Divide and Conquer Data

Advanced Methods for partitioning and sharding data - Latest developments

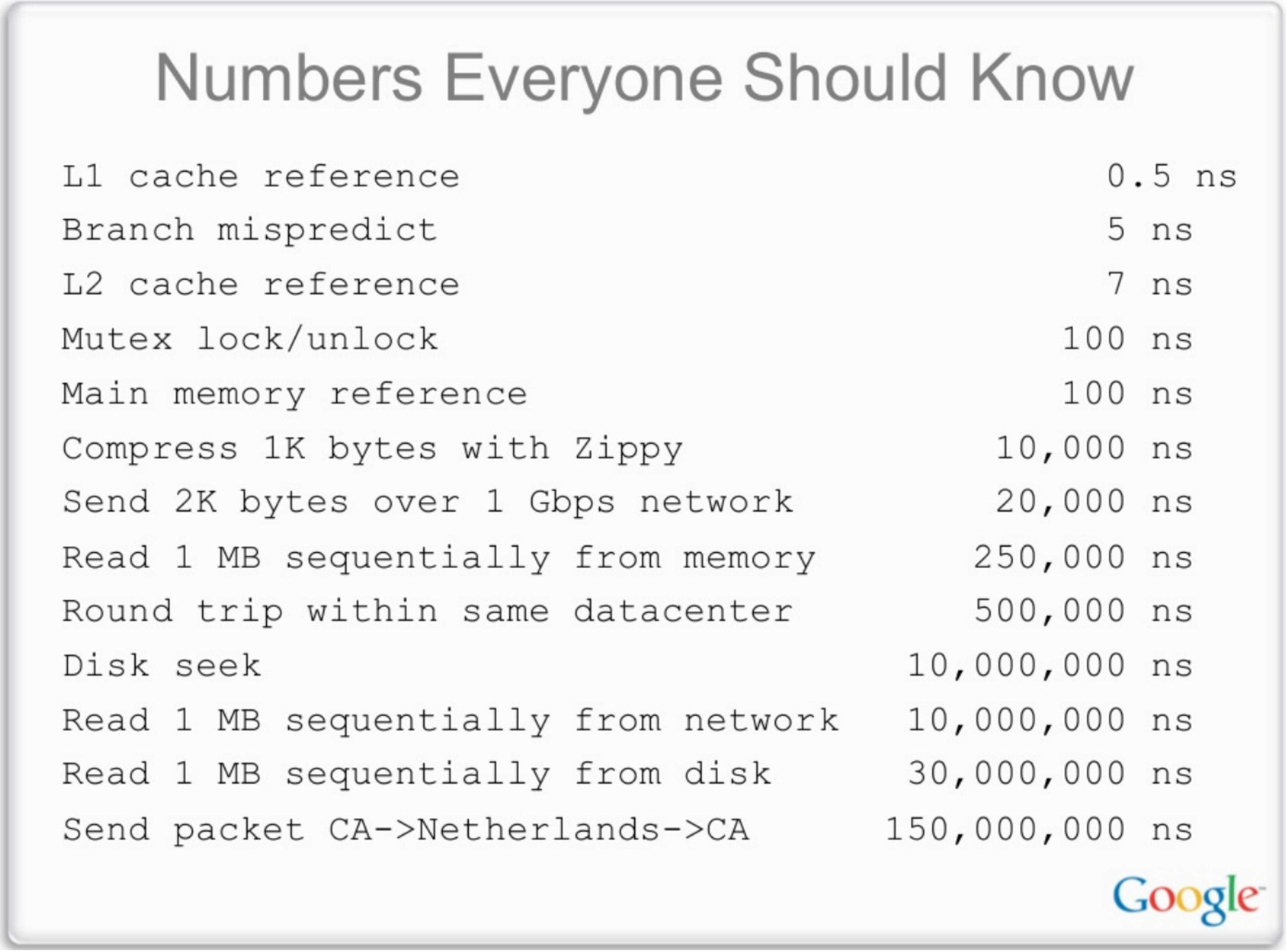
Jobin Augustine

PERCONA

Alternate Schools of thoughts

1. **Expensive Big monolithic systems** **capable of handling huge volumes of data**
2. **Multi master cluster**
   a. Shared disk clusters
   b. Mutual replication clusters

