OLAP も PostgreSQL で！
Swarm64 の FPGA によるDB 高速化ソリューション「S64DA」のご紹介

株式会社マクニカ　アルティマカンパニー
石田 卓也

アジェンダ

- Swarm64 社のソリューション「S64DA」 の紹介
- S64DA のアーキテクチャ
- TPC-H による効果測定
- サマリ

MACNICA

Swarm64 社のソリューション「S64DA」の紹介

Swarm64 の会社紹介

- 創立 : 2013 年
- 本社 : ドイツ, ベルリン　（サポート : アメリカ - シアトル）

- データベースソフトウェア技術者 x FPGA のデータ処理技術者が在籍
- データ解析とトランザクションをリアルタイムに処理

HARDWARE AND DRIVER DESIGN EXPERTS

SKILLED AND SEASONED C++ DEVELOPERS

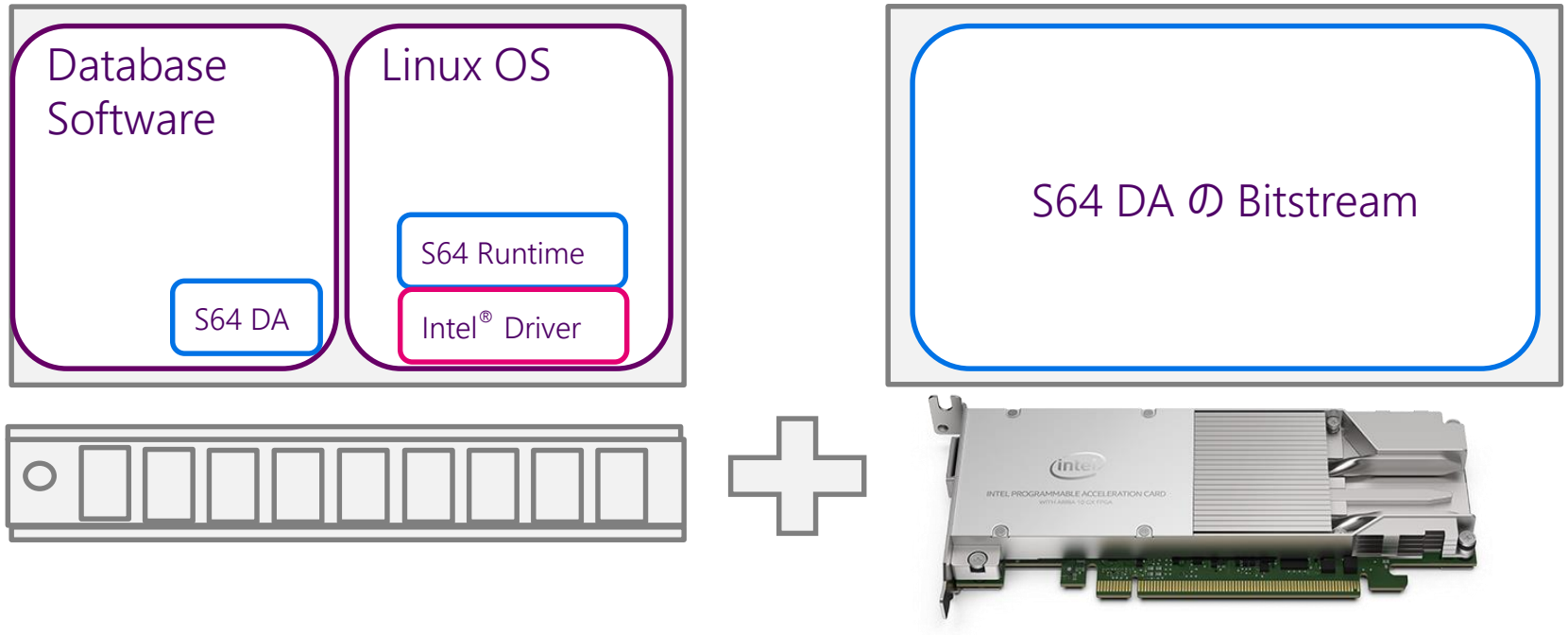
DATABASE PERFORMANCE ENTHUSIASTS

swarm64

Lets You innovate with all your data with Your existing IT and available skills so that You win customers faster

Swarm64 のソリューション S64DA の紹介

- S64DA : Swarm64 Data Accelerator
  - FPGA を使用したデータベース高速化ソリューション
    - ライセンスモデル : サブスクリプション
    - インテル® PAC インテル® Arria® 10 GX FPGA を使用
    - 回路,ソフトウェア/ランタイムは Swarm64 , インテル® から提供

