GPU-Direct SQLでつくるIoT/M2Mログ分析システム

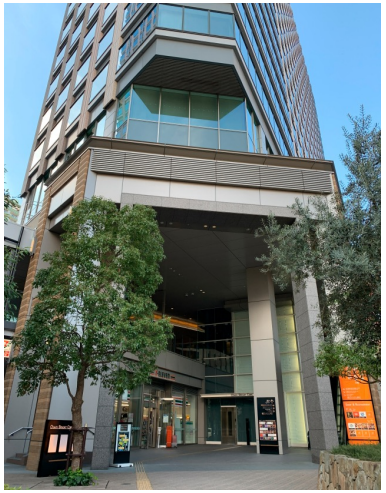
ヘテロＤＢ株式会社
チーフアーキテクト 兼 CEO
海外 浩平 <kaigai@heterodb.com>

自己紹介／会社紹介

ヘテロジニアスコンピューティング技術を**データベース**領域に適用し、
誰もが使いやすく、シンプルで高速な大量データ分析基盤を提供する。

会社概要

- 商号　ヘテロDB株式会社
- 創業　2017年7月4日
- 拠点　品川区北品川5-5-15
  大崎ブライトコア4F
- 事業内容　高速データ処理製品の開発・販売
  GPU＆DB領域の技術コンサルティング



代表者プロフィール

- 海外　浩平（KaiGai Kohei）
- OSS開発者コミュニティにおいて、PostgreSQLやLinux kernelの開発に15年以上従事。主にセキュリティ・FDW等の分野でPostgreSQLのメインライン機能の強化に貢献。
- IPA未踏ソフト事業において“天才プログラマー”認定 (2006)
- GPU Technology Conference Japan 2017でInception Awardを受賞

