海量数据下分布式数据库设计实践

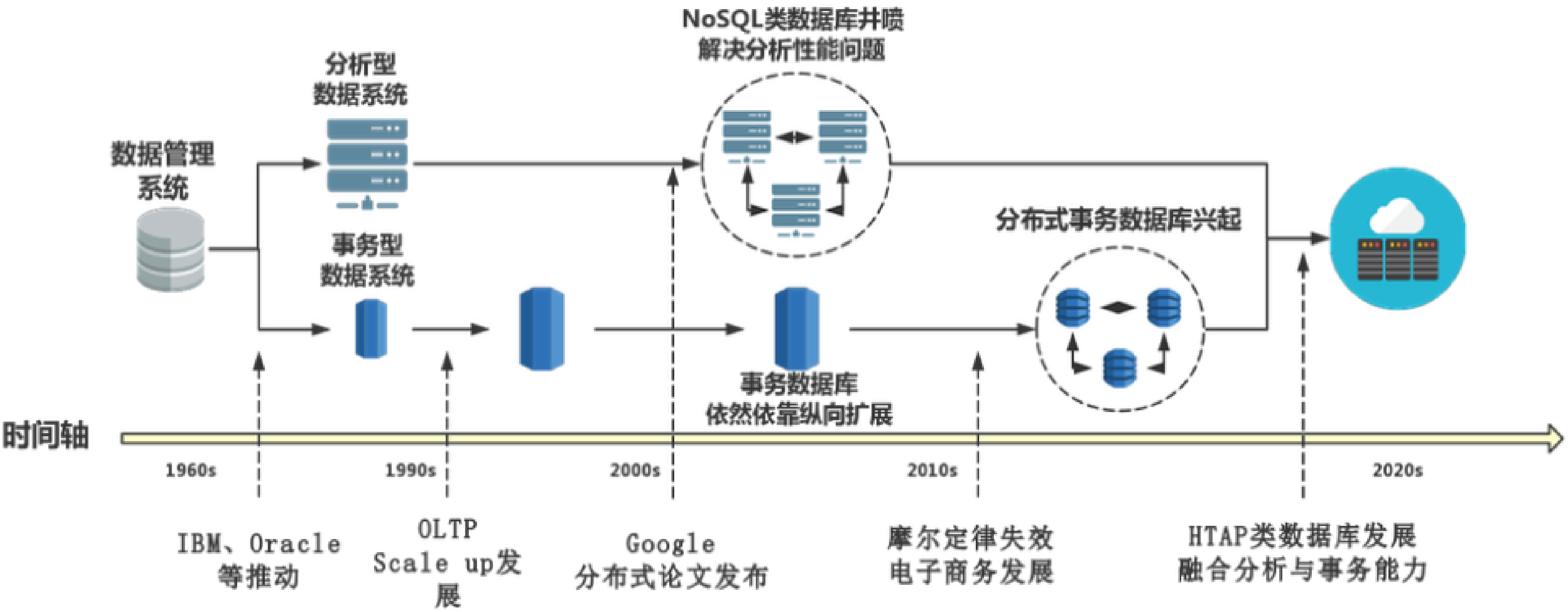
SequoiaDB North America Research Lab
巨杉数据库北美研发实验室

Danny Chen, Calvin Wong

Agenda

- History of distributed database
- Dive in distributed database technologies
- Comparison of different technologies
- Introduction to SequoiaDB

Evolution of distributed database



数据管理系统

分析型数据系统

事务型数据系统

NoSQL类数据库井喷
解决分析性能问题

事务数据库
依然依靠纵向扩展

分布式事务数据库兴起

时间轴

1960s

1990s

2000s

2010s

2020s

IBM、Oracle
等推动

OLTP
Scale up发
展

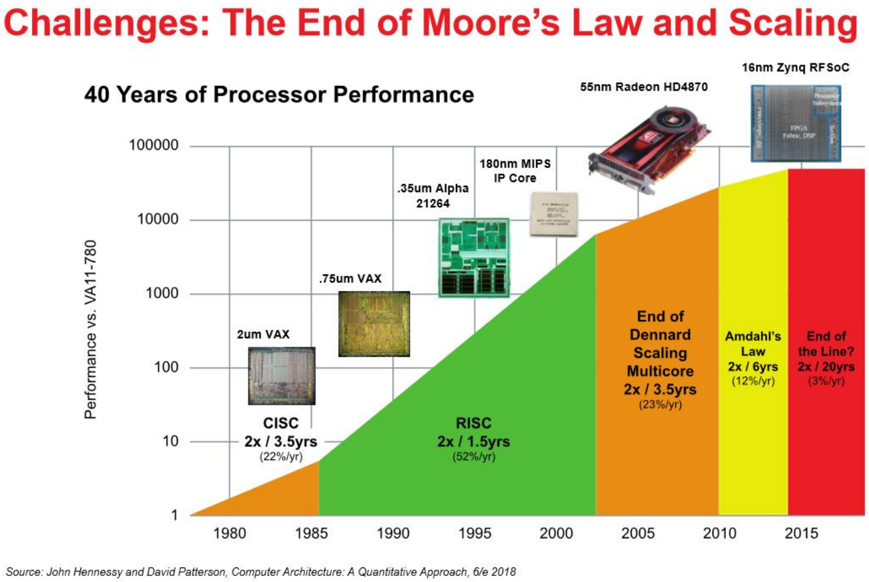
Google
分布式论文发布

摩尔定律失效
电子商务发展

HTAP类数据库发展
融合分析与事务能力

Evolution of distributed database

- Data
  - Storage capacity and throughput
  - Various data format support especially in era of internet
  - Different data source
- Processing
  - Ending of Moore's law
  - Power limitation



