Session:<13-D-1>

データベースの新潮流 -NewSQLとHTAP-

Speaker:ミック

自己紹介

- 名前: ミック
- 経歴: SIerに勤務するエンジニア（だった）。BI/DWHの開発からスタートし、パフォーマンス専門チームで長らく活動。2018年から米国シリコンバレーで技術調査やスタートアップ探索を実施。2025年現在、技術戦略のチームに所属。とりあえず新しい技術を探す何でも屋。
- SNS: X (@copinemickmack)

近刊の紹介

「データベースなんも分からん」人向けの最初の一冊。超入門書。(2024)

脱初級を目指すデータベース設計の改訂版。半年～1年くらいDB経験のある人向け(2024)

コミカルな対話形式で楽しく学ぶSQL中級入門。『達人本』より易しめ (2024)

New！

データベースやSQLの考え方や思想についてのエッセイ。寝転んでも読める軽い一冊。(2025)

読書ガイドマップ

データベースなんもわからん

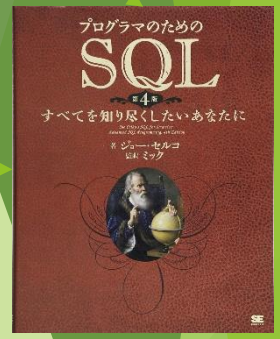
SQLなんもわからん

DB設計チョットわかる

SQLチョットわかる

パフォーマンス気になる

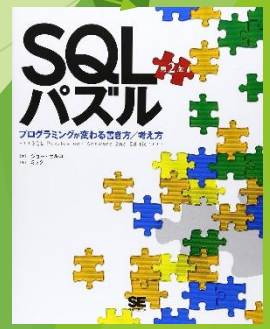
SQL自信あり



物好き

エッセイ・読み物

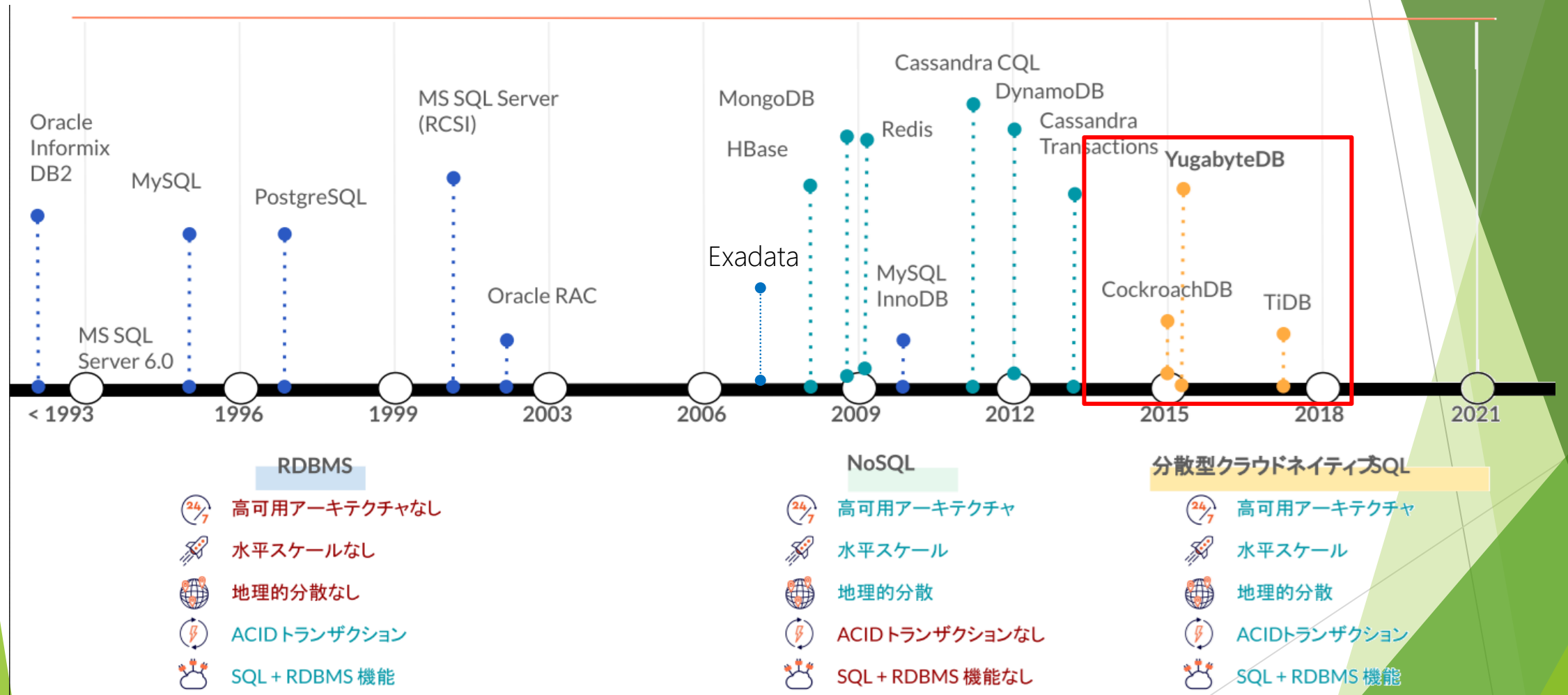
変態



練習問題

データベースの新潮流①
NewSQL ― 統合と分散の螺旋

データベースの変遷と現在地



Source: Yugabyte

NewSQLの何が”New”なのか？

特徴	RDB	NoSQL	NewSQL
リレーショナル	スキーマ定義（テーブル）	スキーマレス(※1)	**スキーマ定義（テーブル）**
ACID	ACID保証あり	ACID保証なし	**ACID保証あり**
SQL	サポート	なし(※2)	**サポート**
スケーラビリティ	限界あり	水平方向にスケーラブル	**水平方向にスケーラブル**
分散データベース	No	Yes	**Yes**

※1 近年はスキーマ定義を行えるものもある。
※2 近年は独自のクエリ言語を持つものもある。

名前に反して、NewSQLに”新しい機能”はない

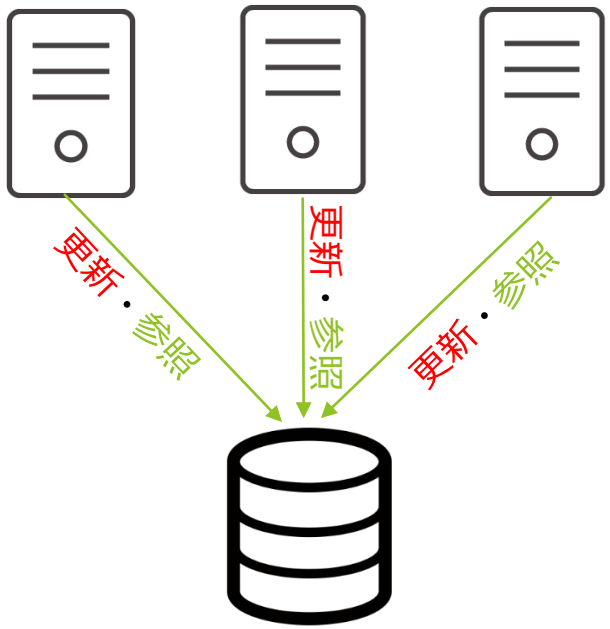
2017年の予言（？）

> NoSQLというのは、将来的にはRDBの機能の一つを指す言葉になることも、十分に考えられます。

Source: ミック, “RDBとNoSQLにみるDB近現代史 データベースに破壊的イノベーションは二度起きるか？”
https://en-ambi.com/itcontents/entry/2017/11/22/110000/

