



# **Cassandra** - Eine Einführung

Mikio L. Braun  
Leo Jügel

TU Berlin, twimpact

LinuxTag Berlin  
13. Mai 2011



# Was ist NoSQL?

- Für viele Webanwendungen sind “klassische Datenbanken” nicht die richtige Wahl:
  - Datenbank im Wesentlichen nur Speicher für Objekte
  - Konsistenz nicht unbedingt erforderlich
  - Viele gleichzeitige Zugriffe

# NoSQL im Vergleich

Klassische Datenbanken	NoSQL
Mächtige Querysprache	Sehr einfache Querysprache
Skaliert durch größere Server ("Scaling Up")	Skaliert auf einem Cluster ("Scaling Out")
Änderungen des Datenbankschemas aufwendig	Kein festes Datenbankschema
ACID: Atomicity, Consistency, Isolation, Durability	In der Regel nur "eventually consistent".
Transaktionen, Locking, etc.	In der Regel keine Unterstützung für Transaktionen o.ä.

# Brewer's CAP Theorem

- **CAP: Consistency, Availability, Partition Tolerance**
  - **Consistency:** Man erhält keine veralteten Daten.
  - **Availability:** Lese/Schreiboperationen sind immer möglich.
  - **Partition Tolerance:** Bei Netzwerk oder Rechnerausfällen bleiben die restlichen Garantien erhalten.
- Man kann nur jeweils zwei davon garantieren!

---

Gilbert, Lynch, *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*, ACM SIGACT News, Volume 33, Issue 2, June 2002



Homepage <http://cassandra.apache.org>

Programmiersprache	Java
--------------------	------

Geschichte	<ul style="list-style-type: none"><li>• Bei Facebook zur Indexsuche entwickelt, im Juli 2008 als Open Source freigegeben</li><li>• Seit März 2009 Apache Incubator</li><li>• Seit Februar 2010 Apache Top-Level</li></ul>
------------	---

Haupteigenschaften	<ul style="list-style-type: none"><li>• strukturierter Key-Value-Store</li><li>• “<b>eventually consistent</b>”</li><li>• völlig gleichberechtigte Knoten</li><li>• Cluster ohne Neustart modifizierbar</li></ul>
--------------------	---

Support	DataStax ( <a href="http://datastax.com">http://datastax.com</a> )
---------	--

Lizenz	Apache 2.0
--------	------------

# Version 0.6.x und 0.7.x

- Wichtigsten Änderungen in 0.7.x
  - Konfigurationsdateien in YAML statt XML
  - Schemamodifikationen (ColumnFamilies) im laufenden Betrieb
  - Erste Ansätze für Sekundärindexe.
- Allerdings auch anfängliche **Stabilitätsprobleme**

# Inspirationen für Cassandra

- **Amazon Dynamo**
  - Cluster ohne dezidierte Masterknoten
  - Peer-to-Peer Erkennung, HintedHintoff, etc.
- **Google BigTable**
  - Datenmodell
  - Benötigt zentrale Masterknoten
  - Bietet wesentlich mehr Möglichkeiten der Kontrolle:
    - welche Daten gemeinsam gespeichert werden sollten
    - On-the-fly Kompression, etc.

# Installation

- tar.gz von <http://cassandra.apache.org/download/> herunterladen
- In Verzeichnis auspacken
- ./conf enthält Konfigurationsdateien
- ./bin/cassandra -f startet Cassandra, Ctrl-C zum stoppen



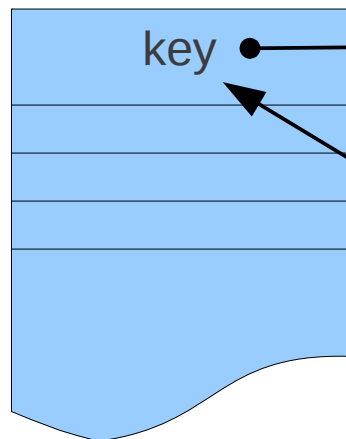
# Konfiguration

- Datenbank
  - Version 0.6.x: `conf/storage-conf.xml`
  - Version 0.7.x: `conf/cassandra.yaml`
- JVM Parameter
  - Version 0.6.x: `bin/cassandra.in.sh`
  - Version 0.7.x: `conf/cassandra-env.sh`

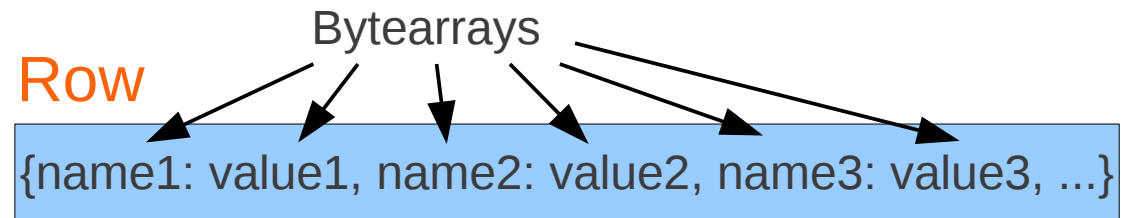
# Cassandras Datenmodell

**Keyspace** (= Datenbank)

**Column Family** (= Tabelle)



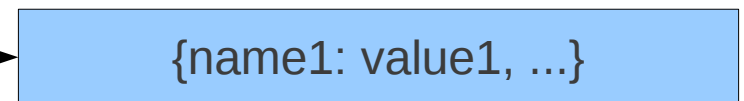
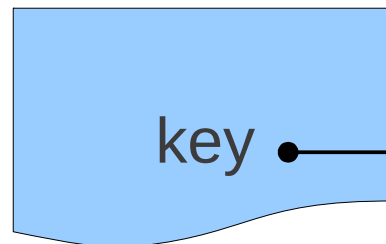
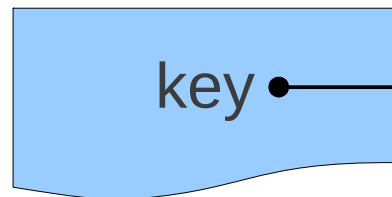
**Row**



**Column**

Sortiert nach Namen!

