dbtech showcase 2020

ペタバイト級IoTデータを高速に処理するスケールアウト型データベースGridDB

TOSHIBA

東芝デジタルソリューションズ株式会社
服部　雅一

日本データベース学会の第4回「業績賞」受賞

「GridDB®」の研究・開発、製品化への取り組みが、DB技術と産業の発展に貢献したと評価

業績賞

東芝デジタルソリューションズ株式会社殿
株式会社東芝 生産技術センター殿

貴社は高頻度で大量に発生するデータの管理に適したスケールアウト型のデータベースとして「GridDB」の開発・製品化を行うとともに更にその適用範囲を広げ我が国におけるデータベース技術と産業の発展に多大な貢献をされましたここにその業績を永くたたえ本会業績賞規程により表彰します

令和二年三月二日
日本データベース学会
会長 喜連川 優

01 背景

02 GridDBの特長

03 ラムダアーキテクチャからの脱却

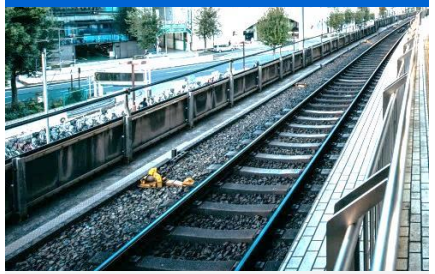
04 OSSについて

01

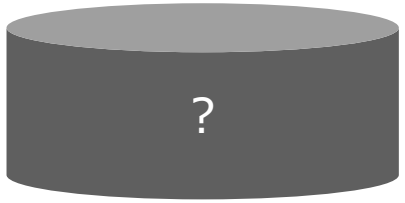
背景

社会インフラでも大規模なIoTデータが出現

- ビジネスの価値源泉は”データ”

電　力	製造業	鉄道・交通	物　流	地域・社会
				
・電力託送の使用量計算・料金計算 ・発電の遠隔監視	・産業用機器の監視 ・製造工場の品質管理 ・製造工場の歩留り向上、生産性向上	・鉄道の輸送計画 ・運行監視	・配送用トラック荷物の監視・分析	・スマートXXX ・地域の家庭、ビル、工場施設等の統合管理

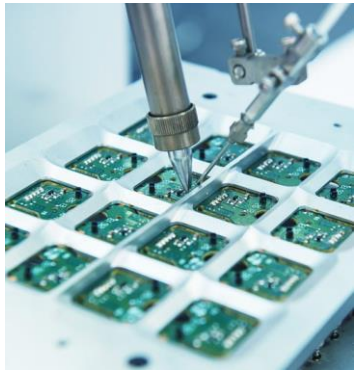
- 支える側の技術進化

収集層	蓄積層	分析・利活用層
・IT機器の普及 ・5Gなどネットワーク技術		・ＡＩや大規模データ分析処理技術 ・GPU大規模コンピューティング

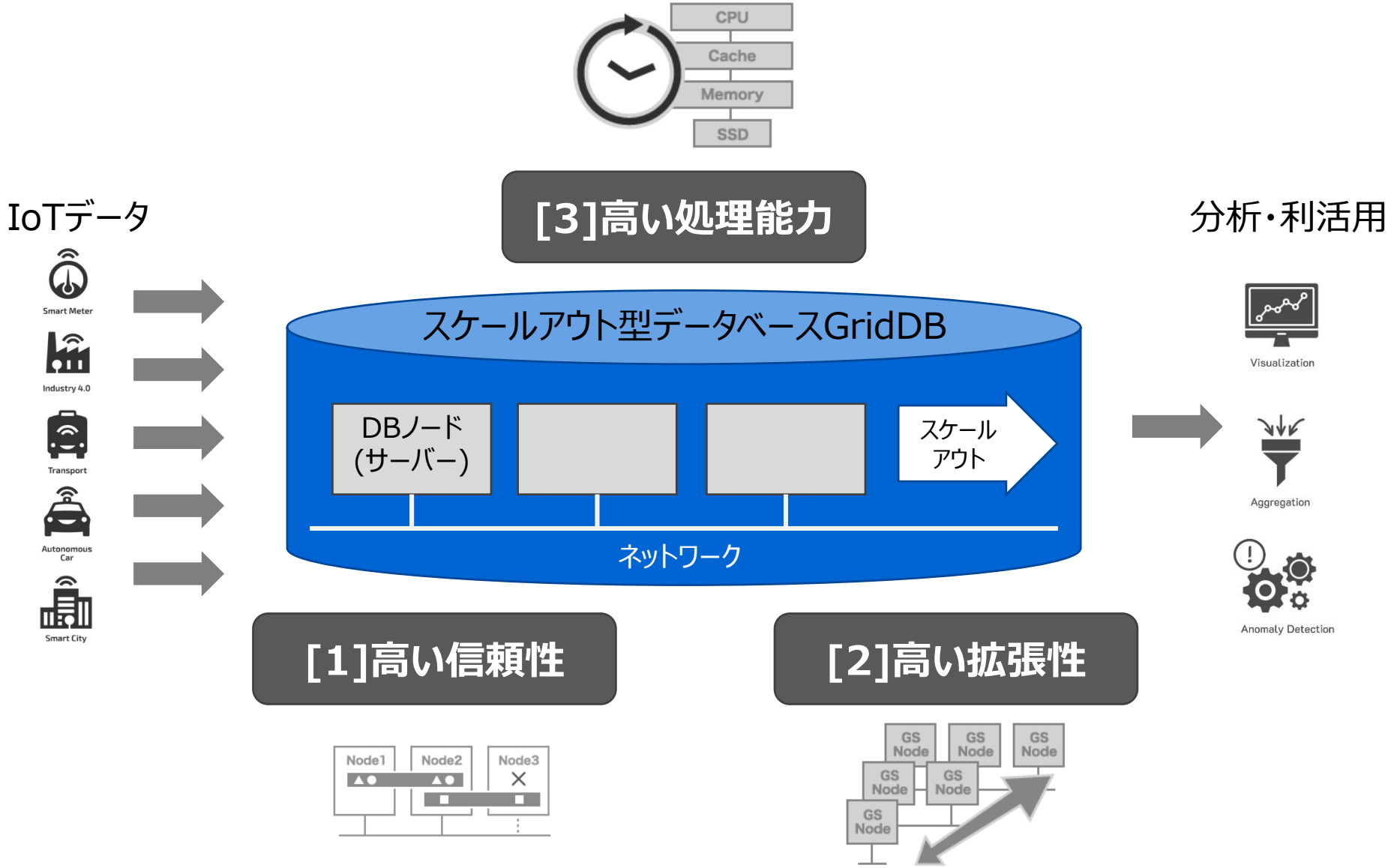
Why … GridDB ?

［Ⅰ］高い信頼性	• アプリの多くはミッションクリティカル • ベストエフォートはNG
［Ⅱ］高い拡張性	• 増大し続けるデータ量 • 台数の増加による処理性能の向上→”スケールアウト”
［Ⅲ］高い処理能力	• 秒やミリ秒の間隔で発生する大量のデータ • 発生直後からリアルタイムに参照や加工などのデータ処理