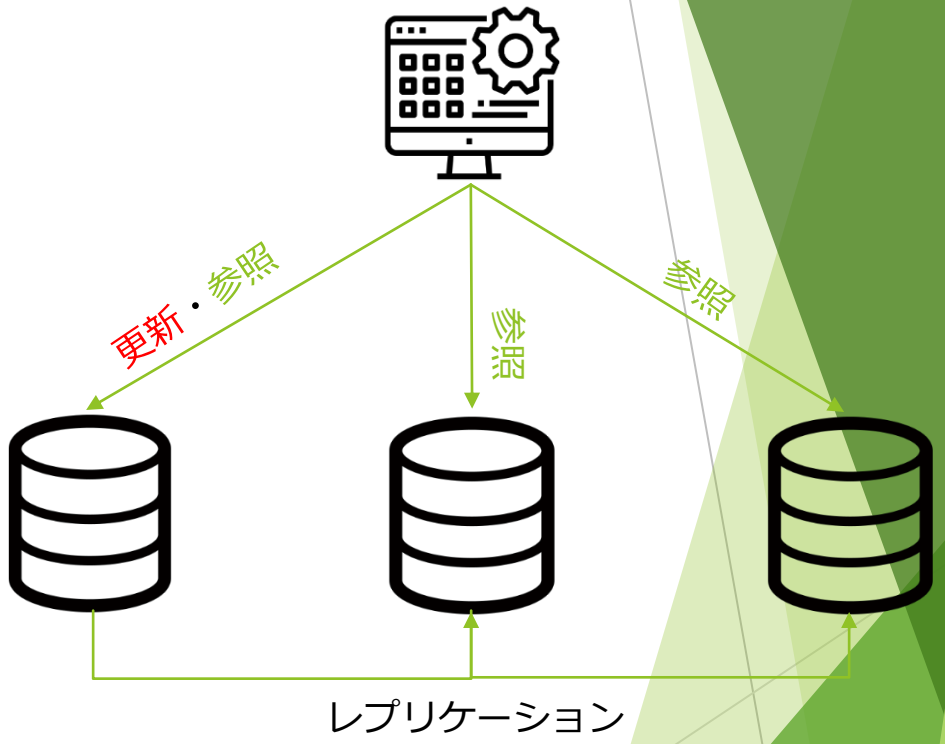
RDBの水平スケーラビリティの限界

シェアードエブリシング方式



Oracle RACで採用される方式。更新と参照をスケールさせられる優れた仕組みだが、共有リソースのストレージがボトルネックになる。

リードレプリカ方式



主にウェブサービスで利用される。参照はスケールするが更新はプライマリノードでしか受けられない。

分散合意アルゴリズムの革新

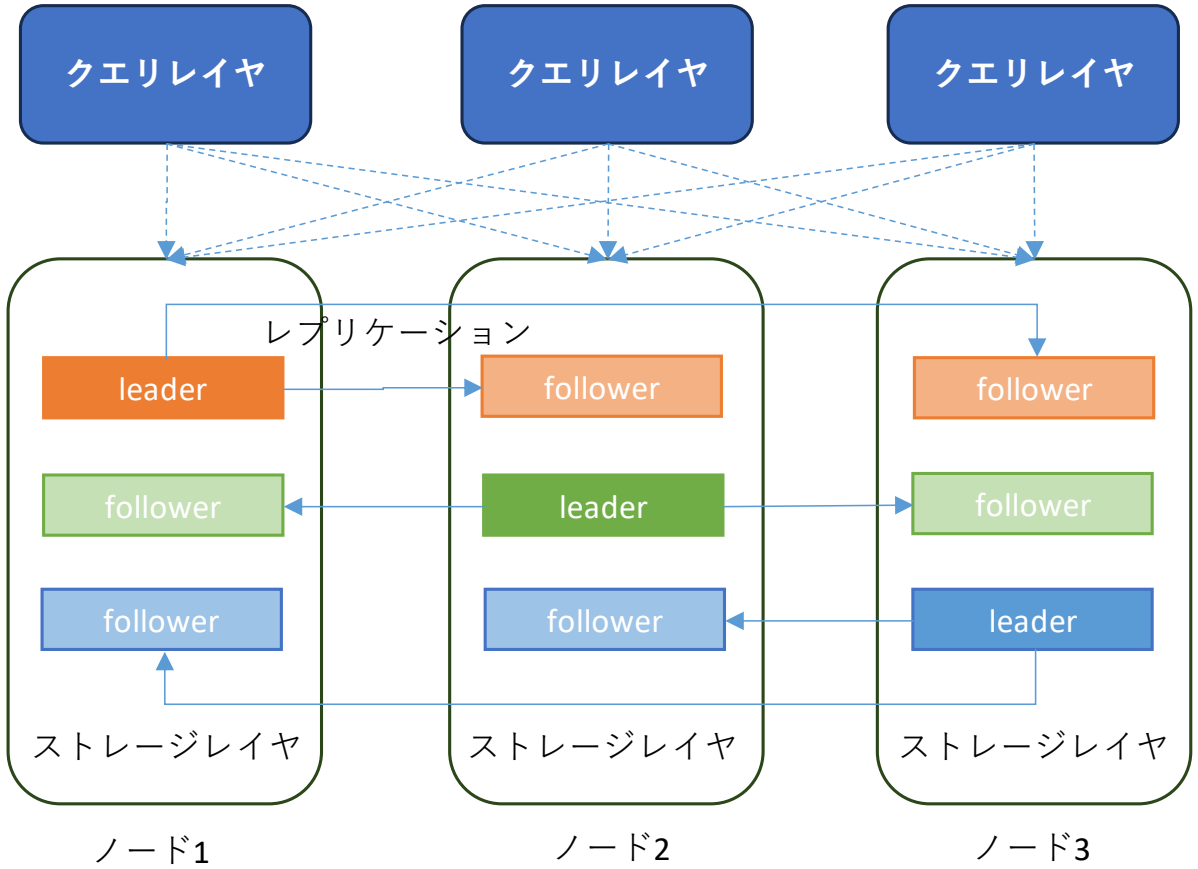
RDBの長所を活かしたままスループットを出すには分散構成を採る必要があることはだいぶ前から議論されていた。分散合意アルゴリズムとしては、従来**Paxos**が知られていたが（論文の登場は1998年）、複雑で実装が難しいためなかなかデータベースへの適用が進まなかった。そこでPaxos系アルゴリズムの代替として設計された分散合意アルゴリズムが**Raft**(2013年登場)。

- ログレプリケーション
- リーダー選挙

CockroachDB、TiDB、YugabyteDBなど多くのNewSQL製品で利用される。

Source: “In Search of an Understandable Consensus Algorithm” (2013)
https://raft.github.io/raft.pdf

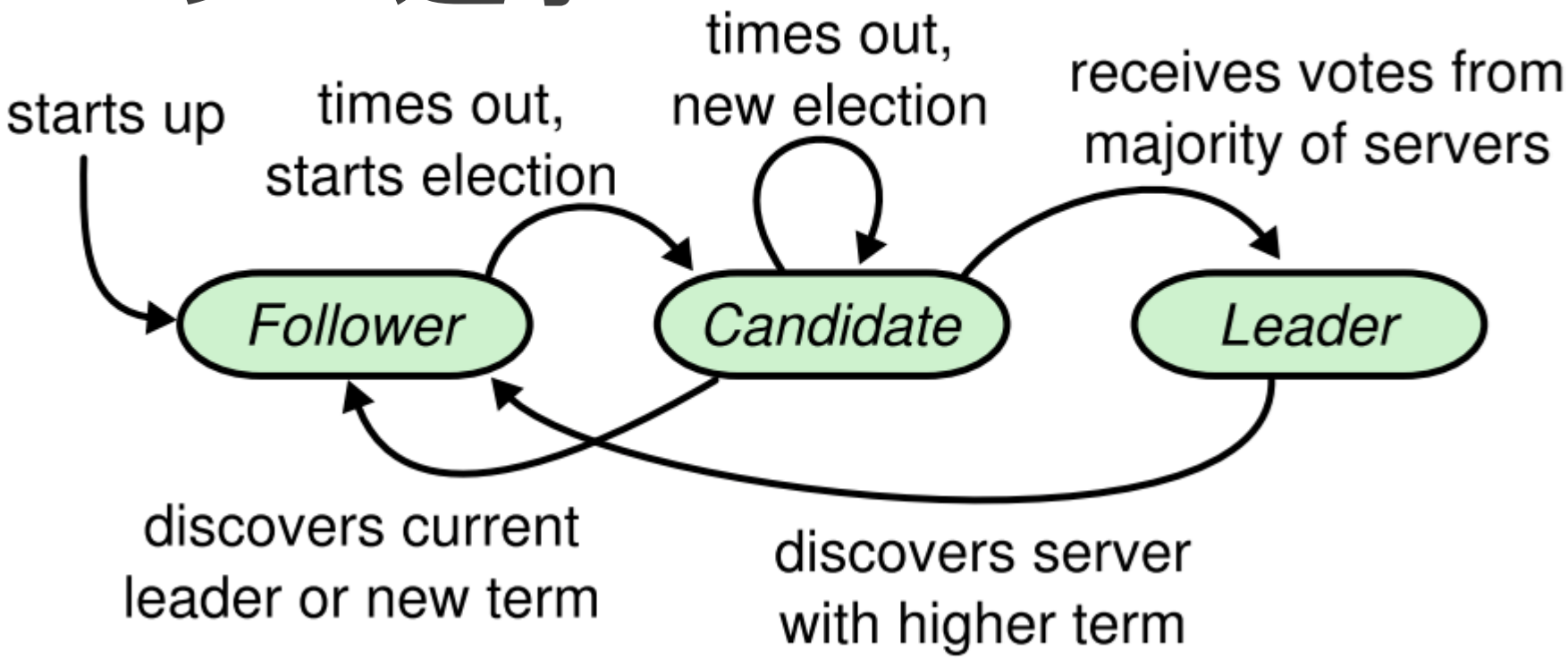
分散データベースの実装イメージ



コンピュートとストレージのノードが分離されており、個別にスケールアウトが可能。書き込み時にはリーダーがまず書き込みを受け付け、フォロワーノードにログレプリケーションでデータ同期を行う。**Raft**で合意された結果は覆えることはなく永続化される。

Source: NTTデータ, “「NewSQL」とは？改めて知りたいデータベースの最新動”
https://www.nttdata.com/jp/ja/trends/data-insight/2023/1219/

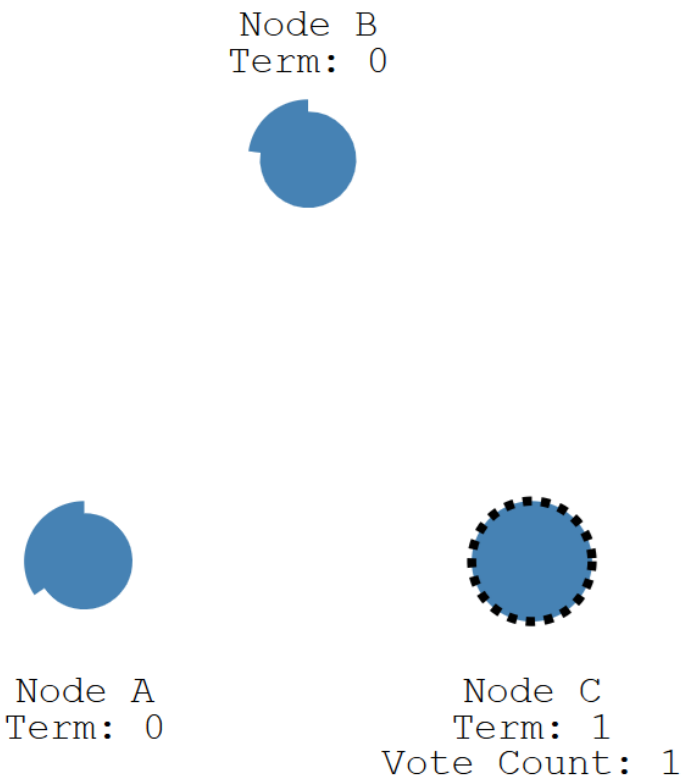
リーダー選挙



まずノードはFollowerから始まり、他のノードとの通信が一定時間ないと「**選挙**」が開始され、Candidateへ変化する。もし**過半数**のノードから投票を得たらLeaderに遷移する。

Source: Zixuan Zhang, “Raft Algorithm, Explained - Part 1 — Leader Election”
https://towardsdatascience.com/raft-algorithm-explained-a7c856529f40

Raftの動きを理解するアニメーション



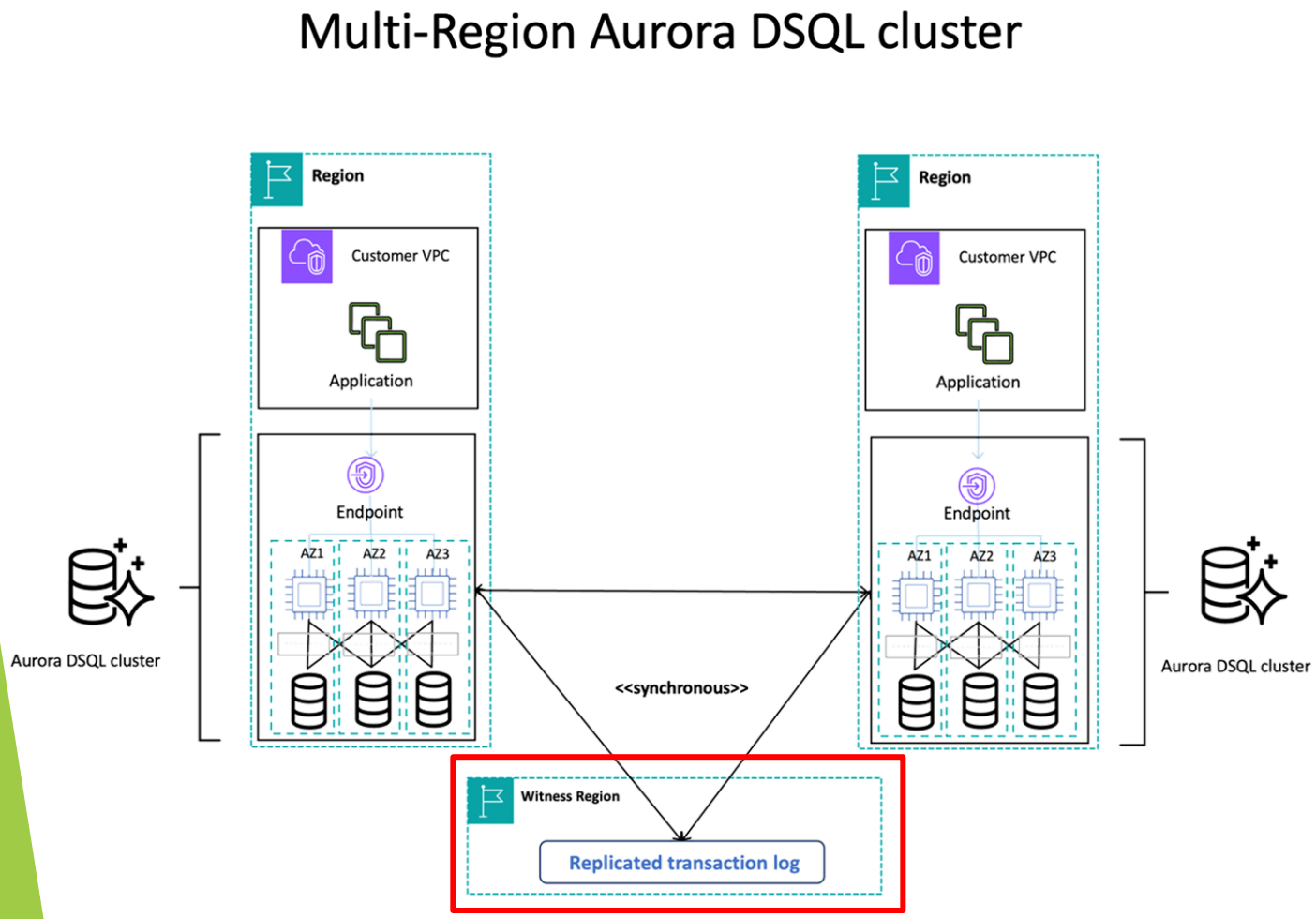
ログレプリケーションとリーダー選挙のアニメーションを見ることのできるウェブサイト。とりあえずRaftを理解するならばこのサイトを見るとイメージがつかみやすい。