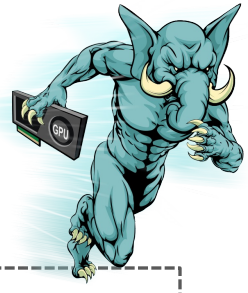
PG-Strom – GPUを用いてSQLを並列処理する

実行結果

SQL パーサ

SQL オプティマイザ

SQL エグゼキュータ



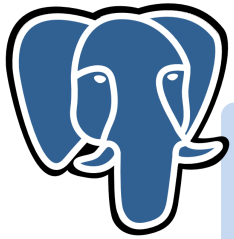
PG-Strom拡張モジュール

代替実行計画作成

GPUコード生成

データの読み出し

GPUカーネル実行



トランザクション

共有バッファ

各種関数・演算子

プロセス間通信

PostgreSQLの各種インフラ

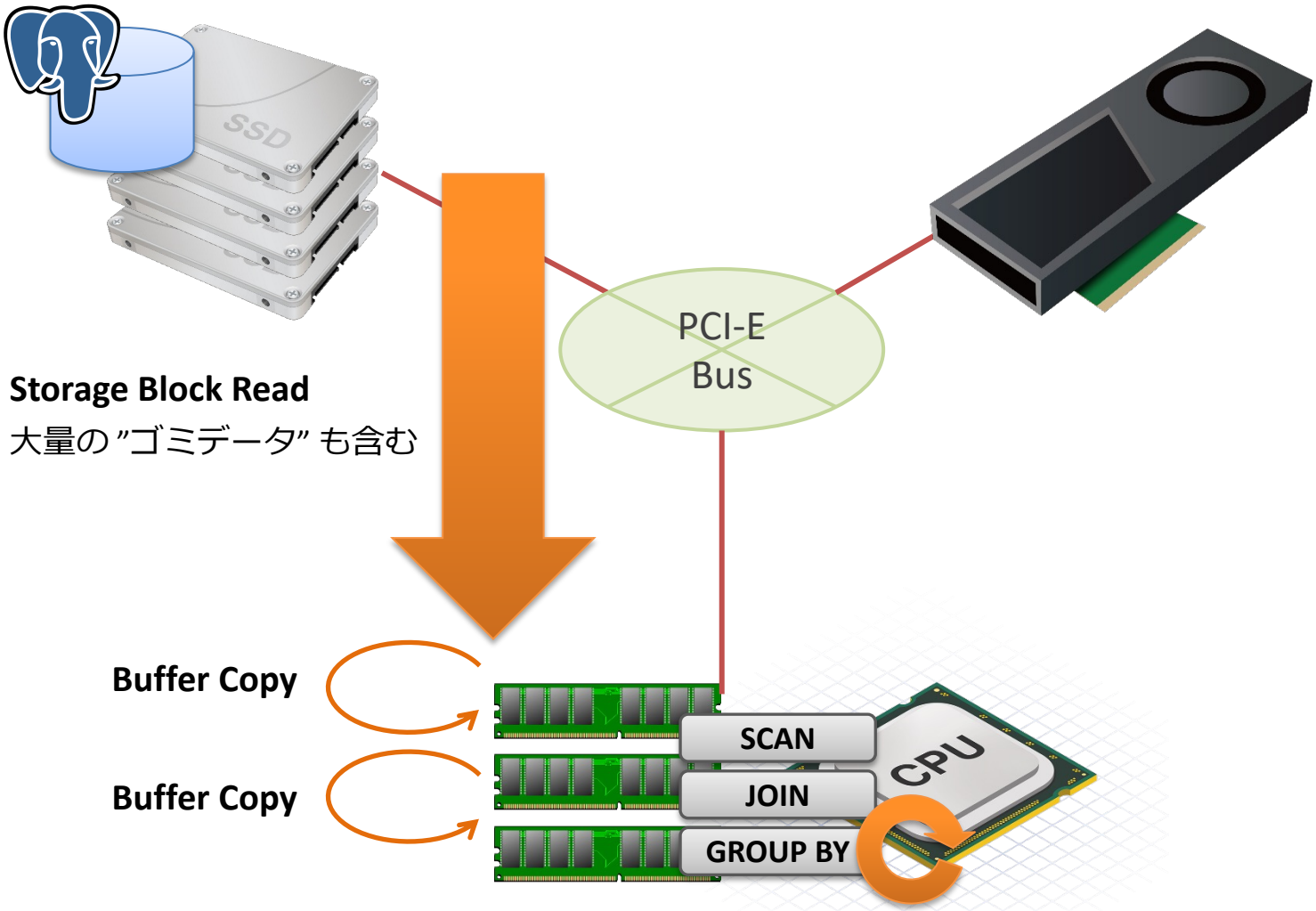
GPUデバイス管理

JITコンパイル

GPU-Direct SQL

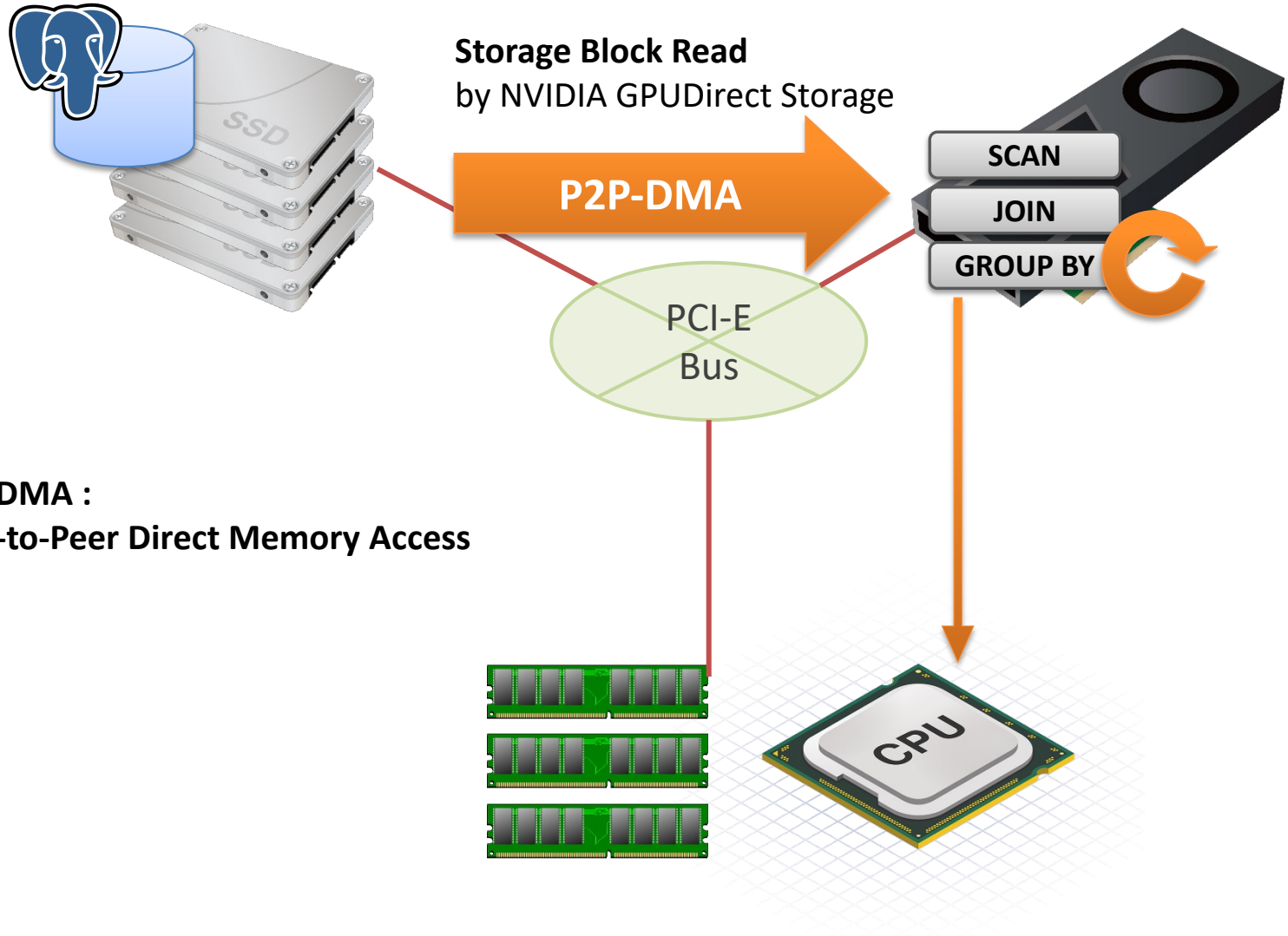
PG-Stromのインフラ機能

GPU-Direct SQL（1/5）



GPU-Direct SQL（2/5）

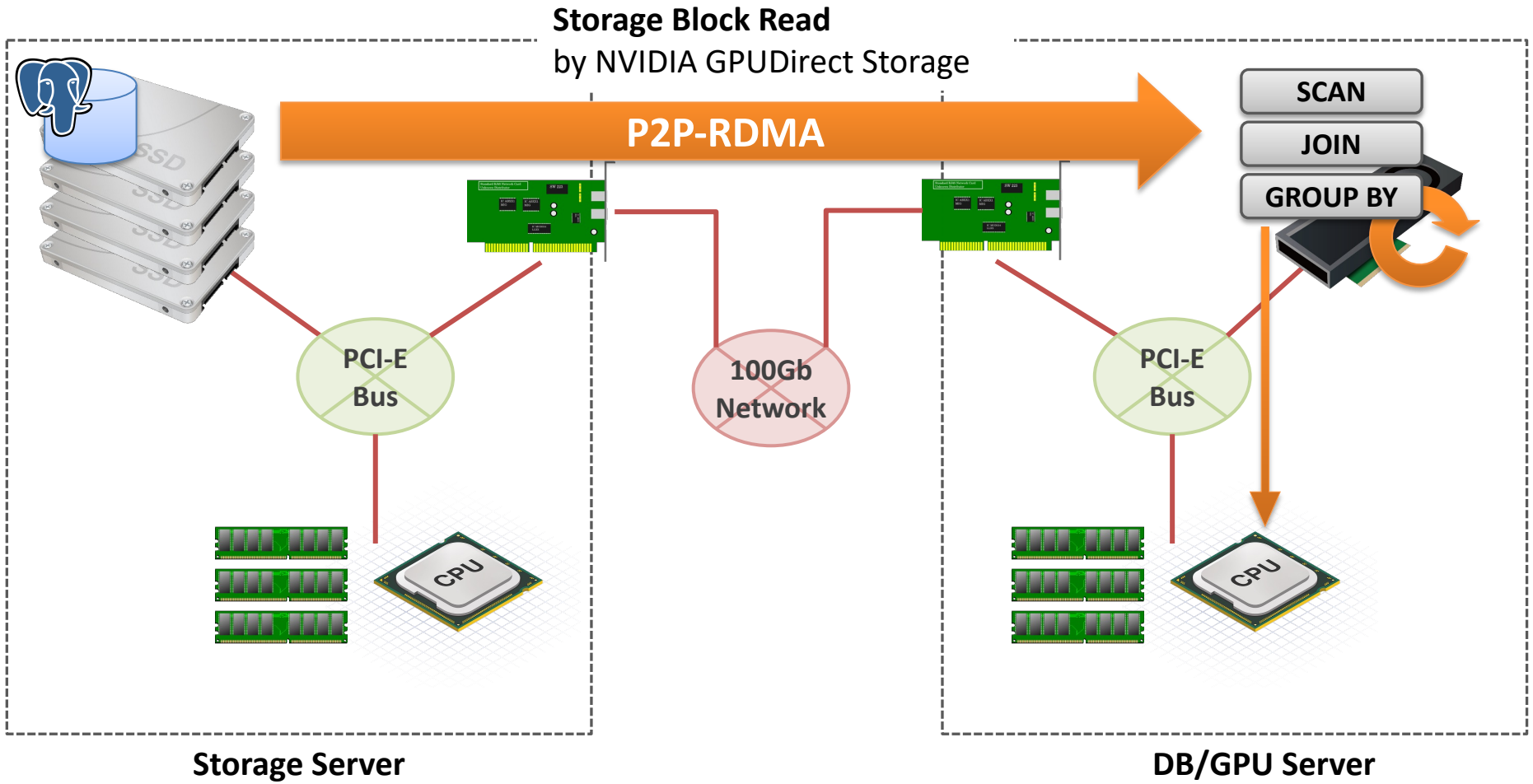
P2P-DMAを利用し、NVME-SSDとGPUを直結してデータ転送



P2P-DMA :
Peer-to-Peer Direct Memory Access

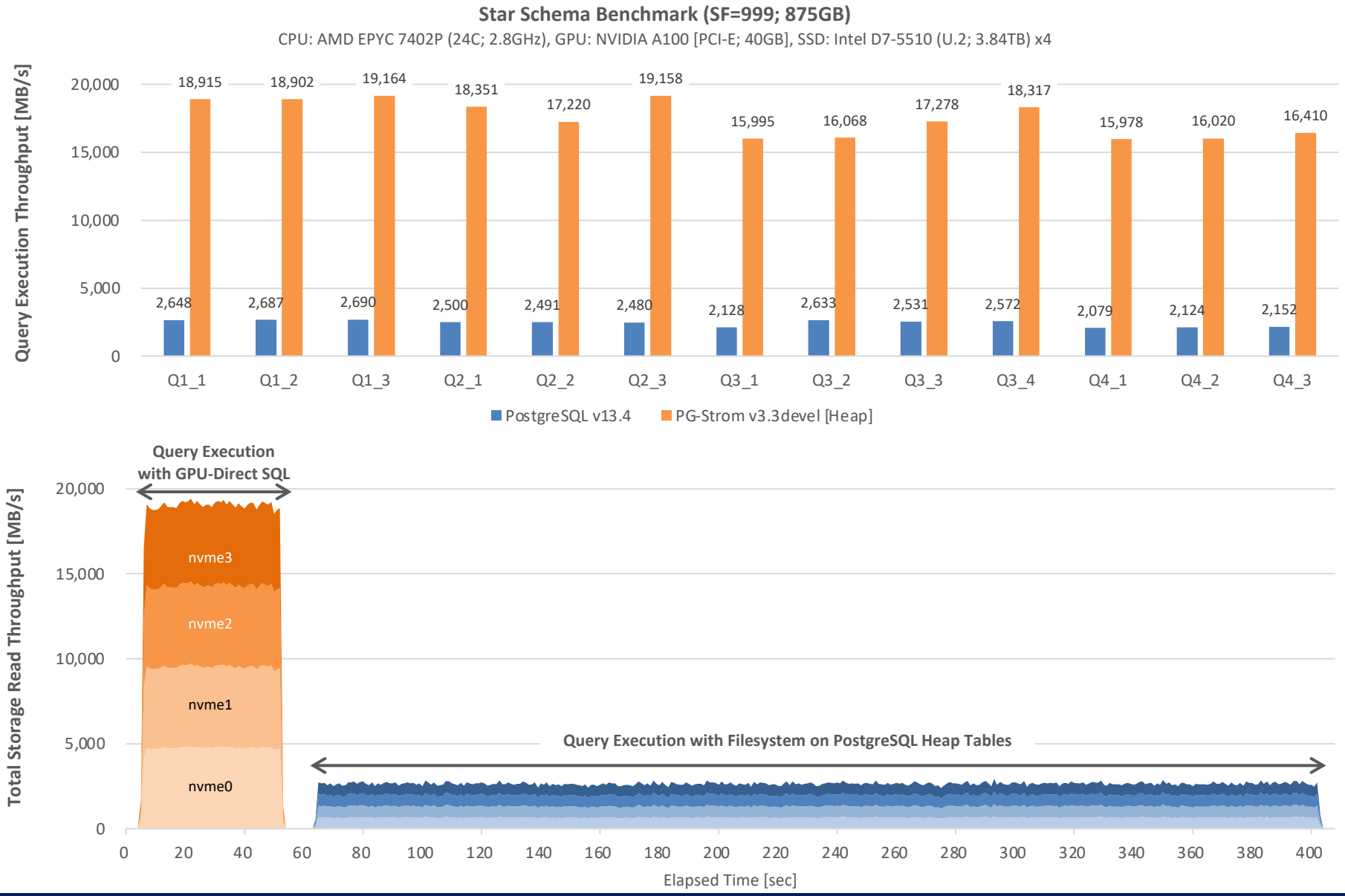
