



Cassandra for Twitter Trend Analysis

Mikio L. Braun
Leo Jugel

TU Berlin, twimpact

Cassandra Hackathon, #berlinbuzzwords
June 9, 2011





TWIMPACT: An Overview

- Real-time analysis of Twitter
- Trend analysis based on retweets
- Very high data rate (several million tweets per day, about 50 per second)



TWIMPACT: twimpact.jp

直近1時間 | 直近24時間 | 直近1週間(100) | 直近1ヶ月(100) | Top 100 ユーザー

RTによるTwitterトレン



検索 ログイン

RTによるTwitterトレンド [直近1時間]



kthepop 「トッサッキ〜ントっ、トサキント☆トサキント☆トサキント☆トッサッキ〜ントっ」 この声が正しく再生された人は公式RT
なう | +48 / 99回のRT | このTweetにRTする!



seishun0 RT 高校の時、ある男性産婦人医が性教育講演に。妊娠中絶の際必ず相手の男を手術に立ち合わせその殆どの男は途中で泡吹いて気絶したそう
な。講演では中絶用医療器具を並べて事細かに説明、騒いでた男子が段々静か
になってた。最後に「ここまでやらんと男はわからん」。
なう | +42 / 109回のRT | このTweetにRTする!



xhinata00 思い切り笑いたい人は見ろ。むしろみんな腹痛くなれ。RT回っ
たけど見てなかった俺がばかだった <http://bit.ly/avKNI9>
なう | +39 / 177回のRT | このTweetにRTする!



178tei このツイートを公式RTしない人はホモ

Top 20 ユーザー

- 大喜利 274 **ogiri_tweet**
- 言葉 194 **kotoba_bot**
- masason** 189
- meigenbot** 183
- 555hamako** 151



TWIMPACT: twimpact.com

TWIMPACT Real-time Monitoring v1.9 Follow us on Twitter

Home | JP Nuclear Disaster | JP Earthquake Relief | JP Earthquake | Libya | Egypt | Iran | About | ?

jpdisaster



Live Ticker

no translation

powered by Google™



boku_pakura あやこ @boku_pakura
 まじで——(ㄟ ≡ ㄟ)そんなんないし！
 すげー...東京はすげー... wRT @amyu3: 飲み会とかない？新大久保は佛のソジャがガンガンパしとるよ。あたしは交流会でパグになるが... RT @boku_pakura: 突然ナンパするわけにもいかんやん...
 just now ゆちよのノド仏



Vitamin717
 びたみん7さんとその飼い主の黒猫
 @moriyukogiin
 森さん、頑張って！ RT @moriyukogiin: 今日の部会の資料は酷かった。... 100mSvの被曝でもリスクは大したことがないという「国立がんセンター」、「日本医学放射線学会」、「ICRP」、そして「心配しすぎてしまうと、かえって心身の不調を起こす」。
 just now Tokyo



GRAN_ARAYA hideaki araya
 @mikihiyamada
 RT @masason: RT @mikihiyamada: @swissinfo スイス連邦保険局クリストフ・ミュリット氏「福島のような事故が起きては許さないと各閣僚が総辞職を...

Keywords

#INES IAEA **fukushima**
 nuclear power plant radiation reactor sievert シーベルト
 メルトダウン レベル 保安院
 冷却水 原子力 **原発**
 放射線 放射能 **東京**
 東京電力 東電 枝野 汚染 溶融
福島 線量 臨界 茨城 菅
 被ばく 被曝 被爆 避難 除染

Hashtags

#genpatsu
 #fukushima #genpatsu
 #save_fukushima #jishin
 #nhk_news #Japan
 #iwakamiyasumi2 #nuclear
 #seiji #news #IPHONE
 #nuclearJP #2ch #fb
 #iwakamiyasumi #Jobs
 #greentea #chiba #fail

User Mentions

save **fukushim**
 kikko no blog

Trending (rate/h)



Mentioned Locations



Application Profile

- Information about tweets, users, and retweets
- Text matching for non-API-retweets
- Retweet frequency and user impact
- Operation profile:

	get_slice (all)	get	get_slice (range)	batch_mutate (one row)	insert	batch_mutate	remove
Fraction	50.1%	6.0%	0.1%	14.9%	21.5%	6.8%	0.8%
Duration	1.1ms	1.7ms	0.8ms	0.9ms	1.1ms	0.8ms	1.2ms