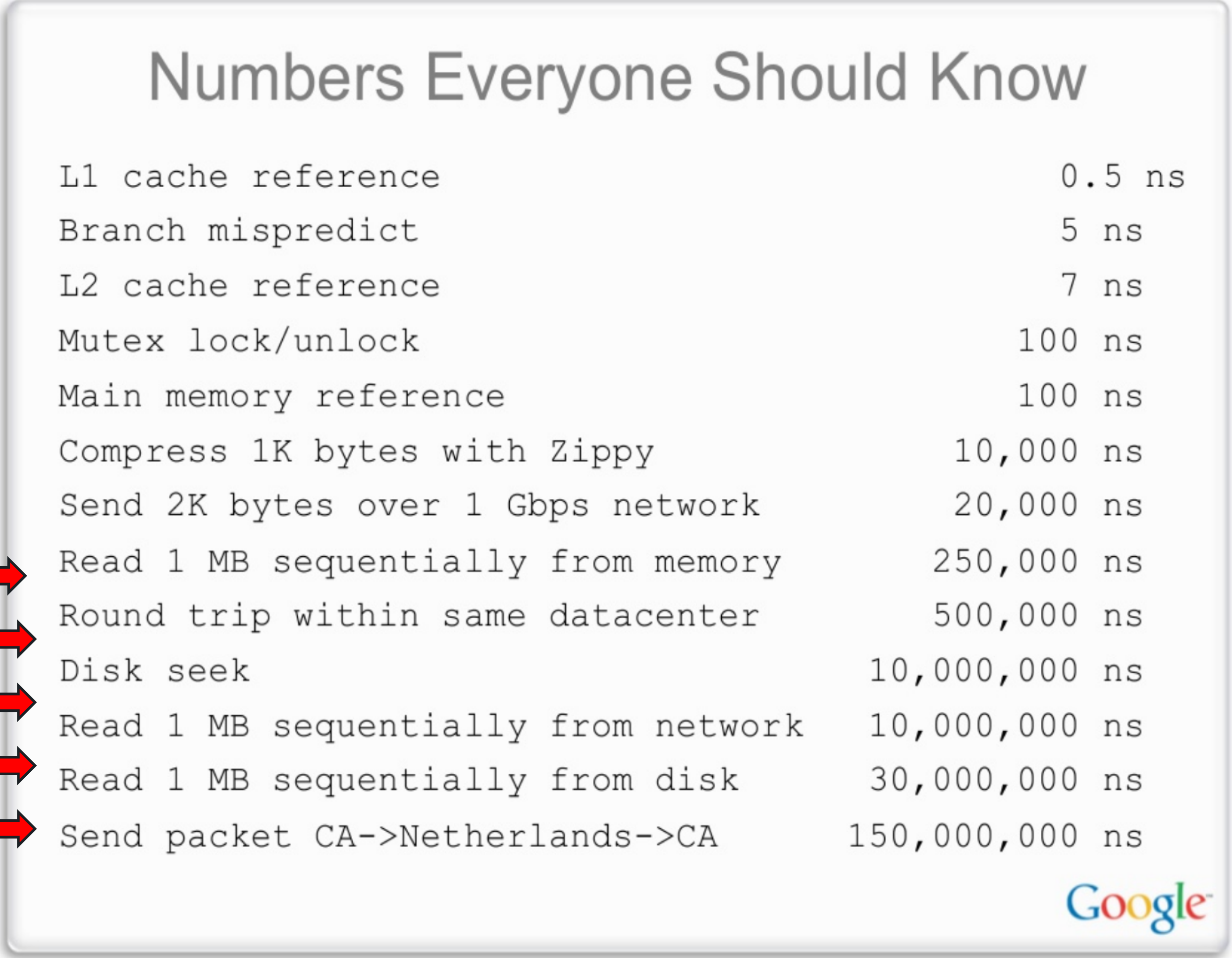
Ever evolving technology



Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Courtesy : Jeff Dean, Google

