HW3: Comparison between ElasticSearch & Solr

403410034 資工四 黃鈺程

Outline

- Environment
- Preprocessing
  - Mixed Encoding
  - Multi-Processing
  - Queries Generating
- Benchmark
  - Indexing
  - Disk Usage
  - Querying
- Conclusion

Environment

- Fedora 27
  - Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz
  - 4 Cores
  - 4 GB
- Python 3
  - Jupyter, Numpy, Pandas, Matplotlib
- Elastic Search 6.2.2
- Solr 7.3.0
- Test with 500,000 Records
  - Disk Not Enough

Mixed Encoding

- File is not completely UTF-8
  - Python Decoding Error
  - Solved by Specifying “errors” when Opening File

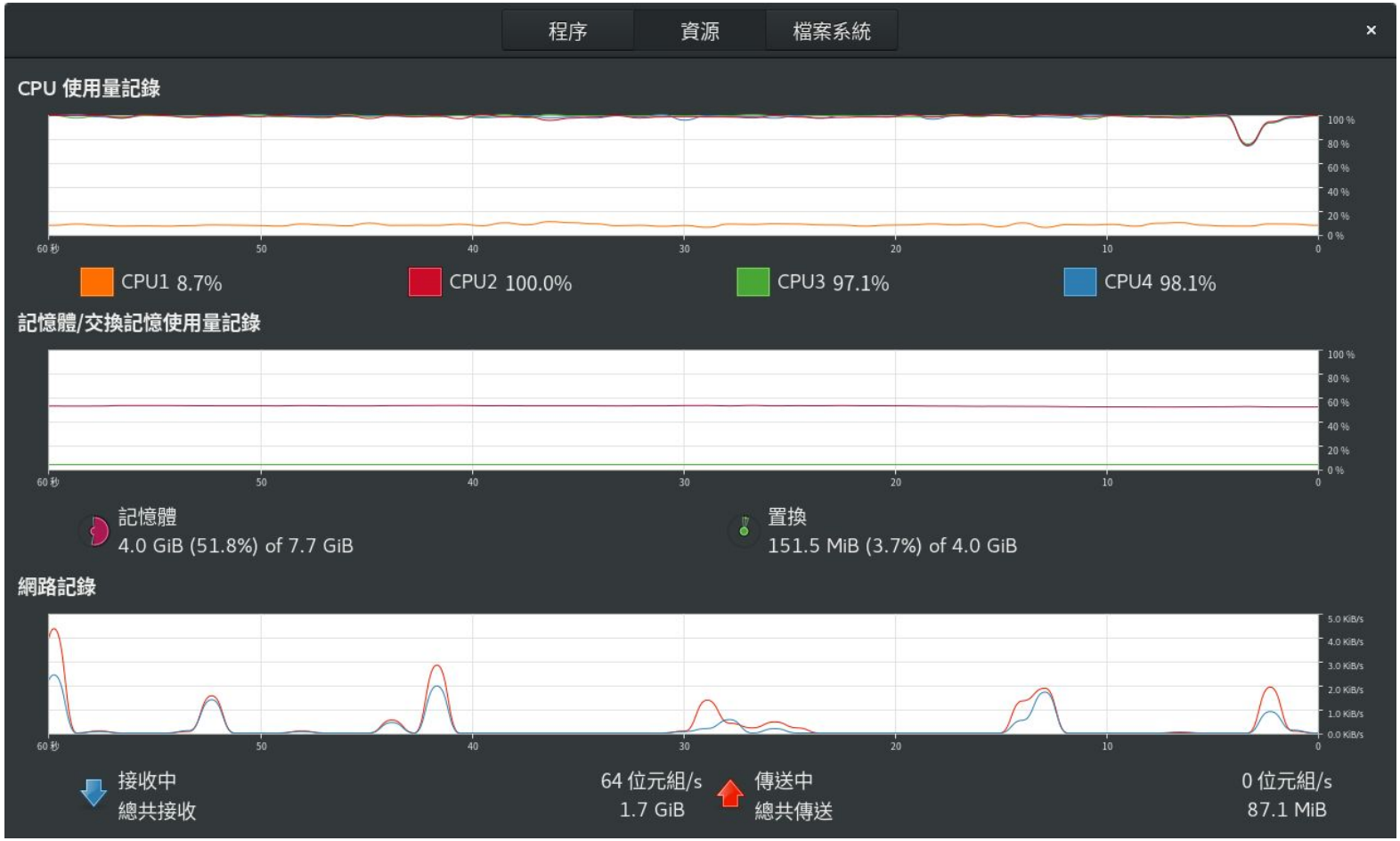
```
with open(raw, encoding='utf-8', errors='ignore') as f:
    with mp.Pool(3) as p:
        record_gen = enumerate(extract_record(f))
        firstn_gen = islice(record_gen, 500_000)
        result_gen = p.imap(segmentize, firstn_gen)
        for idx in result_gen:
            if (idx + 1) % 20_000 == 0:
                print(f'Processed {idx+1:10d} records')
```

