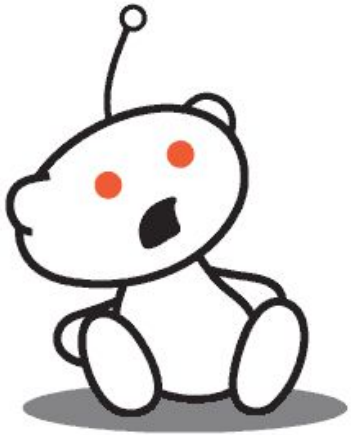
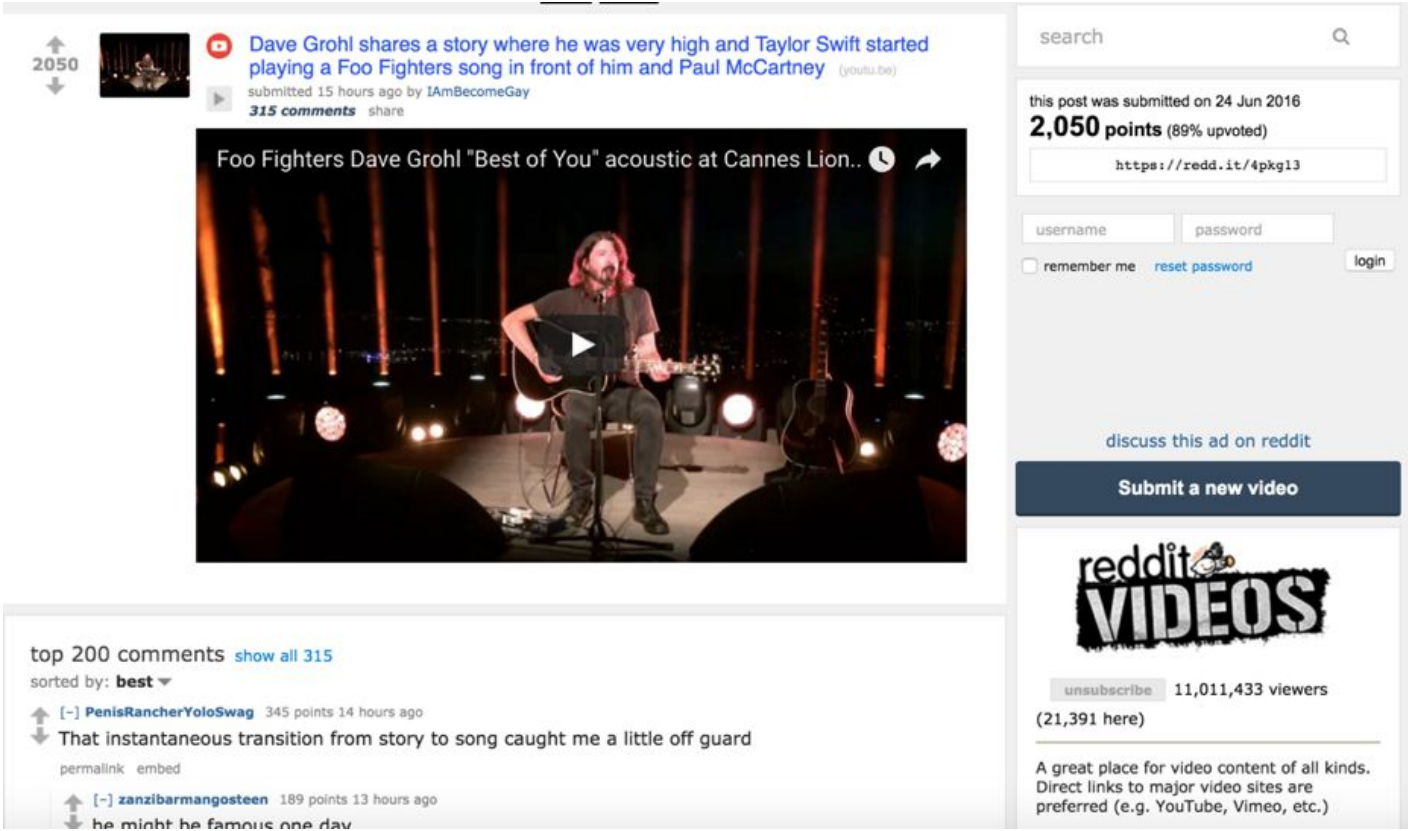
Benchmark MinHash + LSH Algorithm on Spark