Challenges: The End of Moore's Law and Scaling

Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e 2018

Evolution of distributed database

Scale up v.s. scale out

- In common: Parallelism
- Degree of scale (Geno sequence 25m vs 10 m)
- Cost
- Vendor lock in
- Platform
- Eco-system

VS

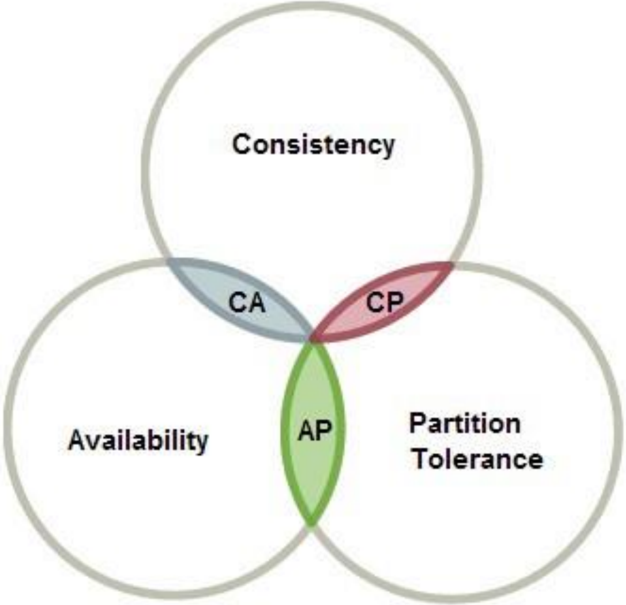
Agenda

- History of distributed database
- Dive in distributed database technologies
- Comparison of different technologies
- Introduction to SequoiaDB

Dive in technologies

CAP Theorem

- High Consistency
- High Availability
- Partition Tolerance
- Only satisfy 2 of 3



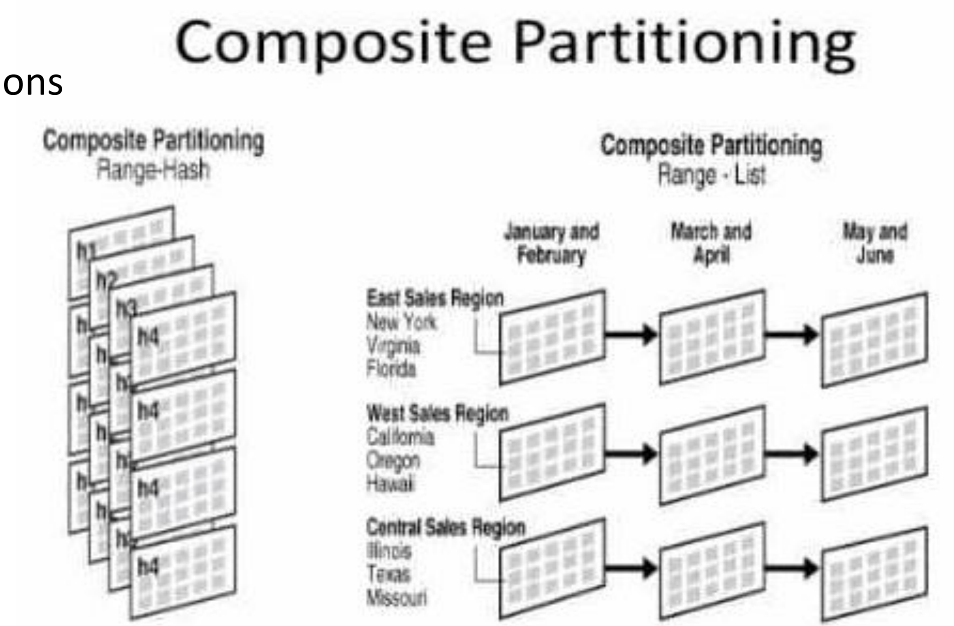
Dive in technologies

- Types of applications
  - OLTP
  - OLAP
  - HTAP
- Solution
  - Appropriate trade off
  - Capability to support

	OLTP	OLAP	HTAP
Data volume	low	high	high
Throughput	low	medium	high
Response speed	high	high	low
Concurrency	medium	high	low
TC	high	medium	low
consistency	high	medium	low
reliability	high	medium	low

Dive in technologies

- Distribution
  - Partitioning/Fragmentation/Sharding, horizontally assign each record to 1/n partitions
  - Vertically break down of the schema
  - Transparent on fragmentation, location, replication, local mapping, naming
  - Composite partition(Multi-dimension)
- Components
  - Query parsing, access plan creation and rule lookup
  - Rule based distribution, connection handler
  - Result aggregation
- Challenges: change