Source : The Secret Lives of Data
https://thesecretlivesofdata.com/raft/

NewSQLの主要プレイヤー

	Cloud Spanner	yugabyteDB	TiDB	CockroachDB
互換性	プロプライエタリ（PostgreSQL）	PostgreSQL	**MySQL**	PostgreSQL
調達額	N/A	$291M (Series C)	$341.6M (Series D)	**$633.1M** (Series F)
開発元と日本進出	米国 日本法人あり	米国 日本法人あり	米国 日本法人あり	米国 **日本法人なし**
可用性	**99.999%** (マルチリージョン)	99.99%	99.99%	**99.999%** (マルチリージョン)

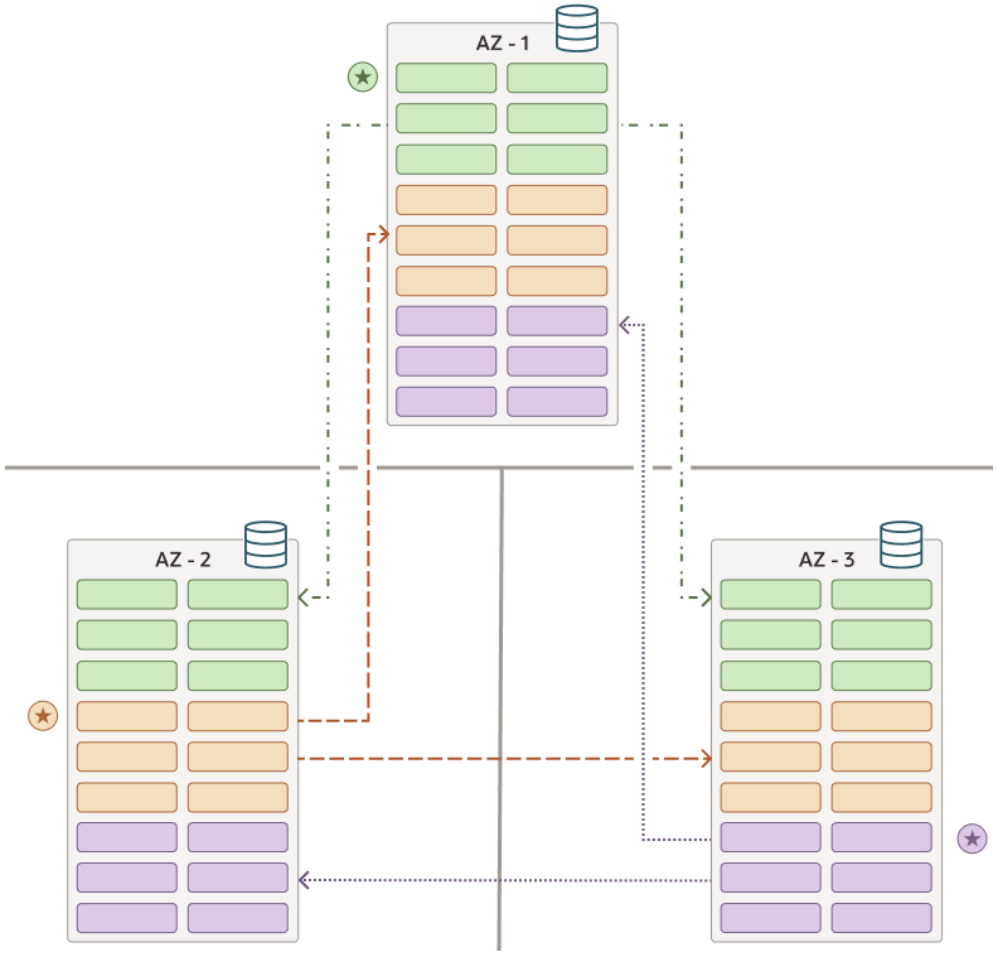
ビッグベンダのNewSQL① - AWS



re:Invent 2024で発表されたAurora DSQL。マルチリージョン採用時に**99.999%**の可用性。ハイスケーラビリティと高可用性の両立を目指す。発表セッションの中で珍しくSpannerに言及しておりライバル視していることがうかがえる。日本で利用する際は、**第三リージョン**(Witness)をどこにするかの問題がある。

Source: AWS, “Amazon Aurora DSQL の紹介”
https://aws.amazon.com/jp/blogs/news/introducing-amazon-aurora-dsql/

ビッグベンダのNewSQL② - Oracle



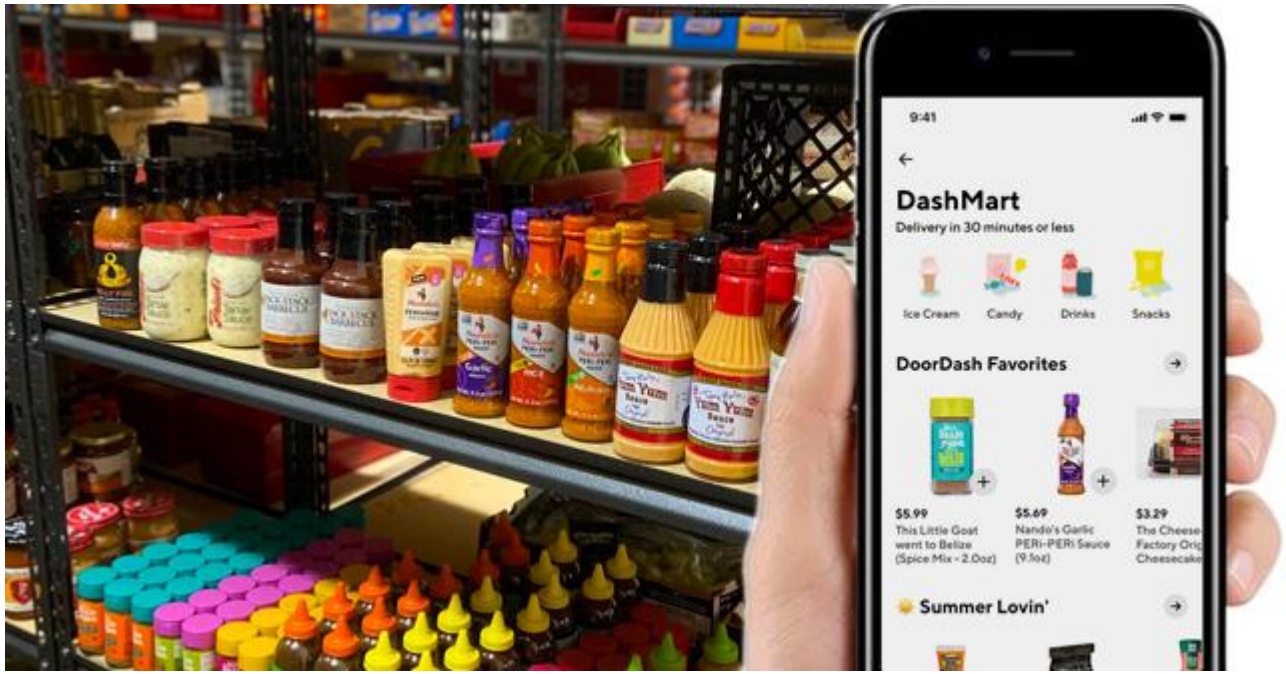
Oracle 23aiよりRaftベースのレプリケーション機能をサポート。
「**データ損失ゼロで高速な自動フェイルオーバー**」を実現する。
すべてのシャードが同じデータ・センターにある場合は、1秒未満のフェイルオーバーを実現できる。
アプリケーションからは通常のOracleのように扱える。

Source: Oracle, “Oracle Globally Distributed DatabaseでのRaftレプリケーションの使用”
https://docs.oracle.com/cd/F82042_01/shard/raft-replication-concepts.html

NewSQLの導入実績

2020年代に入ってから急速に採用が進んでいる。日本ではここ2-3年でアーリーアダプタによる採用が始まったところ。

NewSQLのユースケース(1)
高負荷のリテールEC - DoorDash

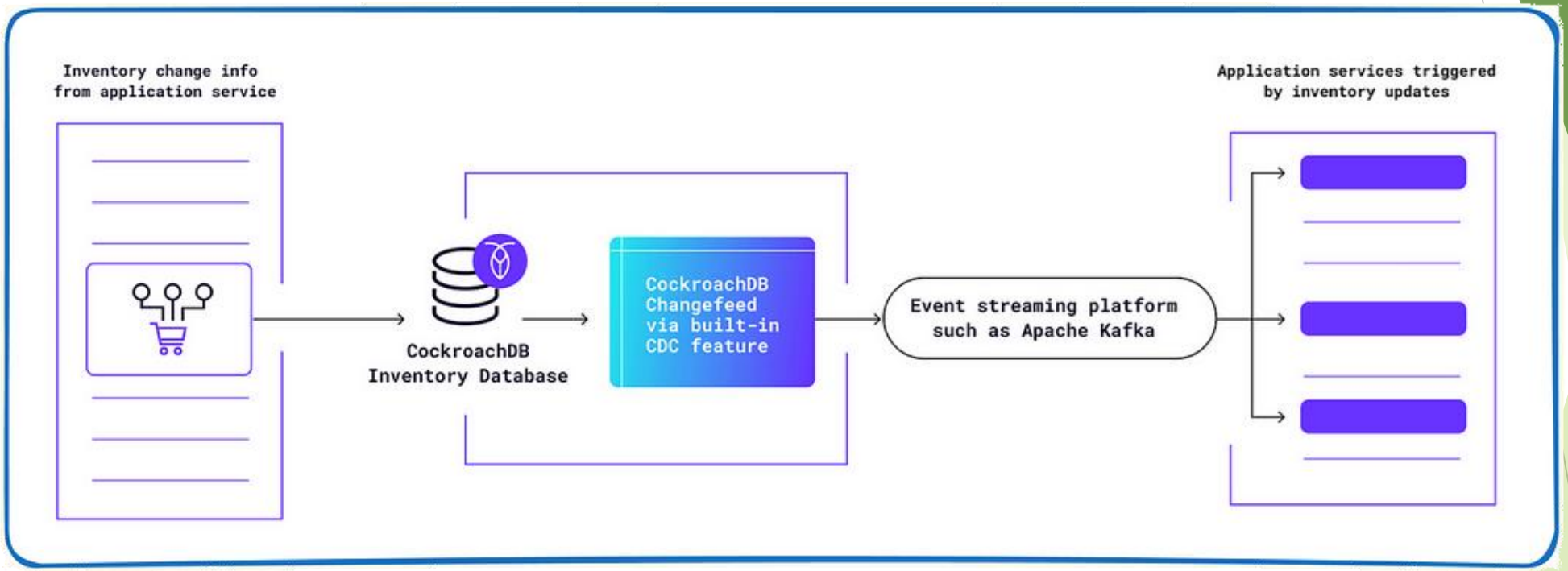


フードデリバリ大手のDoorDashはコンビニなどと協力して生鮮食品を中心とした超高速配送サービスDashmartを開始。これがコロナの巣ごもり需要にマッチし大ヒット。しかし、あまりの高負荷に