**Super Column Family**



# Beispiel: Einfacher Objektspeicher

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

Felder in Bytearrays umwandeln

Keyspace "MyDatabase":

ColumnFamily "Person":

"1": {"id": "1", "name": "Mikio Braun", "affiliation": "TU Berlin"}

# Beispiel: Index

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

Keyspace "MyDatabase"  
ColumnFamily "Pages"

"3": {"id": 3, ...}  
"4": {"id": 4, ...}

ColumnFamily "Links"

"1": {"id": 1, "url": ...}  
"17": {"id": 17, "url": ...}

ColumnFamily "LinksPerPageByNumberOfHits"

"3": { "00000132:00000001", "000025: 00000017": ... }  
"4": { "00000044:00000024", ... }

Datenfelder der Objekte

Sowohl für die Verlinkung,  
als auch zum Indizieren!

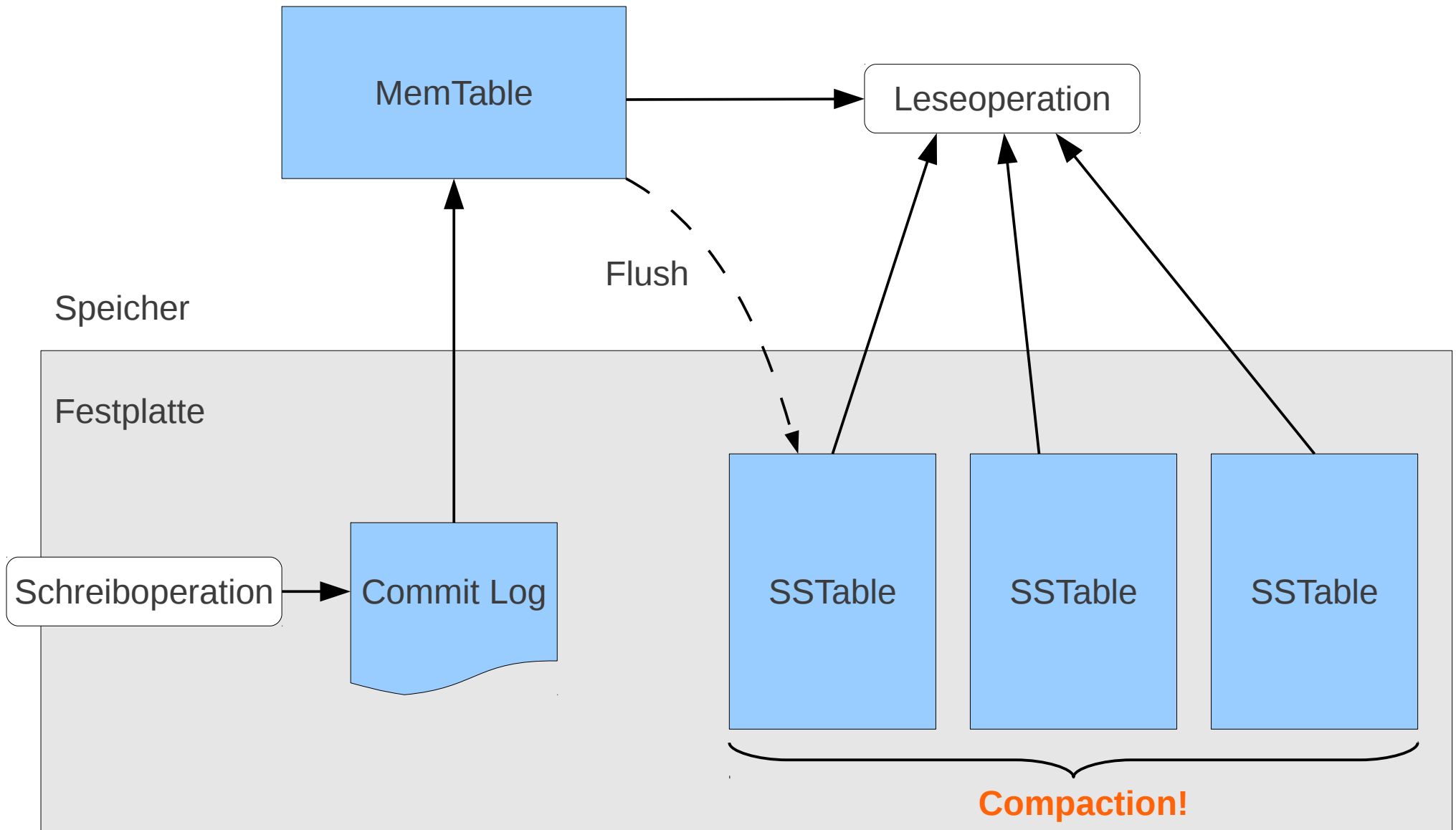
Hier wird ausgenutzt,  
dass Columns nach  
Namen sortiert werden!

Natürlich alles auf Bytearrayebene!