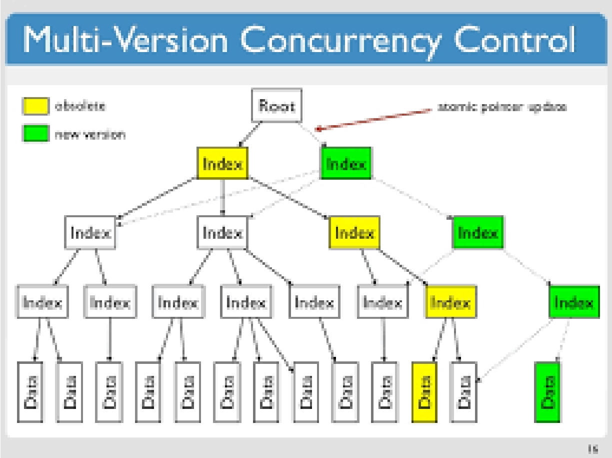


Dive in technologies (Transaction Isolation level)

- RC and RR are most commonly used
- Lock based
  - Simpler to implement by using different lock mode
  - Read/write can block each other
  - 2-PL and deadlock detection , more challenge in distributed environment

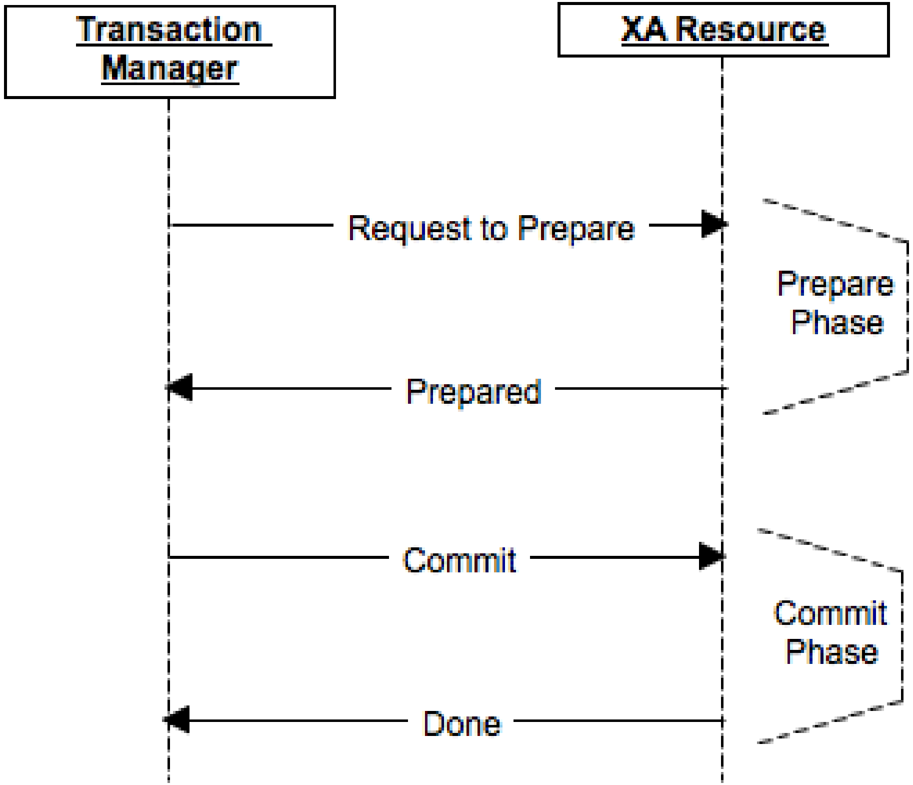
- MVCC based provides point-in-time consistent view
  - Read is never blocked
  - Snapshot isolation with vacum process
  - Latest data+undo log
  - Higher memory/storage footprint + CPU overhead

	State of Held Resource								
State Being Requested		None	IN	IS	NS	S	IX	SIX	U
	X	Z	NW						
None	yes	yes	yes	yes	yes	yes	yes	yes	yes
	yes	yes							
IN (Intent None)	yes	yes	yes	yes	yes	yes	yes	yes	yes
	no	yes							
IS (Intent Share)	yes	yes	yes	yes	yes	yes	yes	yes	no
	no	no							
NS (Scan Share)	yes	yes	yes	yes	yes	no	no	yes	no
	no	yes							
S (Share)	yes	yes	yes	yes	yes	no	no	yes	no
	no	no							
IX (Intent Exclusive)		yes	yes	yes	no	no	yes	no	no
	no	no	no						
SIX (Share with Intent Exclusive)		yes	yes	yes	no	no	no	no	no
	no	no	no						
U (Update)	yes	yes	yes	yes	yes	no	no	no	no
	no	no							
X (Exclusive)	yes	yes	no	no	no	no	no	no	no
	no	no							
Z (Super Exclusive)		yes	no	no	no	no	no	no	no
	no	no	no						
NW (Next Key Weak Exclusive)		yes	yes	no	yes	no	no	no	no
	no	no	no						