在庫ステータスの更新が追い付かず、**過剰販売**や**買い控え**を発生させることになり、前者は顧客体験の悪化に、後者は機会損失につながっていた。

Source: RetailWire, “DoorDash delivers a virtual convenience store”
https://retailwire.com/discussion/doordash-delivers-a-virtual-convenience-store/

NewSQLのユースケース(1)
更新負荷に対してスケールさせる

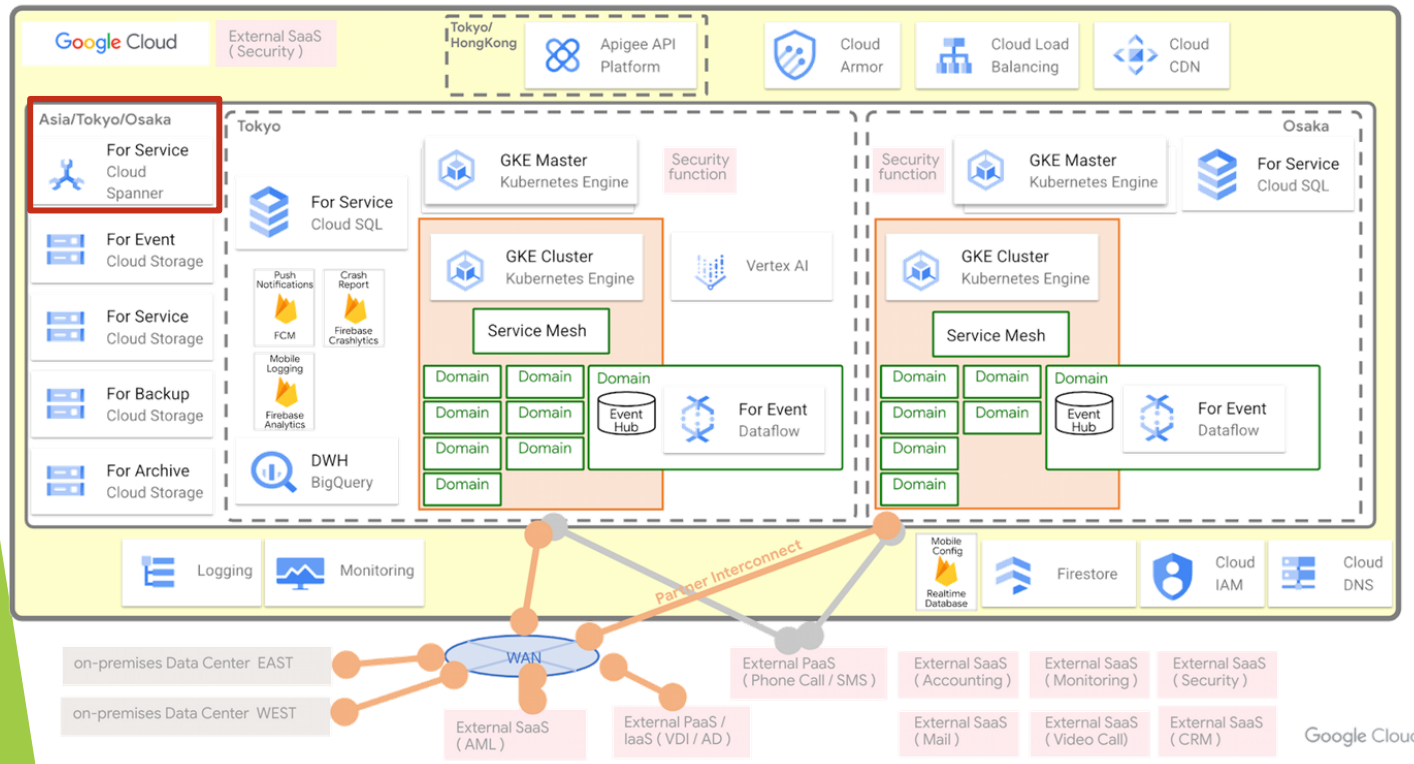


大量の在庫の更新情報を一度CockroachDBで永続化し、CDCで後続のKafkaに流してニアリアルタイムの在庫ステータス反映を実現。**120万QPS**のスループットを実現。

Source: “How DoorDash Used CockroachDB’s Change Feed for Real-Time Inventory Processing？”
https://interviewnoodle.com/how-doordash-used-cockroachdbs-change-feed-for-real-time-inventory-processing-system-design-d06ea70d6a8

NewSQLのユースケース(2)
金融分野における高可用性 - みんなの銀行

みんなの銀行システム～勘定系基盤



> 特に **Cloud Spanner の存在が大きかった**と思います。検討時点ではまだ大きな実績のないプロダクトではあったのですが東京と大阪で同じシステムを同時に動かす "東阪両現用（とうはんりょうげんよう）" は、例えばどちらかで大規模災害が発生した際などでも銀行サービスを停止させないために絶対必要な仕組み。

Source: みんなの銀行：日本初の「デジタルバンク」として Google Cloud に勘定系を構築。Cloud Spanner で銀行基幹システムで求められる可用性を実現
https://cloud.google.com/blog/ja/topics/customers/minna-no-ginko-spanner

NewSQLのユースケース(3)
増えすぎたデータベースの統合 - レバテック

TiDBに移行を決めた背景

”リソース・管理効率”の課題感はかなり高かった

- 開発組織に対してRDSやAuroraのクラスターが増えすぎた
  - システム(マイクロサービス)が増えたことが大きな要因
  - 約80人の開発組織に対して約50個のインスタンス

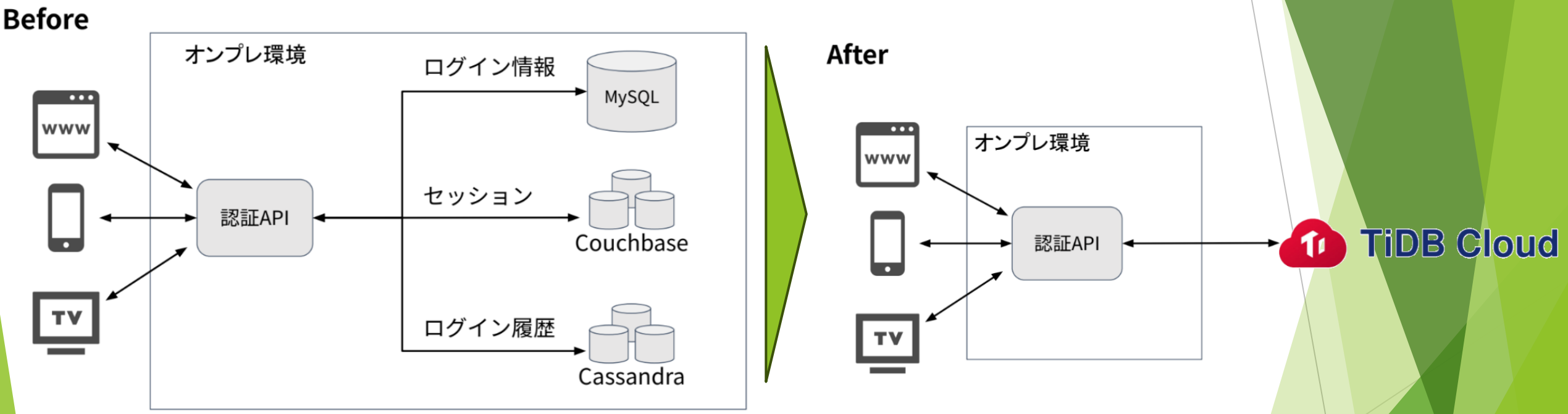
- SREが工数とコストに追われてしまう日々
  - DB運用に問題が発生した際にSREの工数が奪われる
    - クラスターの数だけSREは運用問題と戦う
  - それぞれのクラスターごとに最適なスペック調整が必要
    - 無駄に高いスペックを設定すると無駄なお金のコストになる

SREの人たち

“クラスタを一元管理できて、リソース効率や管理効率が向上できることです。散在していたクラスタをTiDBの1つのクラスタにまとめることができれば管理対象を1つにできます。”

Source: 増えすぎたデータベースと計画停止の苦労を、TiDBへの移行で解決できるか？ レバテックの挑戦
https://www.publickey1.jp/blog/24/tidb_pr.html

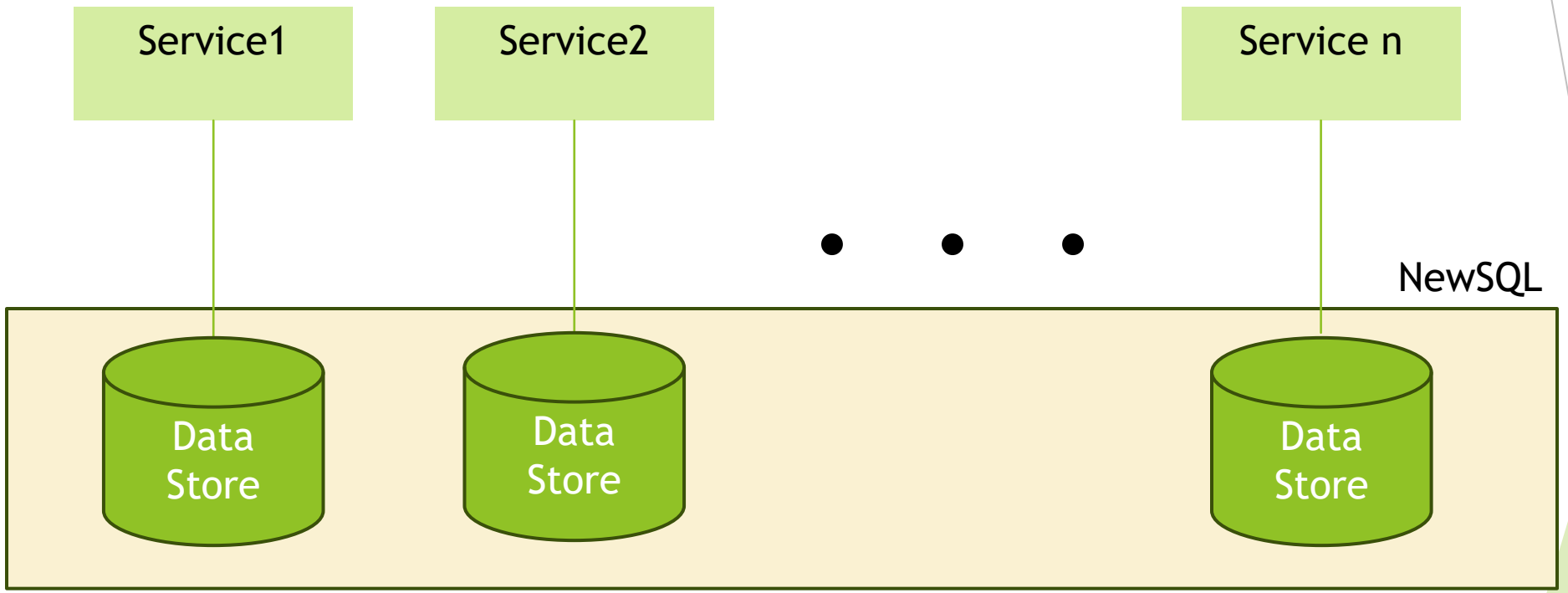
NewSQLのユースケース(4)
異種混合データベースの統合 - DMM.com



3種類のDBに精通しているエンジニアは少ないため習熟のハードルが高く、開発効率が悪かった。それをTiDBで統合することで解決する。

Source:【事例から学ぶ】アーキテクチャ多様化時代にデータベースを「TiDBにまとめる」という選択
https://pingcap.co.jp/case-study/dmm-menu-micoworks/