Insight Data Engineering Fellow Program, Silicon Valley
Xiaoqian Liu
June 2016

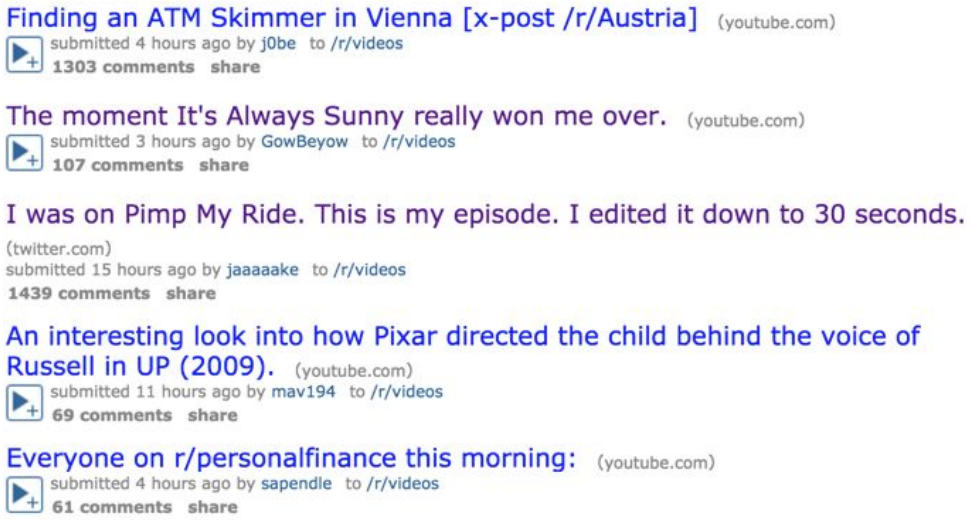
What's next?




2050
Dave Grohl shares a story where he was very high and Taylor Swift started playing a Foo Fighters song in front of him and Paul McCartney (youtu.be)
submitted 15 hours ago by IAmBecomeGay
315 comments share
Foo Fighters Dave Grohl "Best of You" acoustic at Cannes Lion..
top 200 comments show all 315
sorted by: best
[-] PenisRancherYoloSwag 345 points 14 hours ago
That instantaneous transition from story to song caught me a little off guard
permalink embed
[-] zanzibarmangosteen 189 points 13 hours ago
he might be famous one day
search
this post was submitted on 24 Jun 2016
2,050 points (89% upvoted)
https://redd.it/4pkg13
username
password
remember me
reset password
login
discuss this ad on reddit
Submit a new video
reddit VIDEOS
unsubscribe
11,011,433 viewers
(21,391 here)
A great place for video content of all kinds. Direct links to major video sites are preferred (e.g. YouTube, Vimeo, etc.)

Post Recommendation

- Data: Reddit **posts** and **titles** in 12/2014
- Similarity metric: Jaccard Similarity
  - (%common) on titles



Pairwise Similarity Calculation is Expensive!!

- ~700k posts in 12/2014
- Individual lookup: 700K times, O(n)
- Pairwise calculation: 490B times, O(n^2)

MinHash: Dimension Reduction

Post 1	Dave Grohl tells a story
Post 2	Dave Grohl shares a story with Taylor Swift
Post 3	I knew it was trouble when they drove by

4 hash funcs

	Min hash 1	Min hash 2	Min hash 3	Min hash 4
Post 1	932378	11070	107000	195512
Post 2	20930	213012	107000	195512
Post 3	27698	14136	104464	154376

LSH (Locality Sensitive Hashing)

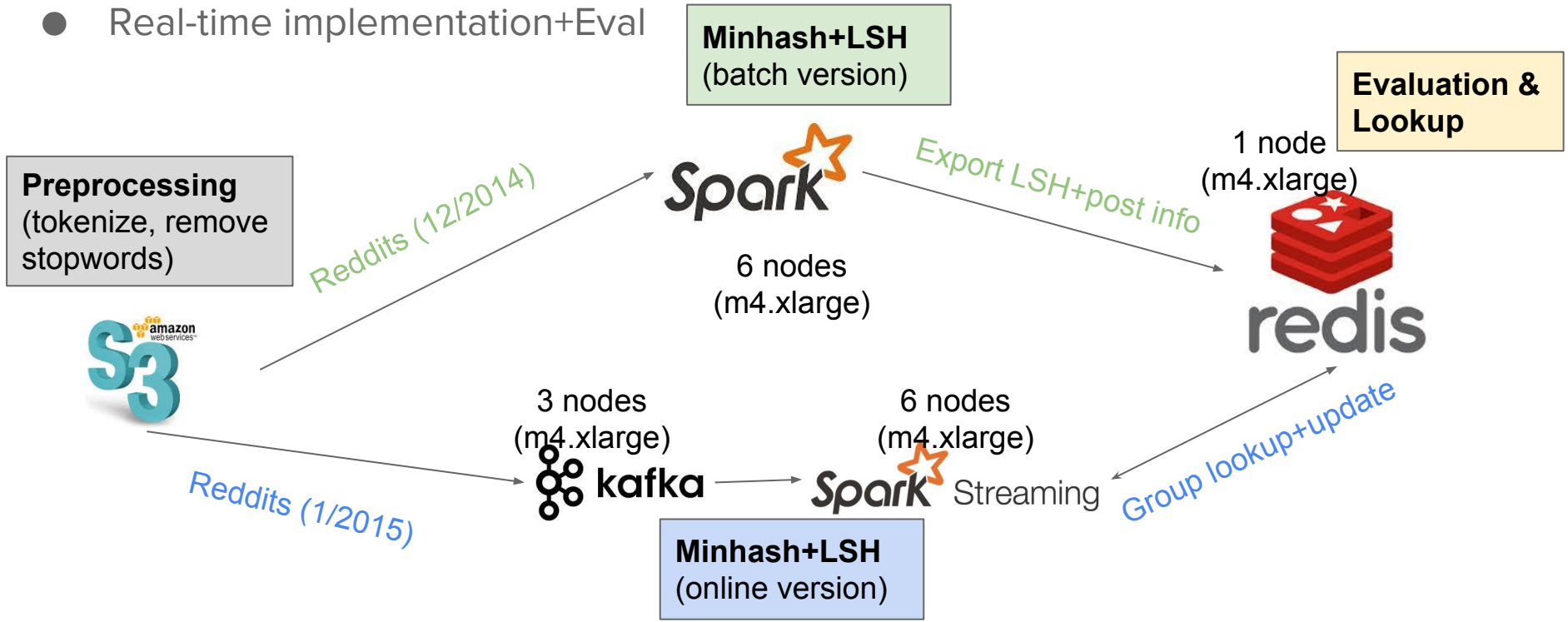
- Further reduce the dimension
- Suppose the table is divided into 2 bands w/ width of 2
- Rehash on each item
- Use (Band id, Band hash) to find similar items