RDB	• 性能が低い。データが増えた時に対応できない。
HDFS	• リアルタイム性能が悪い。信頼性が低い。
NoSQL	• 信頼性が低い。



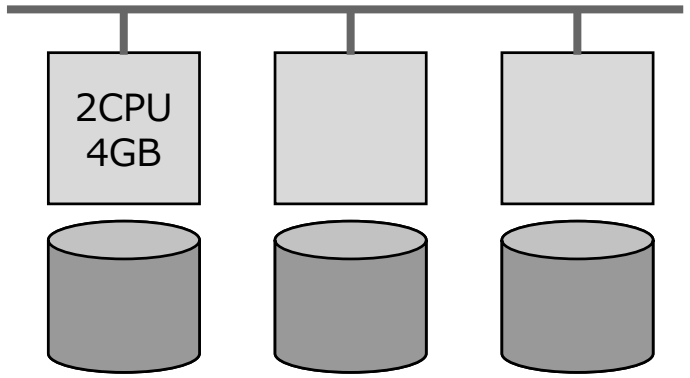
GridDBの位置づけ



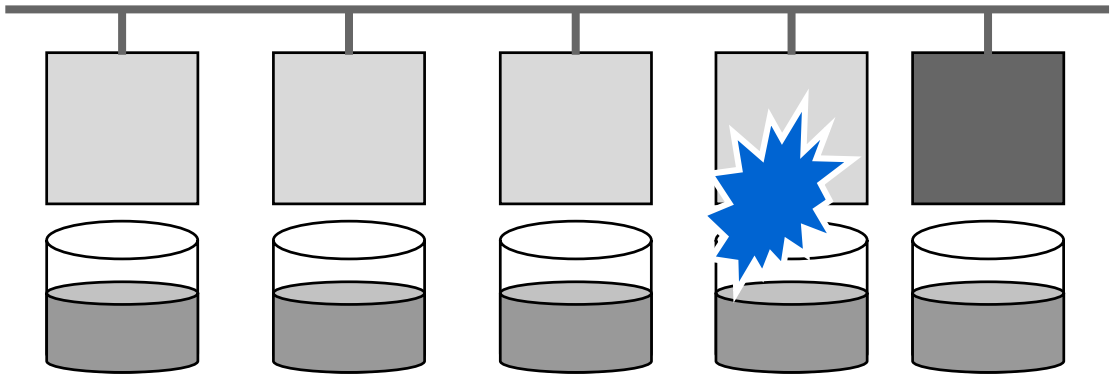
02

GridDBの特長

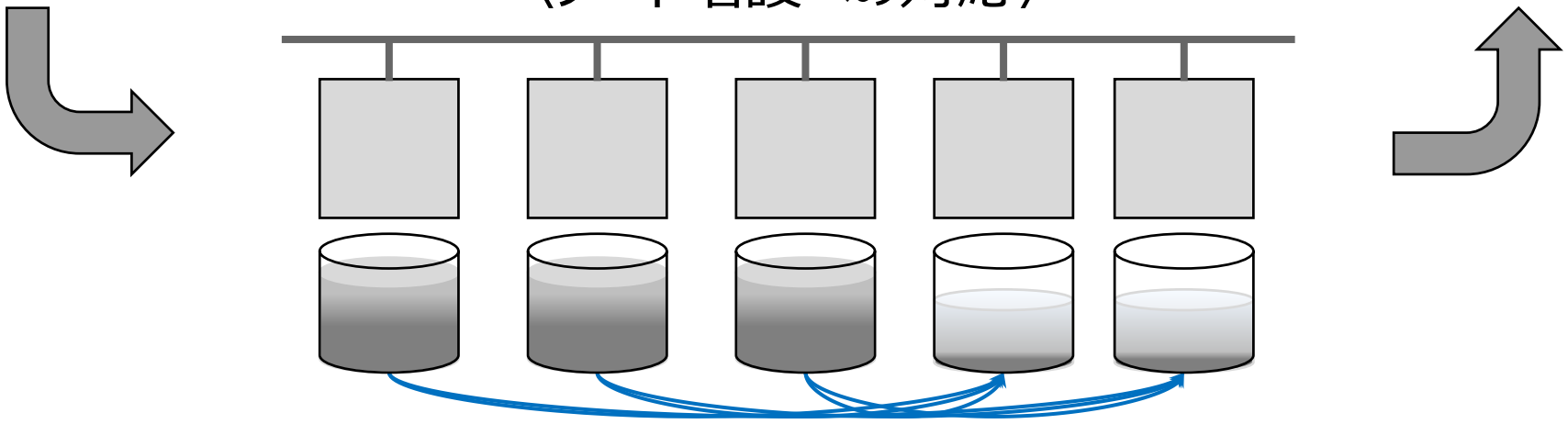
拡張性 “スケールアウト”の課題



〈台数増→ノード障害への対応〉



〈ノード増設への対応〉

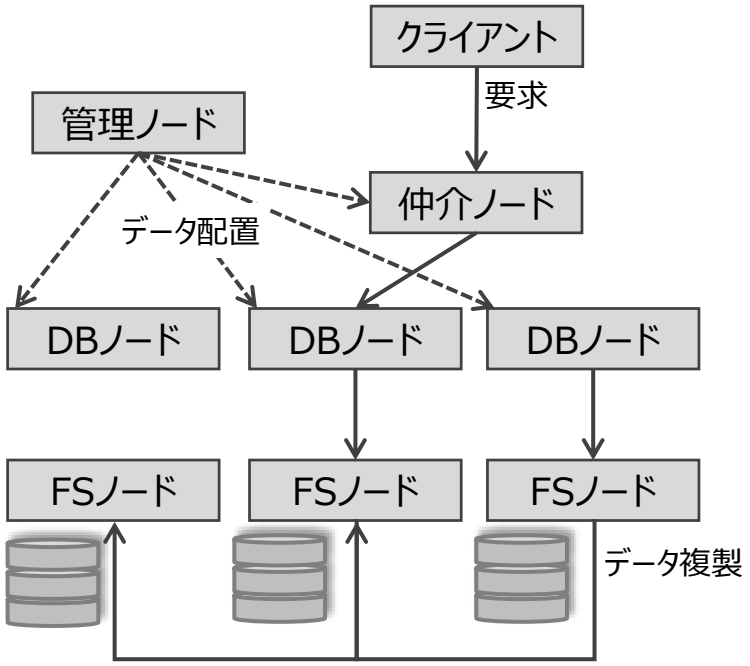


従来のDBクラスタ

- トレードオフ

  「データの分散化」 → データの一貫性が弱くなる

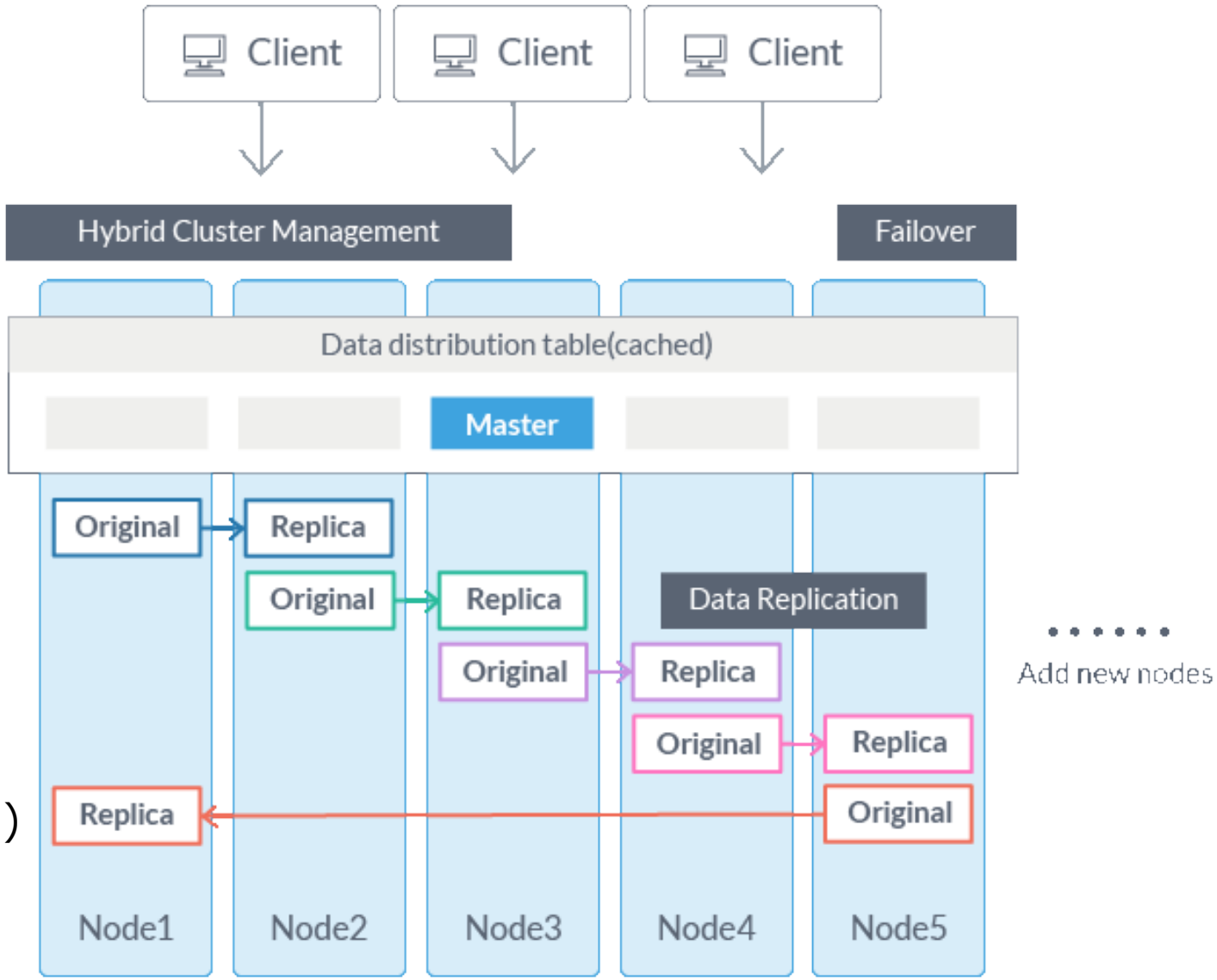
  逆に「一貫性を強める」 → パフォーマンスが落ちる



〈マスタースレーブ方式の場合〉
- データの一貫性を維持しやすい
- クライアントとDBノードの間に存在する管理ノードや仲介ノードがボトルネック

→パフォーマンス低下を引き起こす。

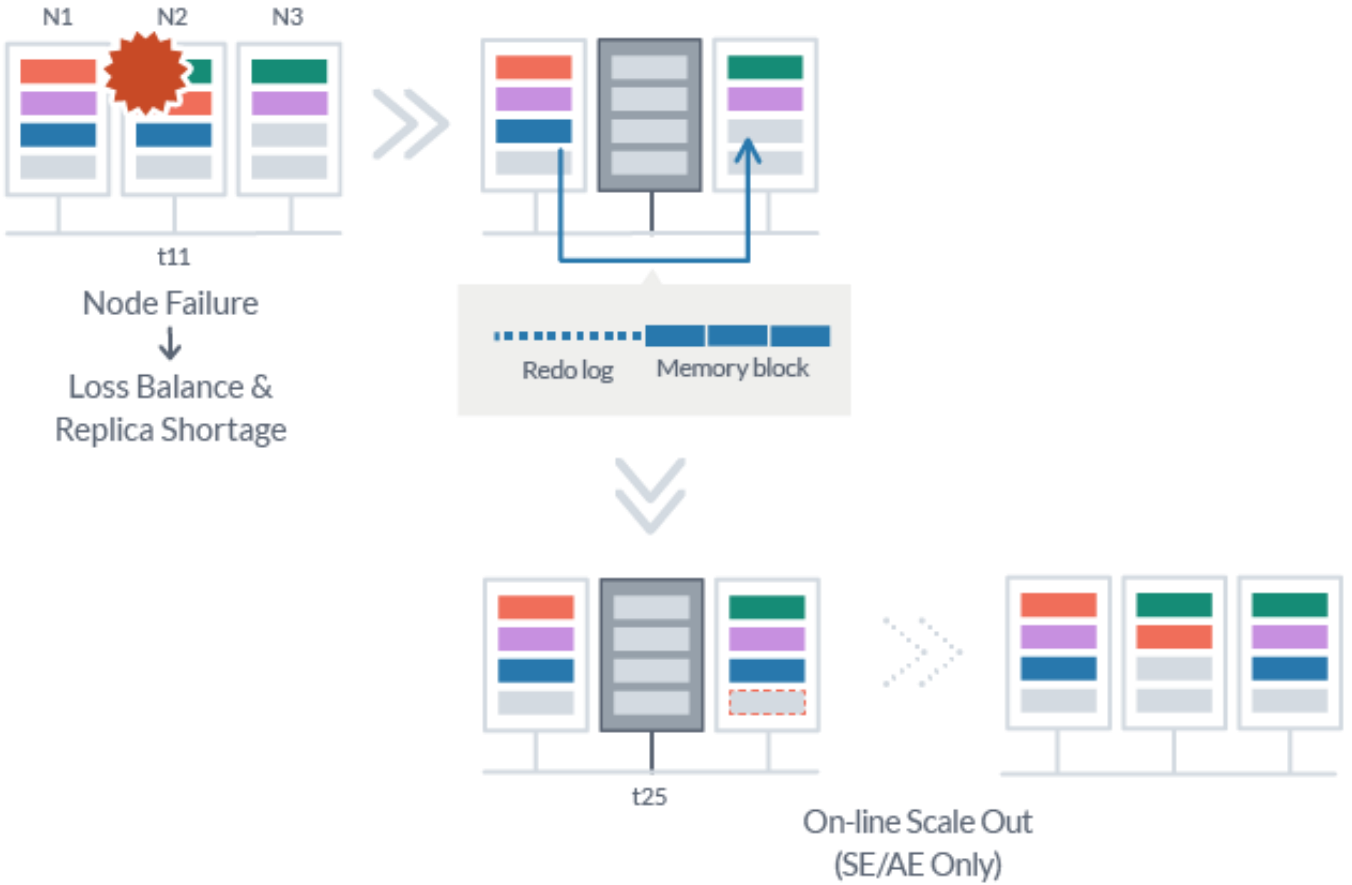
GridDBの自律型DBクラスタ



データ複製(レプリカ)