Example: Simple Object Store

```
class Person {  
    long id;  
    String name;  
    String affiliation;  
}
```

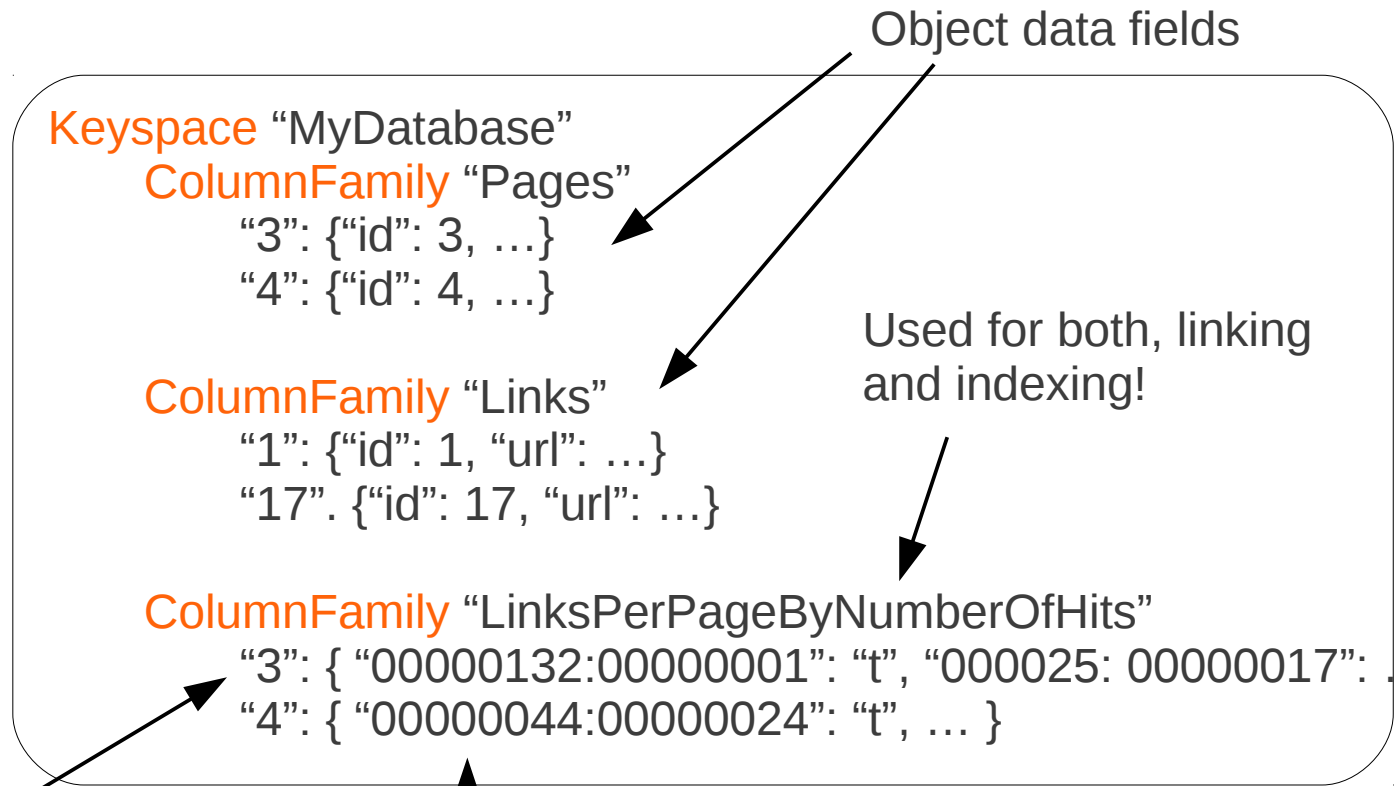
Convert fields to byte arrays



```
Keyspace "MyDatabase":  
    ColumnFamily "Person":  
        "1": {"id": "1", "name": "Mikio Braun", "affiliation": "TU Berlin"}
```

Example: Index

```
class Page {  
    long id;  
    ...  
    List<Links> links;  
}  
  
class Link {  
    long id;  
    ...  
    int numberOfHits;  
}
```



Here we exploit that columns are sorted by their names.