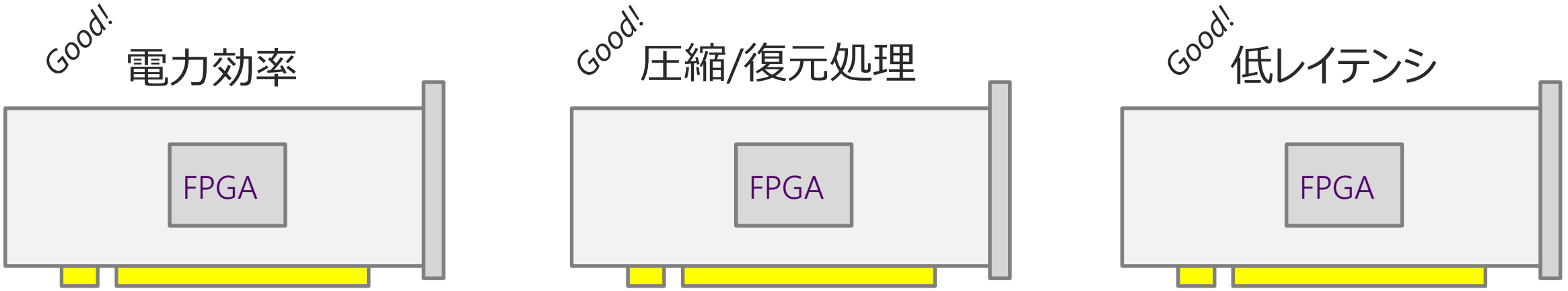


OEM パートナー	アクセラレーション・カード	サーバーモデル
Dell	インテル® PAC インテル® Arria® 10 GX FPGA 搭載版	Dell EMC PowerEdge* R640 Rack Server
Dell	インテル® PAC インテル® Arria® 10 GX FPGA 搭載版	Dell EMC PowerEdge* R740xd Rack Server
Dell	インテル® PAC インテル® Arria® 10 GX FPGA 搭載版	Dell EMC PowerEdge* R740 Rack Server
Dell	インテル® PAC インテル® Arria® 10 GX FPGA 搭載版	Dell EMC PowerEdge* R840 Rack Server
Dell	インテル® PAC インテル® Arria® 10 GX FPGA 搭載版	Dell EMC PowerEdge* R940xa Rack Server
Hewlett Packard Enterprise (HPE)	インテル® PAC インテル® Arria® 10 GX FPGA 搭載版	ProLiant DL360 Gen10
Hewlett Packard Enterprise (HPE)	インテル® PAC インテル® Arria® 10 GX FPGA 搭載版	ProLiant DL380 Gen10
Hewlett Packard Enterprise (HPE)	インテル® PAC インテル® Stratix® 10 SX FPGA 搭載版	2019年1Hリリース予定
富士通	インテル® PAC インテル® Arria® 10 GX FPGA 搭載版	FUJITSU Server PRIMERGY RX2540 M4 (型番:PYR2544RUN)