GridDBの自律型DBクラスタ

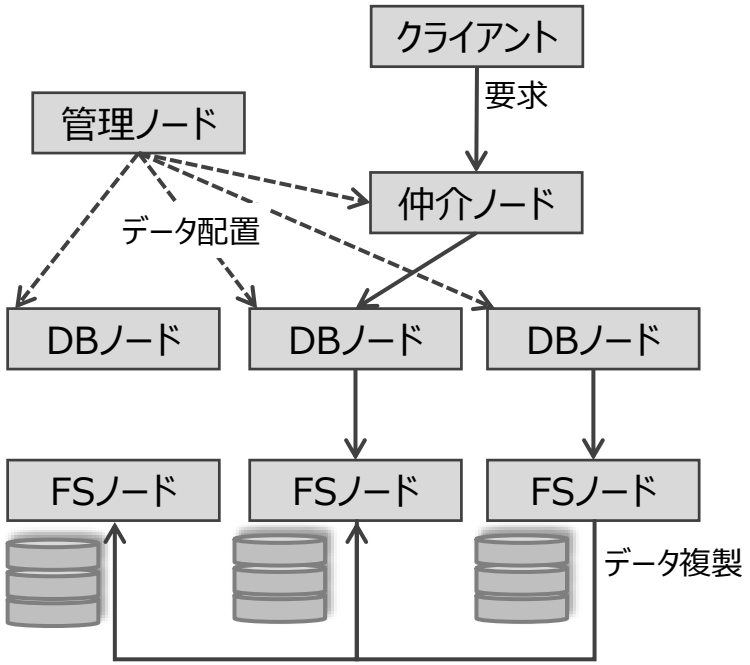
〈ノード障害への対応〉



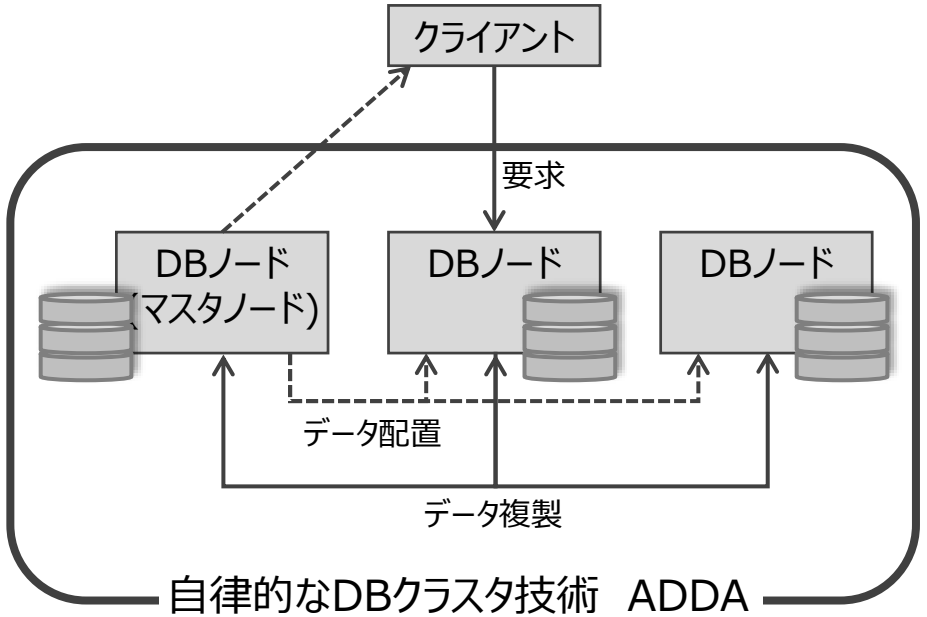
〈ノード増設への対応〉

DBクラスタ構成の比較

〈従来技術〉



〈GridDB〉



- 管理ノードや仲介ノードは存在せず

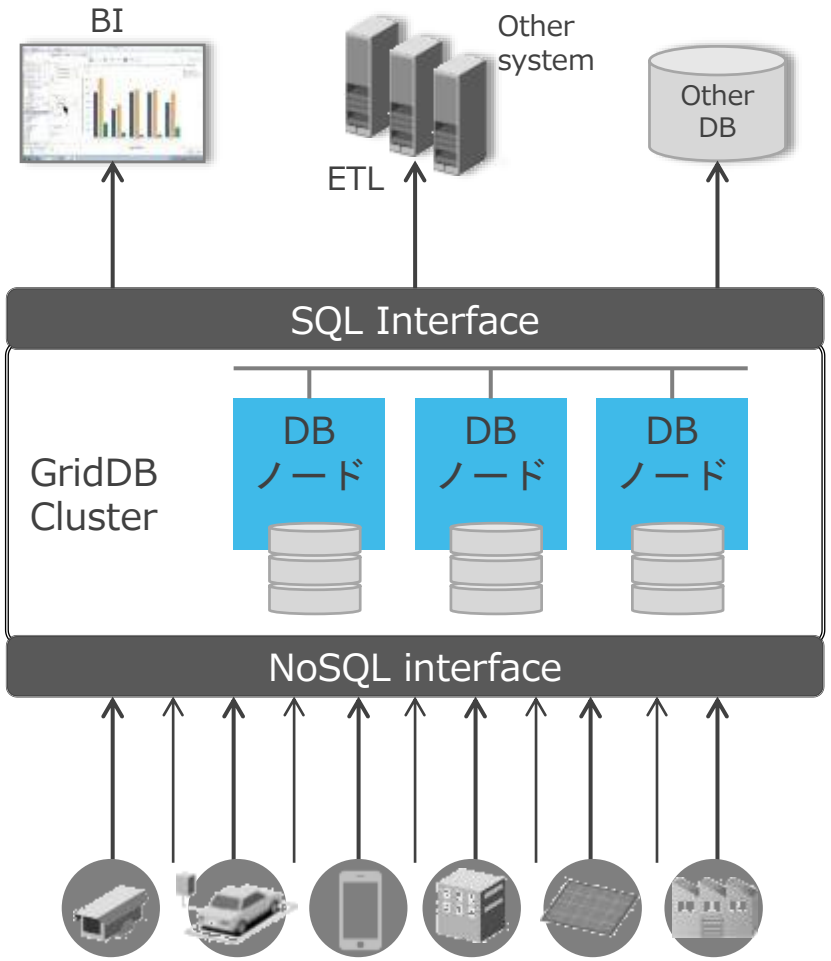
→通信コストなどの間接コストが無い

- ノードの追加削除やレプリカの欠損を検知し、安定的なデータ再配置を行う

→無停止でのスケールアウトを実現

NoSQL・SQLデュアルインターフェイス

違いはクエリ仕様

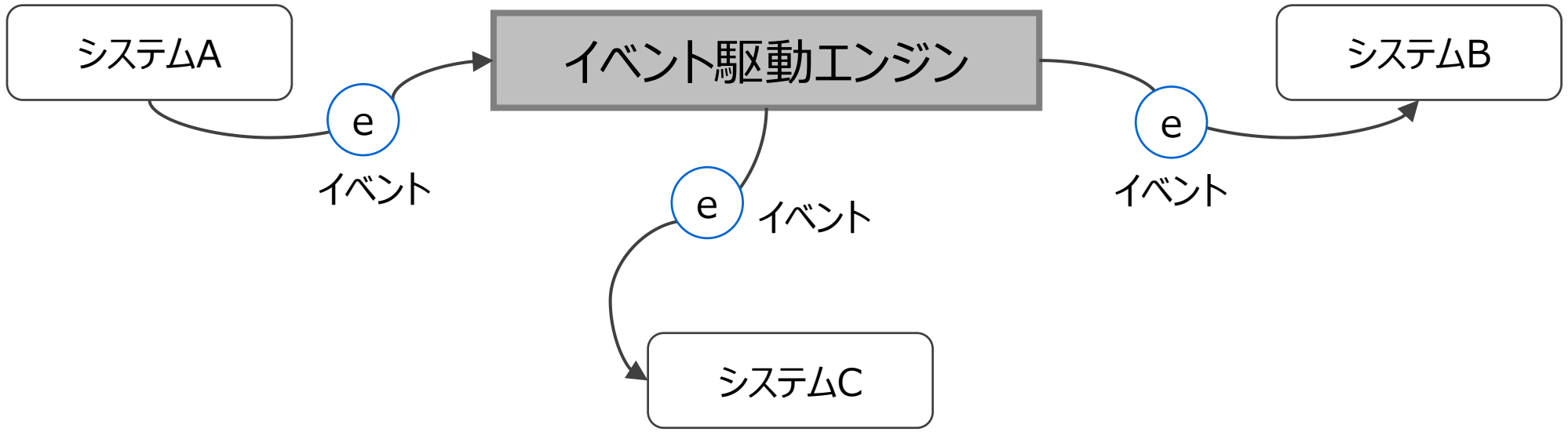



```
Connection connection =
  DriverManager.getConnection(uri, user, password);
final Statement statement =
  connection.createStatement();
ResultSet rs = statement.executeQuery(
         "SELECT * FROM Edge_XX3" +
         " WHERE sid ='XX1089000.001'" +
         " AND timestamp >= TIMESTAMP('2017-01-01T00:00:00Z')" +
         " AND timestamp < TIMESTAMP('2017-01-08T00:00:00Z'));
while (rs.next()) {
   ...
```

```
GridStore store = GridStoreFactory.getInstance().getGridStore(props);
Container<Point> container =
  store.getContainer("Edge_XX3", Point.class);
   Query<Point> query = container.query(
         "SELECT * " +
         " WHERE sid ='XX1089000.001'" +
         " AND timestamp >= TIMESTAMP('2017-01-01T00:00:00Z')" +
         " AND timestamp < TIMESTAMP('2017-01-08T00:00:00Z')"
         );
RowSet<Point> rs = query.fetch();
while (rs.hasNext()) {
   ...
```