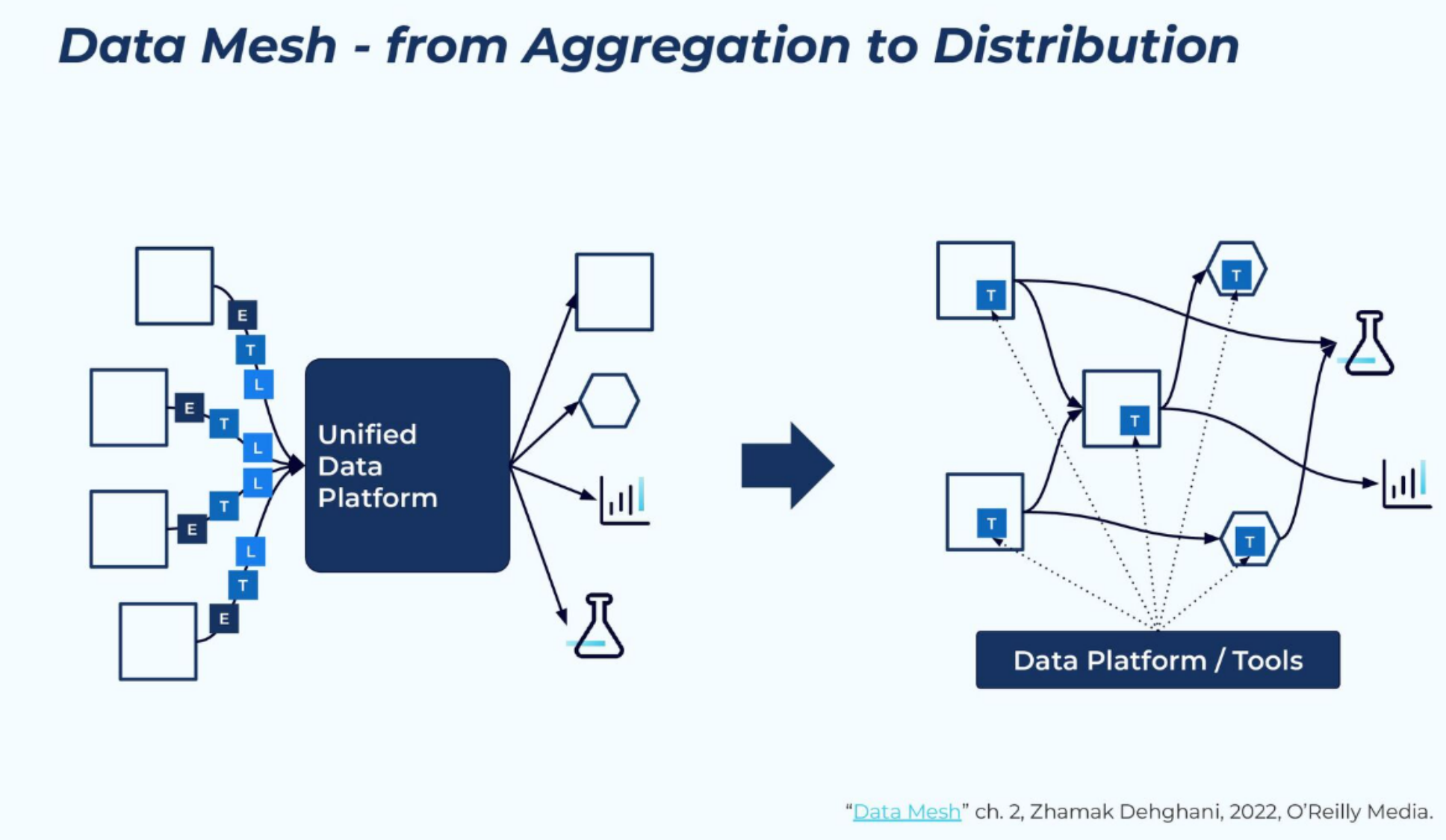
ユースケースの意外なダークホース



マイクロサービスでは通常、サービスごとに求められるキャパシティやデータ特性、ワークロードが異なるため、サービスごとにデータストアを持つことがセオリーとされているが・・・

統合と分散の間で

統合と分散の間で - Data Lake vs Data Mesh



Source: Microservice, Data, and Data Mesh
https://speakerdeck.com/hashitokyo/microservice-data-and-data-mesh

NewSQLに欠点はないのか？

- **レスポンスタイム（レイテンシ）が悪化する**
  これ自体はNewSQLに限らず分散システム全体に共通する難点だが、NW通信のオーバーヘッドが乗ってしまう。

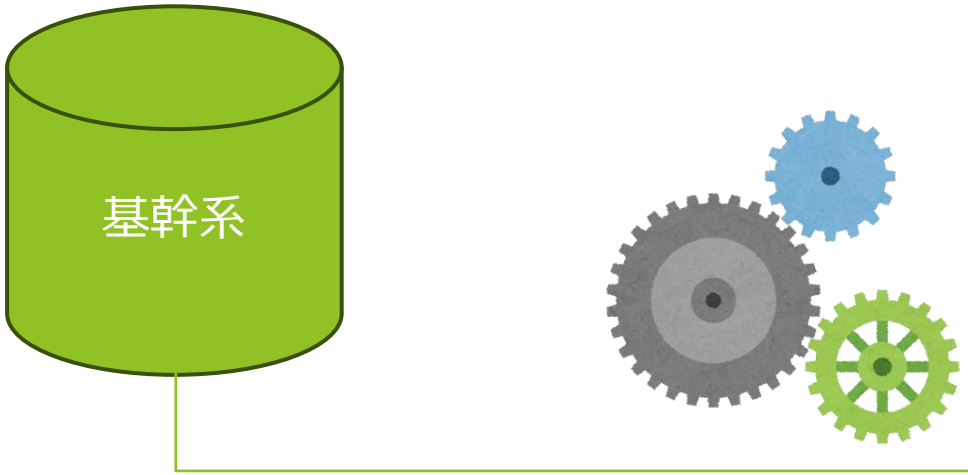
- **まだ比較的割高**
  まだ広く普及しておらず薄利多売モデルが構築できていないのと、R&D費が乗ってしまうため比較的お高め。

- **障害対応が大変**
  分散構成はコンポーネントが増えて複雑になるため、いざ障害や遅延が起きたときの原因調査が大変。近年はマネージドサービス化も進んでいるので、ベンダに投げてしまう選択肢も出てきている。

データベースの新潮流②
HTAP – スーパーデータベースの夢

アーキテクチャの黄金律

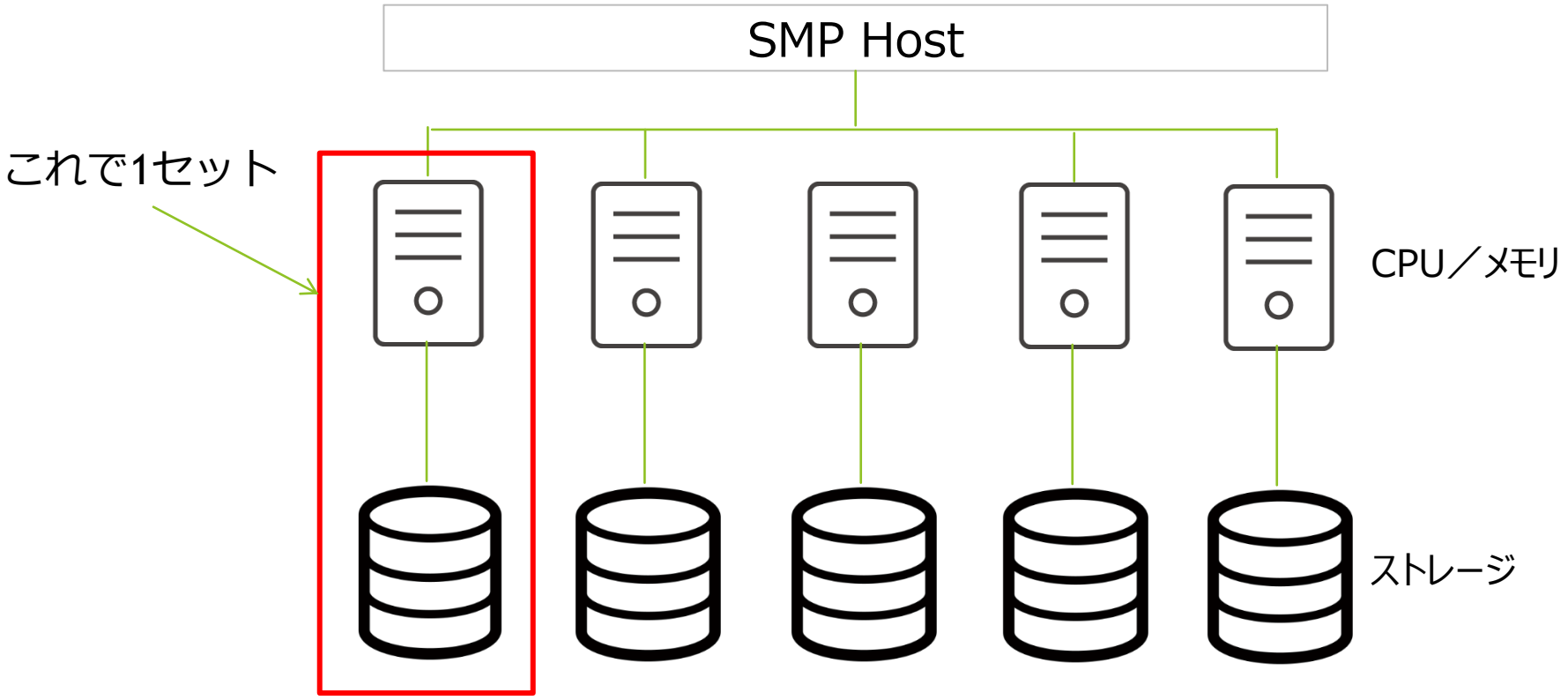
・多数のショートクエリ
・90%ile 3秒
・ダウンするとメチャクチャ怒られる