Swarm64 のソリューション S64DA の紹介

- S64DA : Swarm64 Data Accelerator
  - FPGA を使用したデータベース高速化ソリューション

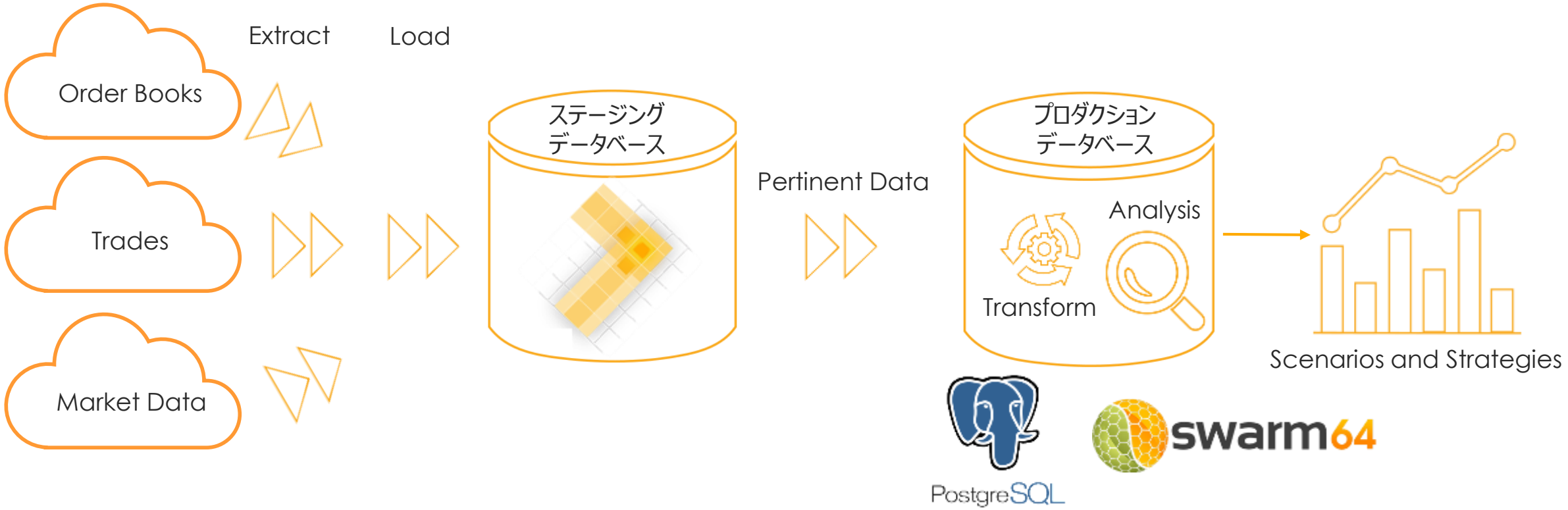
サーバー + データベースに最適！



Swarm64 のソリューション S64DA の紹介

- ソリューションのメリット：データの高速処理
  - PostgreSQL ,MariaDB , MySQL でほぼリアルタイムな解析が可能に
  - 複雑なビジネス分析を標準 PostgreSQL に比べ 2~10倍高速化
  - INSERT 動作は最大 10 倍高速に処理
  - PostgreSQL と完全互換
  - 既存のテーブルを残したまま、高速化したいところだけ変更
  - （アクセラレータを意識することなく使用可能）

PostgreSQL ベースで高速な ELT 処理と分析処理を実現

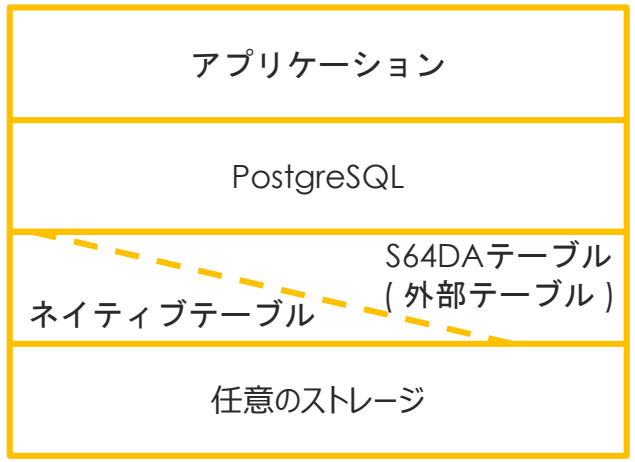


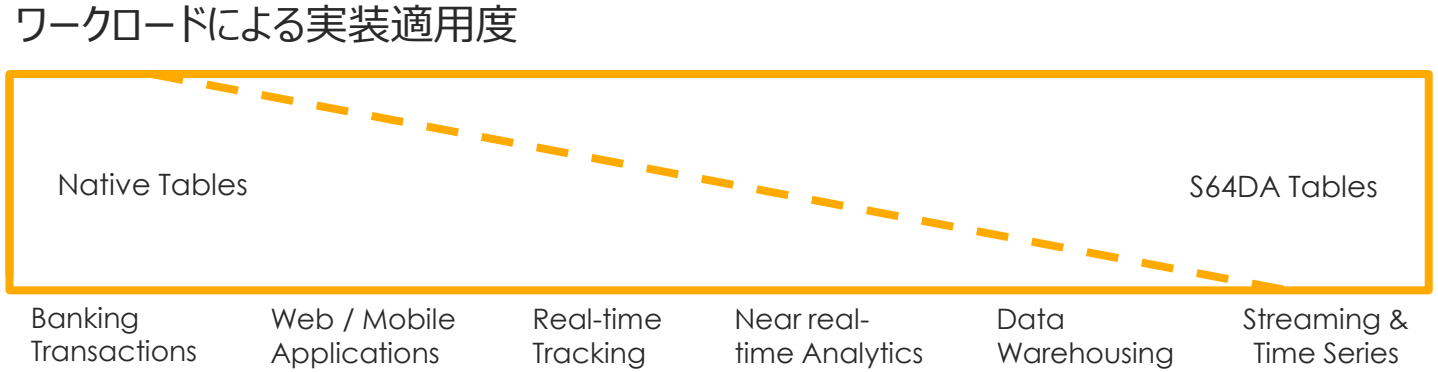
- 入力データ : 注文情報 (タイムスタンプ/売買情報/量)、トレード情報 (タイムスタンプ/取引情報)、マーケット情報 (タイムスタンプ/感情分析/ニュース)、etc
- 変換 (Transform) : 注文や取引、マーケット情報を適したスキーマに変換
- 分析処理 (Analysis) : シナリオ作成と洞察