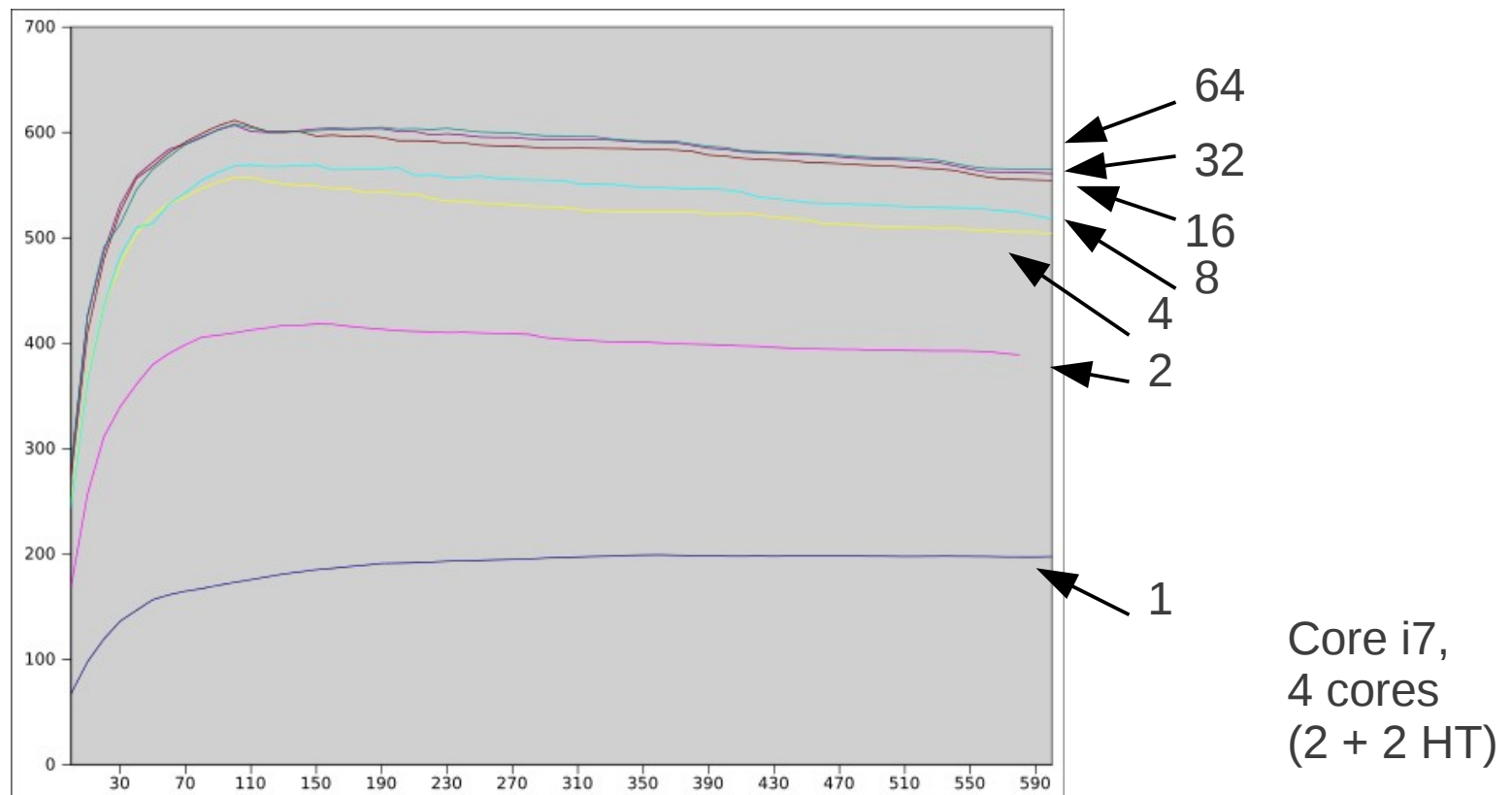
Of course, everything encoded in byte arrays, not ASCII

Practical Experiences with Cassandra

- Very stable
- Read operations relatively expensive
- Multithreading leads to a huge performance increase
- Requires quite extensive tuning
- Clustering doesn't automatically lead to better performance
- Compaction leads to performance decrease of up to 50%