・少数のヘビークエリ
・数十秒～数分でOK
・所詮社内システム

基幹系（OLTP）と情報系（OLAP）の分離は絶対原則

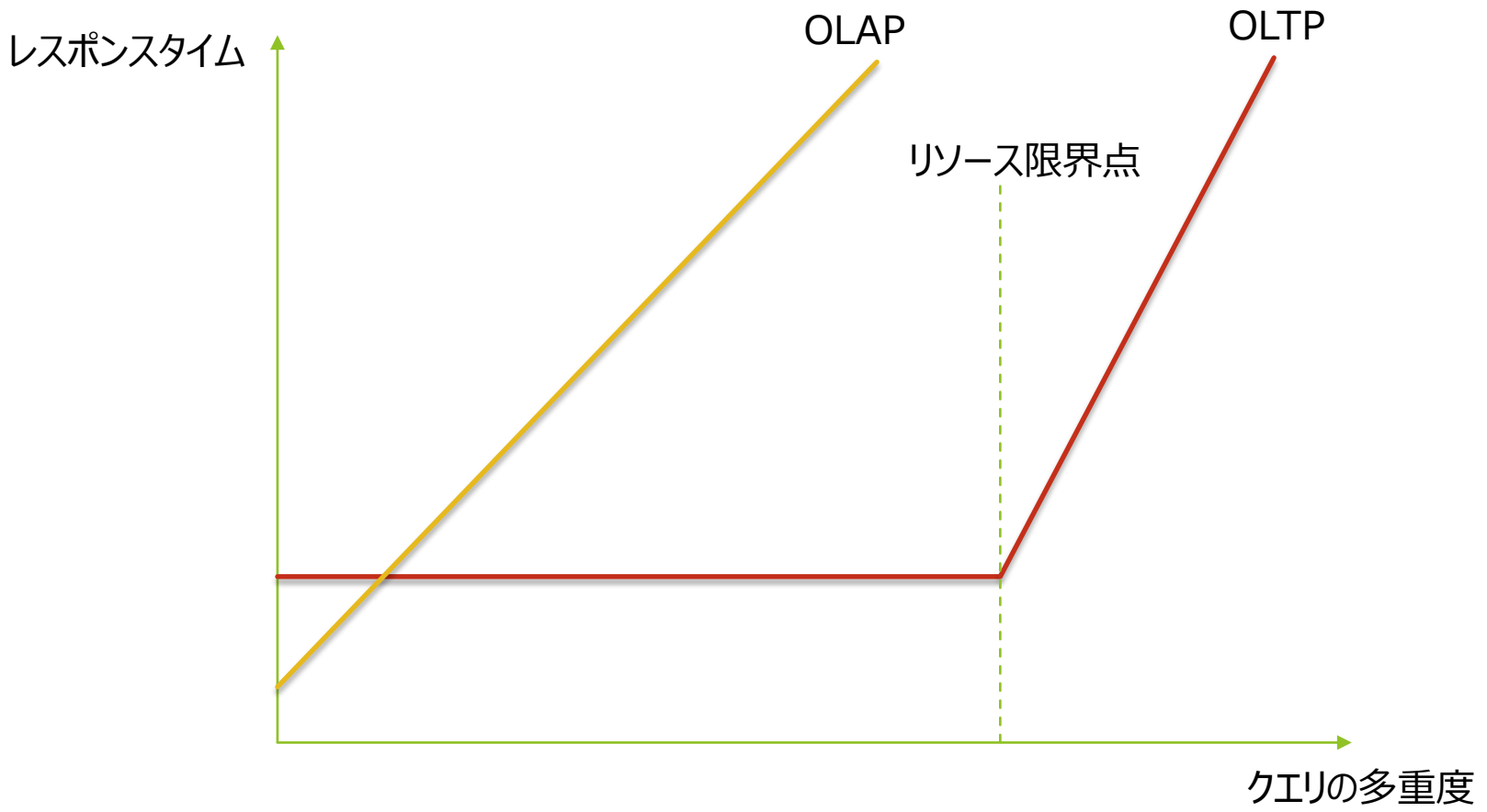
独自の進化を遂げた情報系アーキテクチャ



シェアードナッシングによるMPP構成。すべてのリソースを使い切ることで限界までレスポンス高速化に全振りした思想。

MPP = Massive Parallel Processing (超並列分散処理)

MPPの掟：OLTPに使うのはご法度



MPPは優れたアーキテクチャではあるが、クエリの多重度に比例してクエリのレスポンスタイムは悪化するのでOLTP向きではない。

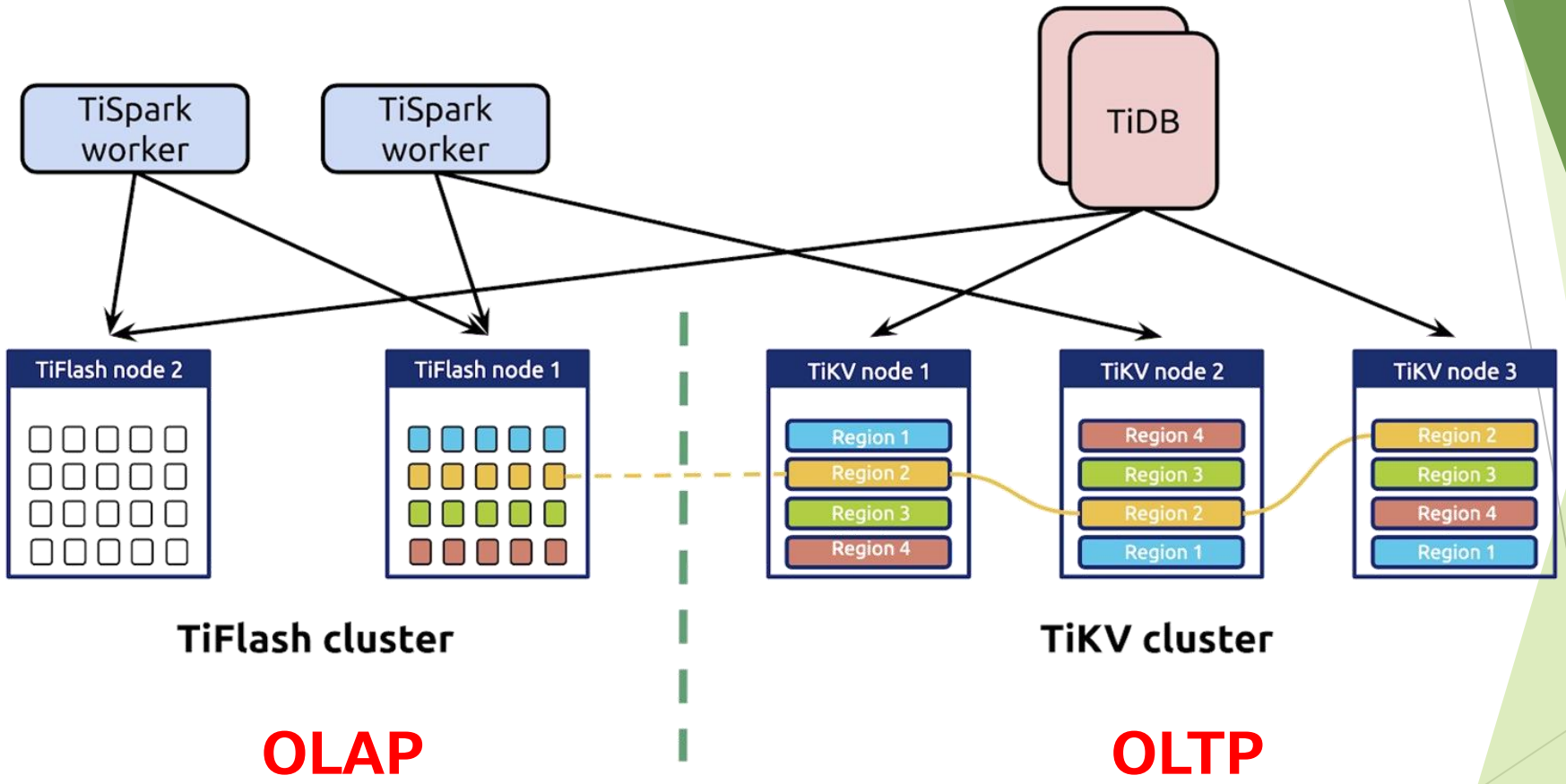
それでもリアルタイム分析がしたい

> 複雑なアナリティクスを、データマートやデータウェアハウスにあるデータを抽出して行うのではなく、**ライブのトランザクションデータ**と同じデータセットを使って実行する

・・・もちろん、それができればそれに越したことはないのだが・・・

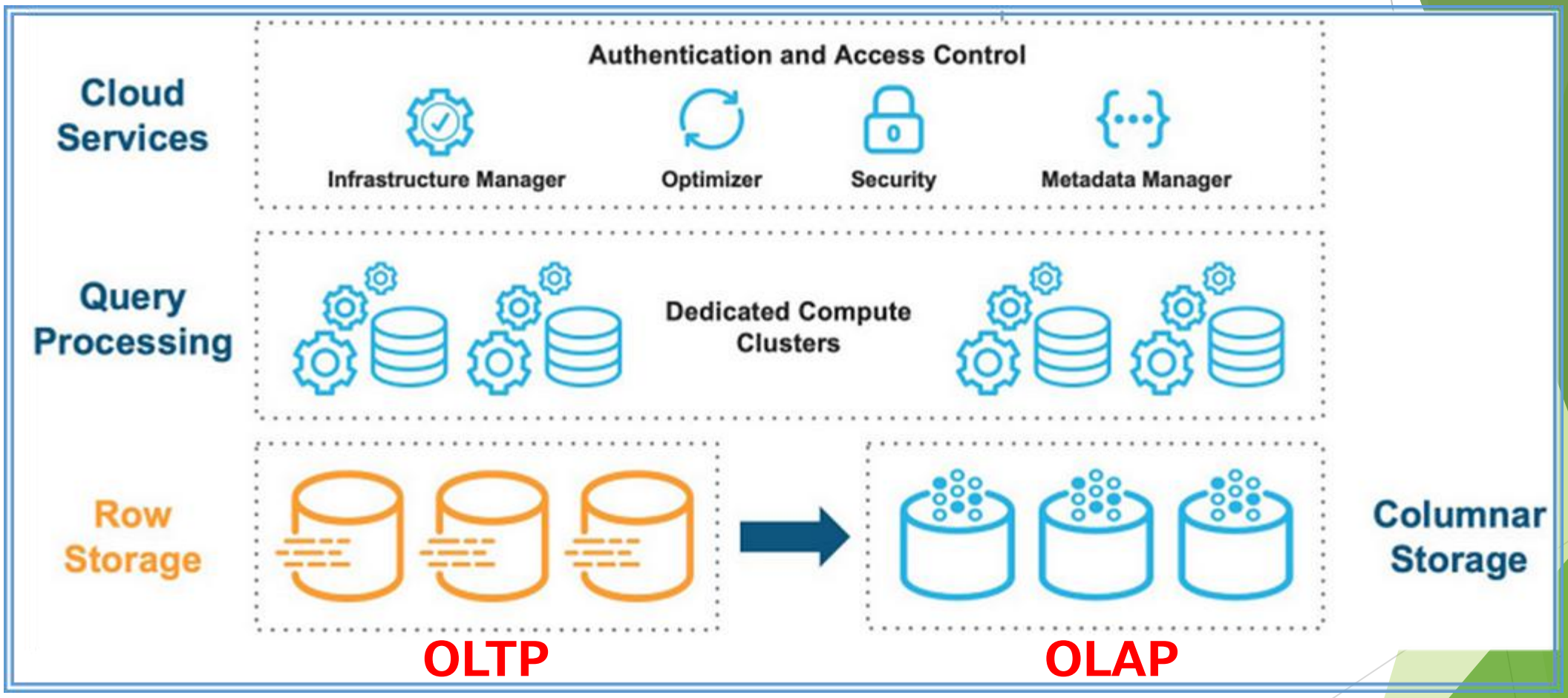
※HTAP(**H**ybrid **T**ransactional and **A**nalytical **P**rocessing)が提唱されたのは2014年、ガートナー社が定義した。