Multi-Processing

- Text Segmentation is a CPU-Bound Task.
  - Use 3 Processes to Speed Up
  - Takes 22:38 in total
  - Speed up Nearly 2.x

```
with open(raw, encoding='utf-8', errors='ignore') as f:
    with mp.Pool(3) as p:
        record_gen = enumerate(extract_record(f))
        firstn_gen = islice(record_gen, 500_000)
        result_gen = p.imap(segmentize, firstn_gen)
        for idx in result_gen:
            if (idx + 1) % 20_000 == 0:
                print(f'Processed {idx+1:10d} records')
```

Multi-Processing



Queries Generating

- To Test Query
  - Not able to Turn Off ALL Cache
    - ES/Solr Internal Cache
    - Disk Cache
  - Bad: Query **Single** Words **Many** Times
  - Fine: Query **Many** Words **Single** Time
  - Good: Query **Many** Words **Many** Times
    - Times proportional to Word Frequency

Queries Generating

- Words Frequency
  - From Jieba (zh-TW)
  - 308, 431 Words
  - `np.random.choice` 100, 000 words

```
print(len(queries))
weights = np.float32(weights)
weights /= np.sum(weights)

def get_queries(n):
    return np.random.choice(queries, size=n, p=weights)

queries = get_queries(100_000)
np.save('../data/queries.npy', queries)
print(queries[:10])
```

```
308431
['終歸' '局' '成績' '聽' '開始' '腕' '設置' '歷程' '女廁' '時尚']
```

Indexing

- ES using Bulk API
  - BATCH_SIZE = 4000
- Solr using Single Large JSON
  - `[ {}, {}, {} ]`
- Note that CPU Usage is not 100%
  - Solr has higher CPU Usage (8x% vs 7x%)

	Pre	Time	it/s
ElasticSearch	00:00	09:00	925.19
Solr	03:24	03:28	1213.59

Disk Usage

- By using `du -sh ./<folder>`
  - Can be embedded in Jupyter

```
def get_es_size():
    res = !du -sh ../elasticsearch-6.2.2/
    return res.s.split()[0]
```