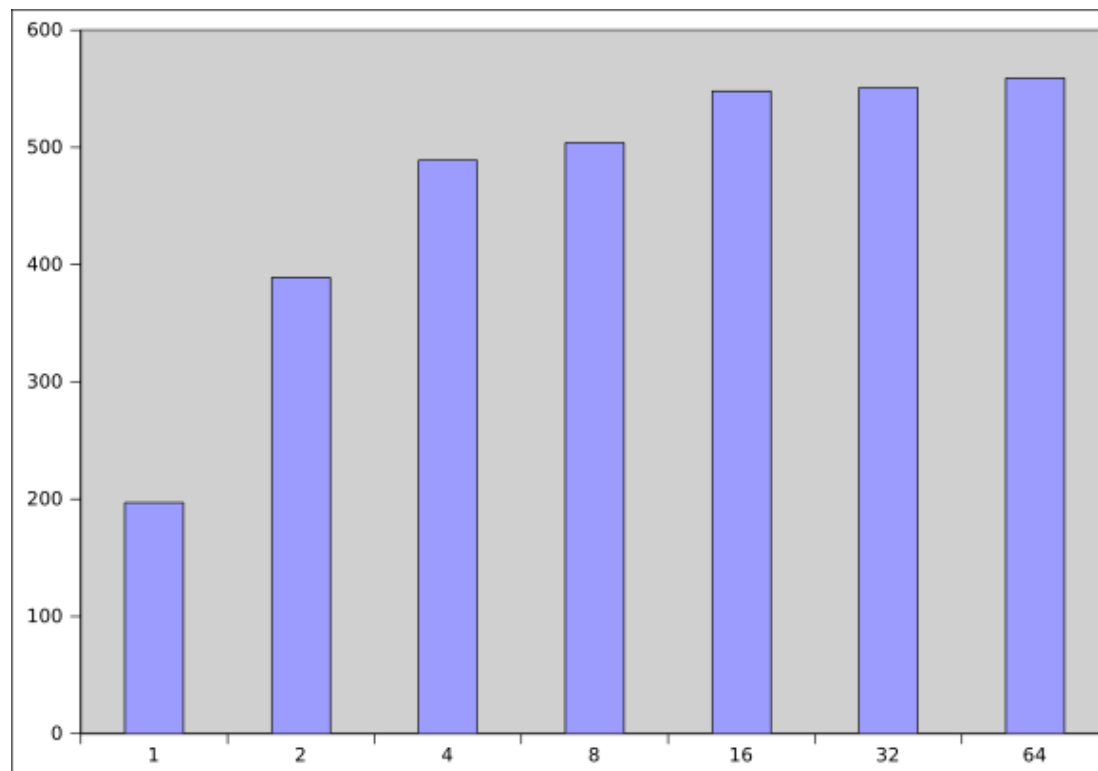
Performance through Multithreading

- Multithreading leads to much higher throughput
- How to achieve multithreading without locking support?