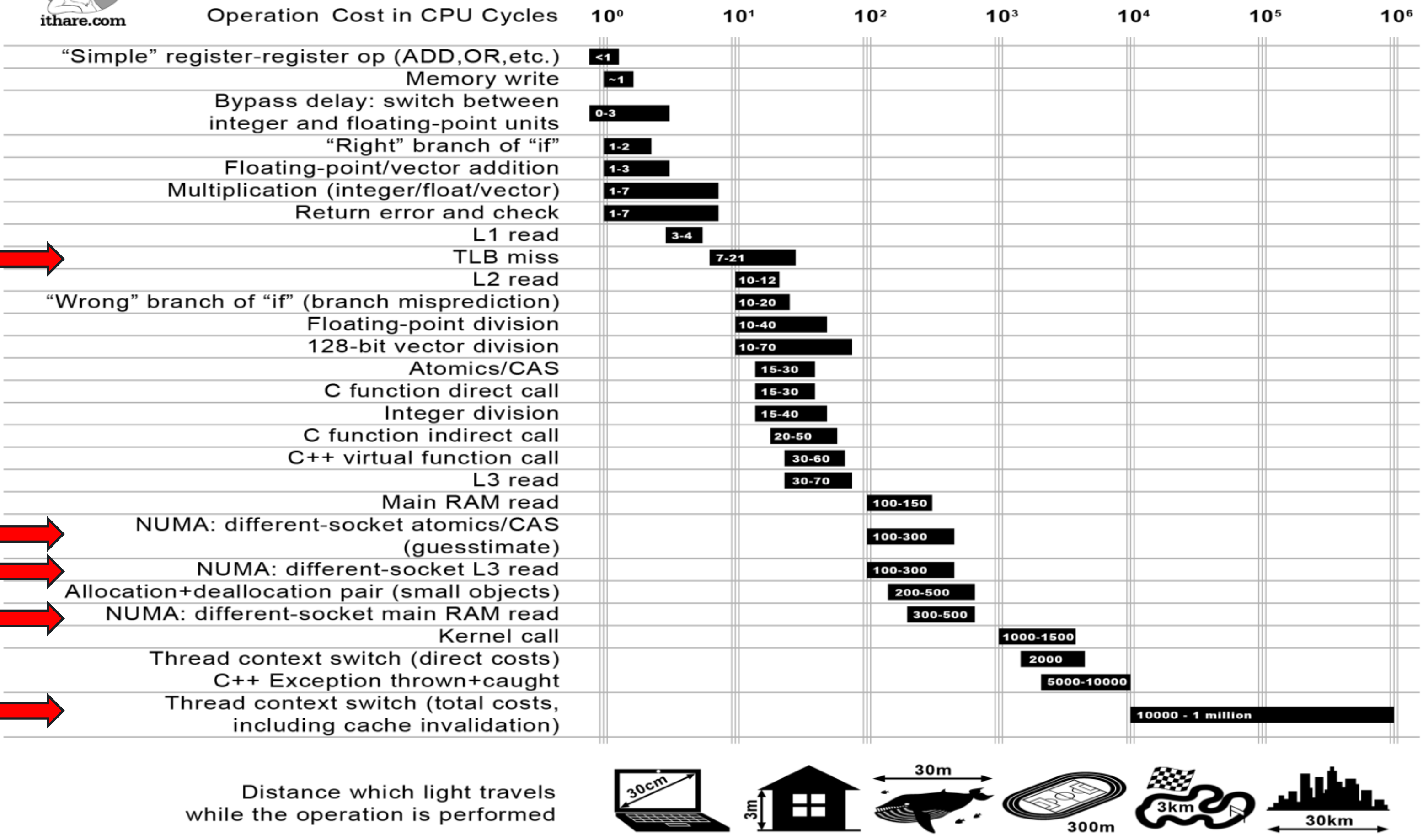


Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

ithare.com

Not all CPU operations are created equal

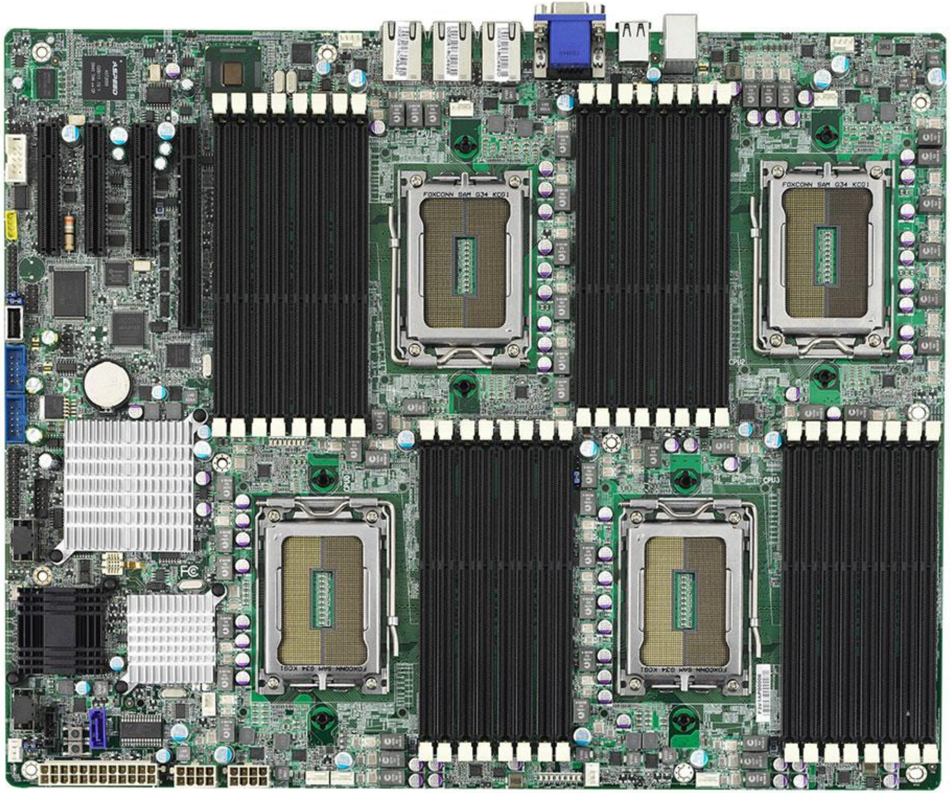


Operation	Cost in CPU Cycles
“Simple” register-register op (ADD,OR,etc.)	<1
Memory write	~1
Bypass delay: switch between integer and floating-point units	0-3
“Right” branch of “if”	1-2
Floating-point/vector addition	1-3
Multiplication (integer/float/vector)	1-7
Return error and check	1-7
L1 read	3-4
TLB miss	7-21
L2 read	10-12
“Wrong” branch of “if” (branch misprediction)	10-20
Floating-point division	10-40
128-bit vector division	10-70
Atomics/CAS	15-30
C function direct call	15-30
Integer division	15-40
C function indirect call	20-50
C++ virtual function call	30-60