# Benötigt man SuperColumnFamilies?

- Meistens kann man SuperColumnFamilies durch mehrere ColumnFamilies ersetzen.
- Da die SuperColumnFamilies die Implementation und das Protokol verkomplizieren gibt es auch Stimmen, die sie ganz abschaffen wollen... .

# Cassandras Architektur



# Cassandras API

- THRIFT-basierte API

Leseoperationen	
get	Einzelne Column
get_slice	Reihe von Columns
multiget_slice	Reihe von Columns in verschiedenen Rows
get_count	Anzahl von Columns
get_range_slice	Reihe von Columns in einer Reihe von Rows
get_indexed_slices	Reihe von Columns mit Index

Schreiboperationen	
insert	Einzelne Column
batch_mutate	Mehrere Columns in verschiedenen Rows
remove	Einzelne Column
truncate	Ganze Column Family

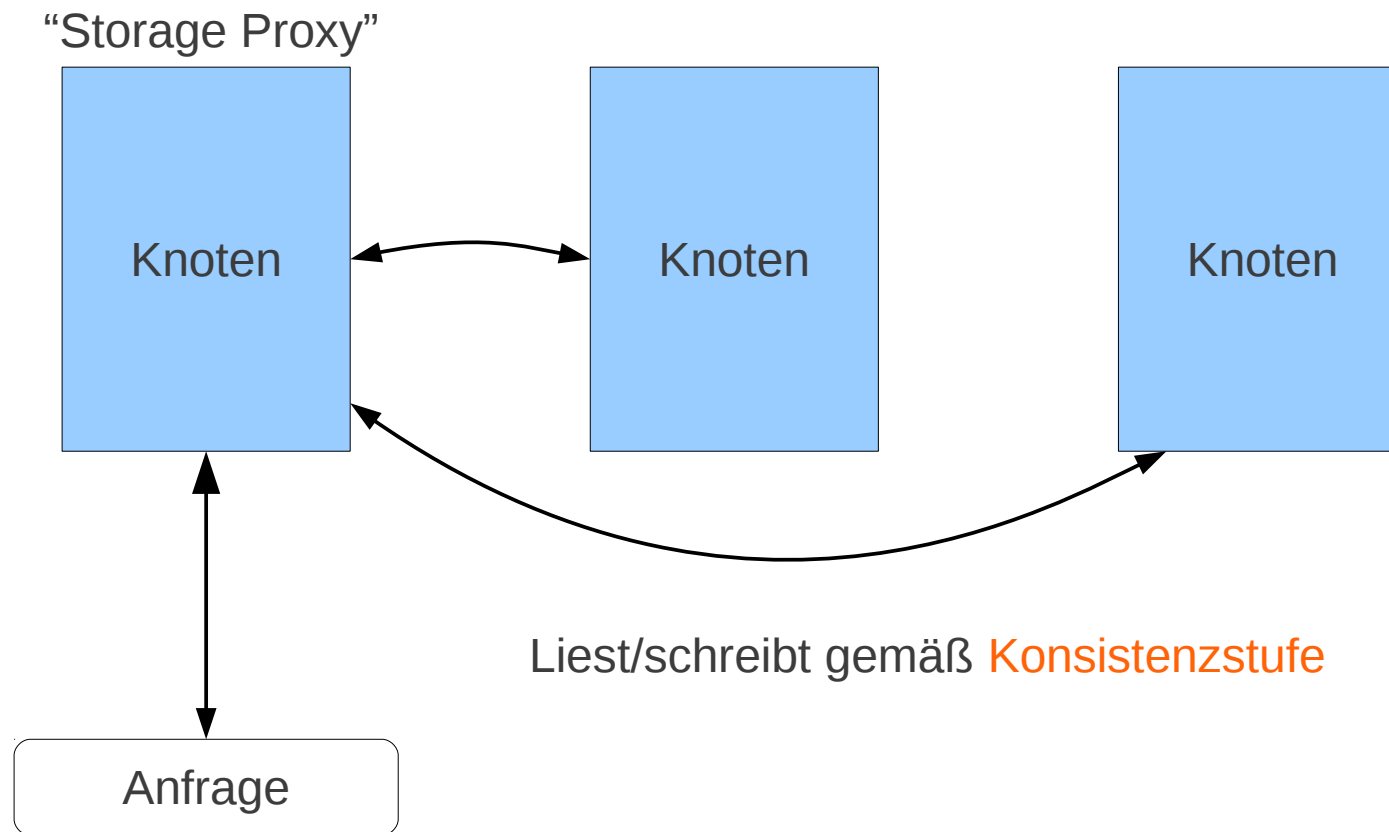
## Sonstige

login, describe\_\*, add/drop column family/keyspace

erst in 0.7.x

# Cassandra im Cluster

- Völlig gleichberechtigte Knoten
- Zum Bootstrappen ein Seed nötig





# Konsistenzstufen und Replikationsfaktor

- Replikationsfaktor: Auf wie vielen Knoten werden die Daten gespeichert?
- Konsistenzstufen:

Konsistenzlevel	
ANY	Ein Knoten hat die Operation bekommen, selbst im Fall von HintedHandoff
ONE	Auf einem Knoten wurde die Operation ausgeführt.
QUORUM	Operation für die Mehrzahl der Knoten ausgeführt / Neuestes Resultat wird zurückgegeben
LOCAL_QUORUM	QUORUM im lokalen Datenzentrum
GLOBAL_QUORUM	QUORUM im globalen Datenzentrum
ALL	Warte, bis alle Knoten die Operation ausgeführt haben.