HTAPのアーキテクチャ



OLTP向けの行ベーストアとOLAP向けのカラムベーストアを両方用意してデータ同期を行うことで両方のクエリを扱う。

Snowflake Unistore

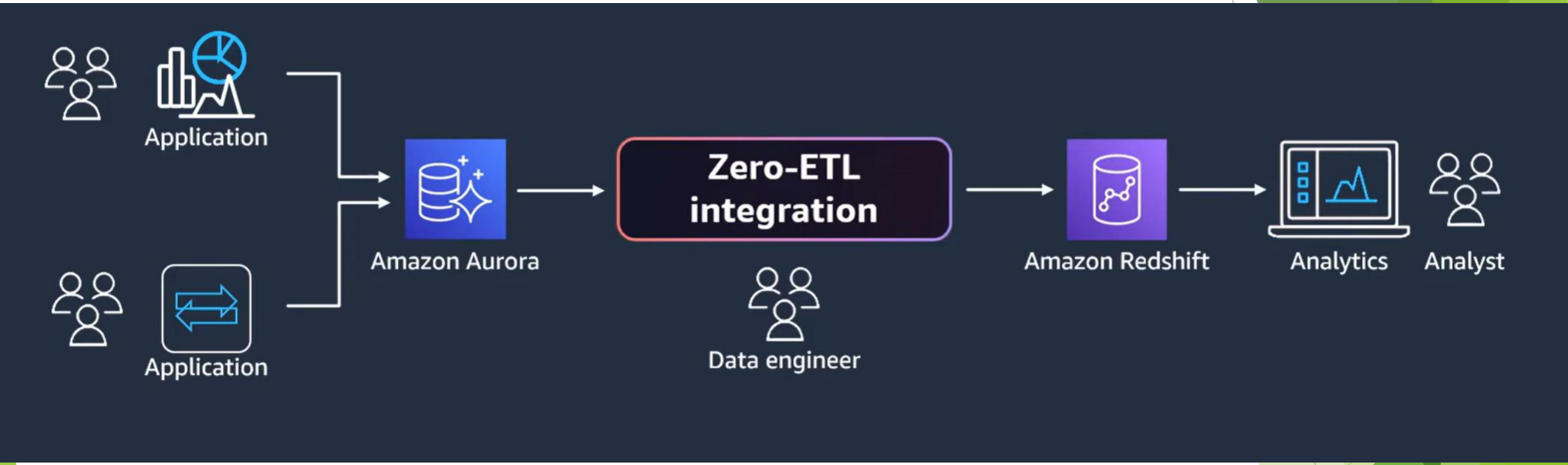


Snowflake社のUnistoreも発想としては同じで、行ベースとカラムベースのストアを両方持つというアーキテクチャ。

HTAPの懸念点

- **性能**：OLAP向けのクエリによって本当にOLTP向けのクエリを遅延させたりスループットが悪化することはないのだろうか？ リソースは完全に分離されているのだろうか？
- **信頼性**：OLAP向けのアーキテクチャで障害が起きたことによってOLTP側のシステムも巻き込み事故で障害が発生することはないだろうか？
- **データ量**：OLTP向けとOLAP向けでデータのコピーを持ちあうとかなり全体のデータ量が増えるのではないか？ (特にNewSQL的なDBはデータのコピーをノード間で持ちあうため)

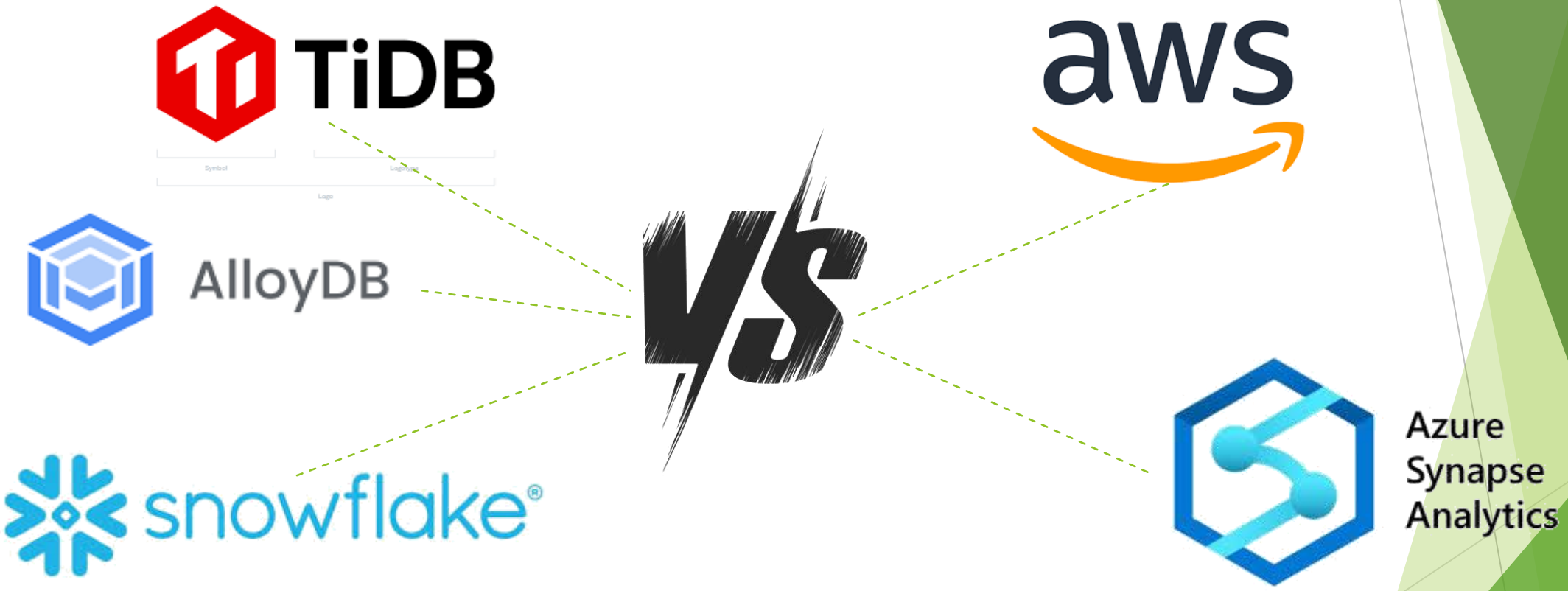
疎結合型HTAP - AWSからの回答



OLTP（Aurora）とOLAP（Redshift）はあくまでデータベースを**分離**しておく。その間のETLをCDCによるレプリケーションで簡略化する。

Source: How Infosys used Amazon Aurora zero-ETL integration with Amazon Redshift for near real-time analytics and insights
https://aws.amazon.com/jp/blogs/database/how-infosys-used-amazon-aurora-zero-etl-to-amazon-redshift-for-near-real-time-analytics-and-insights/

HTAP 密結合型 vs 疎結合型

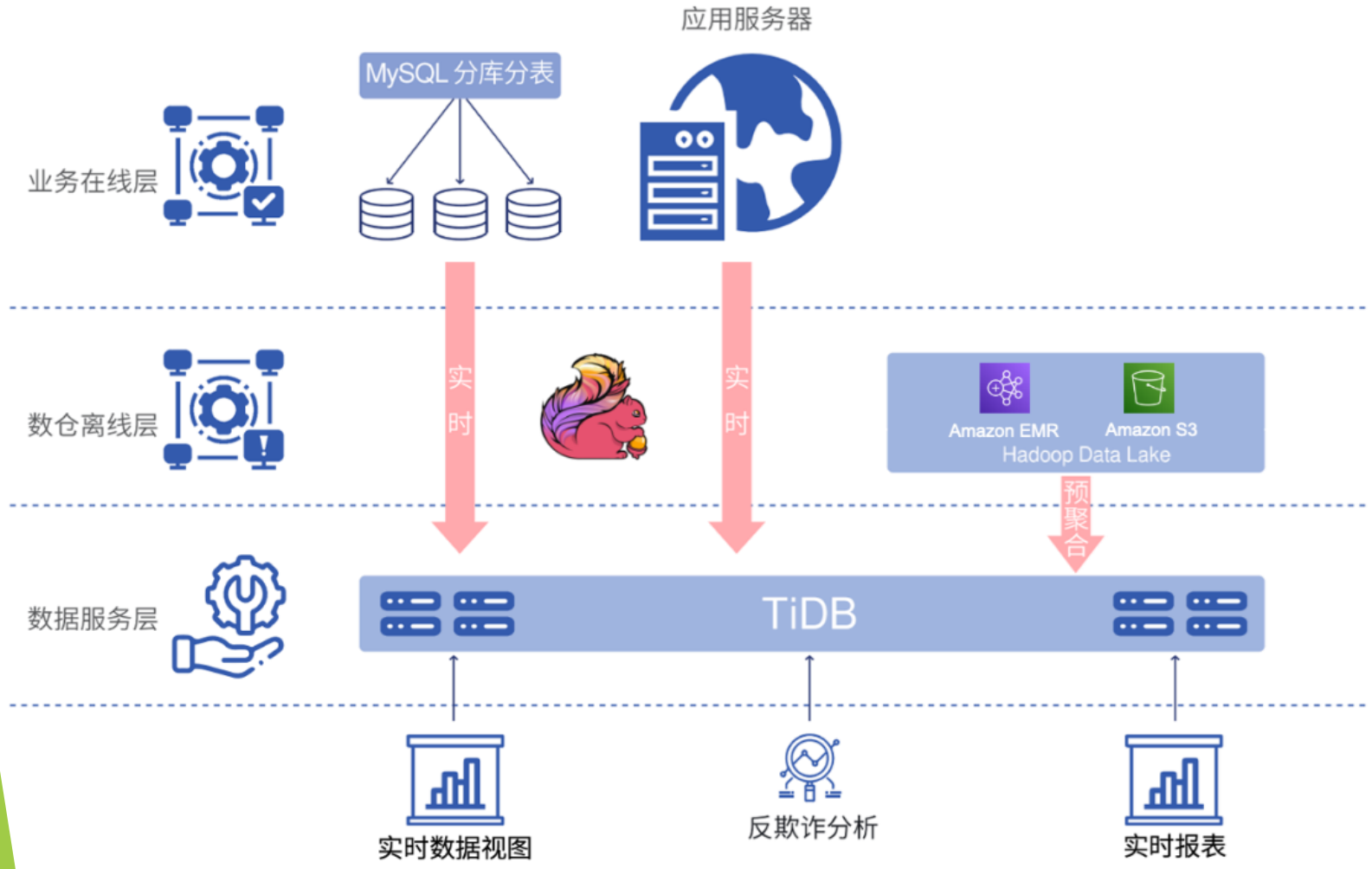


密結合型も疎結合型も、どちらもまだデファクトの位置を占めるには至っていない。日本で受け入れられるのはどちらか？

HTAPのユースケース

- **リアルタイムマーケティング**：ライブコマースにおいてリアルタイムで売り上げや離脱率などを把握しパーソナライズド・レコメンデーションを行う。
- **不正検知**：金融におけるアンチ・マネーロンダリングや送金・決済時における犯罪行為の検知と即時ブロック。
- **需要予測**：製造業における製品の在庫数をリアルタイムで更新し、過去の販売データを基に需要予測を行う。
- **IoT予測分析**：デバイスから収集したセンサーデータをリアルタイム分析することで機器の故障予測や予兆検知を行う。