GPU-Direct SQL（3/5）

P2P-RDMAを用いて、他ノードのストレージから直接読み出しも可



P2P-RDMA :
Peer-to-Peer Remote Direct Memory Access

GPU-Direct SQL（4/5）



GPU-Direct SQL（5/5）

ユニーク機能として、各種カンファレンスで注目を集める

最優秀はヘテロDB　米エヌビディアが

nikkei.com/article/DGXMZO24542480S7A211C...

日本経済新聞

朝刊・夕刊　ストーリー　Myニュース　日経会社情報　人事ウオ

トップ　速報　マネー　経済・金融　政治　ビジネス　マーケット　テクノロジー　国際　オピニオン　スポ

最優秀はヘテロDB　米エヌビディアがAIイベント

2017/12/12 18:37

保存　共有　印刷　その他

米エヌビディアが東京・お台場で開いたイベント「GTCジャパン」で、同社が支援する人工知能（AI）関連スタートアップが事業内容を競うプレゼンテーション大会が開かれた。最優秀企業には、画像処理半導体（GPU）を活用してデータベースを構築するスタートアップのヘテロDB（東京・品川、海外浩平社長）が選ばれた。

■得意分野をGPUに任せる

プレゼン大会は、エヌビディア製のGPUを使い、ディープラニング（深層学習）などを利用したAIサービスや製品を開発する企業を支援する「インセプション・プログラム」に選ばれた約80社のうち19社のAI関連スタートアップが参加。事業内容や今後の開発計画などを説明した。

画像の拡大

最優秀企業賞を受賞したヘテロDB（東京・品川）の海外浩平社長＝12日、東京・お台場

最優秀企業に輝いたヘテロDBは、GPUを活用したデータベース構築を手掛ける。GPUはこれまでハイエンドのゲーム用コンピューターやハイパフォーマンスコンピューティング（HPC）に活用されることが多かったが、ヘテロDBではデータベースの一部にGPUを活用し、処理の高速化とシステム運用を単純化するサービスを展開する。

海外社長は「データベースの処理の中にはGPUが得意な部分もある。ゲーミングやHPCなど自動車レースのF1のような領域で使われていたGPUを広く使える存在にしたい」と話した。選考したエヌビディアの山田泰永シニアマネージャーは「GPUの新たな活用領域を切り開いた」と評価した。

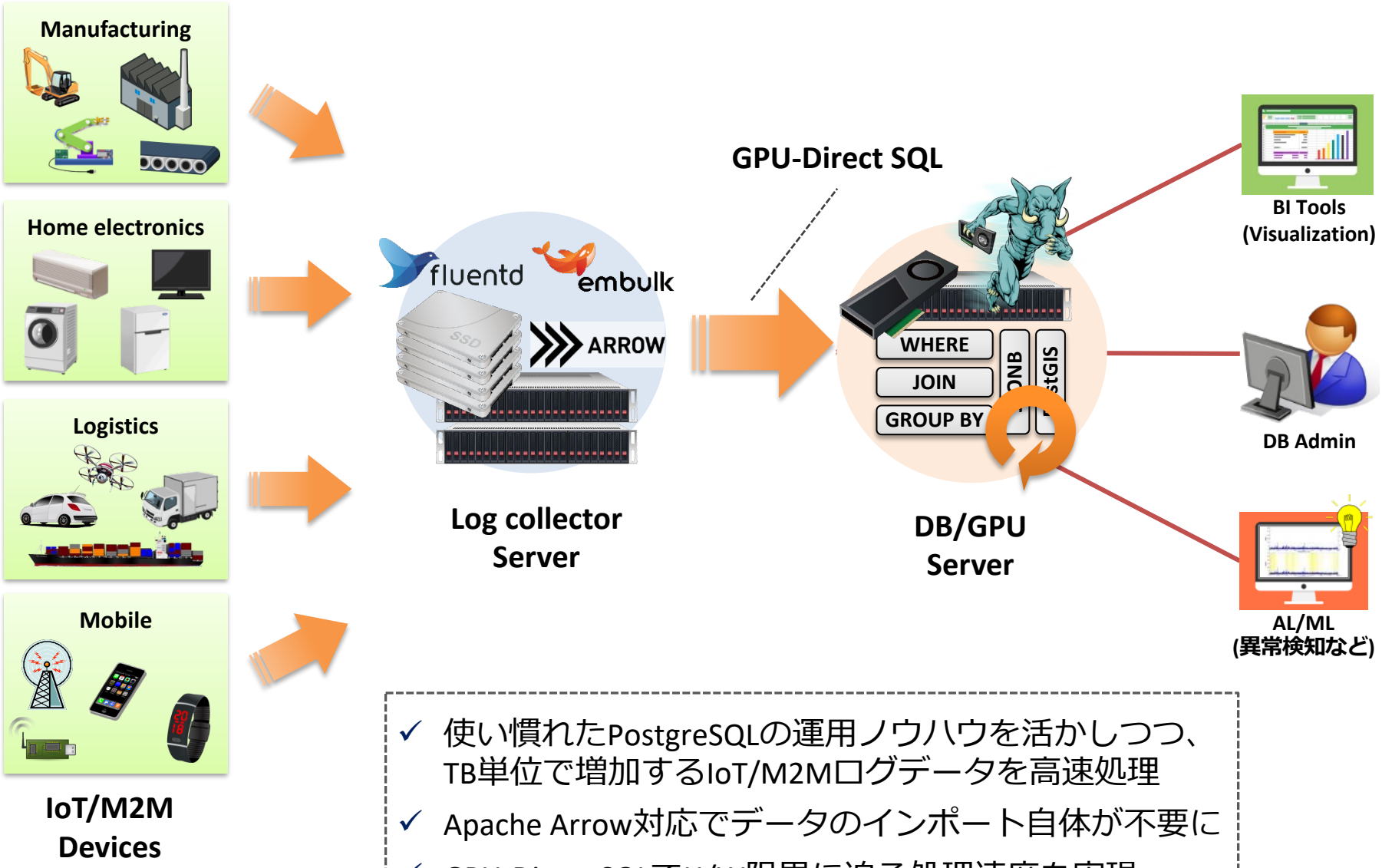
GPU Technology Conference Japan **2017**において、
AIベンチャー20社の中から、最優秀賞である
Inception Awardを受賞

