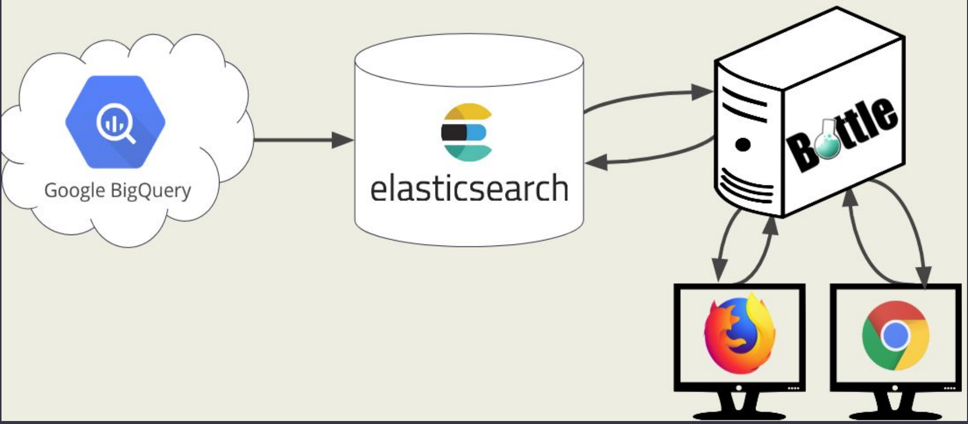
- Most issues can be resolved with time and/or money

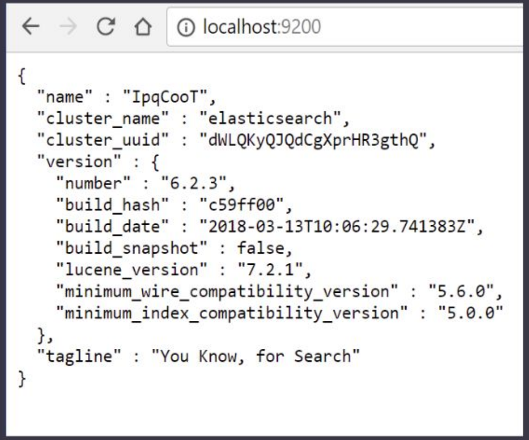
Project goals

We want to create a tool that:

- can be used to search for expert developers
- provides search criteria for developers' skills and experience
- bases results on actual contributions
- uses a very large dataset of possible experts (20+ million)
- search queries are reasonably fast
- can be extended to include more criteria
  - *(code quality, bugs introduced, etc.)*

The whole system

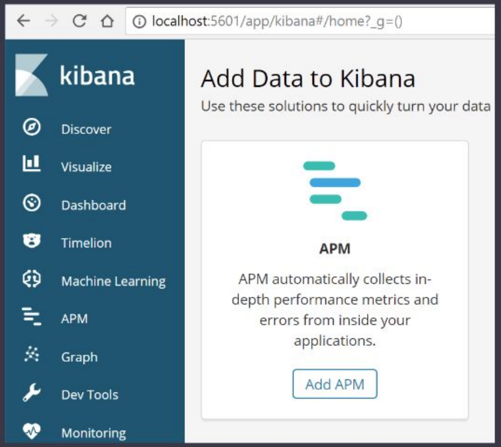




```
localhost:9200

{
  "name" : "IpqCooT",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "dWLQKyQJQdCgXprHR3gthQ",
  "version" : {
    "number" : "6.2.3",
    "build_hash" : "c59ff00",
    "build_date" : "2018-03-13T10:06:29.741383Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Elasticsearch Server


localhost:5601/app/kibana#/home?_g=()
kibana
Discover
Visualize
Dashboard
Timelion
Machine Learning
APM
Graph
Dev Tools
Monitoring
Add Data to Kibana
Use these solutions to quickly turn your data
APM
APM automatically collects in-depth performance metrics and errors from inside your applications.
Add APM

Kibana

Nested Query

- Nested query to refine search criteria:
  - *curl -H "Content-Type: application/json" --user elastic:elastic*

    -XGET "http://localhost:9200/_search?pretty"

    -d"{"query": { "nested": {"path": "languages",

    "query": { "bool" : { "must": [{ "match": { "languages.name" : "Objective-C"} },

    { "range": { "languages.duration" : {"gt": 3567688 }} }]}}}}}}"

Thank you! Questions?