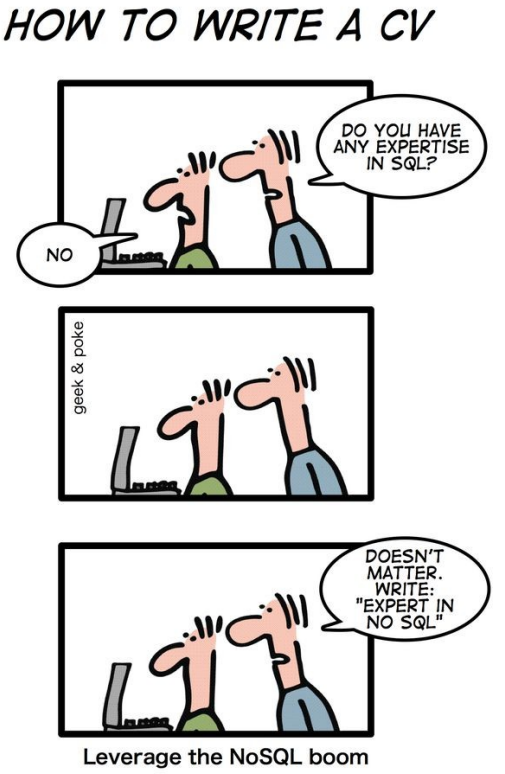
BGD2

NoSQL - Motivation

Andreas Scheibenpflug

NoSQL

- NoSQL DBs sind Datenbanken, die kein relationales Datenmodell verwenden
- Oft finden sie Anwendung bei
  - großen Datenmengen
  - Echtzeitsystemen
  - oder wenn Performance, Skalierbarkeit oder Ausfallsicherheit wichtiger ist als Konsistenz



http://geek-and-poke.com/geekandpoke/2011/1/27/nosql.html

Etwas Geschichte < 1972

- Anwendungen implementieren ihre eigene „Datenbank“
  - Nachteile: Fehleranfällig, Kostenintensiv
- Ende 60er – Erste Datenbankmanagementsysteme (DMBS)
  - IBM Information Management System (IMS)
    - Hierarchisches Datenmodell
  - Integrated Database Management System (IDMS)
    - Netzwerkdatenmodell

Etwas Geschichte < 2005

- 1970 Edgar F. Codd: Relationales Datenmodell, Normalformen
- 70er: Transaktionen (ACID) und SQL
- 1974 IBM System R
  - Vorgänger DB2
  - Vorgänger SQL
- 1978 Oracle
  - Erste kommerzielle DB die SQL unterstützt
- 1989 Postgres, 1995 MySQL,…

Etwas Geschichte > 2005

- Google
  - 2003 Google File System (GFS)
  - 2004 Map Reduce
  - 2006 Big Table
    - Spalten-basiertes Datenmodell (Wide column store)
    - Wird für viele Google Produkte inkl. Suche (Web Indexing) verwendet
  - 2006 Hadoop
    - Open Source Map Reduce Implementierung

Etwas Geschichte > 2005

- Amazon
  - 2008 Dynamo
  - Reaktion auf die Ausfälle von Amazons Weihnachtsgeschäft 2004
  - Verteilte Key-Value DB (Distributed Hash Table - DHT)
  - Radikal anders zu klassischen Datenbanken
    - Consistent Hashing zur Partitionierung der Daten
    - Sloppy Quorum, Eventual Consistency
    - Gossipping, keinen Master, jederzeit skalierbar, hochverfügbar
    - Keine Abfragesprache, keine komplexen Datenmodelle

Etwas Geschichte > 2005

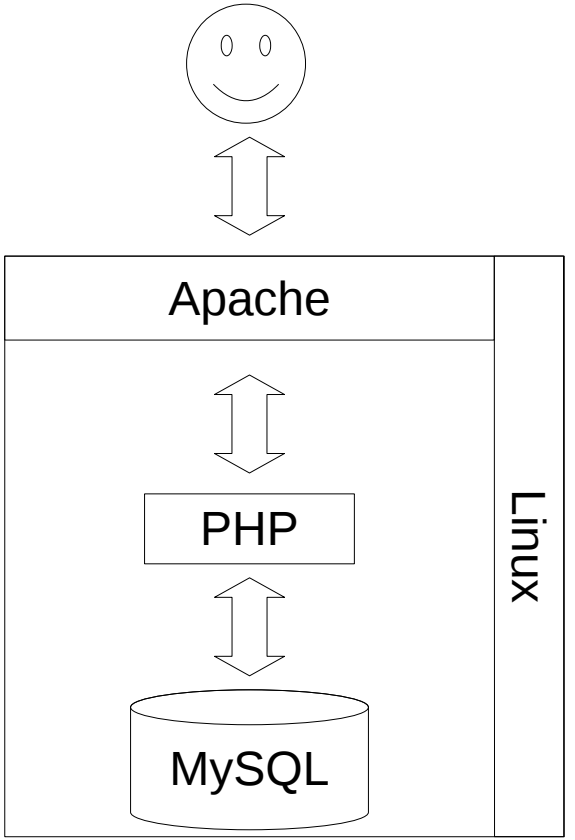
- 2005 CouchDB
  - Dokumenten-basiertes Datenmodell (JSON)
- 2008 Apache Cassandra
  - Spalten-basiertes Datenmodell
- 2009 MongoDB
  - Dokumenten-basiertes Datenmodell (JSON)
- 2011 Riak
  - Key-Value Store (Dynamo-inspired)