Performance through Multithreading

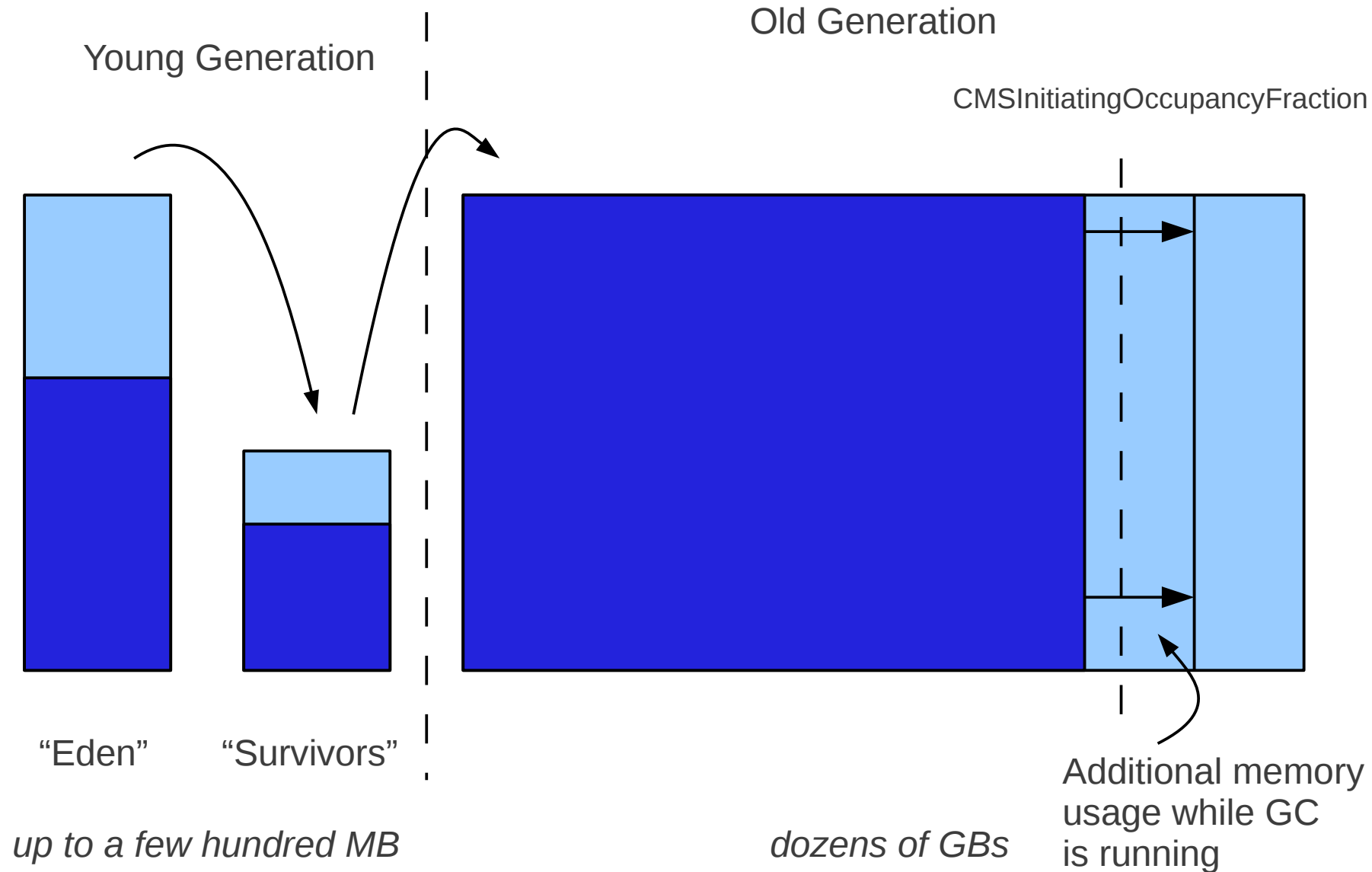
- Multithreading leads to much higher throughput
- How to achieve multithreading without locking support?