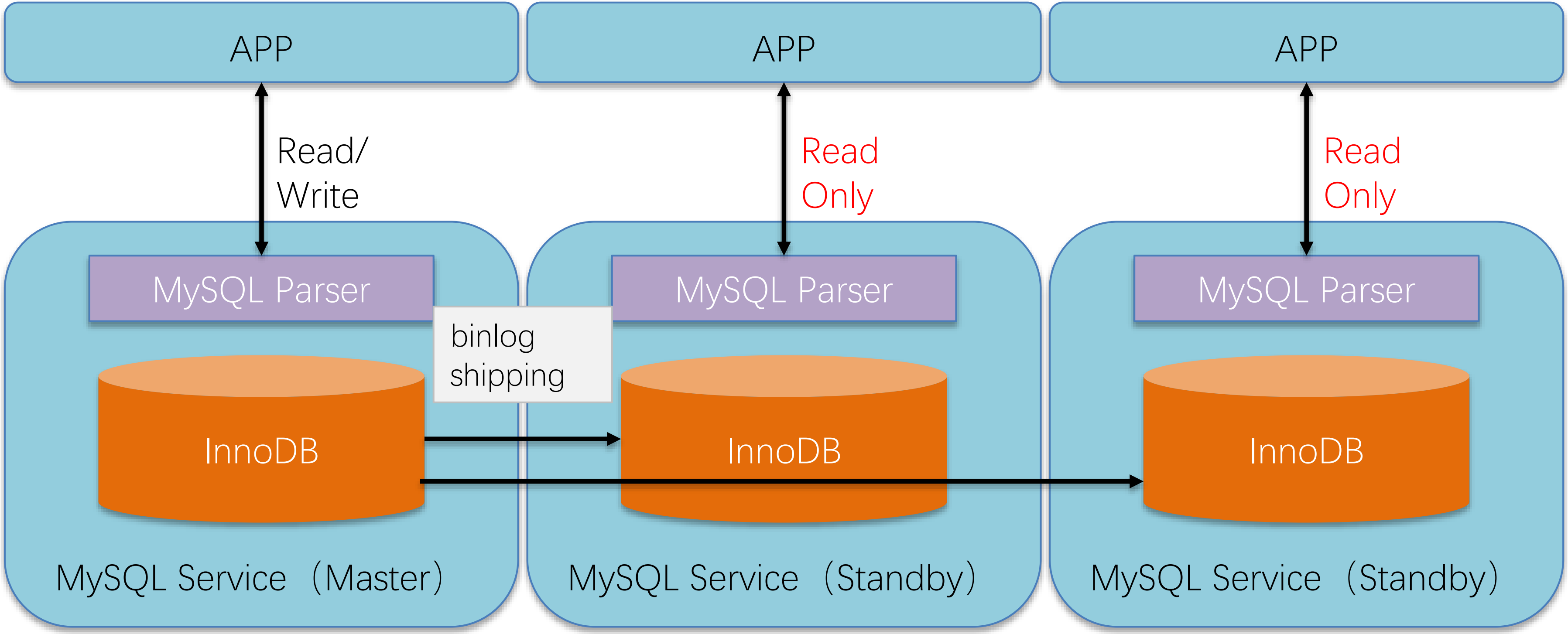


Dive in technologies (Distributed Transaction)

- Transaction
  - Global transaction management
  - Get all resource first, 2 phase commit
  - Transparent to application



Traditional MySQL replication strategy



Traditional MySQL replication strategy

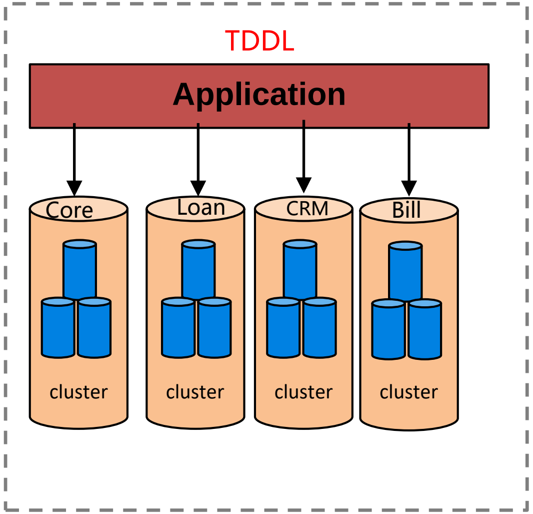
- Synchronous/Semi-Sync/Async Replication
  - Data duplication
  - sync/semi-sync slow
  - async fast but risk of data loss
- Failure detection and take over process

Agenda

- History of distributed database
- Dive in distributed database technologies
- Comparison of different technologies
- Introduction to SequoiaDB

Different distribution implementation

Application separation



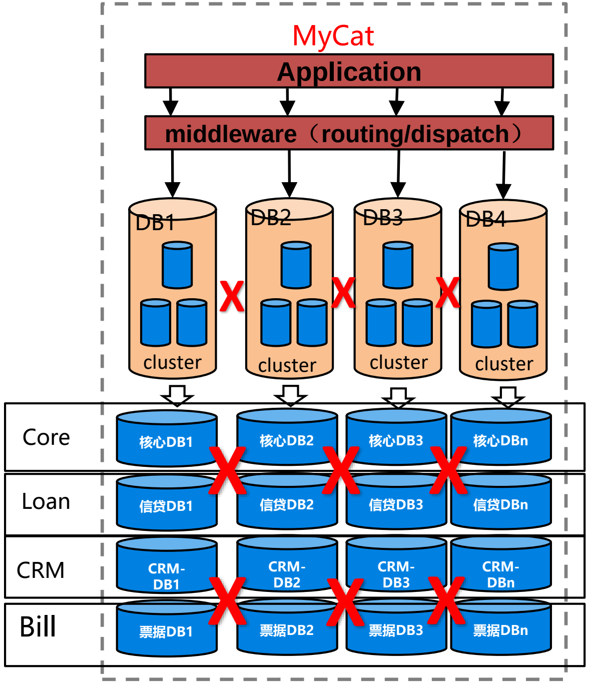
将不同模块的数据表分库存储，库间不相互关联查询，如果有，必须通过数据冗余或在应用层二次加工来解决，对应用程序侵入较大。

How to choose database with big table scenario.