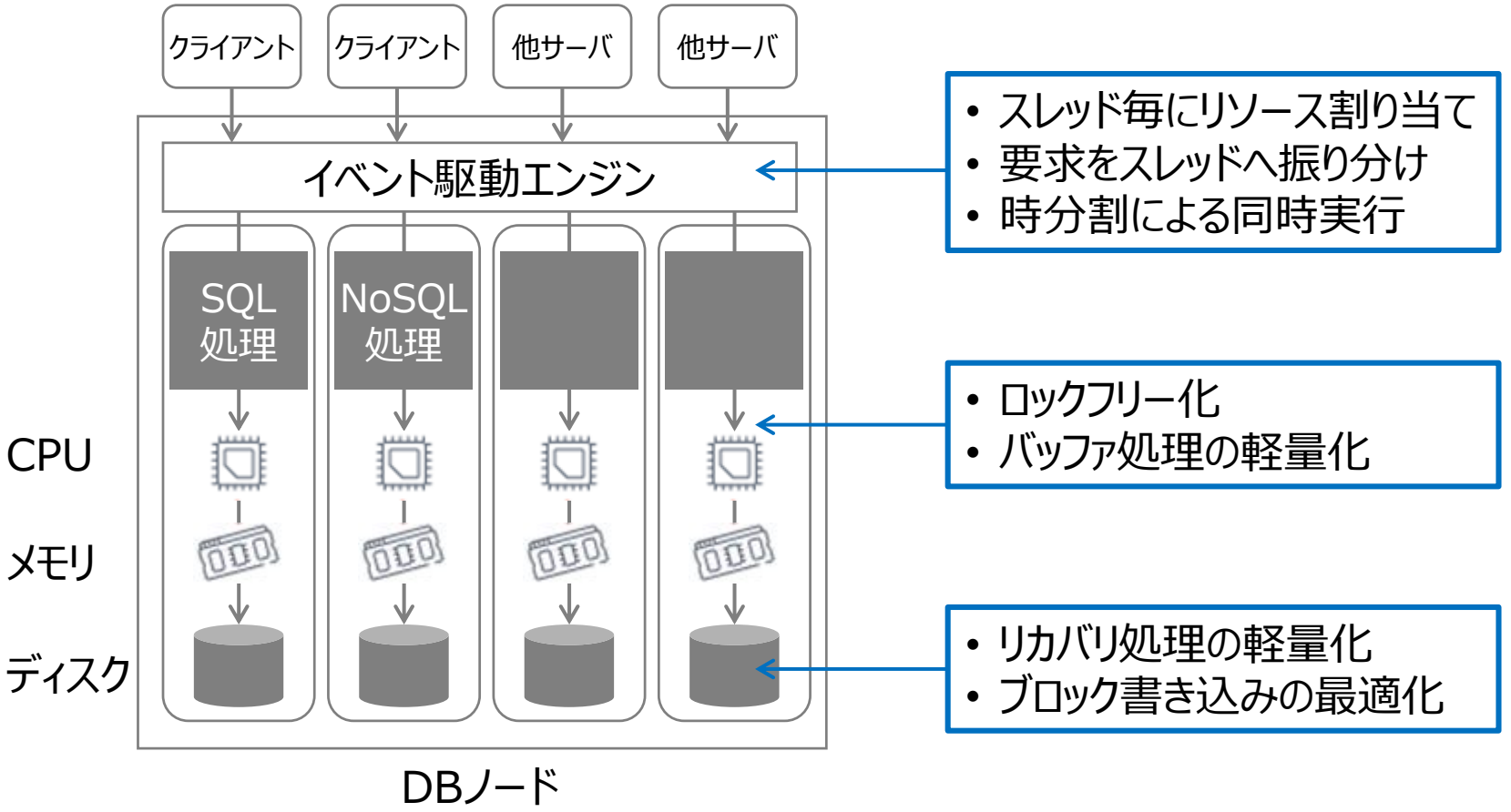
イベントドリブン

イベントに基づいて処理が動き始める方式

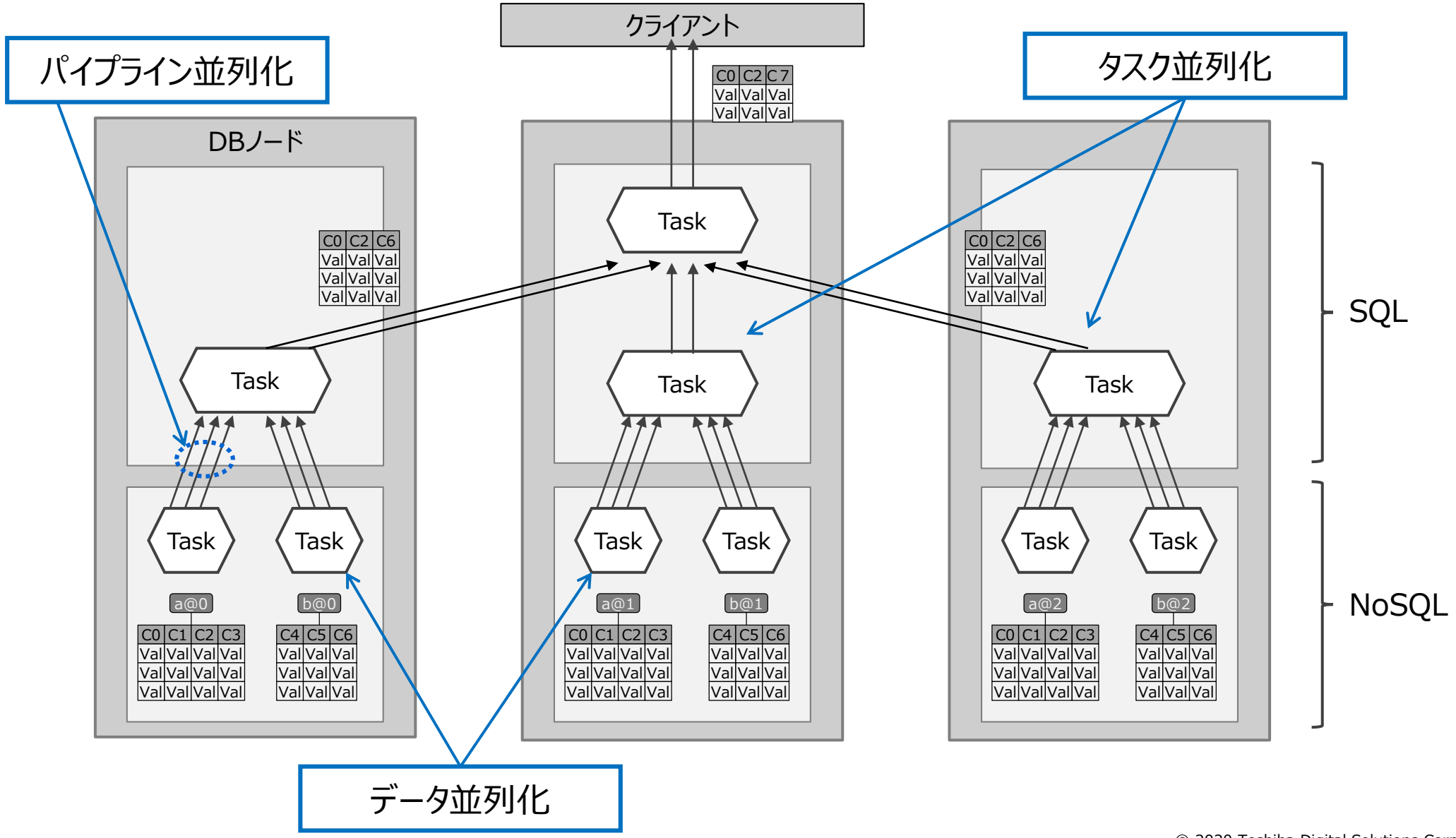


イベント駆動によるNoSQL･SQL処理

- CPUのマルチコア，メニーコア化を前提
- 非同期的なデータ処理を絶え間なく実行するイベント駆動方式



高速データ処理技術

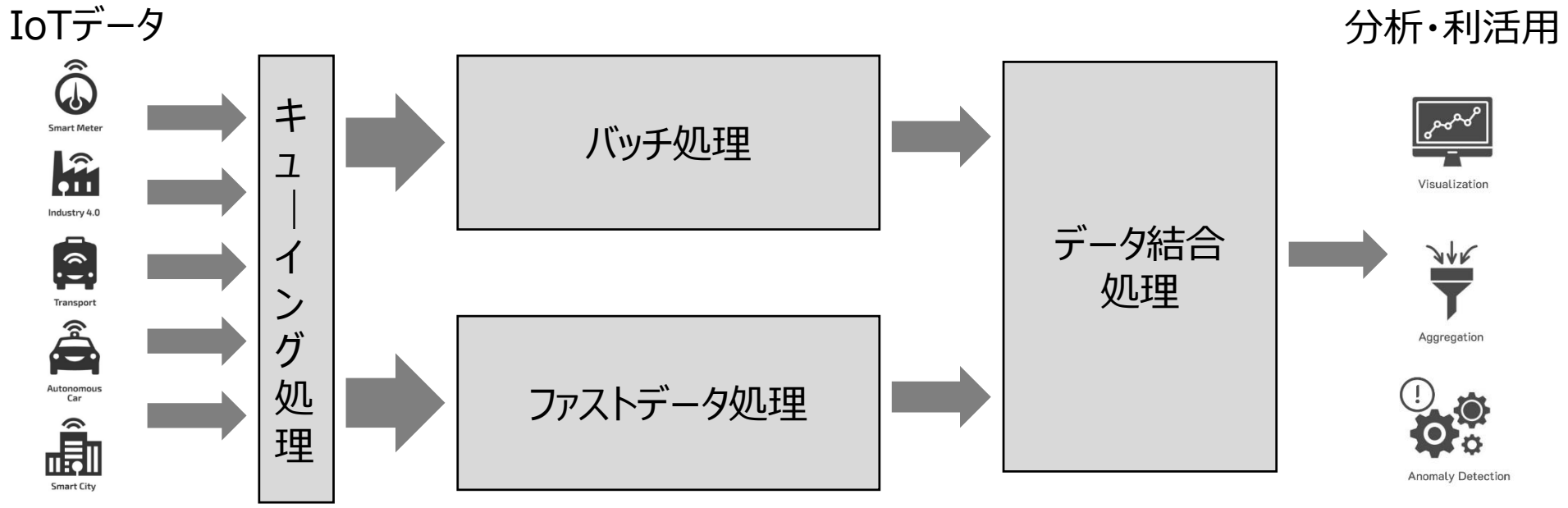


03

ラムダアーキテクチャからの脱却

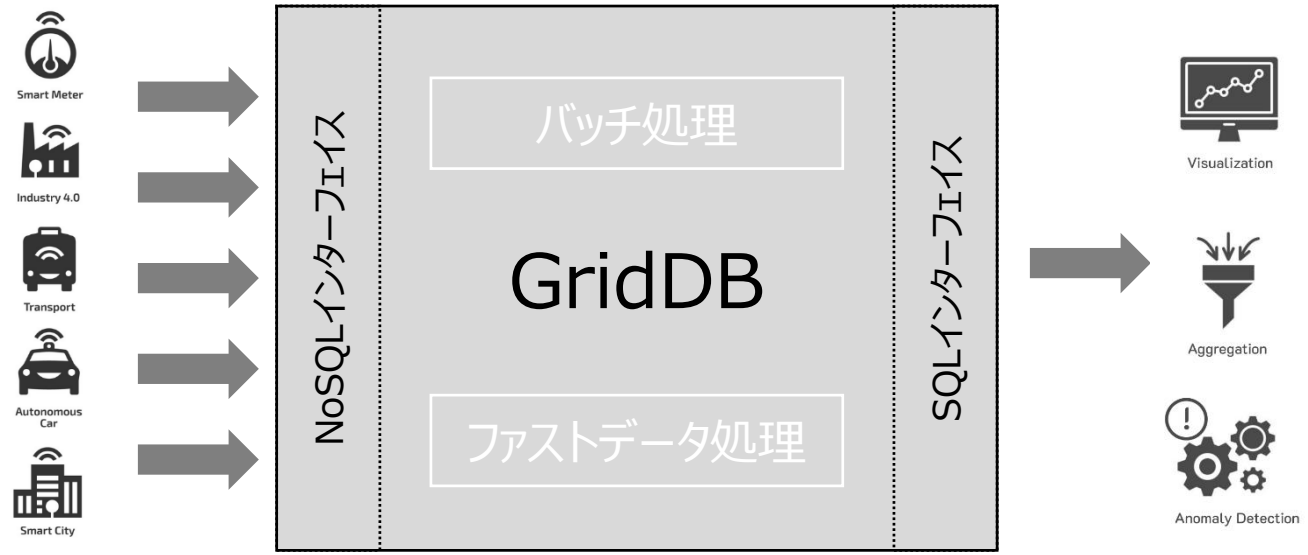
ファストデータとビッグデータの処理統合

バッチ的な分析処理＋リアルタイム性の高い処理（ファストデータ処理）
＝ラムダアーキテクチャー

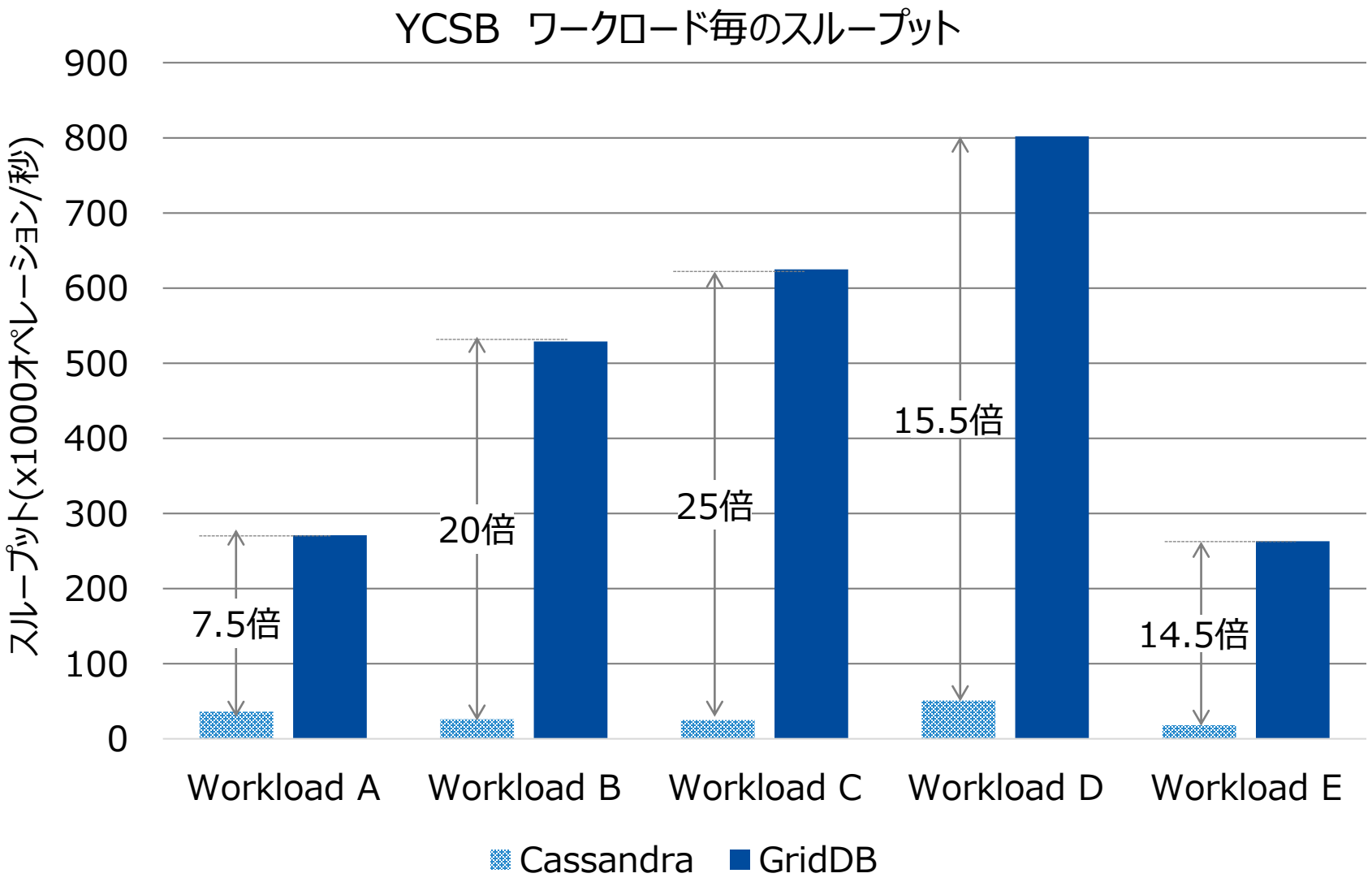


→構築・運用コストの上昇やシステムの複雑化

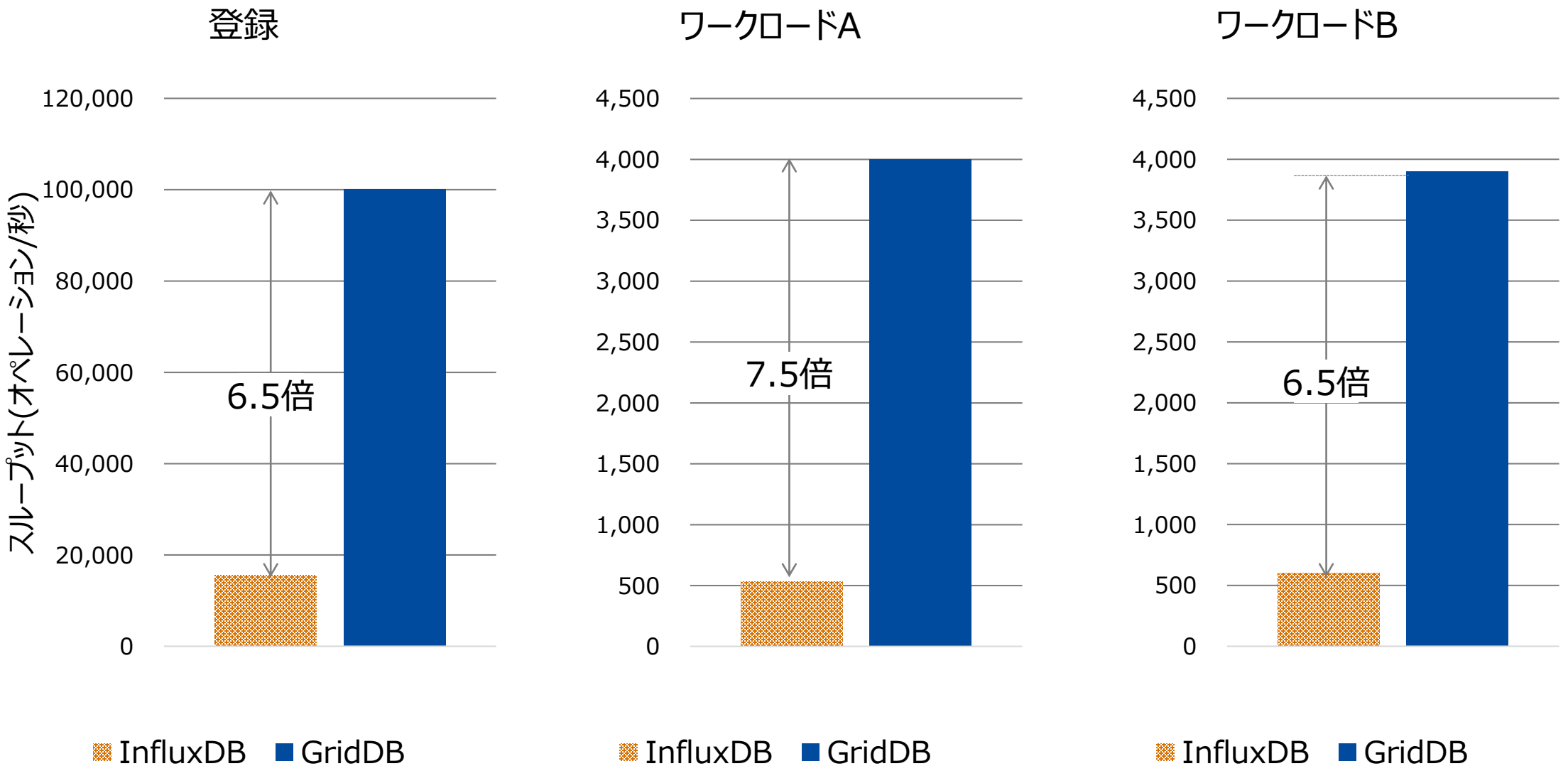
ファストデータとビッグデータの処理統合



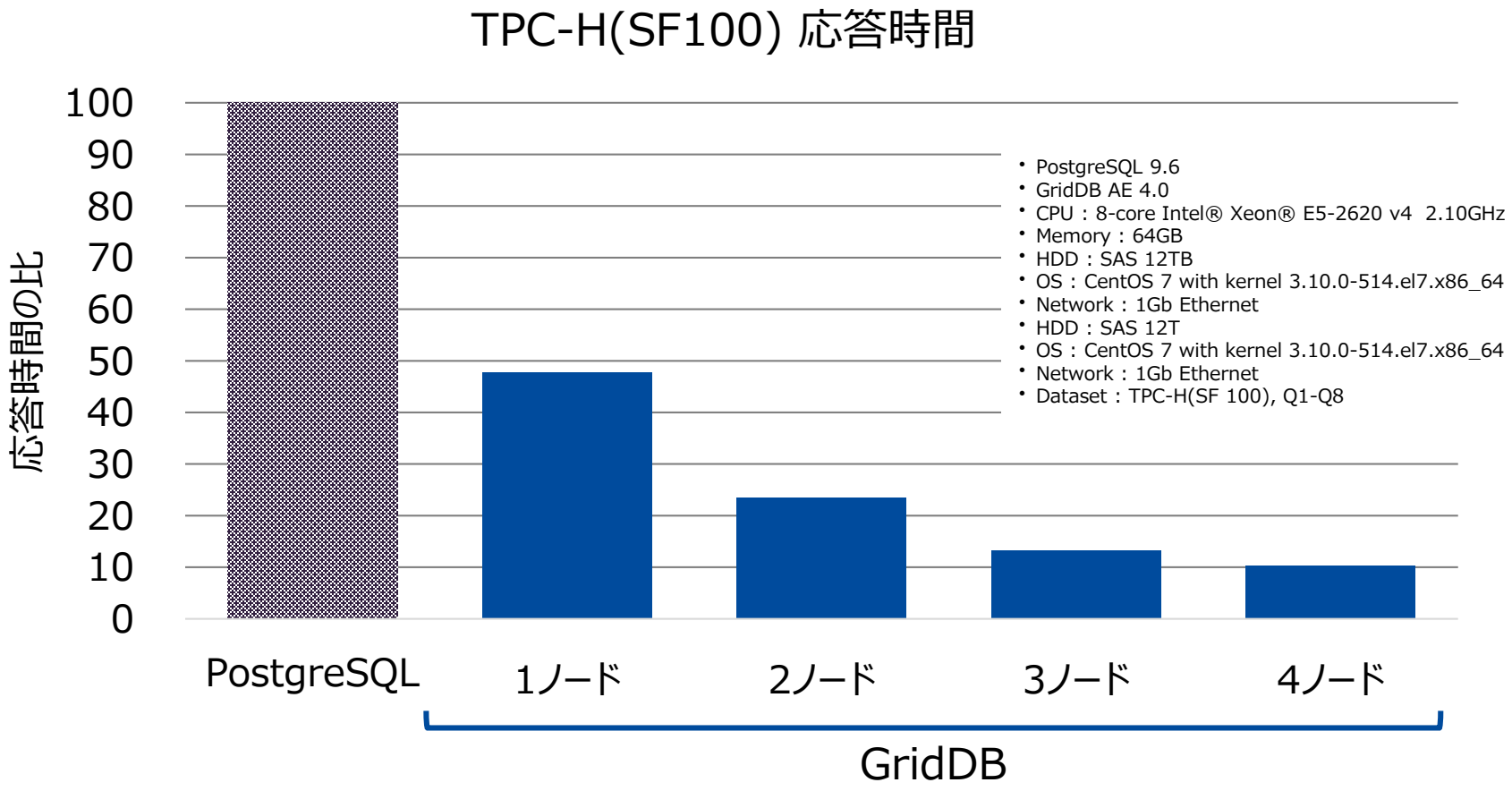
ファーストデータ処理の性能



ファーストデータ処理の性能

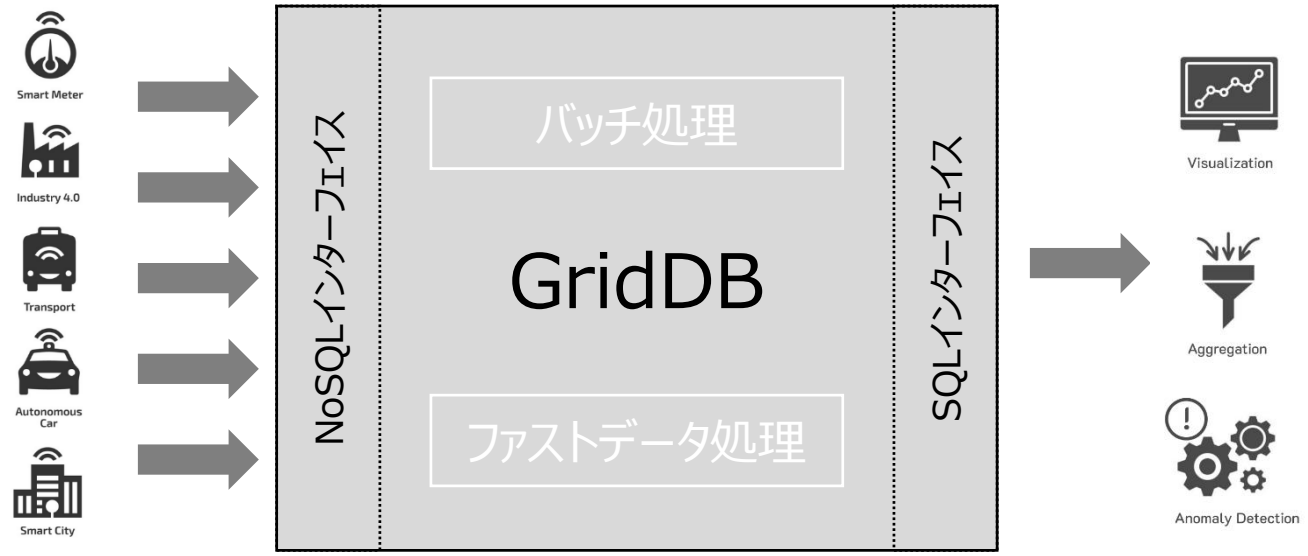


ビッグデータ処理の性能



ファストデータとビッグデータの処理統合

- リアルタイム性の高い登録や参照にはNoSQL
- 大規模なデータ集約・加工にはSQL

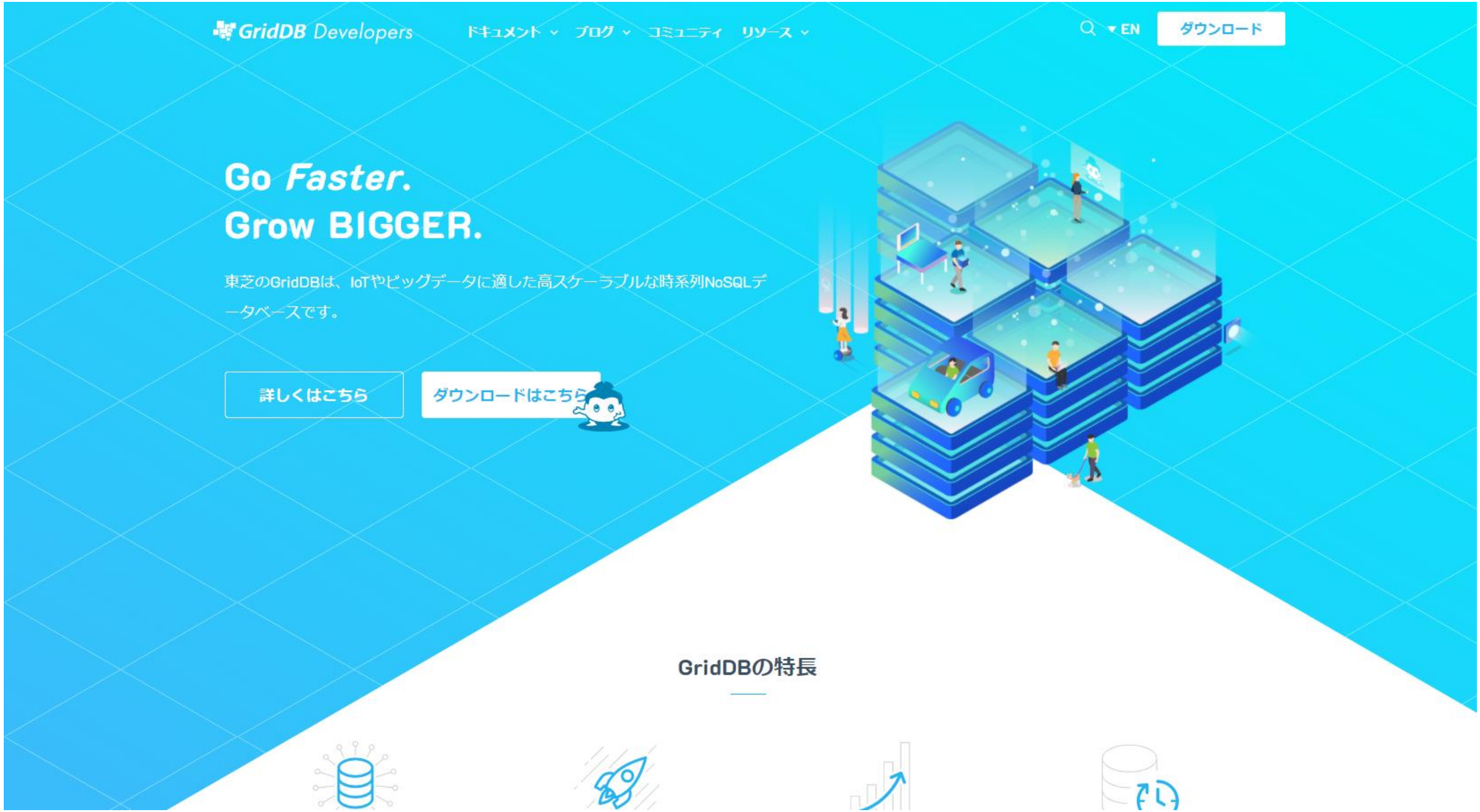


という使い方により，ファストデータ処理とビッグデータ処理を一つのDBで扱える。

04

OSSについて

https://github.com/griddb



有力OSSとのコネクティビティ

Data Processing