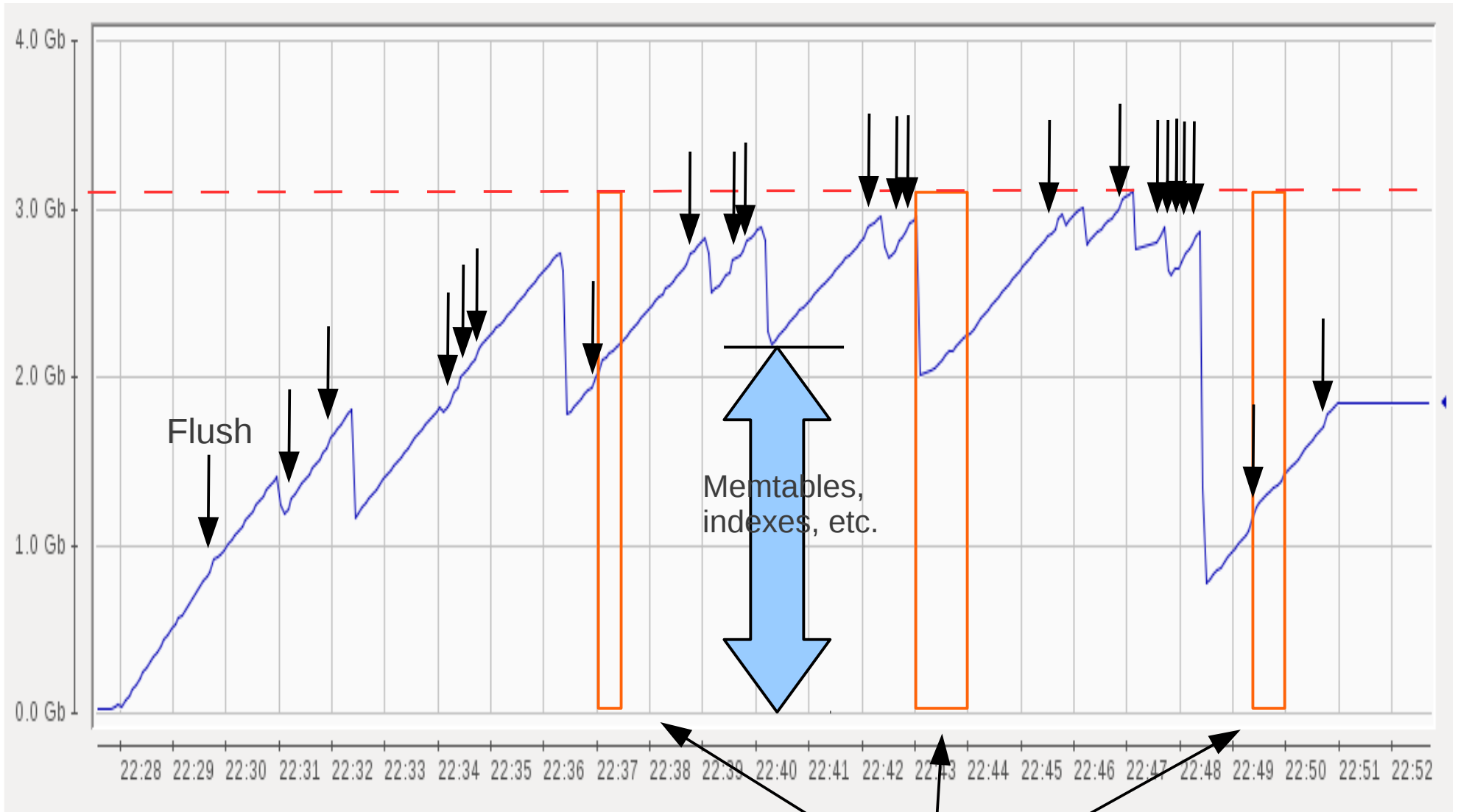
Cassandra Tuning

- Tuning opportunities:
 - Size of memtables, thresholds for flushes
 - Size of JVM Heap
 - Frequency and depth of compaction
- Where?
 - MemTableThresholds etc. in `conf/cassandra.yaml`
 - JVM Parameters in `conf/cassandra-env.sh`