	垂直分库
优势	- Early starting point, strong application control, deep customization - No special requirements for the underlying database, and the sharding is completely internal to the application
劣势	- Application logic is extremely intrusive, and applications need complex logic to make reasonable data distribution - It is very painful to adjust or expand the topology. It is almost impossible to complete online expansion. - It is difficult to support cross-database transactions

Different distribution implementation

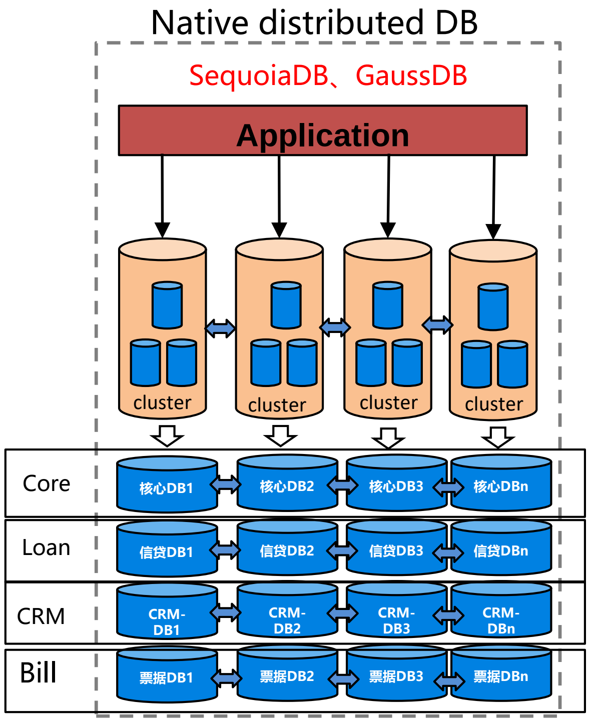
Use agent to handle different DB/Table



Written in JAVA, does support distribution, RW separation, support weak XA, fail over. However, single point failure,compute bottleneck, error-pron HA.

	分库分表
优势	• Build the intermediate SQL parsing layer, split the standard SQL into multiple subqueries and push it to the lower database, and assemble the results in the SQL layer • No special requirements for the underlying database, SQL segmentation in the middleware (support XA is fine) • Partially compatible with traditional SQL, application development is less difficult than vertical sharding
劣势	• Application logic is extremely intrusive, and applications need complex logic to make reasonable data distribution • It is very painful to adjust or expand the topology. It is almost impossible to complete online expansion. • It is difficult to support cross-database transactions

Different distribution implementation



将表分布到不同机器的库上，减轻数据库的压力
物理机的CPU、内存、网络IO负载分摊。支持分布式事务。

	原生分布式
优势	• The database internally handles distributed transaction and data segmentation logic, completely transparent to the application, without the need to perceive the underlying data distribution • The database natively supports distributed transactions, and the performance is much higher than sharding • High availability and data recovery are natively supported by the database kernel without additional assistance
劣势	• New technology, relatively few mature cases in industry • Relatively few auxiliary tools, the ecological environment needs to be improved