MACNICA

Swarm64 社 S64DA 特長

- 柔軟なシステムを構成可能
  - 業務処理はネイティブテーブルで実装し CPU で処理、
    分析処理は外部テーブルで S64DA テーブルを実装し FPGA で処理など
- FPGA を活用して データを大幅に圧縮
  - 圧縮したデータを FPGA でリアルタイムに伸張するため DB サイズを削減
- メインメモリ内の処理及びキャッシュの要件を大幅に削減
  - 価格が低く、消費電力が少ないサーバー上でも同じレベルの性能を実現



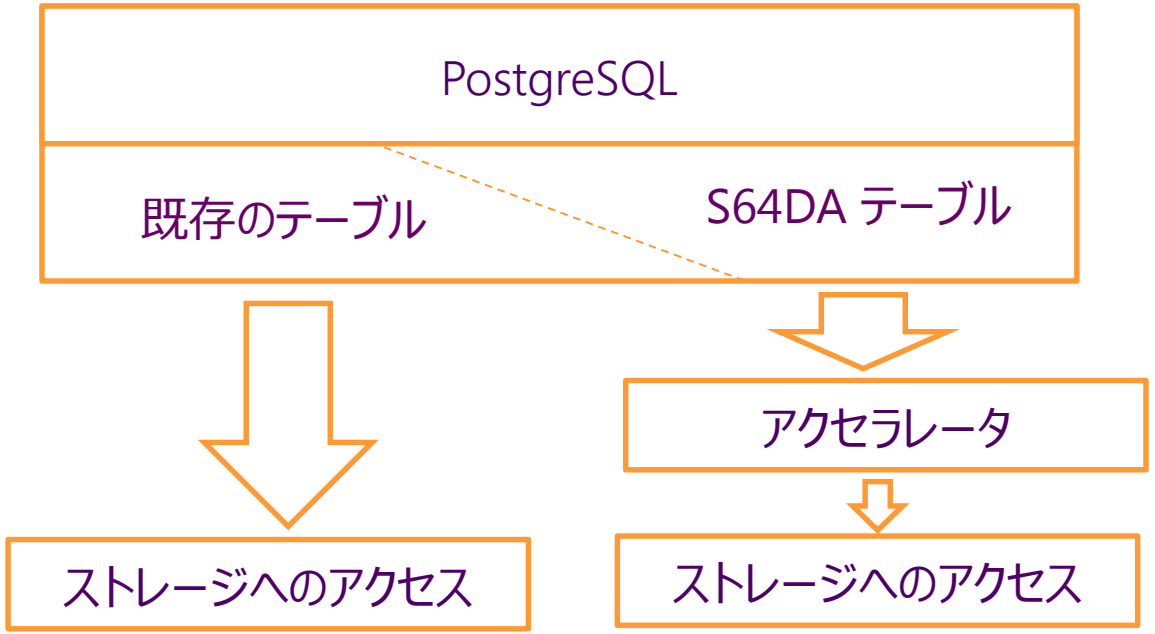


MACNICA

S64DA のアーキテクチャ

S64DA のアーキテクチャ

- Swarm64 DA は PostgreSQL の Extension として使用
  - 外部テーブルを作成しアクセス時に自動でアクセラレータが呼ばれ処理
    - FDW（外部データラッパ） としてプラグイン
    - ストレージのアクセスを最適化
- データ伸長による I/O アクセスの効率化
  - 既存と比べデータ量の削減
  - 大きなブロックサイズでアクセス
- ‘Optimized Column’ で高速化
  - 参照頻度が高いカラムを最適化
    - 発行したクエリに関連するデータのみアクセス
    - I/O スループットの効率化



S64DA のアーキテクチャ

- 実際に Optimized Column として作成するには？

→ CREATE FOREGIN TABLE 内で Optimized Column としたい
　カラムを選択するだけ！

```
CREATE FOREIGN TABLE my_table (col0 INT, col1 TIMESTAMP NOT NULL,
 col2 VARCHAR(30)) SERVER swarm64da_server OPTIONS (optimized_columns'col0, col1');
```