Overview of JVM GC



Cassandra's Memory Usage



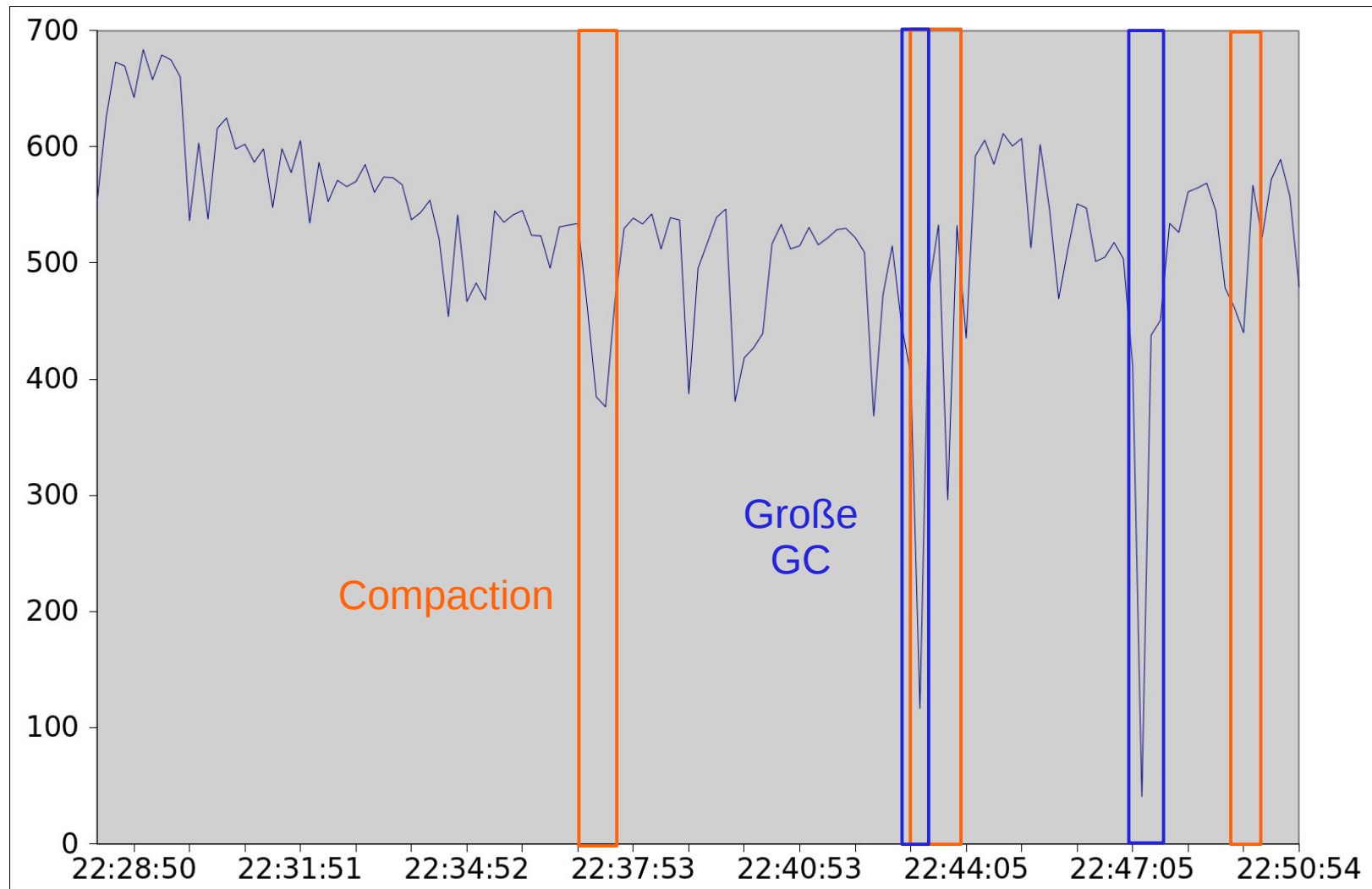
Size of Memtable: 128M, JVM Heap: 3G, #CF: 12

Compaction

Cassandra's Memory Usage

- Memtables may survive for a very long time (up to several hours)
 - are placed in old generation
 - GC has to process several dozen GBs
 - heap too small, GC triggered too late
⇒ “GC storm”
- Trade-off:
 - I/O load vs. memory usage
- Do not neglect compaction!

The Effects of GC and Compactions



Cluster vs Single Node

- Our set-up:
 - 1 node cluster with six-core CPU and RAID 5 with 6 hard disks
 - 4 node cluster with six-core CPU and RAID 0 with 2 hard disks
- Single node consistently performs **1,5-3 times better**.
- Possible causes:
 - Overhead through network communication/consistency levels, etc.
 - Hard disk performance significant
 - Cluster still too small
- Effectively available disk space:
 - 1 Cluster: $6 * 500 \text{ GB} = 3\text{TB}$ with RAID 5 = 2.5 TB **(83%)**
 - 4 Cluster: $4 * 1\text{TB} = 4\text{TB}$ with replication factor 2 = 2TB **(50%)**

Summary: Cassandra

- Platform which scales well
- Active user and developer community
- Read operations quite expensive
- For optimal performance, extensive tuning necessary
- Depending on your application, eventually consistent and lack of transactions/locking might be problematic.

Links

- Apache Cassandra <http://cassandra.apache.org>
- Apache Cassandra Wiki
<http://wiki.apache.org/cassandra/FrontPage>
- DataStax Dokumentation für Cassandra
<http://www.datastax.com/docs/0.7/index>
- My Blog: <http://blog.mikiobraun.de>
- Twimpact: <http://beta.twimpact.com>