L3 read	30-70
Main RAM read	100-150
NUMA: different-socket atomics/CAS (guesstimate)	100-300
NUMA: different-socket L3 read	100-300
Allocation+deallocation pair (small objects)	200-500
NUMA: different-socket main RAM read	300-500
Kernel call	1000-1500
Thread context switch (direct costs)	2000
C++ Exception thrown+caught	5000-10000
Thread context switch (total costs, including cache invalidation)	10000 - 1 million

Everything is evolving rapidly

- CPUs
- Memory
- Storage

NUMA



Storage connectivity

- IDE
- SATA

- HBA Cards

SCSI - *The SCSI standards define commands, protocols, electrical, optical and logical interfaces*

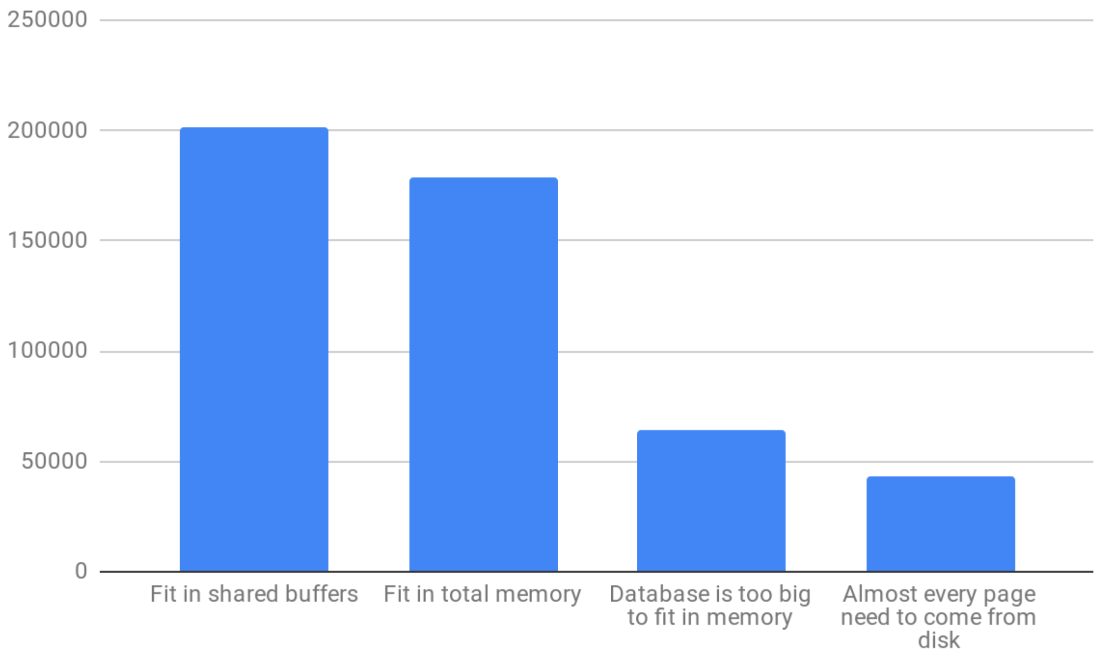
- Cables/Wires and their limitations of transporting data
- Laws of Physics and Noise