# Verhalten bei Ausfall

- Solange Anforderungen der Konsistenzstufe erfüllt werden können, ist alles in Ordnung.
- **Hinted Handoff:**
  - Eine Schreiboperation für einen ausgefallenen Knoten wird auf einem anderen Knoten gemerkt.
  - Diese Daten sind anschließend nicht lesbar!
- **Read Repair:**
  - Selbst nachdem die Leseoperation abgeschlossen ist, werden die Daten überprüft und ggf. aktualisiert.

# Bibliotheken

Python	Pycassa: <a href="http://github.com/pycassa/pycass">http://github.com/pycassa/pycass</a> Telephus: <a href="http://github.com/drifftx/Telephus">http://github.com/drifftx/Telephus</a>
Java	Datanucleus JDO: <a href="http://github.com/tnine/Datanucleus-Cassandra-Plugin">http://github.com/tnine/Datanucleus-Cassandra-Plugin</a> Hector: <a href="http://github.com/rantav/hector">http://github.com/rantav/hector</a> Kundera <a href="http://code.google.com/p/kundera/">http://code.google.com/p/kundera/</a> Pelops: <a href="http://github.com/s7/scale7-pelops">http://github.com/s7/scale7-pelops</a>
Grails	grails-cassandra: <a href="https://github.com/wolpert/grails-cassandra">https://github.com/wolpert/grails-cassandra</a>
.NET	Aquiles: <a href="http://aquiles.codeplex.com/">http://aquiles.codeplex.com/</a> FluentCassandra: <a href="http://github.com/managedfusion/fluentcassandra">http://github.com/managedfusion/fluentcassandra</a>
Ruby	Cassandra: <a href="http://github.com/fauna/cassandra">http://github.com/fauna/cassandra</a>
PHP	phpcassa: <a href="http://github.com/thobbs/phpcassa">http://github.com/thobbs/phpcassa</a> SimpleCassie: <a href="http://code.google.com/p/simpletools-php/wiki/SimpleCassie">http://code.google.com/p/simpletools-php/wiki/SimpleCassie</a>

Oder was selbstgebautes direkt auf THRIFT <http://thrift.apache.org/> :)



# TWIMPACT: Eine Anwendung

- Echtzeitanalyse von Twitter
- Trendanalyse basierend auf Retweets
- Sehr hohes Datenaufkommen (mehrere Millionen Tweets am Tag, ca. 50/s)



# 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トレンド



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。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回って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

1. **大喜利。** **ogiri\_tweet**  
274

2. **言葉** **kotoba\_bot**  
194

3. **masason**  
189

4. **meigenbot**  
183

5. **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——(°Д°≡°Д°)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100%30%がガンガンナンパしとるよ。あたしは交流会で70%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

ゆちょのノド仏



**Vitamin717**  
びたみん717さんとその飼い主の黒猫  
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「心配しすぎてしまおうと、かえって心身の不調を起こす」。

just now

Tokyo



**GRAN\_ARAYA** hideaki araya  
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...」

### 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

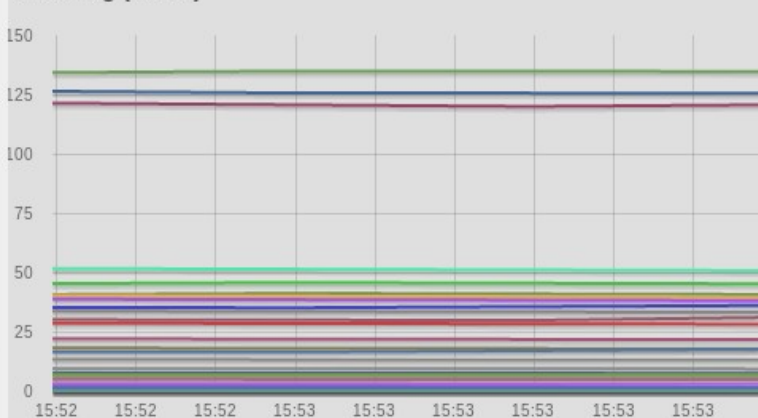
### Hashtags

#genpatsu  
#fukushima #genpatsu  
#save\_fukushima #jishin  
#nhk\_news #Japan  
#iwakamiyasumi2 #nuclear  
#seiji #news #IPHONE  
#nuclearJP #2ch #fb  
#iwakamiyasumi #Jobs  
#greentea #chiba #fail

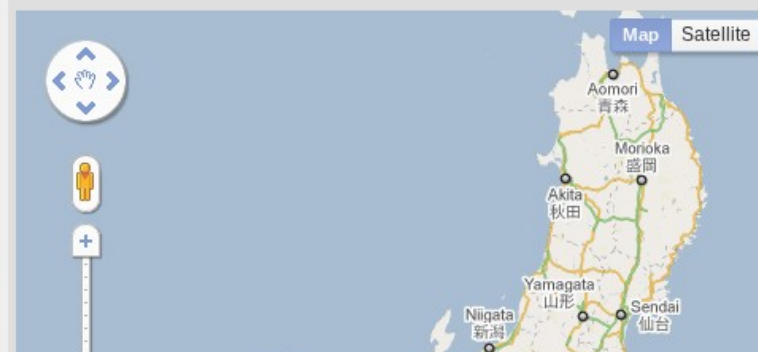
### User Mentions

save fukushima  
kikko no blog

### Trending (rate/h)



### Mentioned Locations



# Anwendungsprofil

- Informationen über Tweets, User, Retweets
- Textmatching für nicht-API-Retweets
- Retweetshäufigkeiten und Userimpact
- Operationsprofil:

	get_slice (alle)	get	get_slice (Bereich)	batch_mutate (eine Row)	insert	batch_mutate	remove
Anteil	50.1%	6.0%	0.1%	14.9%	21.5%	6.8%	0.8%
Dauer	1.1ms	1.7ms	0.8ms	0.9ms	1.1ms	0.8ms	1.2ms