Visualization

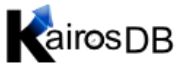
Client Libraries

GridDB

Analytics

Data Collection

Data Storage



Others

2020年6月、SQLインターフェースのソースコード無償公開

従来のNoSQL＋

2020年6月17日【テクノロジー】

東芝DSL、IoT向けDBのSQLインターフェースを無償公開

NEXT MOBILITY編集部

Tweet

https://github.com/griddb

TPC-H Q20

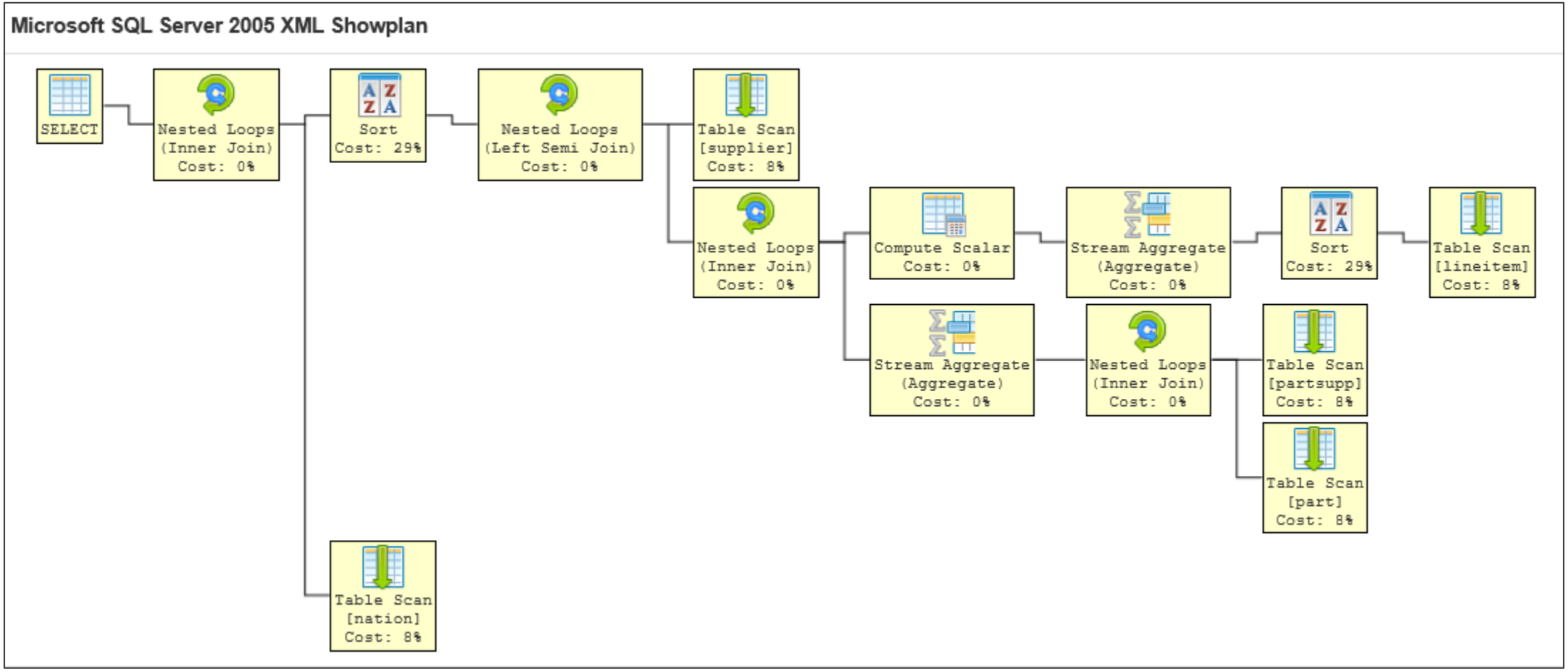
```
SELECT s_name,
       s_address
FROM   supplier,
       nation
WHERE  s_suppkey IN (SELECT ps_suppkey
                     FROM   partsupp
                     WHERE  ps_partkey IN (SELECT p_partkey
                                           FROM   part
                                           WHERE  p_name LIKE 'forest%')
                        AND ps_availqty > (SELECT 0.5 * Sum(l_quantity)
                                           FROM   lineitem
                                           WHERE  l_partkey = ps_partkey