nvm EXPRESS

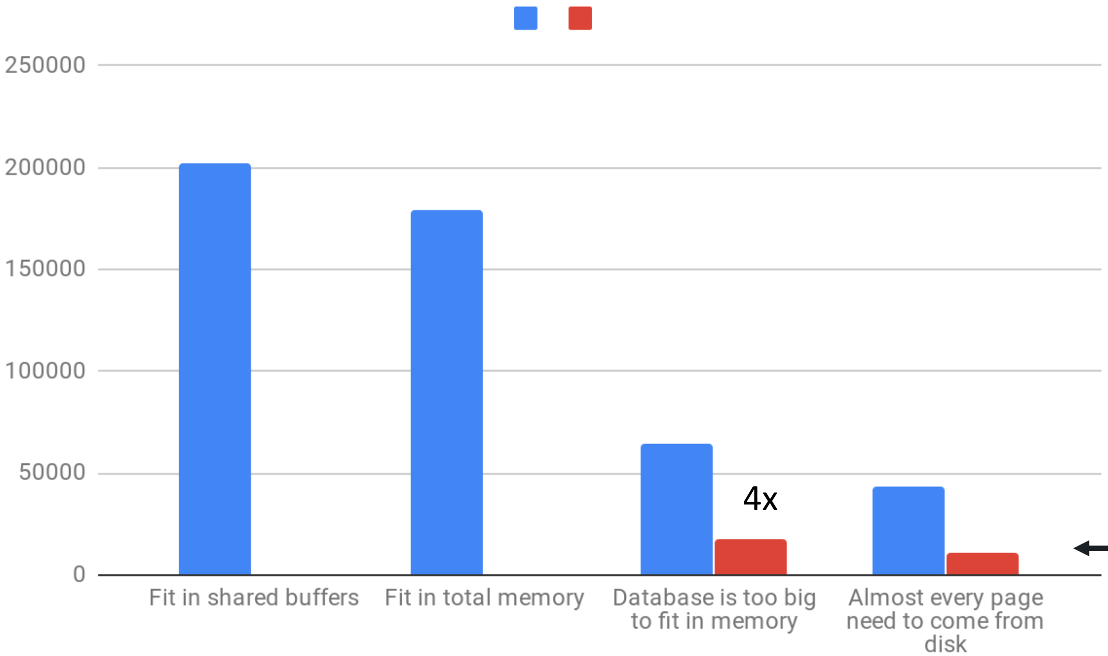
- 500k to I Million IOPs
- M.2 overcoming the Limitations of older interface

They can bring data closer to processing reducing latency

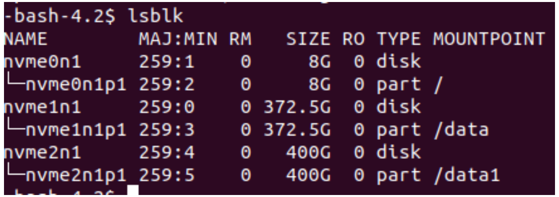
Single node NVMe



local vs remote storage



10K



```
-bash-4.2$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
nvme0n1     259:1    0     8G  0 disk
└─nvme0n1p1 259:2    0     8G  0 part /
nvme1n1     259:0    0 372.5G  0 disk
└─nvme1n1p1 259:3    0 372.5G  0 part /data
nvme2n1     259:4    0   400G  0 disk
└─nvme2n1p1 259:5    0   400G  0 part /data1
```

Small Nodes + Splitting

kubernetes

hadoop
HDFS

Key points

- Importance of Small computing units
- Bigger memory is not efficient
- **Storages are getting faster**
    - Remote storage is less attractive day by day.
    - **Local Storage is getting more and more attractive**

Partitioning

Getting maximum out of single node

Partitioning Advantages

Partition pruning

Added Advantages:

- Small Working-set of data
- Small indexes
- Vacuum benefits
- Retention policies
- Tablespaces and different disks

pg_partman

```
SELECT create_parent(table_name …)
```

- Partitioning for older versions of pg.
- Currently supports native partitioning
- Adds and deletes partitioning
- Background worker for partition maintenance

pg_partmaint - Super Simple partition maintenance for native partioning

Impact on Vacuum

- Typically vacuum kicks in when you have 20% dead tuples.
- Bigger maintenance_work_mem and lots of data it need to hold and process.
- Lots of dead tuples will be scanned but discarded for each query. Ex: 100 GB table can have 20GB dead tuples.