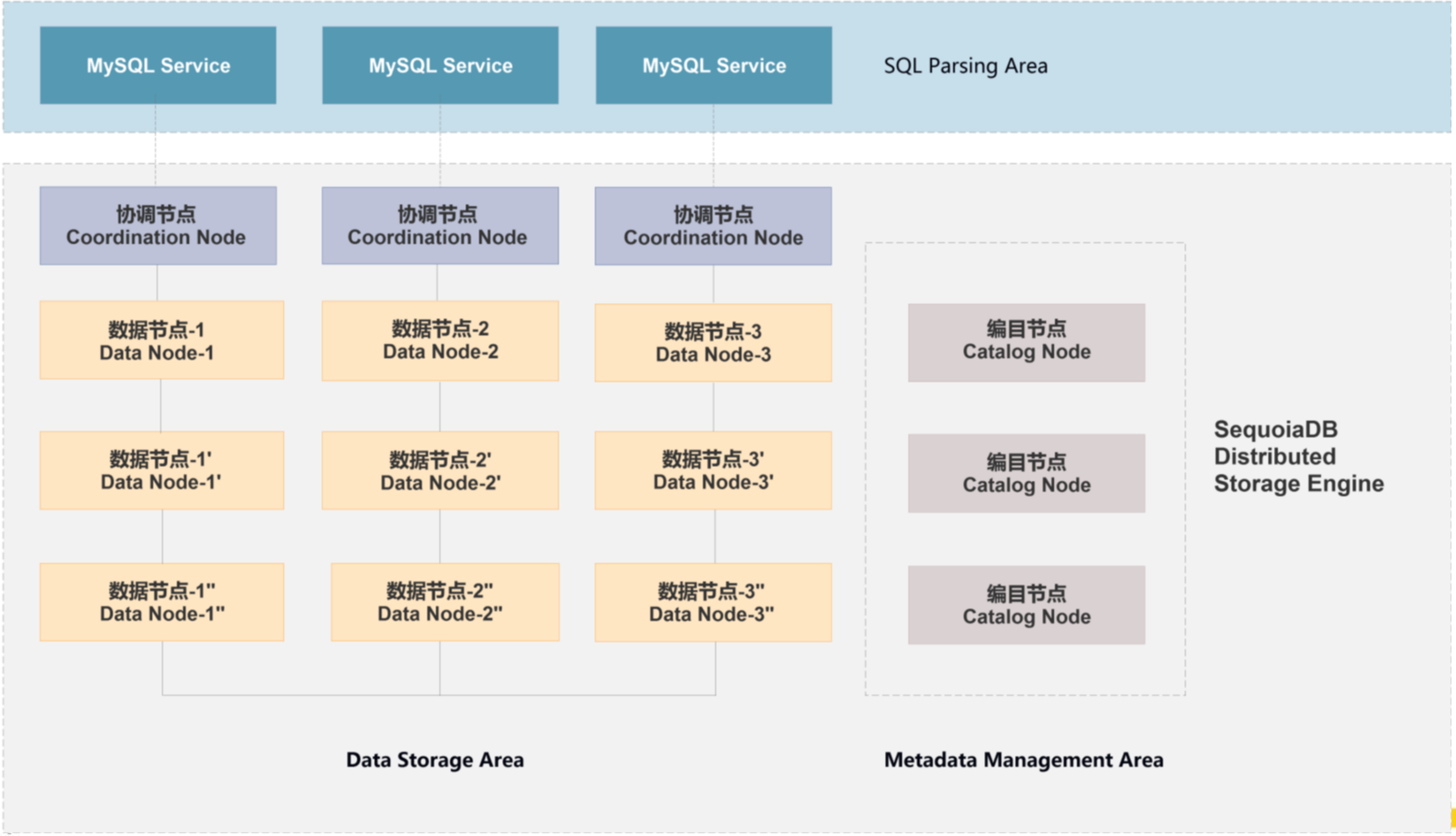
Bank distributed databases' requirement for CAP

在线交易（OLTP）	统计分析（OLAP）	联机服务（Operational）
• For high concurrency and low latency transaction business • Require CAP, where CP is fully satisfied, and A is infinitely close to 100% • Compatible with traditional SQL development models as much as possible, reducing application migration costs and reducing learning process • For new microservice architectures, multiple consistent hybrid support, multi-tenant and physical isolation capabilities	• For low concurrency and high latency back office business • No requirement for meeting CAP, data can be regenerated and imported • Maximize throughput, row and column hybrid storage mode • Reasonable introduction of big data technology, simultaneous application of structured and unstructured • MPP architecture	• For high concurrency and low latency operational business without transaction • Require AP, and data can be written in batches and can be re-imported • Minimize latency, and maximize concurrency • Hybrid use of structured and unstructured • Mainly for historical data, real-time read-only services, image management platforms, and etc