外部サーバとして
swarm64da_server を選択

Optimized Column として
動作させたいカラムを選択

FPGA で高速に処理させるには

- CREATE FOREIGN TABLE でテーブルの作成が必要
  - テーブルを作成する
    - リストア時にスキーマを変更
      CREATE TABLE → CREATE **FOREIGN** TABLE .... SERVER swarm64da_server
  - テーブルをコピーする
    - Table 'X' の載せ替えの場合
      CREATE **FOREIGN** TABLE .... SERVER swarm64da_server
      INSERT INTO X_Swarm (SELECT * FROM X)
    - 最後に 'X_Swarm' の名前を 'X' に変更
      （既存 Table 'X' はリネームまたは削除）

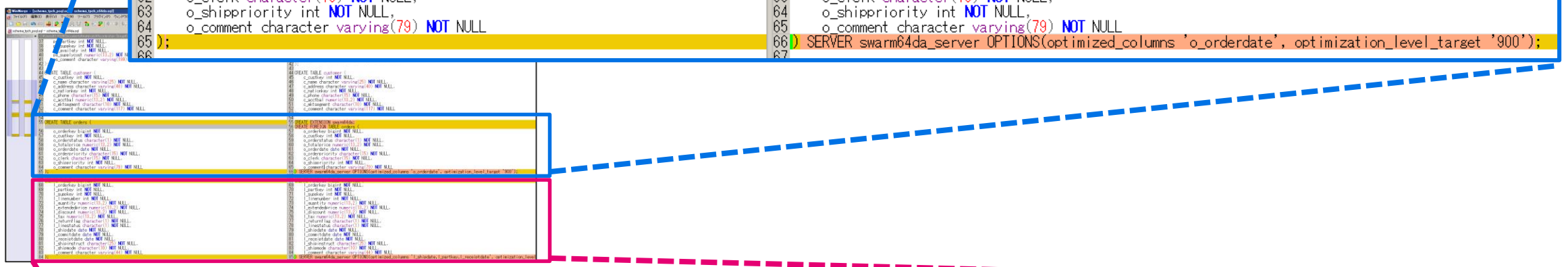
テーブル作成時のクエリ

Native Postgre用 テーブルのクリエイト

```
CREATE TABLE orders (
    o_orderkey bigint NOT NULL,
    o_custkey int NOT NULL,
    o_orderstatus character(1) NOT NULL,
    o_totalprice numeric(13,2) NOT NULL,
    o_orderdate date NOT NULL,
    o_orderpriority character(15) NOT NULL,
    o_clerk character(15) NOT NULL,
    o_shippriority int NOT NULL,
    o_comment character varying(79) NOT NULL
);
```

S64DA用 テーブルのクリエイト

```
CREATE EXTENSION swarm64da;
CREATE FOREIGN TABLE orders (
    o_orderkey bigint NOT NULL,
    o_custkey int NOT NULL,
    o_orderstatus character(1) NOT NULL,
    o_totalprice numeric(13,2) NOT NULL,
    o_orderdate date NOT NULL,
    o_orderpriority character(15) NOT NULL,
    o_clerk character(15) NOT NULL,
    o_shippriority int NOT NULL,
    o_comment character varying(79) NOT NULL
) SERVER swarm64da_server OPTIONS(optimized_columns 'o_orderdate', optimization_level_target '900');
```



```
CREATE TABLE lineitem (
    l_orderkey bigint NOT NULL,
    l_partkey int NOT NULL,
    l_suppkey int NOT NULL,
    l_linenumber int NOT NULL,
    l_quantity numeric(13,2) NOT NULL,
    l_extendedprice numeric(13,2) NOT NULL,
    l_discount numeric(13,2) NOT NULL,
    l_tax numeric(13,2) NOT NULL,
    l_returnflag character(1) NOT NULL,
    l_linestatus character(1) NOT NULL,
    l_shipdate date NOT NULL,
    l_commitdate date NOT NULL,
    l_receiptdate date NOT NULL,
    l_shipinstruct character(25) NOT NULL,
    l_shipmode character(10) NOT NULL,
    l_comment character varying(44) NOT NULL
);
```

```
CREATE FOREIGN TABLE lineitem (
    l_orderkey bigint NOT NULL,
    l_partkey int NOT NULL,
    l_suppkey int NOT NULL,
    l_linenumber int NOT NULL,
    l_quantity numeric(13,2) NOT NULL,
    l_extendedprice numeric(13,2) NOT NULL,
    l_discount numeric(13,2) NOT NULL,
    l_tax numeric(13,2) NOT NULL,
    l_returnflag character(1) NOT NULL,
    l_linestatus character(1) NOT NULL,
    l_shipdate date NOT NULL,
    l_commitdate date NOT NULL,
    l_receiptdate date NOT NULL,
    l_shipinstruct character(25) NOT NULL,
    l_shipmode character(10) NOT NULL,
    l_comment character varying(44) NOT NULL
) SERVER swarm64da_server OPTIONS(optimized_columns 'l_shipdate,l_partkey,l_receiptdate', optimization_level_target '900');
```