	Band 1	**Band 2**	
Post 1	Hash (932378,11070)	Hash (107000,195512)	← **Dave Grohl**
Post 2	Hash (20930, 213012)	Hash (107000,195512)	← **Dave Grohl**
Post 3	Hash (27698,14136)	Hash (104464,154376)	← **Trouble**

*Algorithm source: Mining of Massive Datasets (Rajaraman,Leskovec)

Infrastructure for Evaluation

- Batch implementation+Eval
- Real-time implementation+Eval

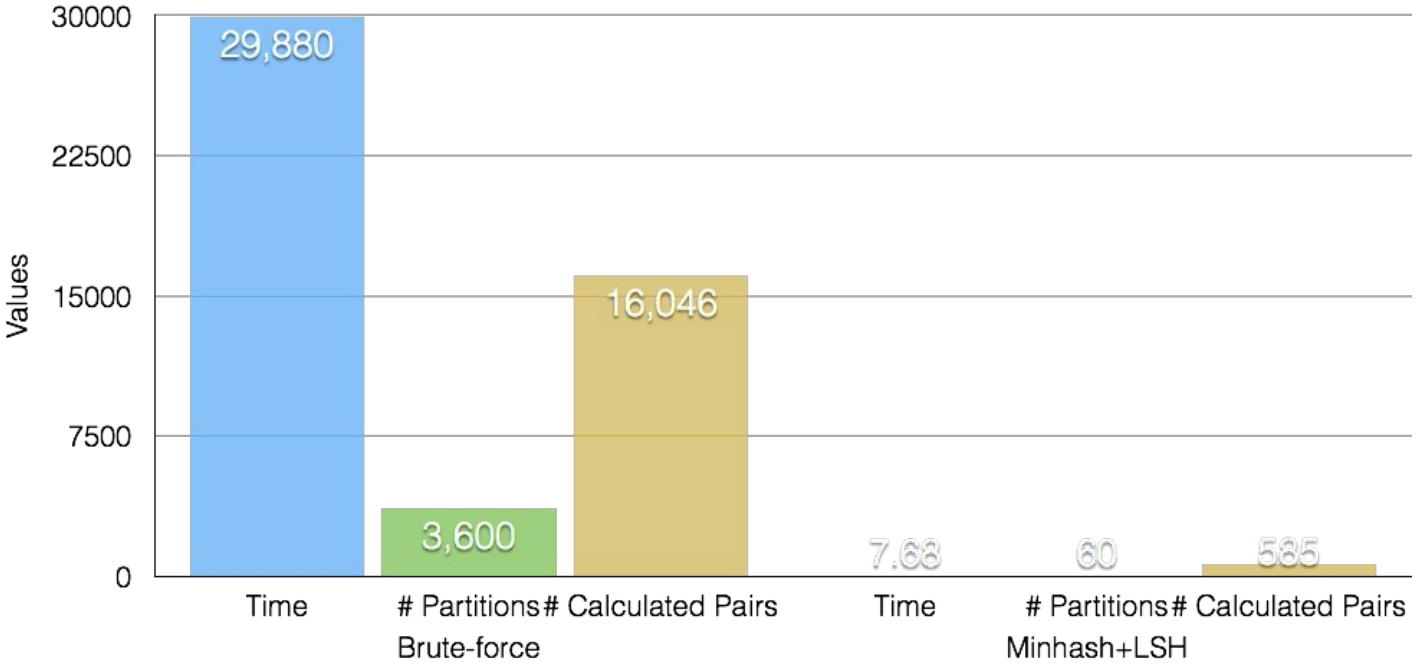


Batch Processing Optimization on Spark

- SparkSQL join, cartesian product
- Reduce Shuffle times for joining two different datasets:
  - Co-partition before joining
- Persist the data before actions
  - Storage level depends on the RDD size
- Filter results before joining and calculating similarities
  - filter(), reducebyKey()

Batch Processing: Brute-force vs Minhash+LSH (10 hash funcs, 2 bands)