GPU Technology Conference **2017** (San Jose) にて、
ポスター発表が Top-5 Finalist としてエントリー
（会場投票で惜しくも次点）

IoT/M2M領域での利用を考える

～Apache Arrow/Fluentdとの連携～

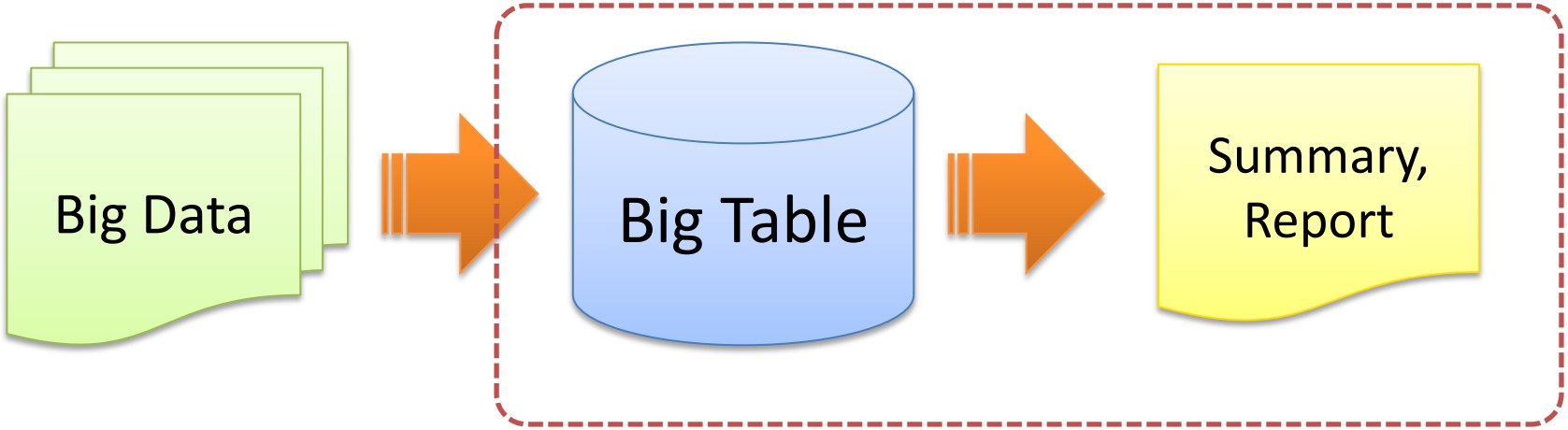
想定利用シーン - IoT/M2Mログデータ処理基盤



- ✓ 使い慣れたPostgreSQLの運用ノウハウを活かしつつ、TB単位で増加するIoT/M2Mログデータを高速処理
- ✓ Apache Arrow対応でデータのインポート自体が不要に
- ✓ GPU-Direct SQLでH/W限界に迫る処理速度を実現。

大量データの処理を考える（1/2）

検索・集計処理がツラい…。



ワークロードの特性

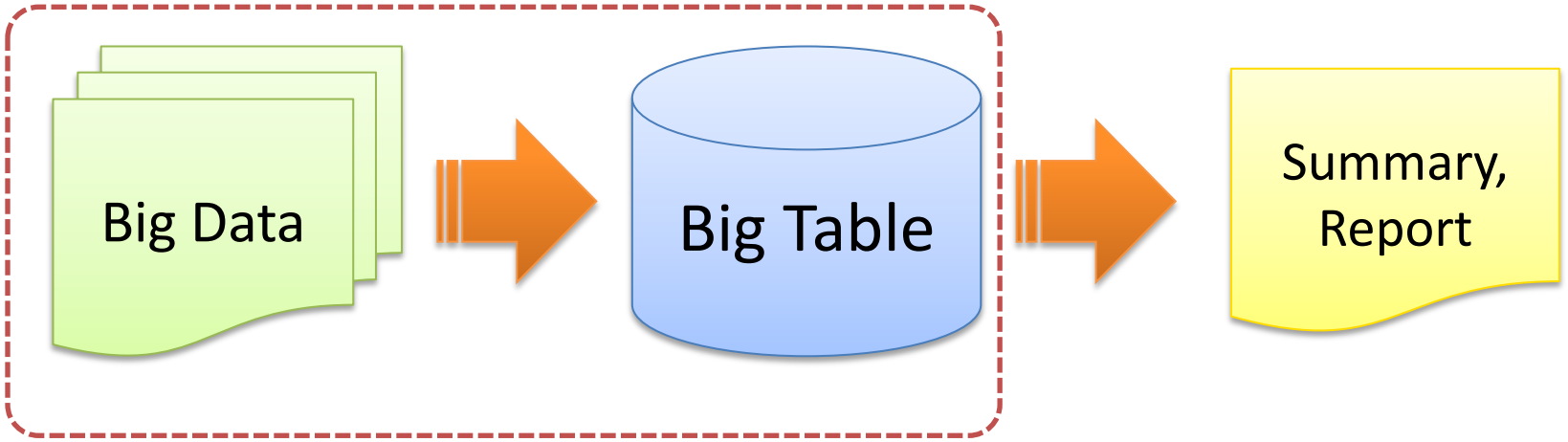
- 多くの場合、テーブルのフルスキャンを伴う。
- テーブル単位のデータ分割が一般的（パーティションなど）
- ストレージからの読み出し速度で律速