テーブル作成時のクエリ

- Native Postgre用 テーブルのクリエイト

```
....

CREATE TABLE orders (
    o_orderkey bigint NOT NULL,
    o_custkey int NOT NULL,
    o_orderstatus character(1) NOT NULL,
    o_totalprice numeric(13,2) NOT NULL,
    o_orderdate date NOT NULL,
    o_orderpriority character(15) NOT NULL,
    o_clerk character(15) NOT NULL,
    o_shippriority int NOT NULL,
    o_comment character varying(79) NOT NULL
);


....
```

- S64DA用 テーブルのクリエイト

```
....

CREATE FOREIGN TABLE orders (
    o_orderkey bigint NOT NULL,
    o_custkey int NOT NULL,
    o_orderstatus character(1) NOT NULL,
    o_totalprice numeric(13,2) NOT NULL,
    o_orderdate date NOT NULL,
    o_orderpriority character(15) NOT NULL,
    o_clerk character(15) NOT NULL,
    o_shippriority int NOT NULL,
    o_comment character varying(79) NOT NULL
) SERVER swarm64da_server OPTIONS(
optimized_columns 'o_orderdate', optimization_level_target '900');

.....
```

MACNICA

TPC-H による効果測定

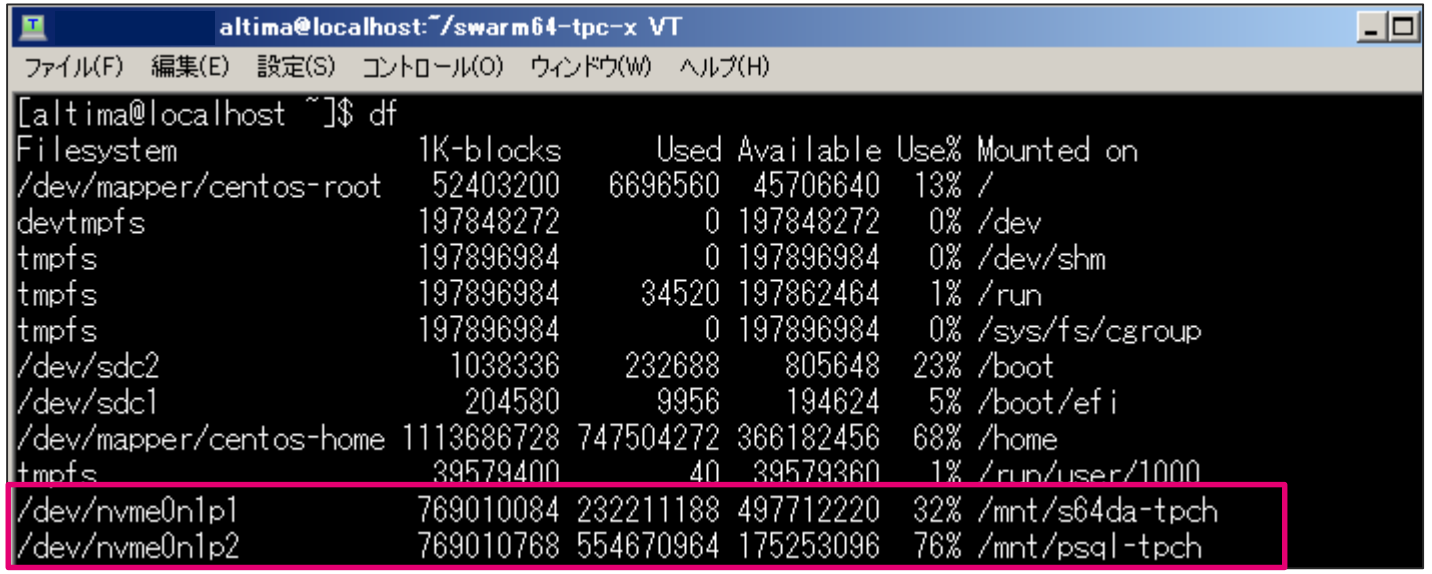
使用した環境について

パーツ	型番
CPU	Xeon Silver 4114　2.2 GHz (Cache13.75 MB)　x 2
RAM	DDR4-2666　384GB
OS	Cent OS 7.4
Storage(OS)	SAS HDD　1.2TB
Storage(DATA)	NVMe　1.6TB
	- Native PSQL11　（770GB分）
	- S64DA with PSQL11（770GB分）
Accelerator Card	インテル® PAC インテル® Arria® 10 GX FPGA
S64DA version	v1.6.0
DCP version	v1.2
Test Data	TPC-H @SF300 Power Test