100k entries, 12/2014 Reddits



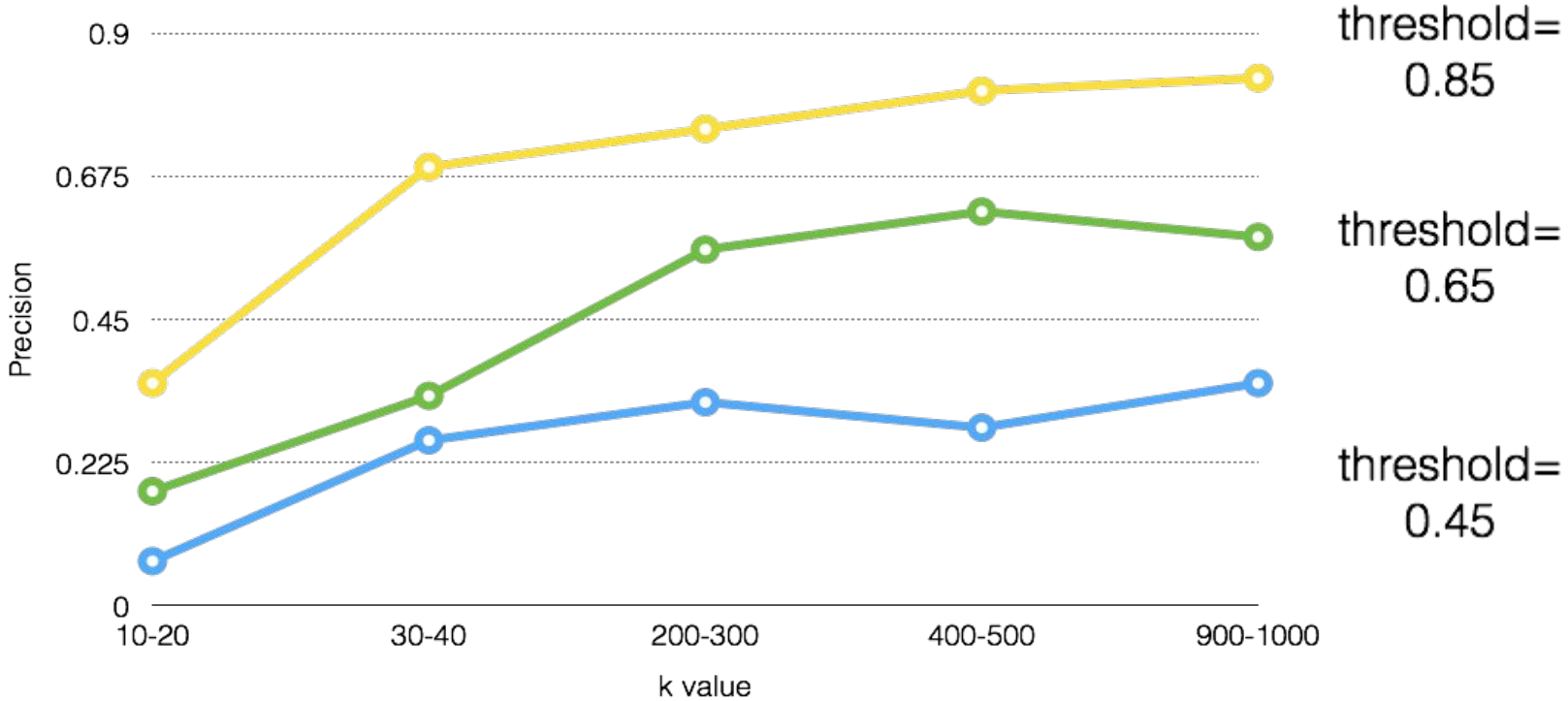
Precision and Recall

- 100k entries, estimated threshold = 0.44

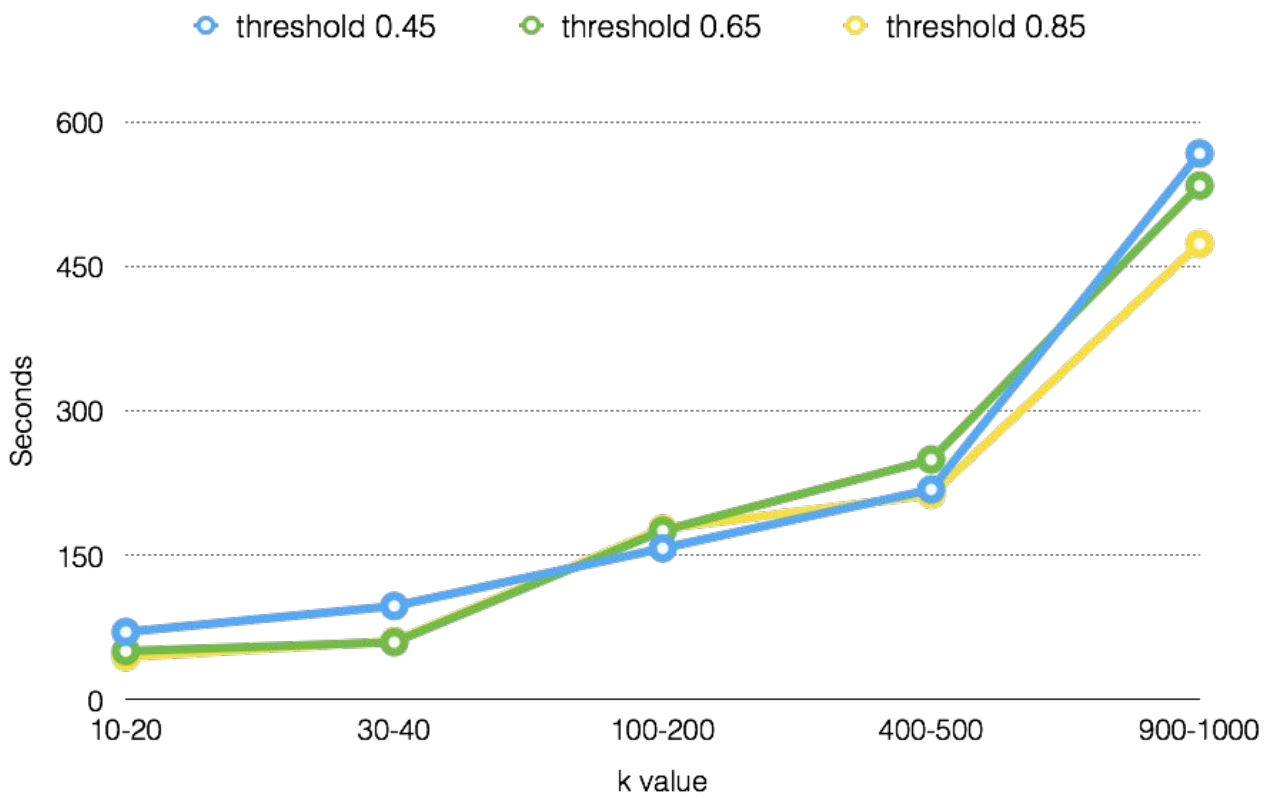
Parameters	Items >=threshold	Total count	Time (sec)	Precision	Recall	num partitions
Brute-force	16,046	9.99B	29,880	1	1	3,600
k=10, b=2	585	65,353	7.68	0.009	0.036	60

780k reddit posts, precision vs k values

K = # hash functions

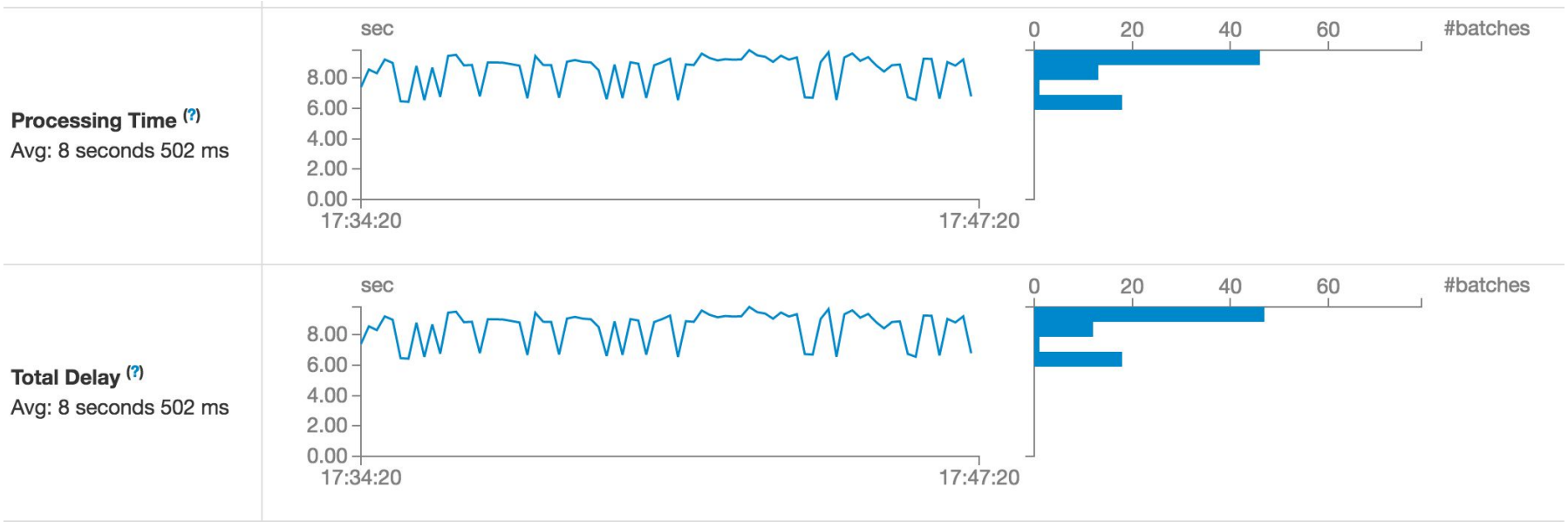


780k reddit posts, time vs k values



Streaming: Average Time

- Throughput: 315 events/sec, 10 sec time window
- 8 sec/microbatch, 6 nodes,



Conclusion

- Effectively speed up on batch processing
- Use 400-500 hash functions, set the threshold above .65
  - Filter out pairs w/ low similarities
  - Linear scan for pairs w/ 0 neighbors
- Only for Jaccard Similarity.
  - For cosine similarity: LSH + random projection

About Me

- BS, MS in Systems Engineering (CS minor), UVA
- Operations/Data Science Intern, Samsung Austin R&D Center
- ML, NLP at scale
- Music, Singing