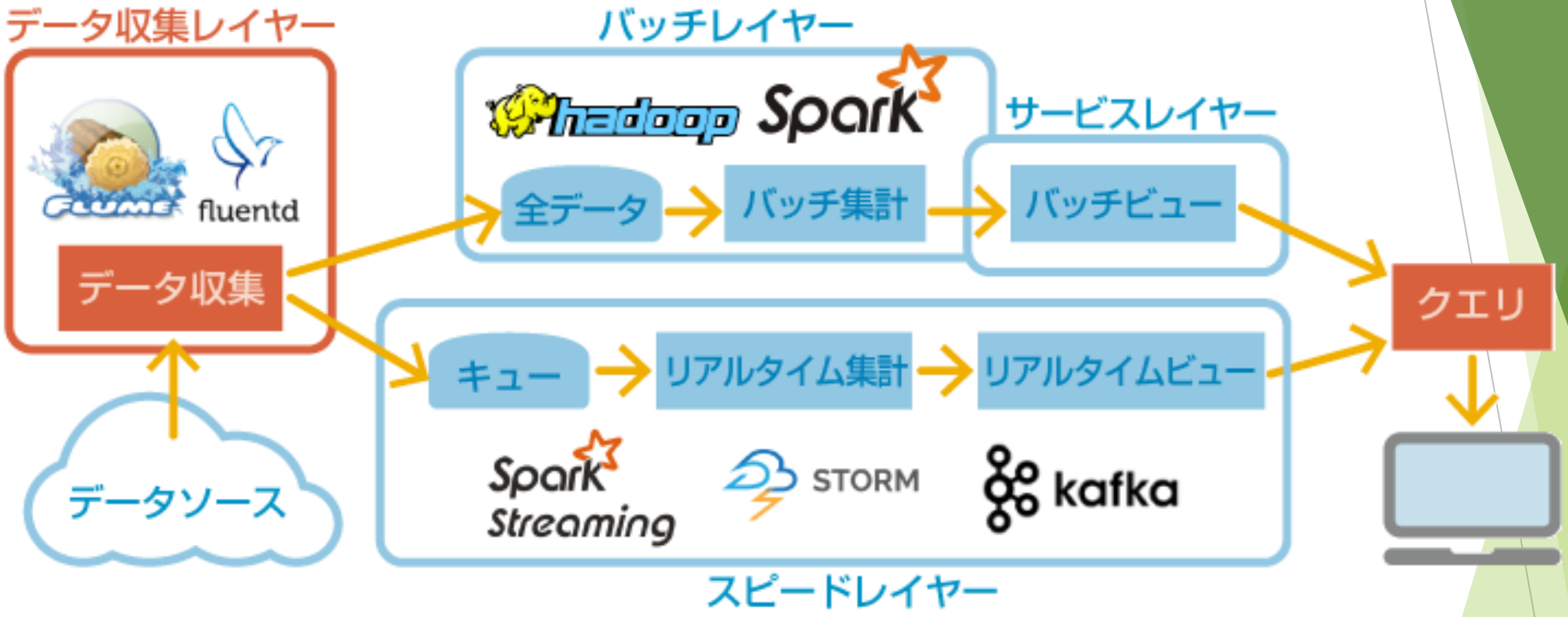
ECにおける不正検知とマーケティング



登録ユーザ数**3億人**の超巨大SNS/ECプットフォーム。ライブコマースを特徴としている。不正検知やマーケティングに従来Hadoopを使っていたところ、リアルタイム性を求めてTiDBを選択。

Source: PingCAP, “TiDB x 小红书 | TiDB HTAP 助力小红书业务升级”
https://cn.pingcap.com/case/user-case-xiaohongshu/

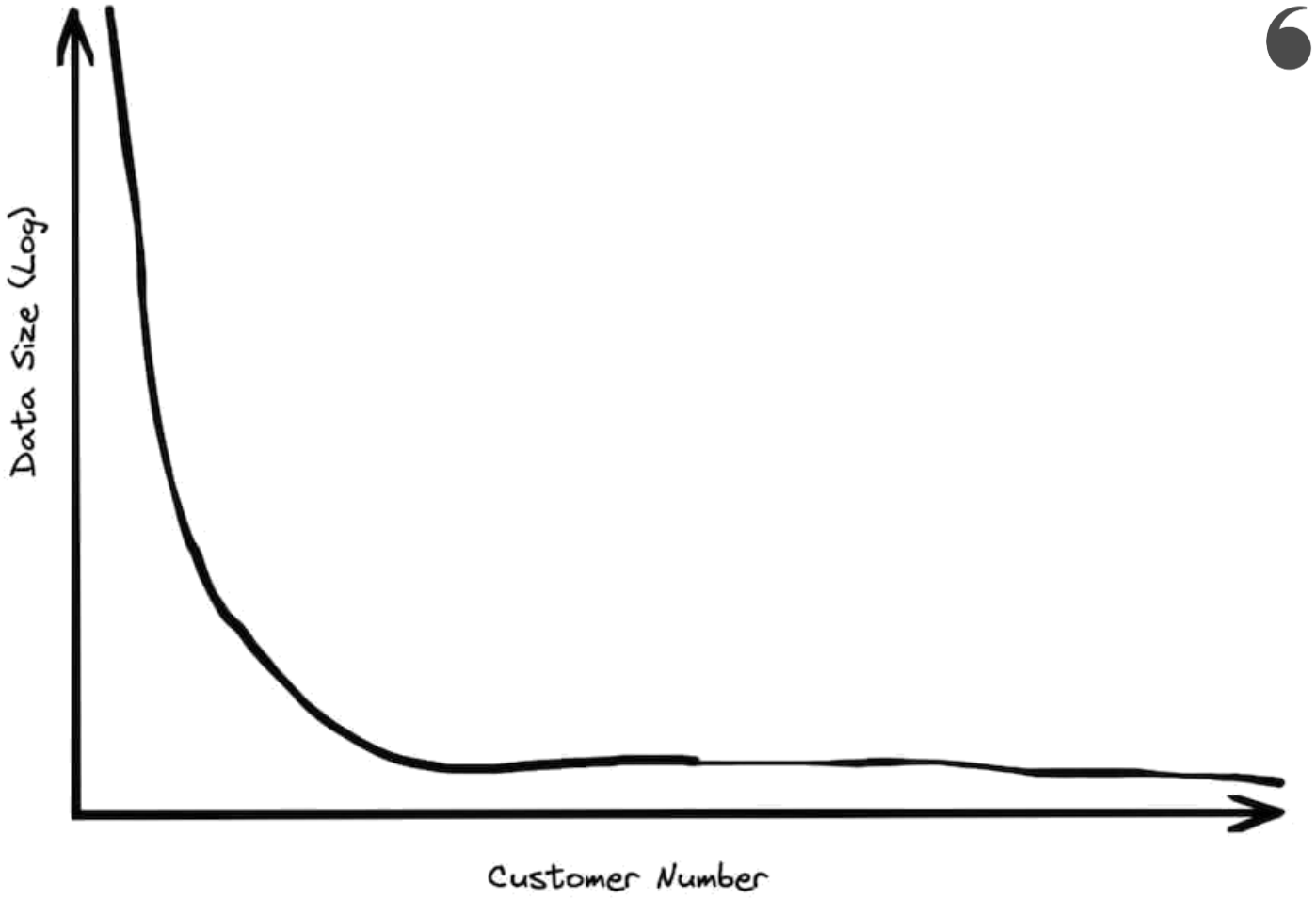
ラムダアーキテクチャはなぜ流行らなかったのか



2012年に提唱されたラムダアーキテクチャは、速報系と確報系の二系統のデータストリームを利用することでリアルタイム分析を実現しようとしたのだが・・・。

Source: NTTデータ先端技術, “Big Data + Fast Data = ラムダアーキテクチャー！”
https://www.intellilink.co.jp/column/bigdata/2015/041500.aspx

分析すべきビッグデータなどなかった



Source: Jordan Tigani, “BIG DATA IS DEAD”
https://motherduck.com/blog/big-data-is-dead/

> なぜビッグデータが稀なのかを理解するためには、データが実際にどう生成されるかを考えると分かりやすい。あなたが中規模企業で、1000人の顧客を抱えているとする。各顧客が毎日100行の新規注文をするとしよう。これはありそうなシナリオだが、それでも1日に生成されるデータは1メガバイト以下だろう。テラバイトに達するには**千年**かかる。

Snowflakeがなぜ広まったのかを考えよ

従来、情報系というのは多額の初期投資が必要な一部の大企業の贅沢品だった。それを中小企業にも門戸を開いた「**データの民主化**」がSnowflake最大の功績。ここから分かることは、**99%のマス層**である中小企業が使いたくなるサービスでなければ、サービスや製品としては普及しない。

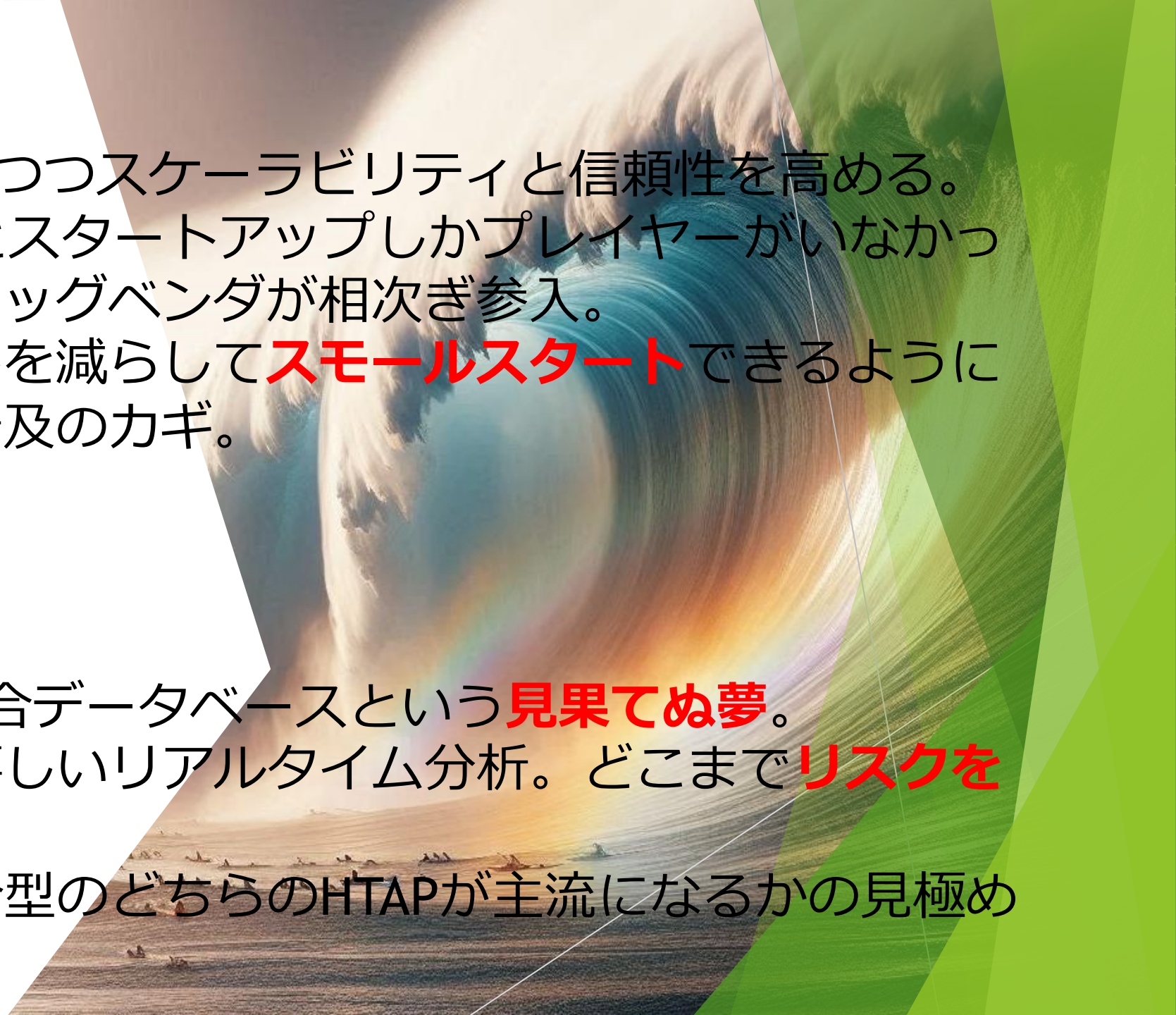
Source: 日立ソリューションズ, “Snowflakeとは”
https://www.hitachi-solutions.co.jp/snowflake/

【NewSQL】

- RDBの長所を残しつつスケーラビリティと信頼性を高める。
- 今まではGoogleとスタートアップしかプレイヤーがいなかったが、昨年からビッグベンダが相次ぎ参入。
- 運用負荷とコストを減らして**スモールスタート**できるようになるかが今後の普及のカギ。

【HTAP】

- OLTPとOLAPの統合データベースという**見果てぬ夢**。
- できたらみんな嬉しいリアルタイム分析。どこまで**リスクを許容**するか？
- 密結合型と疎結合型のどちらのHTAPが主流になるかの見極めが重要。



END