- Fix : DBAs increases the vacuum frequency.
- IO overhead of scanning the table and indexes more frequently

Impact on Memory

Handling bigger tables and associated bigger index requires more memory.

Undivided data = Bigger active data set.

Strategy of fitting active dataset into shared_buffers

Risk of falling from the cliff of bigger shared_buffers.

Simple Shards

Application level shards and postgres_fdw as a sharding solution

Application level shards

- Application awareness
- Avoid statement routing.
- Isolating unavailability.
- Application + DB scaling.

Built-in Sharding features

Advacements in :

Postgres_fdw + Partitioning + Parallelism

- Declarative table partitioning where individual partitions can be foreign tables
- Parallel execution
- Remote DMLs
- Intelligent planner
  - Predicate pushdown
  - Aggregate pushdown
  - Join pushdown

```
CREATE FOREIGN TABLE [ IF NOT
EXISTS ] table_name
  PARTITION OF parent_table [ (
  { column_name [ WITH OPTIONS ] [
column_constraint [ ... ] ]
    | table_constraint }
    [, ... ]
) ] partition_bound_spec
  SERVER server_name
[ OPTIONS ( option 'value' [, ... ] ) ]
```

Advanced Sharding

Extending PostgreSQL

Extensions for PostgreSQL

- Pg_shard and Citus data
- Timescale DB
- External databases and FDWs

pg_shard

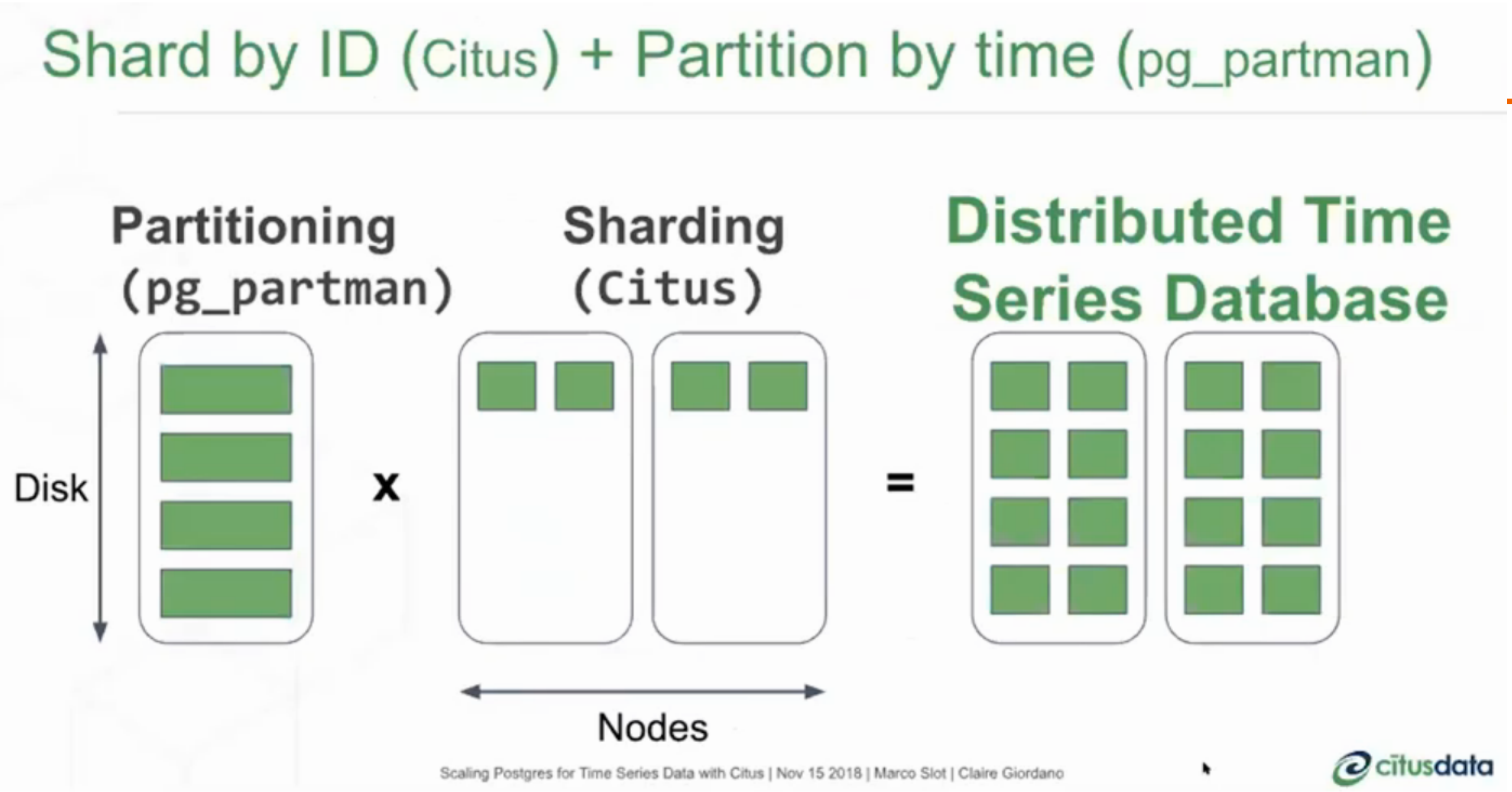
- **Data is cut into small chunks and distributed into worker nodes**
  - Each table is splitted into many shards.
- **Worker nodes stores data.**
  - One shard of a table is one table in the worker node.
  - Automatically shard tables are named
- **Metadata server - coordinator node**
  - Holds repository about shards (only few MBs)
  - where we create extension and shard table.
  - Place to send queries
  - Queries are analyzed to find out the right shard.