"We can have a party, just listening to music"

Backup Slides

Limits & Future Work

- Investigate recall values vs parameters/time/...
  - More recall and precision comparison btw Brute-Force and LSH+MinHash
  - More comparison between different parameter comparisons
- Benchmark for batch processing:
  - Size vs Time
- More detailed benchmark on real-time processing
- More runs of experiments:
  - More representative data
- Optimize resource utilization

MapReduce version of MinHash+LSH

- Mapper side: for each post
  - Calculate min hash values
  - Create bands and band hashes

```
def calcMinHash(row,hash_funcs):
    return [min(map(lambda x:((x*hash_func[0]+hash_func[1])%mod_val), row)) for hash_func in hash_funcs]

def createBands(row,band_row_width):
    return [(i/band_row_width,hash(frozenset(row[i:i+band_row_width]))) for i in xrange(0,len(row),band_row_width)]
```

- Reducer side:
  - Get similar items grouped by (band id, band hash)
  - Calculate jaccard similarity on each item combination -> find the most similar pair

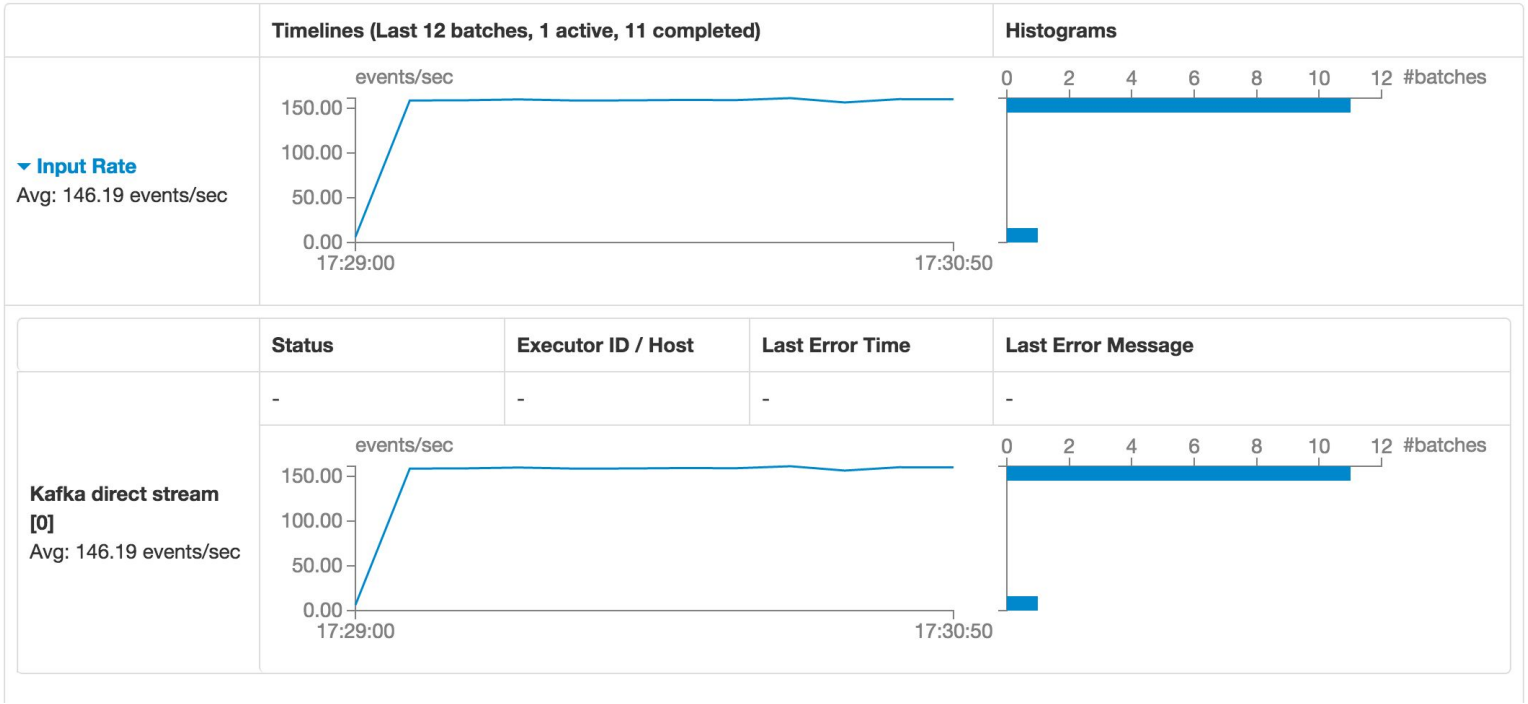
```
bands = minhash.map(lambda hash_list:(hash_list[0],createBands(hash_list[1],band_row_width),hash_list[1]))
band_hash_list = bands.flatMap(lambda x:[((i,x[1][i]),[x[0]]) for i in xrange(len(x[1]))])
band_hash_list = band_hash_list.reduceByKey(add).filter(lambda x:len(x[1])>1)
```

Threshold of MinHash + LSH

- Estimated Similarity Lower bound for each band:
  - ~(1/#bands)^(1/#rows)
- e.g. k =4, 2 bands and 2 rows. at least 0.70 similar
- Collision
- Higher k, more accurate, but slower