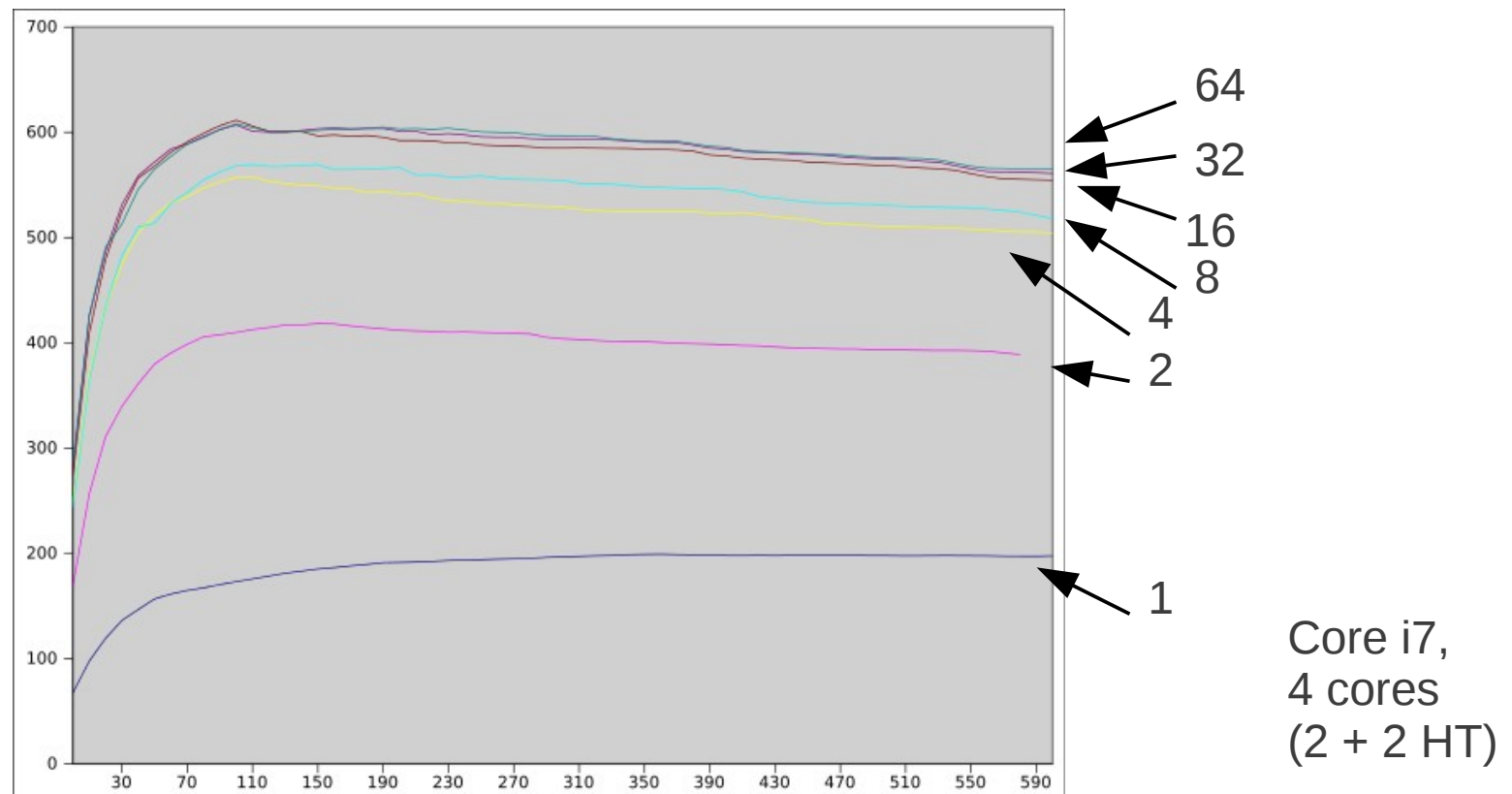
# Praktische Erfahrungen mit Cassandra

- Sehr stabiler Betrieb
- Leseoperationen verhältnismäßig teuer
- Multithreading bringt große Performanceverbesserung
- Relativ aufwendiges Tuning erforderlich
- Clusterbetrieb bringt nicht unbedingt Verbesserung
- Compaction führt zu bis zu 50% Performanceeinbruch