どうする？

- **列指向データ**の利用　…　参照しないデータを読み出さない

大量データの処理を考える（2/2）

そもそもDBへのインポートもツラい…。



ワークロードの特性

- 大量の元データを読み出して、**DBの内部形式**に変換してテーブルへ書き込み。
- データ形式を変換するCPU、ストレージへの書き込み速度で律速
- 並列度を上げにくいことも多い（順次読み出し前提のデータ形式）

どうする？

- **外部データ**を “そのまま” 読み出す　…　インポート自体の必要性をなくす

Apache Arrowデータ形式（1/2）

列指向の構造化データ形式

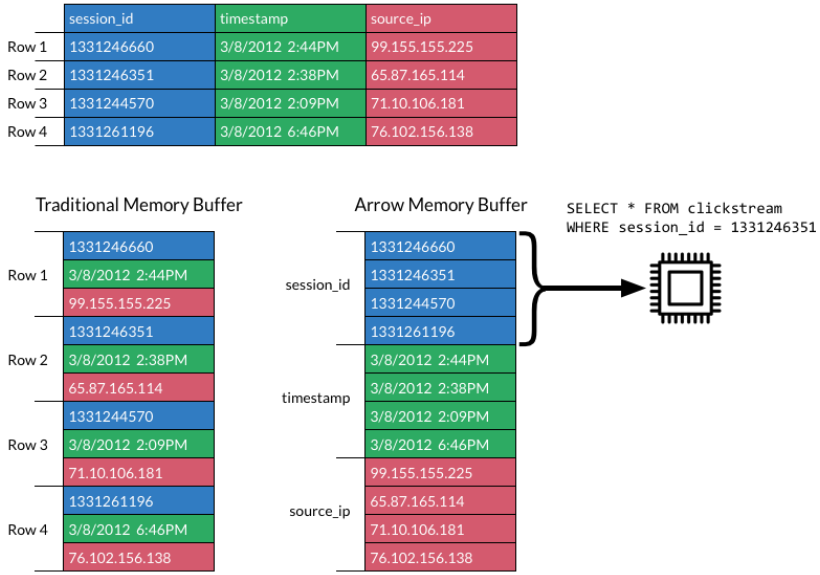
- 被参照列のみ読み出す（= I/O量の削減）が可能
- データが隣接しているため、SIMD命令やGPUでの処理性能を引き出しやすい