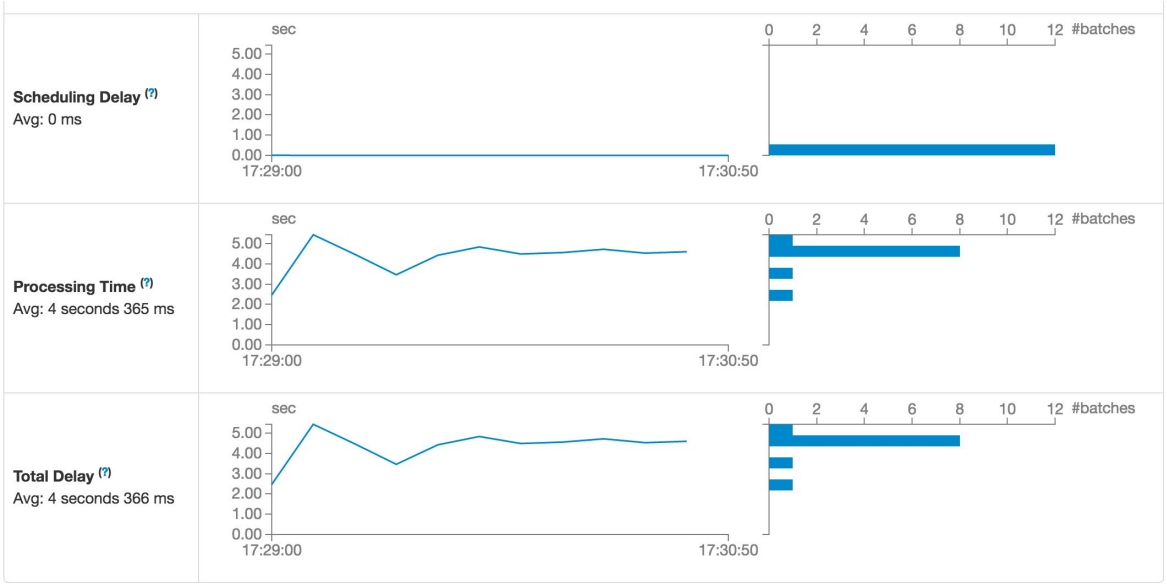
Streaming: Kafka

- Throughput: 146 events/sec, 10 ms time window



Streaming: Average Time

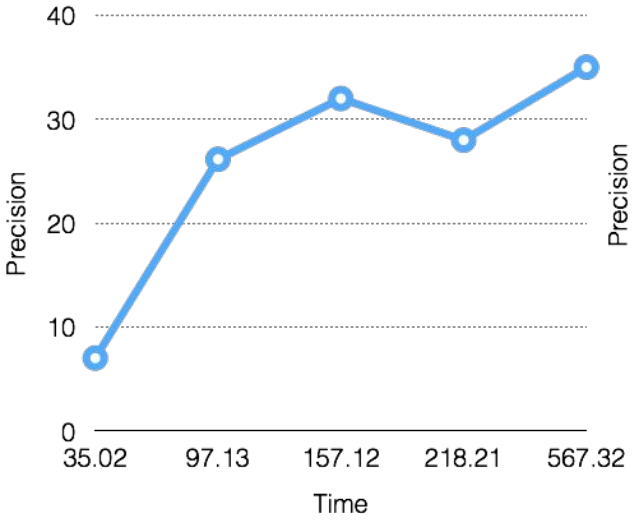
- Throughput:120 events/sec, 10 ms time window
- 8 sec/microbatch, 6 nodes, 1024 MB memory/node