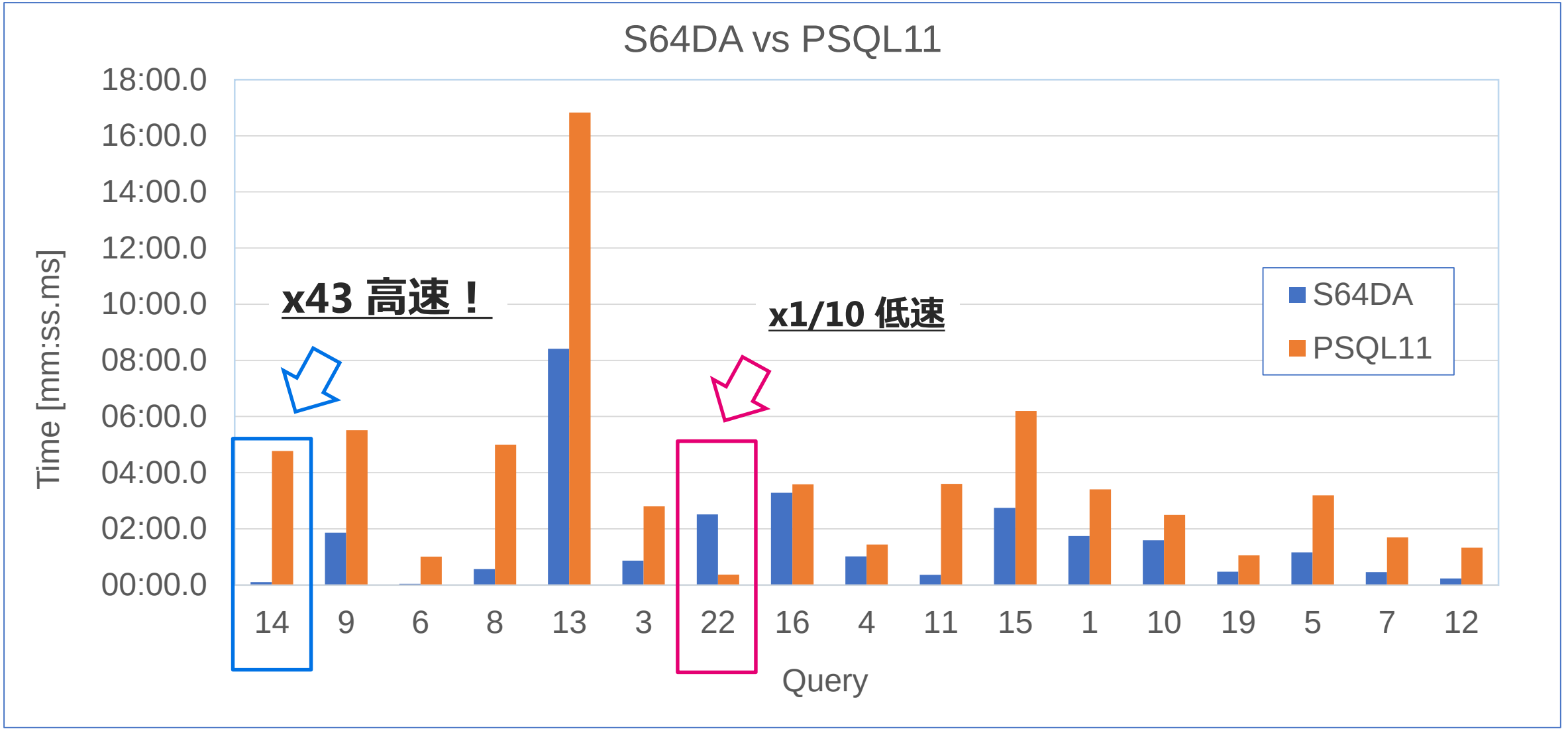
使用した環境について

- SF300 でのストレージの使用率＝ S64DA の場合 **50% 以上削減**
  - S64DA : 232 GB (770 GB の 32%)
    - FPGA を通してデータが圧縮
  - Native PostgreSQL ：554 GB (770 GB の 76%)
    - インデックス使用分のデータがプラス