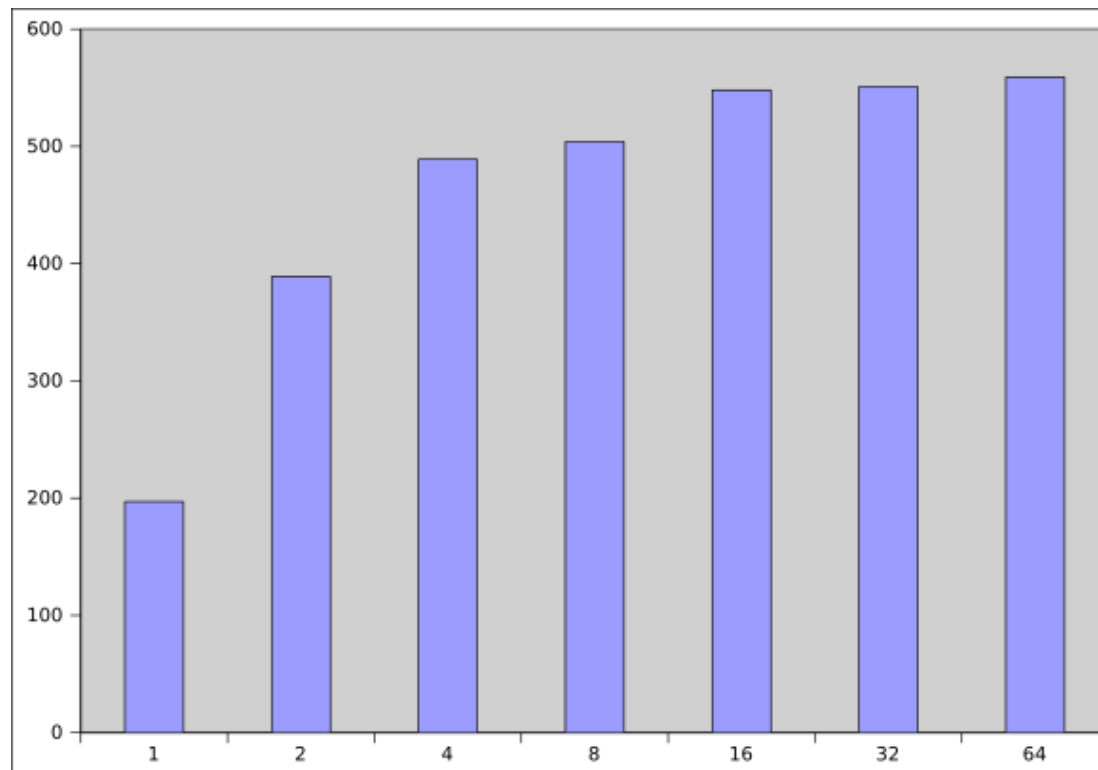
# Performance durch Multithreading

- Multithreading bringt erheblich mehr Performance.
- Multithreading ohne Transaktionen/Locking?



# Performance durch Multithreading

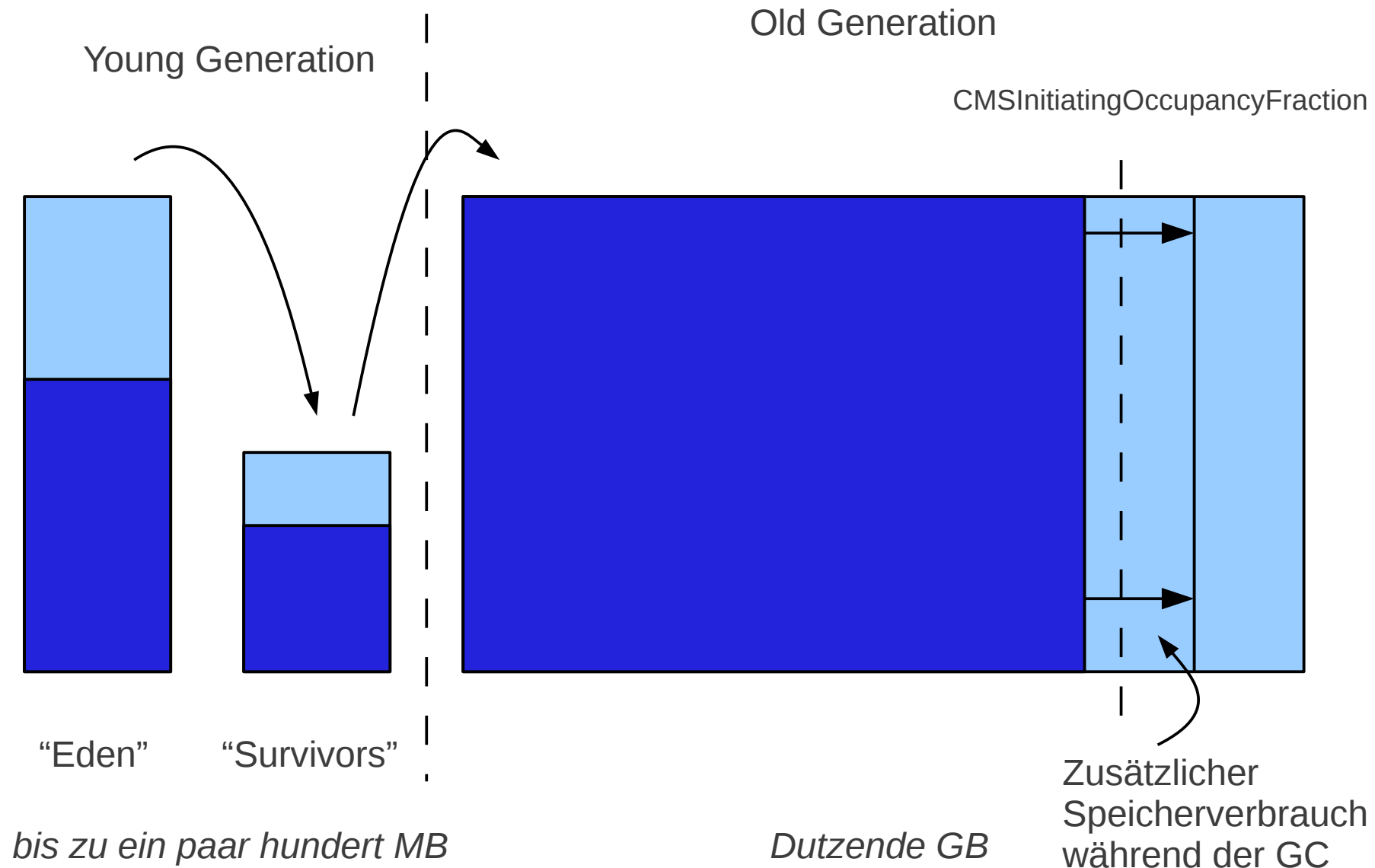
- Multithreading bringt erheblich mehr Performance.
- Multithreading ohne Transaktionen/Locking?



