Datastore Axes
Choosing your scalability direction

Nicolai Plum
Genesis - Autonomy

- Congratulations, Developers: you have autonomy to design and build your own product.
- Developers must tell Database Engineering:
  - How their product data will grow
  - How its database needs will change
Genesis - Developers

● Help, I have responsibility for build my own product successfully!
● I don’t know what I need!
● I can’t predict the future!
Our database services should be a well-defined product

Clearly defined capabilities

... and compromises
Datastore

- A system which stores and retrieves data
  - Small data (in this talk), maybe very numerous
  - Reliable and Durable
  - Can have multiple components
    - Data servers, routers/proxies, Orchestration, etc
Small systems

- Everything is fine
- New products don’t have performance and scalability issues
- No need for compromises
- 70 years of previous computer industry development solves the problems for you
Large systems

- Nothing is fine
- Performance problems
- Scalability problems, always
- Technology-driven design compromises are necessary
- Established products are large systems and have issues
  - ... and users, and revenue impact
Relating to the real world (business)

- Product users come from the real world
- Ask the developers, product managers, scientists…
- Ask the accountants
  - “Financial Planning”, “Management Accounting”
- Do your own business / industry research
  - Yes, it’s a layering violation, but it will save you
- Keep up your contacts – don’t be isolated
- Look for fundamental controlling parameters
Relating to the real world (tech)

- Technology also comes from the real world too … outside your control.
- Technology capabilities:
  - Hardware metrics and performance
    - … many other good talks at this conference
  - Benchmarks, testing, learning from others, use knowledge and experience
- Look for fundamental controlling parameters
Capacity and load

![Graph showing capacity and load comparison between Server CPU power and App CPU demand.](#)
Capacity and load trouble

Uh-oh!
CPU? But what about…

- **Network**
  - 10Gbit and more, limited by CPU (and Latency)

- **Storage**
  - SSD and NAS, limited by CPU (and Latency)
Fundamentals

- Fundamental controlling parameters
  - Don’t change often
  - Still have to be checked
  - Can be elusive
    - Don’t be misled by second-order consequences

- Find the Constraints on scalability
All of us in this together

- Datastore and application scalability problems are connected
- Core Infrastructure teams have more experience
- Sometimes also more knowledge
- Yet we must educate developers
- … and delegate to them
- In the end, we all get paid from the same enterprise revenues!
Axes of Datastore scalability
Data size

More…

- Rows
- Tables
  - Sharding
  - Data partitioning
Data size

More…
- Rows
- Tables
  - Sharding
  - Data partitioning

Caused by
- Business volume growth
- Analytics
- Logging
- Data retention
Data size

More…
- Rows
- Tables
  - Sharding
  - Data partitioning

Effects
- Non-indexed queries are impossible – less ad-hoc reporting
- Split tables for partitioning
  - Harder to query
  - Client-side joins
- Less CPU per unit data
Schema growth
Schema growth

More of

- Columns
- Tables
- Relations & references
- Data model complexity
- Query complexity
  - Joins
  - Foreign Key constraints
Schema growth

More of

- Columns
- Tables
- Relations & references
- Data model complexity
- Query complexity
  - Joins
  - Foreign Key constraints
Schema growth

More of

- Columns
- Tables
- Relations & references
- Data model complexity
- Query complexity
  - Joins
  - Foreign Key constraints

Caused by

- More customers, products
- Product complexity
- Analytics
- Developers
  - …Frameworks, ORMs
  - …Abstraction layers
Schema growth

More of
- Columns
- Tables
- Relations & references
- Data model complexity
- Query complexity
  - Joins
  - Foreign Key constraints

Effects
- Query optimiser stress
  - Queries go bad, need tuning
- Indexing overhead
  - Time and space
- FKs slow inserts
- More queries per end-user action
Read query rate

More…

- Queries
  …on more rows
- More rows retrieved
  … from disc (or SSD)
- More data to sort and send
Read query rate

More...

- Queries
  ...on more rows
- More rows retrieved
  ... from disc (or SSD)
- More data to sort and send

Caused by

- Business growth
- Customer behaviour changes
- New features, interactivity, richer website
- Read growth disconnected from revenue
Read query rate

More…
- Queries
  …on more rows
- More rows retrieved
  … from disc (or SSD)
- More data to sort and send

Effects
- Server CPU, IOPS increase
- Memory cache strain
- Network traffic increase
- Need more read scale-out
  - Replicas
  - Copy number
Write query rate

More...

- Transactions
- Logging, audit
- Analytics
- ETL & Data Pumps
Write query rate

More…
- Transactions
- Logging, audit
- Analytics
- ETL & Data Pumps

Caused by
- Business growth
- Curiosity, Security, Regulation
- Richer customer experience
  - Saved preferences
  - Breadcrumbs
Write query rate

More…
- Transactions
- Logging, audit
- Analytics
- ETL & Data Pumps

Effects
- Server IOPS, CPU
- Latency increase
- Contention, locking
- Replication stress
Notation
Application coding

- Client-side join and filter, divide effort client- and server-side
  - Use an efficient data model
- Vectorise queries, do not iterate on Database
- Multithreaded, Asynchronous
- Parallelise (if you have to)
  - Map / reduce in your app
- Fast client code
Simplified query support

- Don’t support complex joins
- Don’t support use of known datastore weaknesses
- No Foreign Key constraints
- Enforce good indexing
  - Covering secondary indexes
- Discourage pointless server load
  - ORDER BY without LIMIT!
  - Intensive server side aggregate functions
  - Prefer client-side code over server-side code
- Compromise with developers
Caching

- Much faster data access...
  - Most of the time
  - On a good day
- Bad things hide in averages
  Maybe that’s OK for you
90% Cache Hit Rate

Request time

Average
Compression

- **Storage**
  - Application (JSON, text, blobs; Sereal)
  - Database (InnoDB Page)
  - Disc array (storage controller compression)
- **Network** – MySQL protocol compression
  - Usually huge win, but network usually OK
- **Helps**: Data size, Read query (a bit)
- **Hinders**: Updates, CPU/Latency
Replication

- More copies of data
  - MySQL – many servers each with all data, read only
  - Clusters – more nodes in cluster
  - Increased copy number
- More effort replicating data
  - Writes: Neutral at best, bad at worst
  - Especially in clusters
- May need separate read and write queries
  - You should have that anyway
Cluster databases (Galera, Cassandra, MySQL Group Replication)
MySQL Cluster database
Cluster databases

- **MySQL Cluster**
  - Huge read & write rate
  - Very reliable
  - Restricted data size
  - No complex schema
  - No complex queries

- **Cassandra**
  - High read rate
  - Reliable
  - Huge data size
  - (almost) No schema
  - Very simple queries
Split schemas

Original schema:
- T1
- T2
- T3
- DB

Split into three schemas:
1. **Schema 1**
   - T1
   - DB
2. **Schema 2**
   - T2
   - DB
3. **Schema 3**
   - T3
   - DB
Split Schemas

● (SLO) Declare maximum schema size
● Move some tables out to a new schema
  ● Preserve locality of reference
● Much work for developers
Shard data
Shard data

- Multiple data servers with data distributed between them
- Application complexity == developer work
- Some queries much slower than others
- Auto-sharders – Vitess, Spider
  - Limited query subset
  - Much greater operations complexity
  - Easier on developers!
- Compromise with developers
Example: Core transaction data

● Requires
Example: Core transaction data

- Requires
- Solutions?
  - Split data
  - Auto-sharder (Vitess?)
  - Cluster (MySQL?)