Citus Extension

Implemented as an Extension

- Go deep into PostgreSQL extension API to override query planner
- Query will be planned for shards.
- Data load will get faster to shared cluster (millions of TPS is easy) due to parallel load
- OLAP Load and Roll-up tables

```
SELECT create_distributed_table(table_name,colum_name
```



Shard by ID (Citus) + Partition by time (pg_partman)

Scaling Postgres for Time Series Data with Citus | Nov 15 2018 | Marco Slot | Claire Giordano

Time Series Data

Architecture	Applications	Implication

Architecture

- Past and Present
- Ledger

Applications

- Universally applicable
- IOT
- Monitoring
- Weather
- Satelite

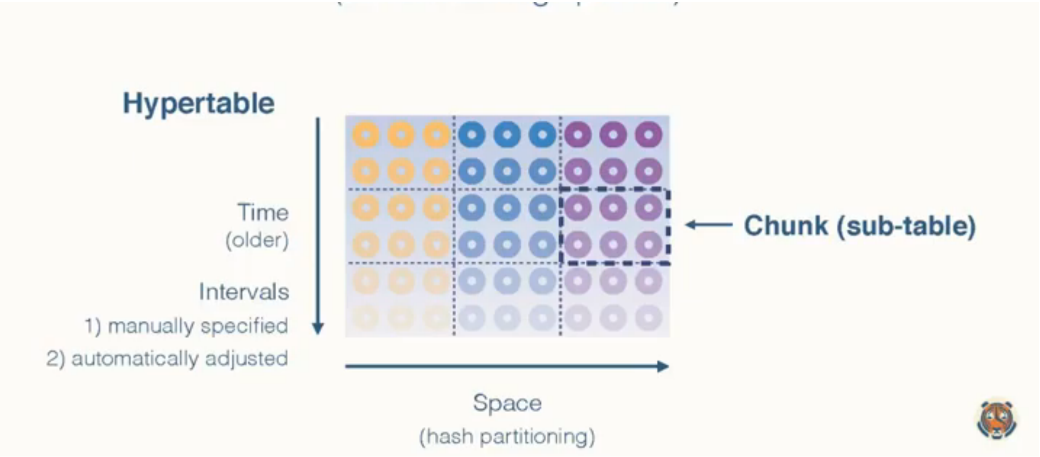
Implication

- Large Volume of data
- Primary key cannot be timestamp in general.
  - *Need an secondary index - B-tree*