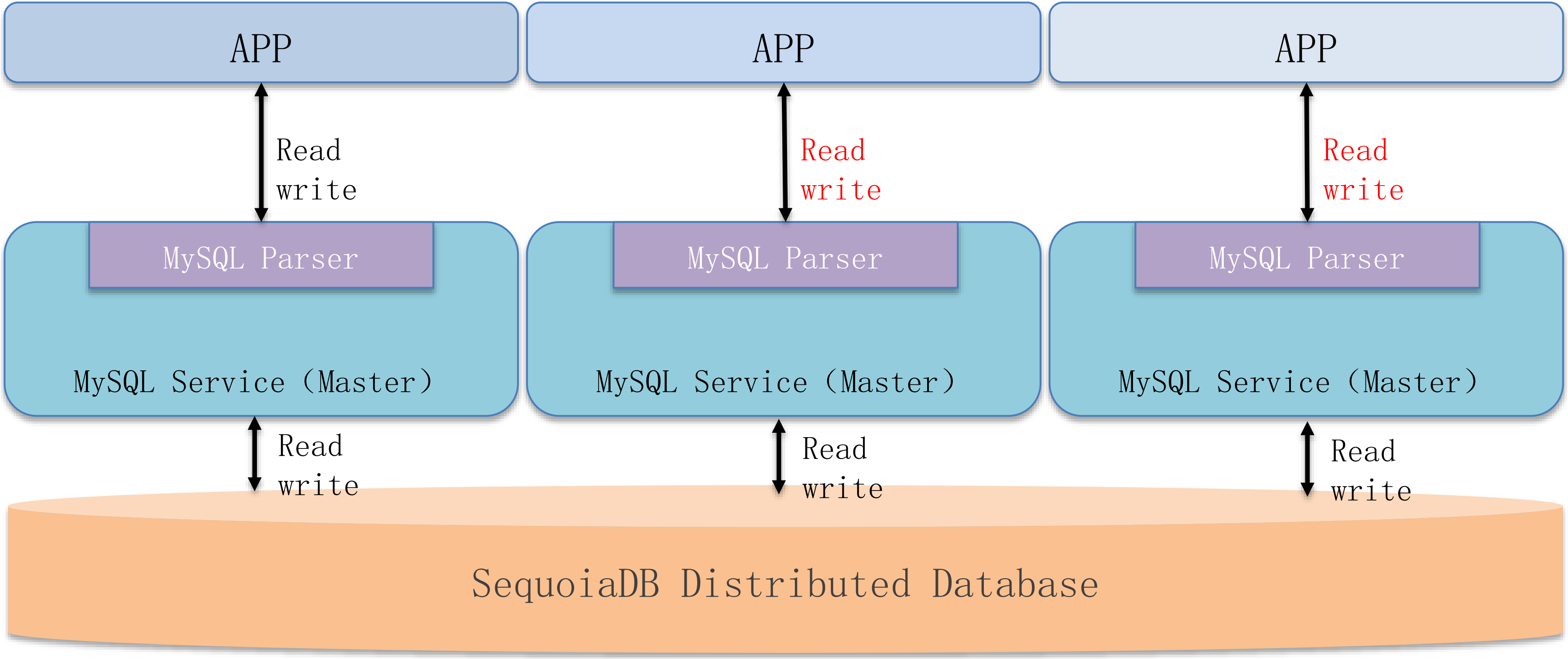
Agenda

- History of distributed database
- Dive in distributed database technologies
- Comparison of different technologies
- Introduction to SequoiaDB

SequoiaDB: Distributed Architecture

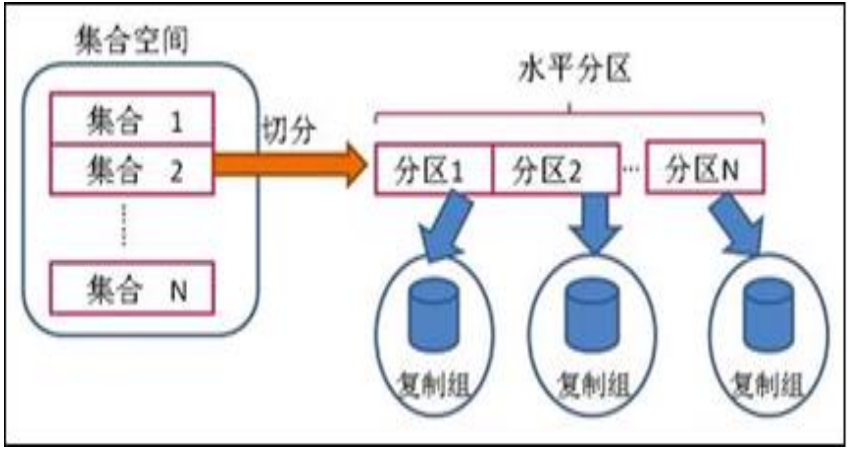


100% MySQL compatible and more

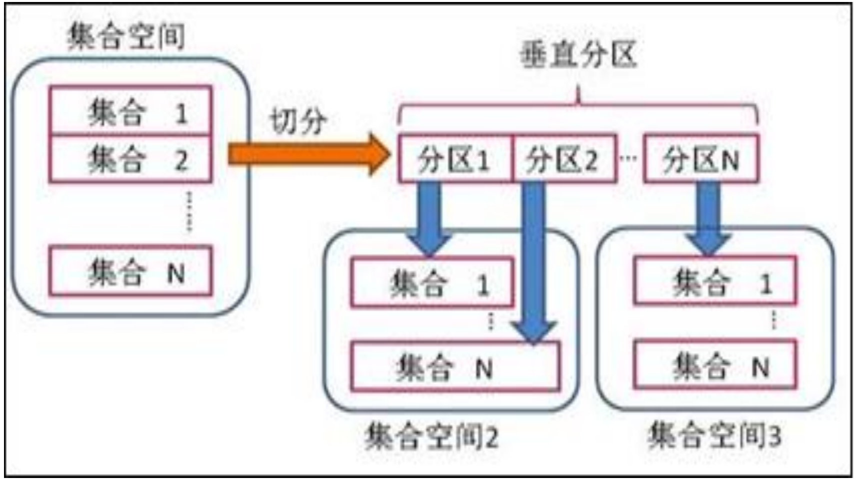


Support two dimensional partition

SequoiaDB support horizontal and vertical partition. Usually choose unique key for horizontal partition，use range cluster key like time-stamp for vertical partition.