                                                  AND l_suppkey = ps_suppkey
                                                  AND l_shipdate >=
                                                      '1994-01-01'
                                                  AND l_shipdate <
                                                      '1995-01-01'))
       AND s_nationkey = n_nationkey
       AND n_name = 'CANADA'
ORDER  BY s_name;
```

TPC-H Q20プラン

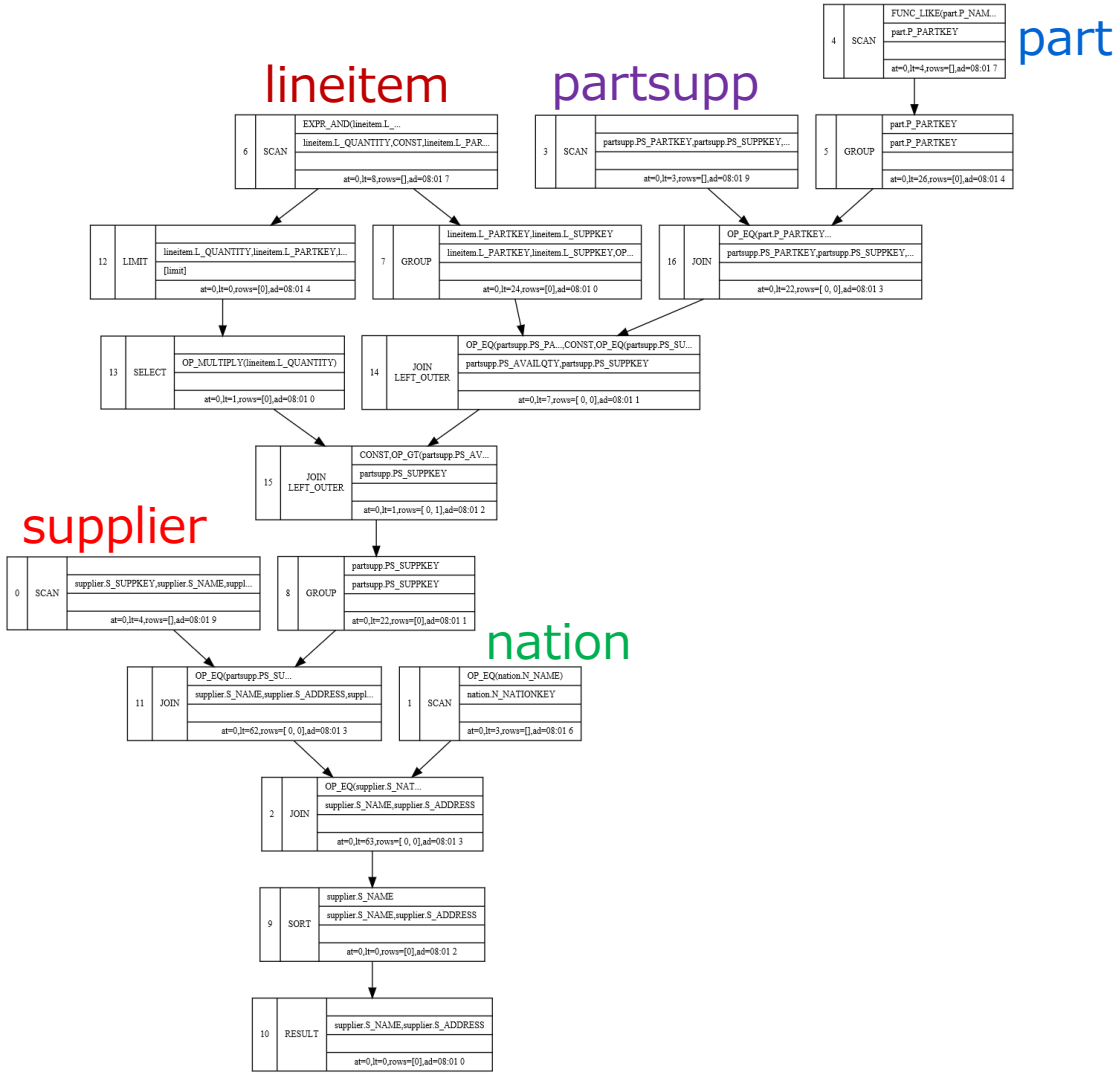
	QUERY PLAN
1	Sort (cost=15218.68..15218.69 rows=5 width=64) (actual time=0.073..0.073 rows=0 loops=1)
2	Sort Key: supplier.s_name
3	Sort Method: quicksort Memory: 25kB
4	-> Nested Loop Semi Join (cost=20.18..15218.62 rows=5 width=64) (actual time=0.016..0.016 rows=0 loops=1)
5	Join Filter: (supplier.s_suppkey = partsupp.ps_suppkey)