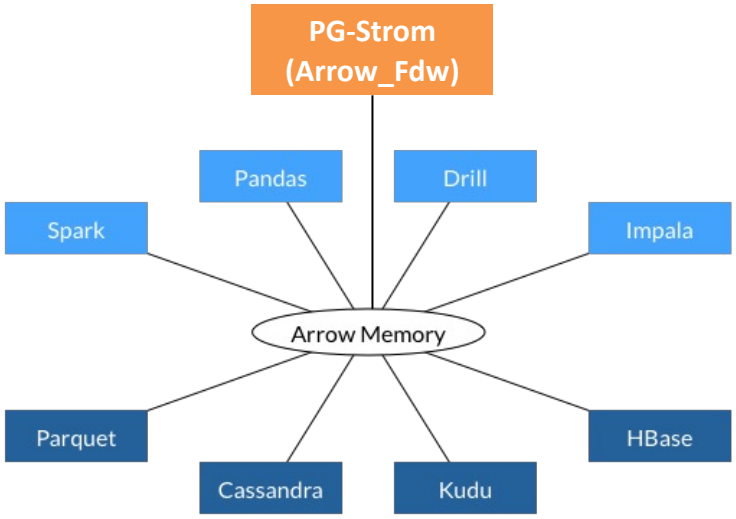
多くのビッグデータ系OSSで対応が進んでいる

- SparkやDrill、Python(PyArrow)など。PG-Strom (Arrow_Fdw) もその一つ。

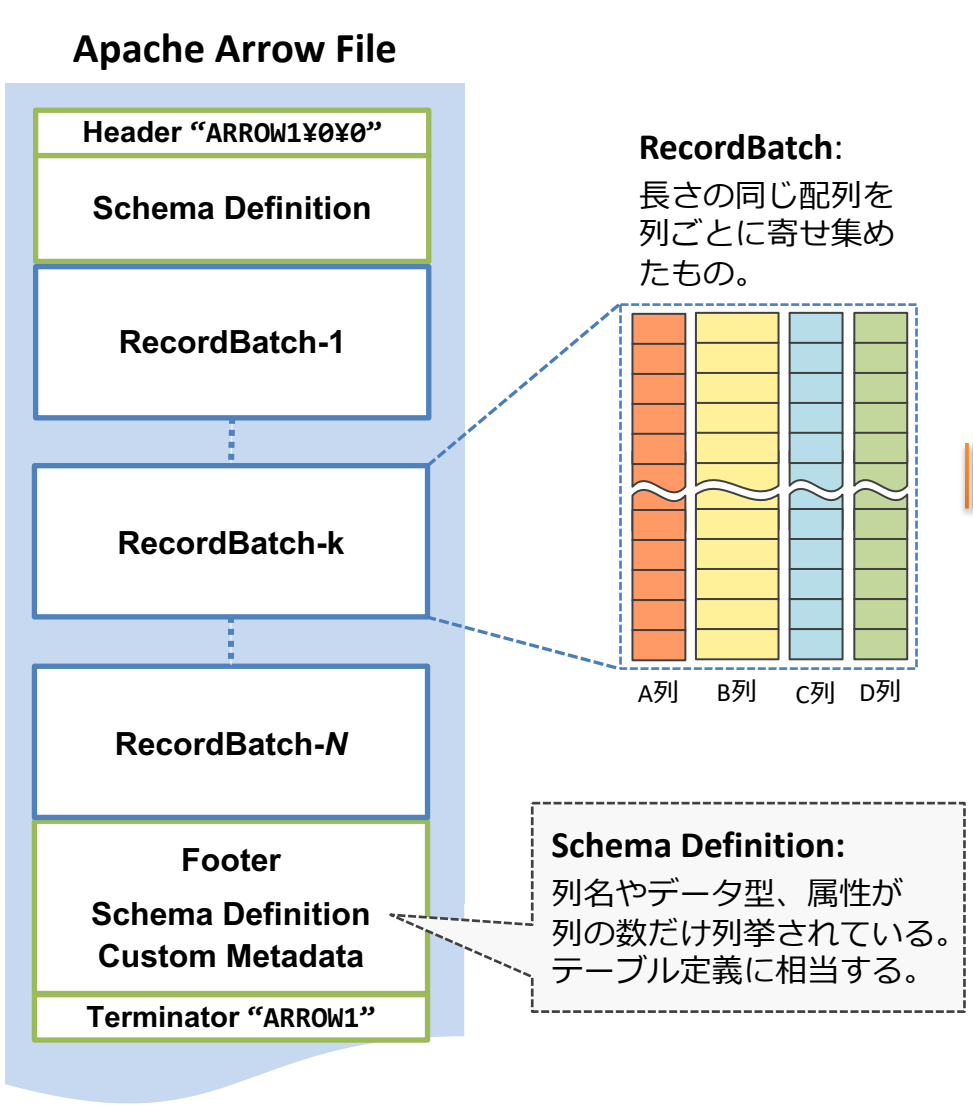
留意点

- 更新はほぼ不可能。追記のみ（Insert-Only）と考えた方がよい





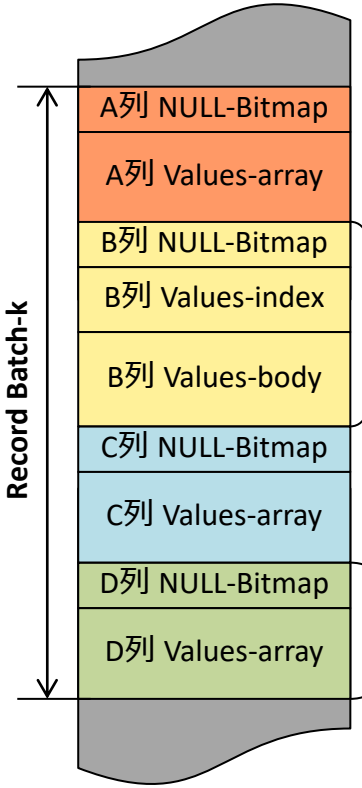
Apache Arrowデータ形式（2/2）



RecordBatch:
長さの同じ配列を
列ごとに寄せ集め
たもの。

Schema Definition:
列名やデータ型、属性が
列の数だけ列挙されている。
テーブル定義に相当する。

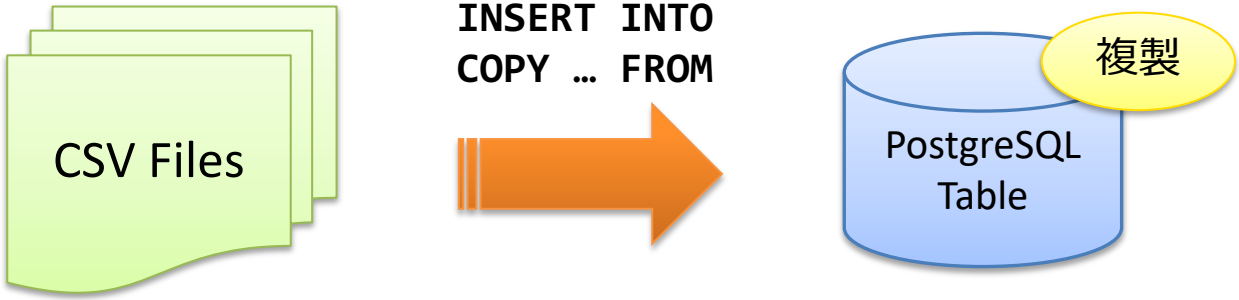
物理的な（ファイル上の）レイアウト



被参照列のみの
ロードが容易

```
SELECT SUM(D)
  FROM f_arrow_tbl
 WHERE B LIKE ‘%abc%’;
```

Apache Arrowならデータのインポートが不要



- ✓ CSVなど外部ファイルを、読み出し、データ形式を変換し、PostgreSQLのテーブルに複製を書き込むことになる。
- ✓ データの件数が多い場合、非常に時間を要する処理になる。

- ✓ PostgreSQLのFDW機能（Arrow_Fdw）を使用して、Arrowファイルに紐付けた外部テーブル（Foreign Table）を定義する。
- ✓ データの移動を伴わないため、一瞬で完了する。（Arrowファイル自体の移動は OS 上のファイルコピー）

Arrow_Fdwによる外部テーブルの定義（1/3）

```
postgres=# drop foreign table f_mytest ;
DROP FOREIGN TABLE
postgres=# CREATE FOREIGN TABLE f_mytest (
             id int,
             ts timestamp,
             x  float8,
             y  text,
             z  numeric
           ) SERVER arrow_fdw
             OPTIONS (file '/tmp/mytest.arrow');
CREATE FOREIGN TABLE

postgres=# SELECT * FROM f_mytest;
 id |            ts             |         x          |      y       |    z
----+---------------------------+--------------------+--------------+---------
  1 | 2022-08-02 00:34:44.210589 |   77.4344383633856 | 65ac7f6      | 38.6218
  2 | 2019-05-11 03:06:16.353798 |  95.33235230265761 | 9105319395   | 51.8267
  3 | 2015-01-04 11:02:25.66779  |  93.67415248121794 | b56930f7834  | 84.9033
  :             :                           :                :           :
 24 | 2022-09-24 06:26:02.058316 | 17.668632938372266 | c5e35        | 55.9739
 25 | 2016-08-08 18:16:12.248363 |   92.2211769466387 | fa889dd51692 |  19.246
(25 rows)
```