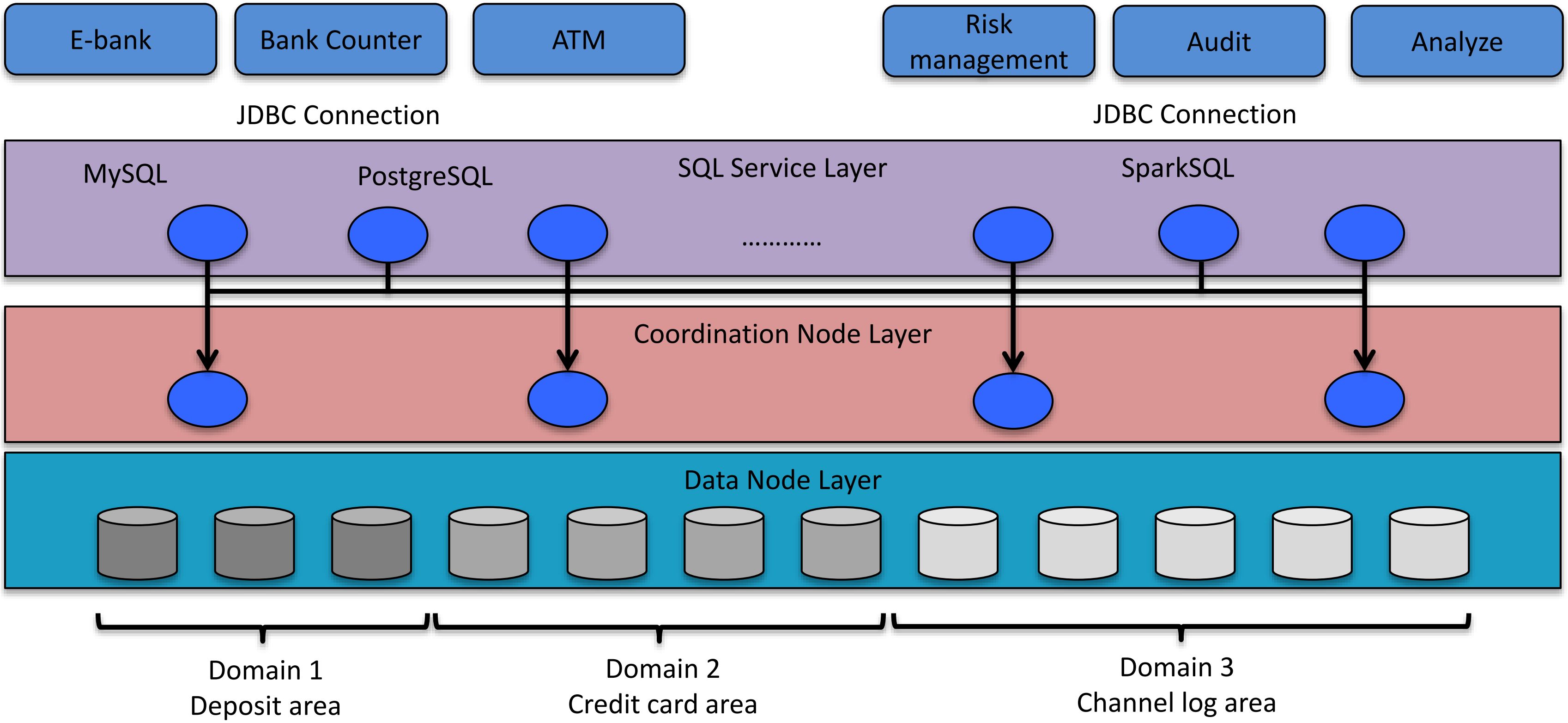


Suit for snapshot data and streaming data respectively



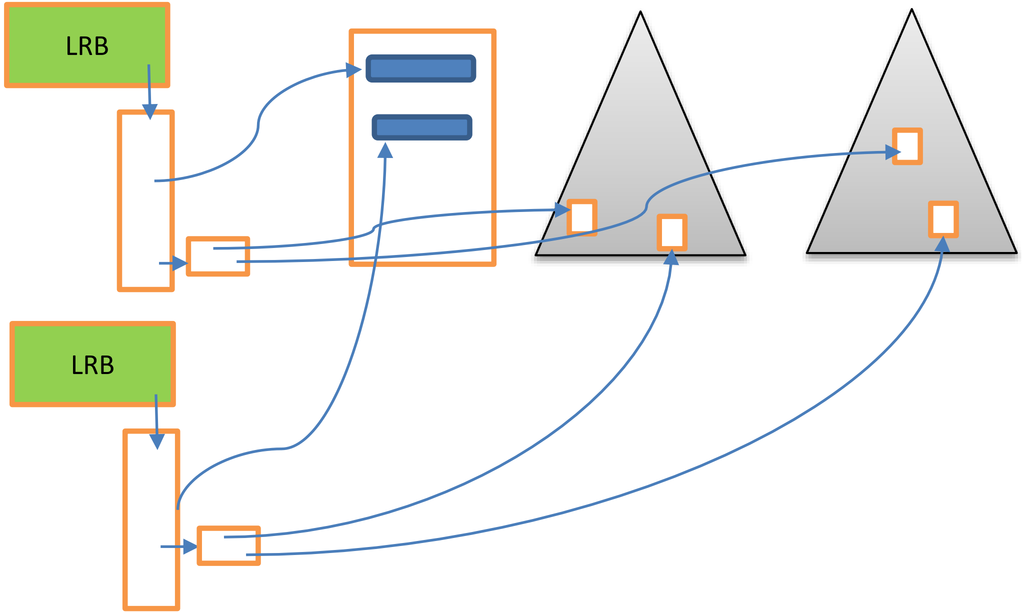
Advantage： linear scale for capacity and performance

Native SQL and HTAP support



Transaction Isolation in 3.2.0

- Support RU/RC/RR
- Combination of locking and versioning
- Read is not blocked
- More in next release



Our Team

- From IBM DB2 and Huawei（数据库老司机）！
- Based in Shenzhen, Beijing and North America Lab
- A leading distributed database team !

- The Coming Plan of SequoiaDB….

Summary

- Listen to the customers
- Use proper technologies for the right job
- Built from scratch and own our road map
- Source code level support

THANKS!
Q&A

SequoiaDB Website：
www.sequoiadb.com

Github：
SequoiaDB/SequoiaDB
SequoiaDB/sequoiasql-mysql

Join SequoiaDB Community
加入SequoiaDB社区！