	Before	After	Delta
ElasticSearch	32M	2.8G	2.77G
Solr	185M	1.8G	1.62G

Querying

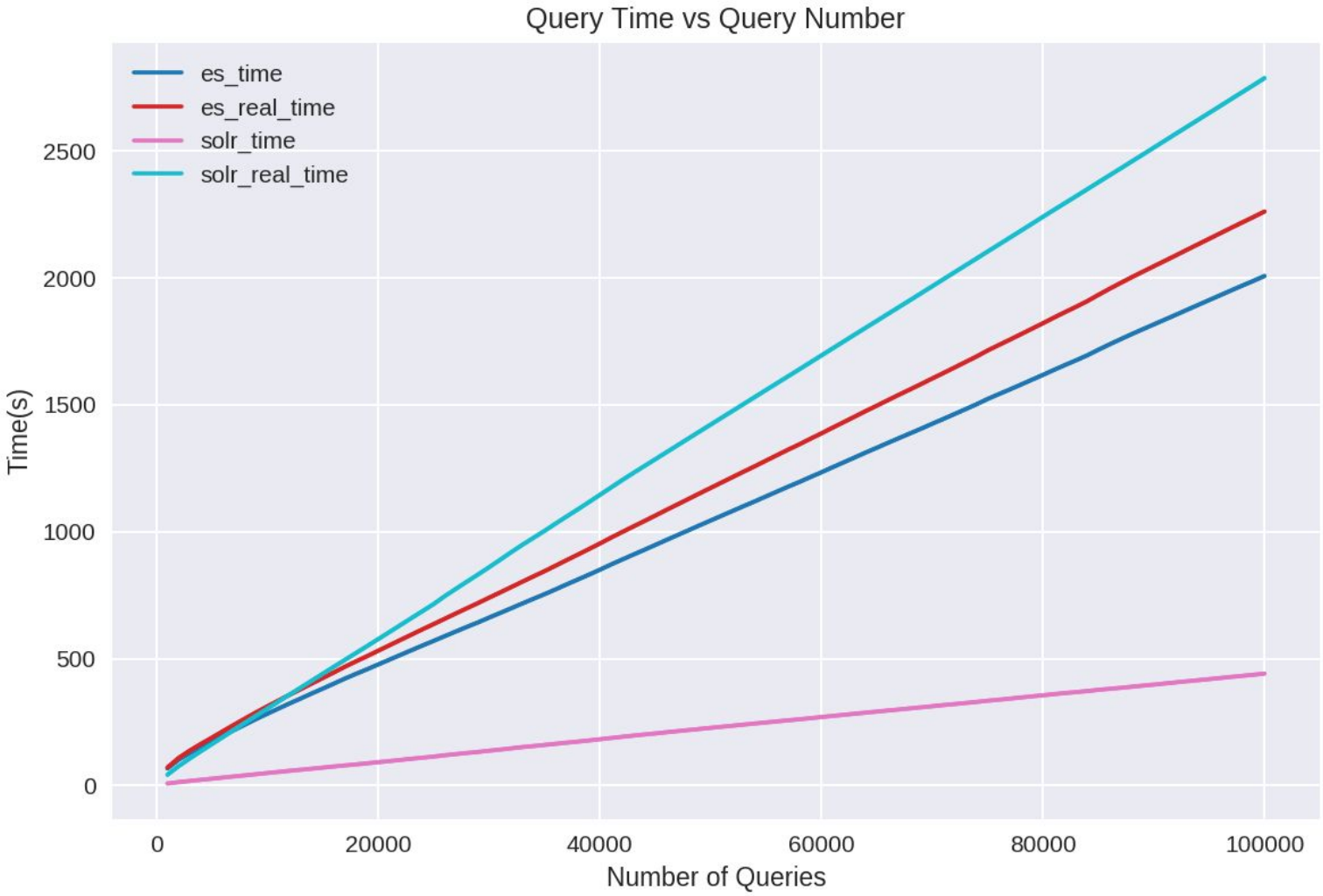
- 100, 000 words
- Note CPU Usage is not 100%
  - May be improved by Sending Queries Async.
- Observe that QPS(Queries per second) is **improving**
  - Logarithmic growth
  - I think it's due to **Cache**

	Total (real)	it/s (start)	it/s (end)
ElasticSearch	37:41	10	44.10
Solr	46:26	6	35.88

Querying

- Query
  - ES: Thru `elastic-search-py`
  - Solr: Thru `check_output (curl)`
- Time:
  - Search Engine Time
    - ES: `res['took']`
    - Solr: `res['Qtime']`
  - Real Time
- Time is Proportional to Number of Queries

Querying


Query Time vs Query Number
es_time
es_real_time
solr_time
solr_real_time
Time(s)
0
500
1000
1500
2000
2500
0
20000
40000
60000
80000
100000
Number of Queries

Conclusion

- Difference between `es_time` & `solr_time`
  - Due to different metrics
- Difference between `x_time` & `x_real_time`
  - Due to Json Encoding, Cache, etc
- Return Result
  - Solr has Cleaner Hierarchy
- Auto Create field
  - Both are able

Conclusion

- All the scripts are put into notebooks
  - Including Starting ES/Solr, Document/Core Creation, Result Visualization, etc
- One Click and Run All
  - Reproducable
  - Easy To Share
- Can be Hosted on Github
  - https://github.com/amoshyc/ccu-search-engine/blob/master/hw3/00_pre.ipynb