→ストレージコストの削減



TPC-H@SF300 の結果



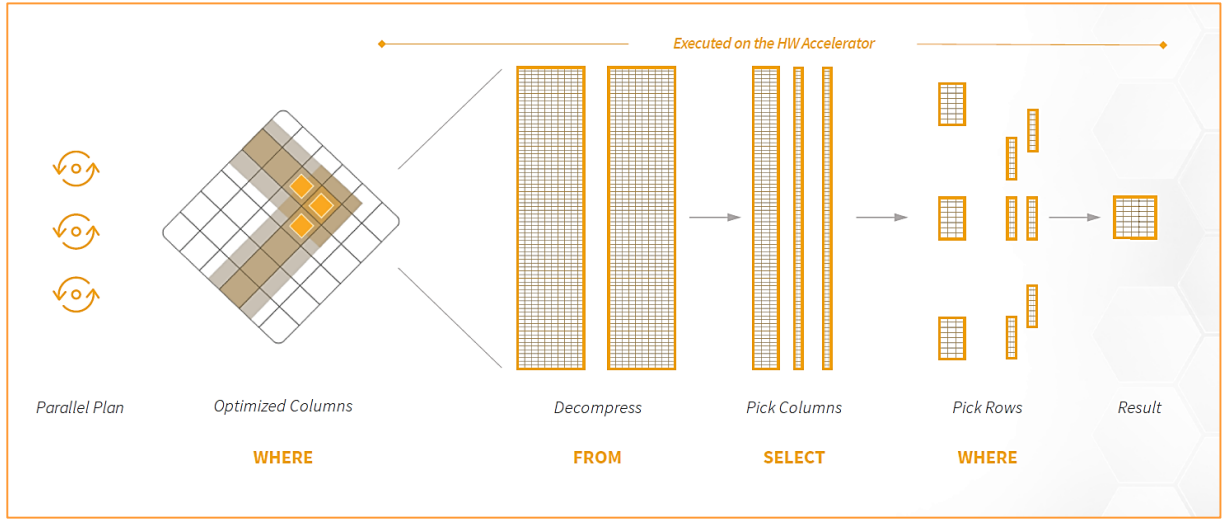
※ 4/18 現時点での弊社での結果となります。

Query 14 が速い理由

```
select
    100.00 * sum(case
        when p_type like 'PROMO%'
        then l_extendedprice * (1 - l_discount)
        else 0
    end) / sum(l_extendedprice * (1 - l_discount))
as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= date '1996-06-01'
    and l_shipdate < date '1996-06-01' + interval
'1' month;
```

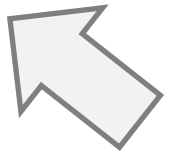
検索がほとんど FPGA で行われる！

- 検索のほとんどがFPGAで行われる。
  - where 句でのカラムの選択/フィルタリング
  - 必要なデータのみ FPGA より提供



Query 22 が遅い理由

```
select
        cntrycode,
    count(*) as numcust,
    sum(c_acctbal) as totacctbal
from
    (
        select
            substring(c_phone from 1 for 2) as cntrycode,
            c_acctbal
        from
            customer
        where
            substring(c_phone from 1 for 2) in
                ('32', '27', '31', '25', '26', '24', '19')
            and c_acctbal > (
                select
                    avg(c_acctbal)
                from
                    customer
                where
                    c_acctbal > 0.00
                    and substring(c_phone from 1 for 2) in
                        ('32', '27', '31', '25', '26', '24', '19')
            )
            and not exists (
                select
                    *
                from
                    orders
                where
                    o_custkey = c_custkey
            )
    ) as custsale
group by
    cntrycode
order by
    cntrycode;
```



この処理でデータが毎回

- FROMのネストされたサブクエリ
  - ネストしたループに必要なすべての中間テーブルが再作成され、常にデータがリロードされる状態に
- Native の場合は、インデックスを使って検索をし、結果を保存することが可能
- 現在メーカではこのクエリを高速化するための解決策を策定中
  - 計画中でのクエリを書き直しを実装予定

MACNICA

サマリ

終わりに

- データアナリティクスも PostgreSQL で高速化するために
  - 結果の参照を迅速に行いたい
  - 何百万行のデータを処理したい

  → S64DA は上記課題を解決！
- 既存の大量のデータをシステム刷新よりもコストを抑えて速く処理
- FPGAの電力効率により、TCE (総消費エネルギー)も削減

サマリ

- 低レイテンシ、高電力効率、ストリーミングの得意な FPGA の使用によりサーバー+データベースにおいて効率的な動作を実現
- データ圧縮によるストレージコストの削減
- 柔軟なシステムを構築できる実装形態
  - 同じシステムに業務処理を行う PostgreSQL も構成可能
- 複雑なビジネス分析を CPU のみと比較して2倍～10倍高速化
- BI ツールへローデータの取り込みを高速化し分析処理を迅速化

MACNICA

ありがとうございました。

Swarm64 のお問い合わせはこちらまで

株式会社マクニカ　アルティマカンパニー
営業窓口: 平部 真彬 (hirabe-m@macnica.co.jp)