6	-> Hash Join (cost=20.18..36.73 rows=9 width=68) (actual time=0.015..0.015 rows=0 loops=1)
7	Hash Cond: (supplier.s_nationkey = nation.n_nationkey)
8	-> Seq Scan on supplier (cost=0.00..14.70 rows=470 width=72) (actual time=0.013..0.013 rows=0 loops=1)
9	-> Hash (cost=20.13..20.13 rows=4 width=4) (never executed)
10	-> Seq Scan on nation (cost=0.00..20.13 rows=4 width=4) (never executed)
11	Filter: ((n_name)::text = 'CANADA'::text)
12	-> Materialize (cost=0.00..15161.91 rows=170 width=4) (never executed)
13	-> Nested Loop Semi Join (cost=0.00..15161.06 rows=170 width=4) (never executed)
14	Join Filter: (partsupp.ps_partkey = part.p_partkey)
15	-> Seq Scan on partsupp (cost=0.00..15136.60 rows=340 width=8) (never executed)
16	Filter: ((ps_availqty)::double precision > (SubPlan 1))
17	SubPlan 1
18	-> Aggregate (cost=14.80..14.82 rows=1 width=8) (never executed)
19	-> Seq Scan on lineitem (cost=0.00..14.80 rows=1 width=8) (never executed)
20	Filter: (((l_shipdate)::text >= '1994-01-01'::text) AND ((l_shipdate)::text < '1995-01-01'::text) AND (l_partkey = partsupp.ps_partkey) AND (l_suppke partsupp.ps_suppkey))
21	-> Materialize (cost=0.00..14.26 rows=2 width=4) (never executed)
22	-> Seq Scan on part (cost=0.00..14.25 rows=2 width=4) (never executed)
23	Filter: ((p_name)::text ~~ 'forest%'::text)
24	Planning time: 3.573 ms
25	Execution time: 0.238 ms

PostgreSQL

TPC-H Q20プラン

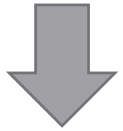


TPC-H Q20プラン

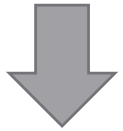


今後のOSSリリース計画

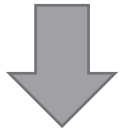
2016年2月　オープンソース化　NoSQLインターフェイスのみ



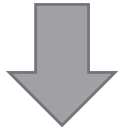
2020年6月　SQLインターフェイス公開



2021年1月　SQLオペレータ高速化や最適化など強化版



2021年3月　ストアー新、V5リリース



SQLコンパイラ刷新予定

TOSHIBA

ご清聴、ありがとうございました。

https://github.com/griddb