When you update a data, you are losing old data

TimescaleDB

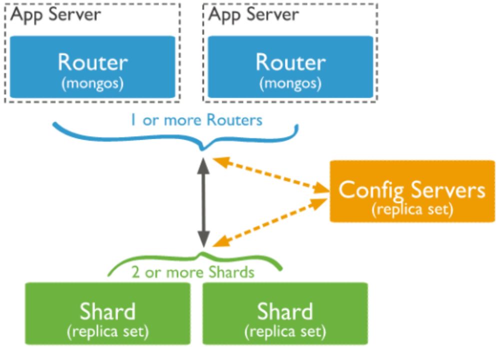
- Addresses many of the limitations of NoSQL databases.
- Full PostgreSQL and SQL features.
- Good Abstraction of underlying complexity and exposes table for application.
- High Insert performance
- Hypertable
- Right-size chunks
- Transparent disk addition
- Intelligent push down
- Custom UDFs



Externally Sharded data

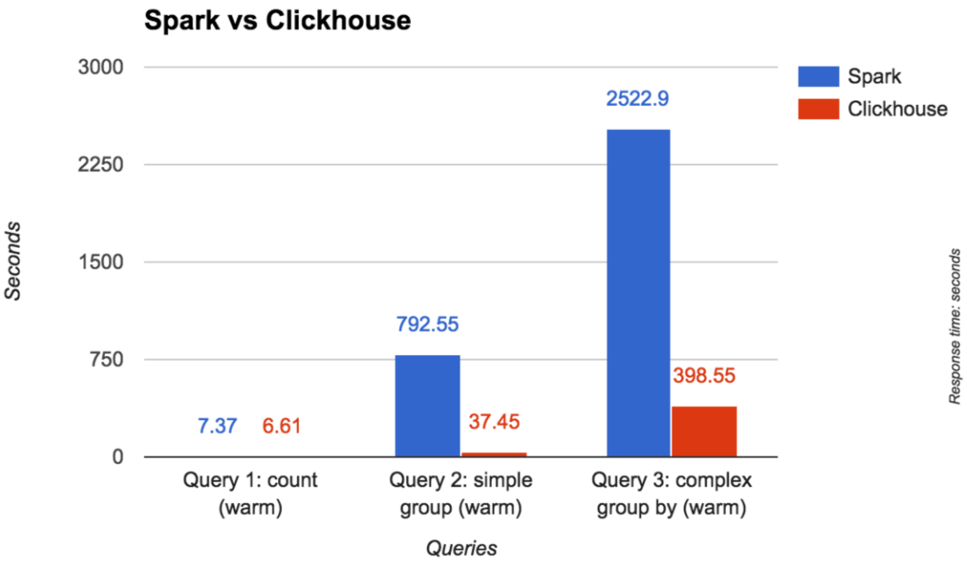
MongoDB and Mongo_fdw

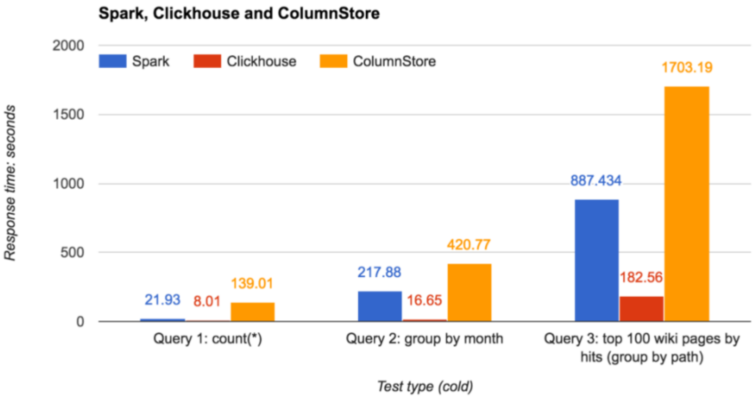
- New in design
- Growing
- Designed for sharding



- Collections as Tables
- Full Capable SQL
- MongoDB sharded cluster as distributed “Storage engine”

Clickhouse db





https://clickhouse.yandex/

Clickhouse db

- Column Store
- Linearly scalable
- High compression
- SIMD instruction

clickhousedb_fdw

References

NVMe Performance : https://www.youtube.com/watch?v=ada_JMsQ3Gk&feature=youtu.be
Table Inheritance : http://evol-monkey.blogspot.com/2018/03/implementing-distributed-reporting.html
Built in sharding : https://www.pgconf.asia/JA/2017/wp-content/uploads/sites/2/2017/12/D2-B1.pdf

Summary

- Dividing the data into small chunks through partitioning and sharding is the way to handle large volume of data.
- PostgreSQL as an ecosystem, offer large varieties of solutions.
- Developments in hardware especially storage, is pushing or small computation units associated storage it required.

Q&A