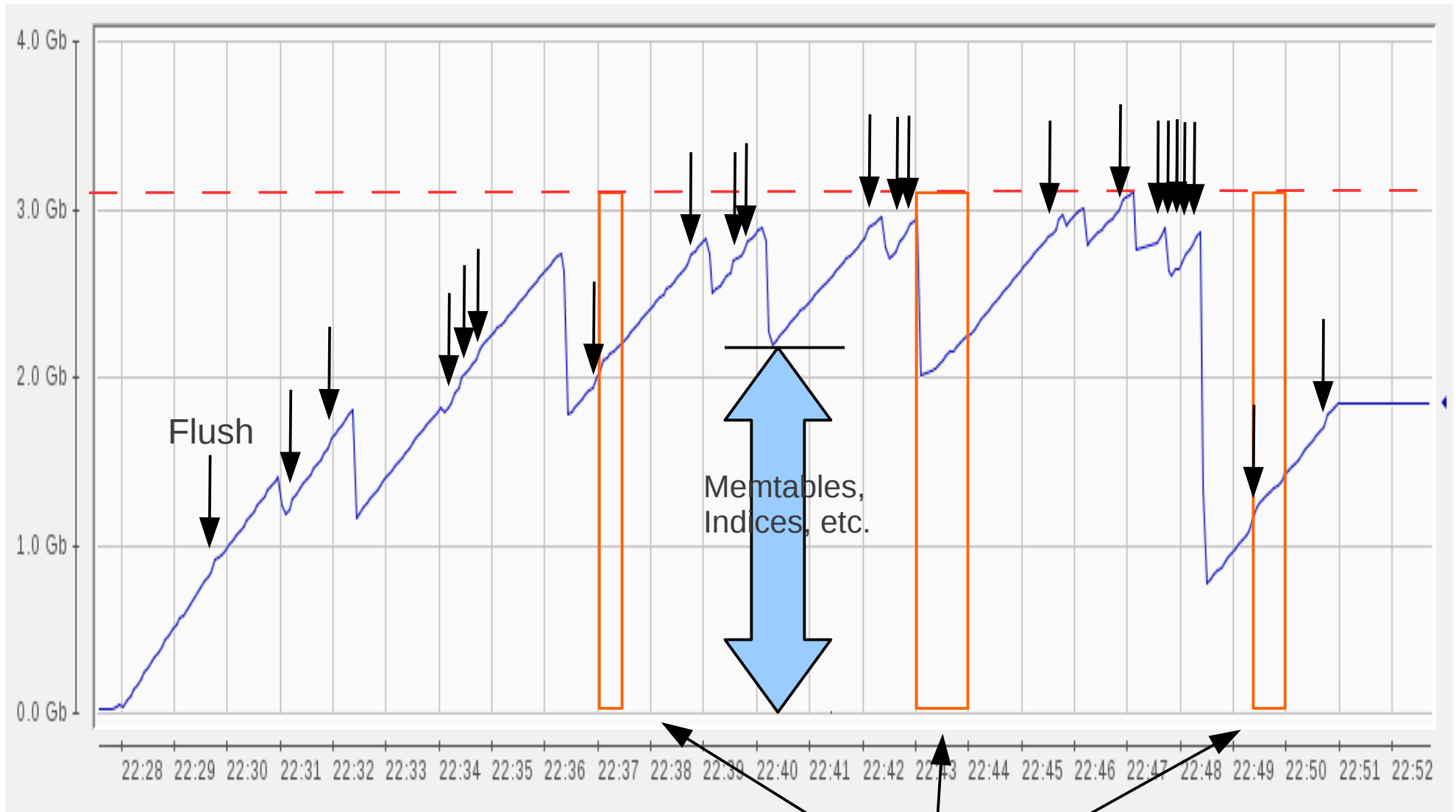
# Cassandra Tuning

- Tuningmöglichkeiten:
  - Größe der Memtables, Thresholds für Flushes
  - Größe des JVM Heaps
  - Häufigkeit, Tiefe der Compaction
- Wo?
  - MemTableThresholds etc. in `conf/cassandra.yaml`
  - JVM Parameter in `conf/cassandra-env.sh`