Klassische Datenbank Architektur

- In den 80ern/Beginn 90ern dominierten Mainframes mit kommerziellen Datenbanksystemen
- Zum Beispiel: Mainframe auf dem die Datenbank mit den Geschäftsdaten (Kunden, Bestellungen, Lagerwirtschaft, usw.) gespeichert ist
  - Benutzer verwendeten Terminals oder später PCs, um sich mit dem Mainframe zu verbinden und Daten zu bearbeiten
- Wir konzentrieren uns im folgenden auf Webanwendungen (Ende 90er Jahre)

LAMP

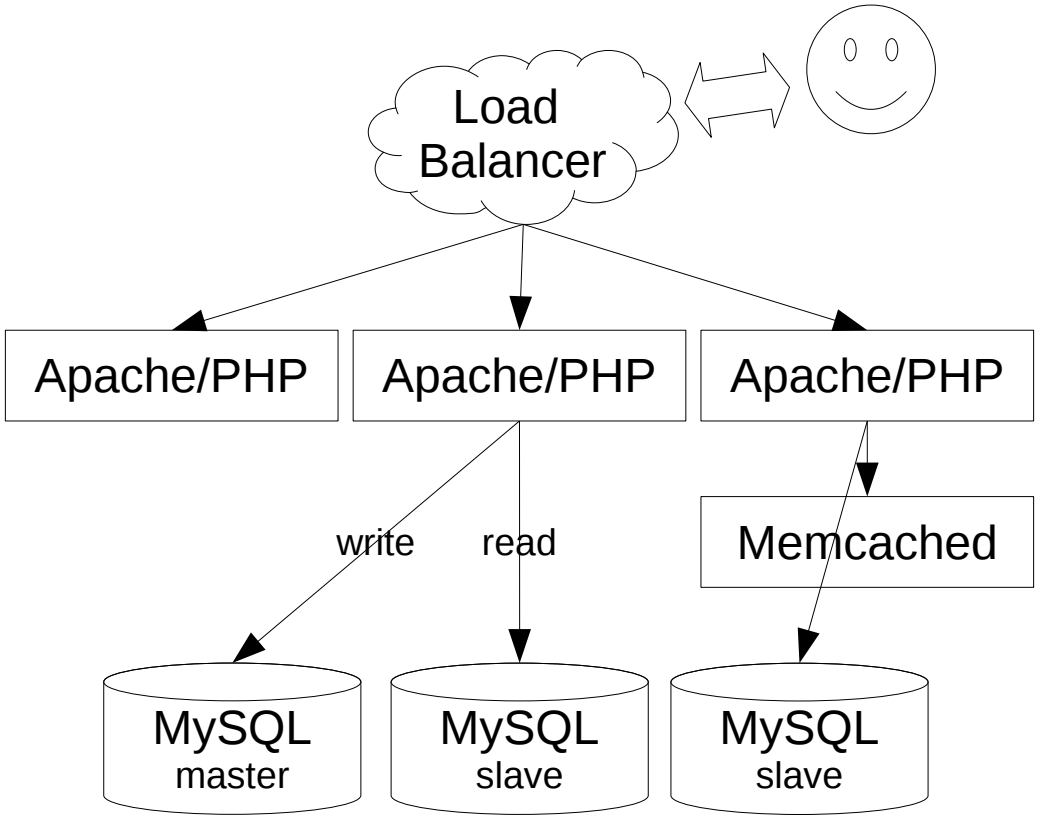
- Komponenten
  - Linux
  - Apache
  - MySQL
  - PHP
- Probleme?



LAMP – Lösungen?

- Anstieg von Last - Skalierung
  - Load Balancer, mehrere Apaches, mehr Hardware
  - Memcached
  - MySQL Replication
- Ausfallsicherheit
  - MySQL Replication
- Anstieg der Datenmenge
  - Sharding

LAMP+


Load
Balancer
Apache/PHP
Apache/PHP
Apache/PHP
write
read
Memcached
MySQL
master
MySQL
slave
MySQL
slave

Database Sharding

- Aufteilung der Daten auf mehrere Datenbankknoten
- Möglichkeit zur horizontalen Skalierung, aber:
  - *The best approach for sharding MySQL tables is to not do it unless it is totally unavoidable to do it.*
    - https://stackoverflow.com/a/5617449
  - *Step 1 – Shard database. Step 2 – shoot yourself.*
    - Twitter, @Dmitriy

Motivation NoSQL

- Performance
- Verwendung von Commodity Hardware
- Horizontale Skalierung
  - Plötzlicher Anstieg an Benutzern, Datenmenge, Last
- Schemalos
  - Daten müssen keine vordefinierte Struktur aufweisen
- Ausfallsicherheit
  - Fault-tolerant systems: System funktioniert noch bei Ausfällen von Netzwerkkomponenten und mehreren Knoten
- Minimierung der objektrelationalen Unverträglichkeit (Object-relational impedance mismatch)

NoSQL - The Dark Side

- Einschränkungen bei Konsistenz und Transaktionen
- Fehlende Abfragesprachen (SQL)
- Teilweise fehlende Funktionalität (im Vergleich zu relationalen Datenbanken)
  - Secondary Indizes
  - Range Queries
  - Beziehungen (vor allem n:m)
- Schemalos
- Beabsichtigte Redundanz von Daten