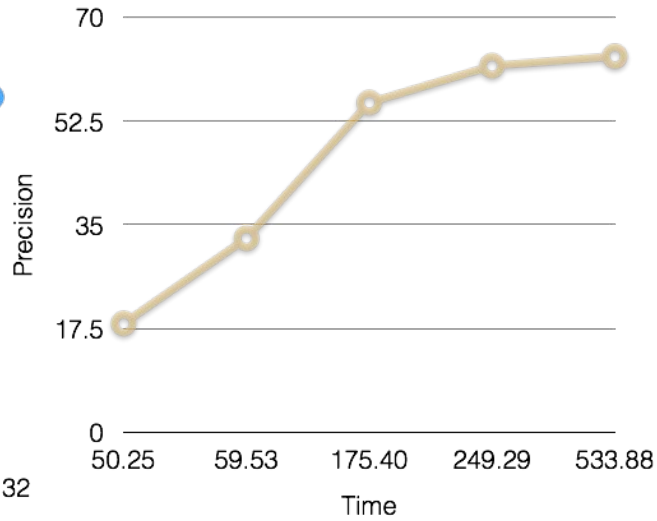
Streaming: Kafka

- Throughput: 315.30 events/sec, 10 ms time window

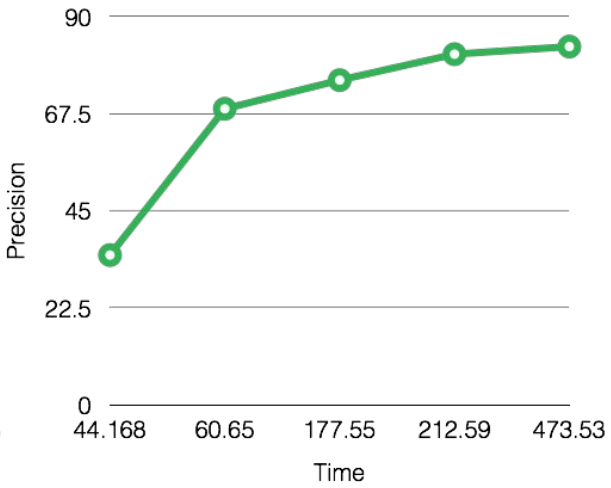
780k reddit posts, precision vs time



Threshold: 0.4-0.5

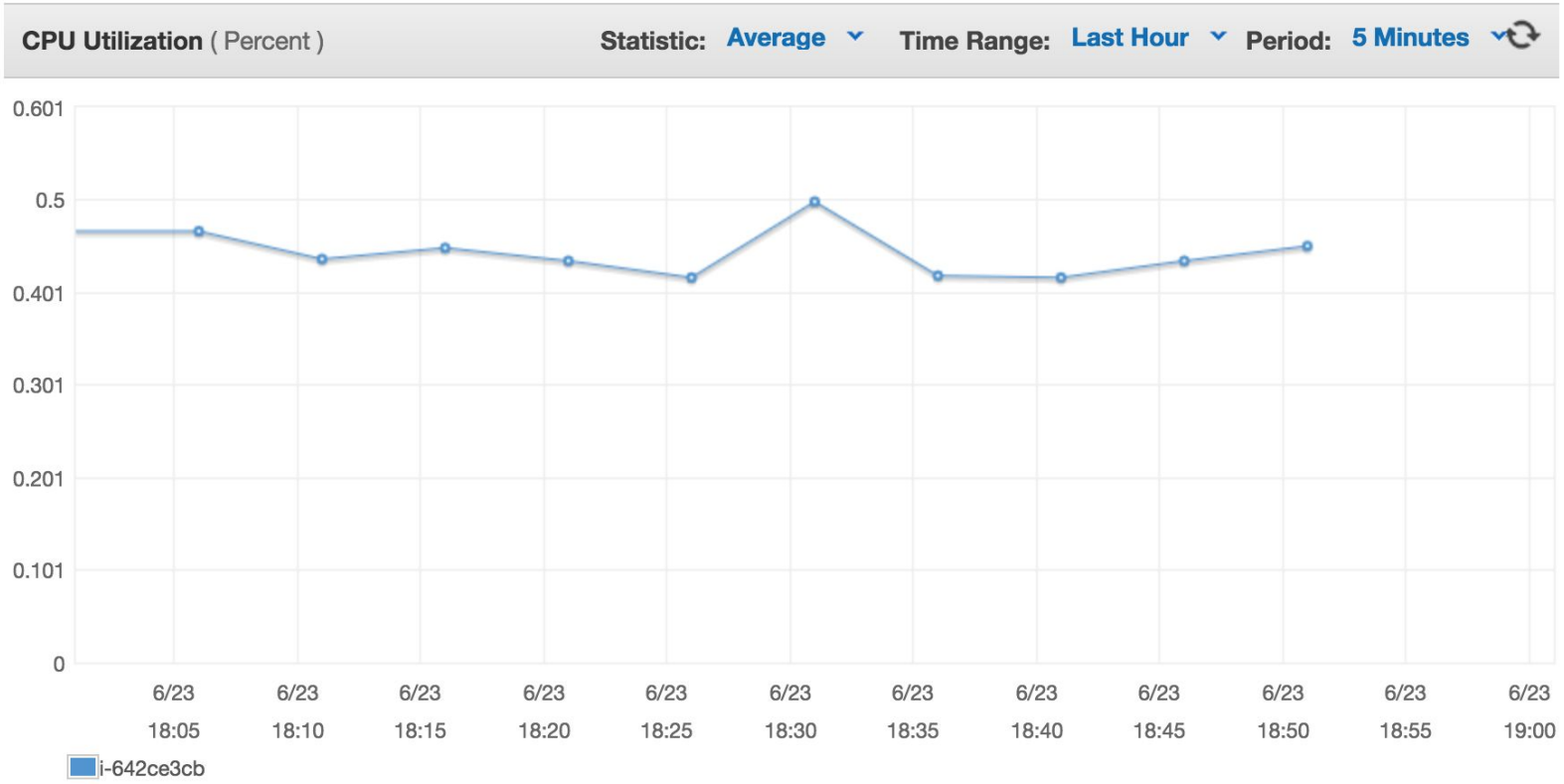


Threshold: 0.6-0.7



Threshold: 0.8-0.9

CPU usage


CPU Utilization (Percent)
Statistic: Average
Time Range: Last Hour
Period: 5 Minutes
0.601
0.5
0.401
0.301
0.201
0.101
0
6/23 18:05
6/23 18:10
6/23 18:15
6/23 18:20
6/23 18:25
6/23 18:30
6/23 18:35
6/23 18:40
6/23 18:45
6/23 18:50
6/23 18:55
6/23 19:00
i-642ce3cb

Task Diagram