# Übersicht über die GC der JVM



# Cassandras Speicherverhalten



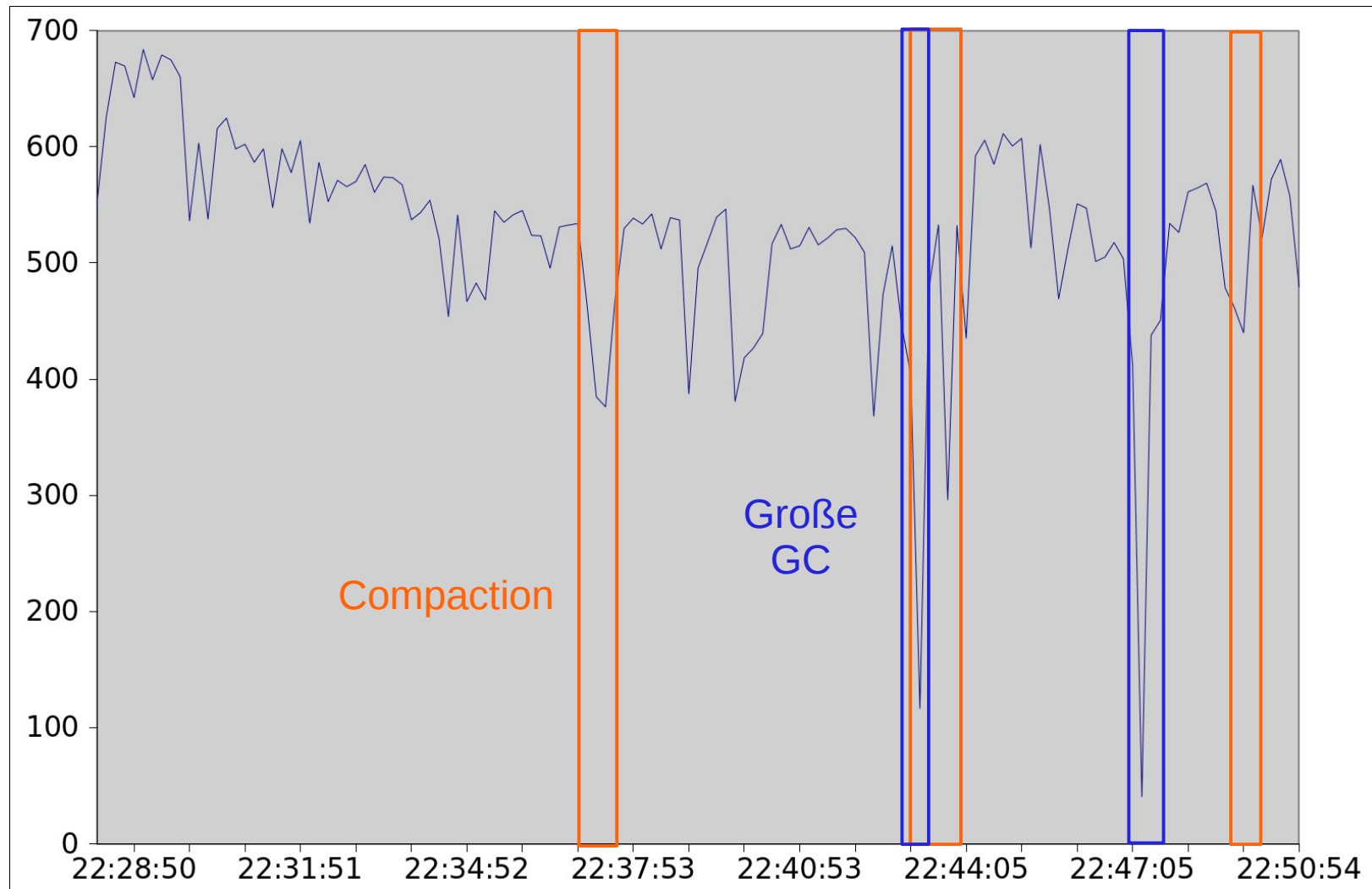
Memtablegröße: 128M, JVM Heap: 3G, #CF: 12

Compaction

# Cassandras Speicherverhalten

- Memtables überleben extrem lange (mehrere Stunden)
  - Landen in der Old Generation
  - Bei GC müssen mehrere Dutzend GB aufgeräumt werden.
  - Heap zu klein, oder GC zu spät ausgelöst  
⇒ “GC storm”
- Tradeoff:
  - I/O Last vs. Speicherbrauch
- Compaction nicht vernachlässigen!

# Die Auswirkungen von GC und Compaction



# Cluster und Einzelknoten

- Im Vergleich:
  - 1 Cluster mit six-core CPU und RAID 5 aus 6 Festplatten
  - 4 Cluster mit six-core CPU mit RAID 0 aus 2 Festplatten
- Einzelcluster lieferte konsistent **1,5-3 Mal mehr Performance**.
- Gründe:
  - Overhead durch Netzwerkkommunikation/Konsistenzstufen, etc.
  - Festplattenperformance ausschlaggebend.
  - Cluster noch zu klein
- Effektiv verfügbarer Plattenplatz:
  - 1 Cluster:  $6 * 500 \text{ GB} = 3\text{TB}$  bei RAID 5 = 2.5 TB **(83%)**
  - 4 Cluster:  $4 * 1\text{TB} = 4\text{TB}$  bei Replikationsfaktor 2 = 2TB **(50%)**



# Alternativen

- MongoDB, CouchDB, redis, sogar memcached... .
- Persistenz: Festplatte oder RAM?
- Replikation: Master/Slave oder Peer-to-Peer?
- Sharding?
- Trend zu komplexeren Querysprachen (Javascript), Map-Reduce Operationen, etc.

# Fazit: Cassandra

- Gut skalierende Plattform
- Aktive Benutzer- und Entwicklercommunity
- Leseoperationen relativ teuer
- Für optimale Leistung detailliertes Tuning nötig
- Je nach Anwendung “eventually consistent” und fehlende Transaktionen/Locking problematisch.

# 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>
- Mein Blog: <http://blog.mikiobraun.de>
- Twimpact: <http://beta.twimpact.com>