Arrow_Fdwによる外部テーブルの定義（2/3）

IMPORT FOREIGN SCHEMAで、Arrowファイルのスキーマ定義ごと取り込むのが楽

```
postgres=# import foreign schema f_mytest
             from server arrow_fdw
             into public options (file '/tmp/mytest.arrow');
IMPORT FOREIGN SCHEMA

postgres=# ¥d f_mytest
                          Foreign table "public.f_mytest"
 Column |            Type             | Collation | Nullable | Default | FDW options
--------+-----------------------------+-----------+----------+---------+-------------
 id     | integer                     |           |          |         |
 ts     | timestamp without time zone |           |          |         |
 x      | double precision            |           |          |         |
 y      | text                        |           |          |         |
 z      | numeric                     |           |          |         |
Server: arrow_fdw
FDW options: (file '/tmp/mytest.arrow’)
```

Arrow_Fdwによる外部テーブルの定義（3/3）

ファイル形式は共通なので、当然、Pythonでも普通に読める

```
$ python3 -q
>>> import pyarrow as pa
>>> X = pa.RecordBatchFileReader('/tmp/mytest.arrow')
>>> X.schema
id: int32
  -- field metadata --
  min_values: '1'
  max_values: '25'
ts: timestamp[us]
x: double
y: string
z: decimal(30, 8)
-- schema metadata --
sql_command: 'SELECT * FROM mytest'
>>> X.get_record_batch(0).to_pandas()
    id                         ts          x             y           z
0    1 2022-08-02 00:34:44.210589  77.434438       65ac7f6  38.62180000
1    2 2019-05-11 03:06:16.353798  95.332352    9105319395  51.82670000
2    3 2015-01-04 11:02:25.667790  93.674152   b56930f7834  84.90330000
3    4 2017-02-10 21:05:26.631906  90.553723          2103  86.78170000
:    :                          :         :             :           :
22  23 2019-05-19 02:06:38.668702  47.295458  6e00a6e037ad  11.02260000
23  24 2022-09-24 06:26:02.058316  17.668633         c5e35  55.97390000
24  25 2016-08-08 18:16:12.248363  92.221177  fa889dd51692  19.24600000
```

Arrowファイルの作り方①　RDBMSの問合せ結果を変換

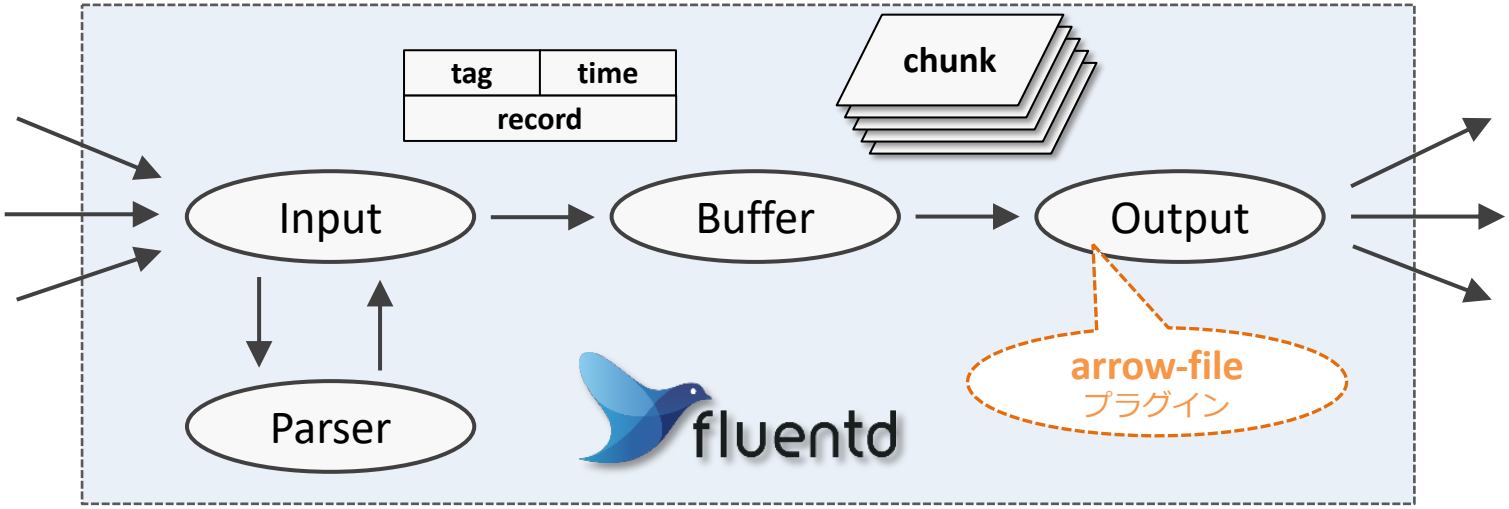
pg2arrowによる PostgreSQL からのダンプ

```
[kaigai@magro ~]$ pg2arrow -d postgres ¥
                          -c "select * from lineorder ¥
                                  where lo_orderpriority in ('1-URGENT','2-HIGH','3-MEDIUM') ¥
                                  order by lo_orderdate" ¥
                          -o /tmp/flineorder.arrow --progress ¥
                          --stat=lo_orderdate
RecordBatch[0]: offset=1712 length=268437080 (meta=920, body=268436160) nitems=1303085
RecordBatch[1]: offset=268438792 length=268437080 (meta=920, body=268436160) nitems=1303085
RecordBatch[2]: offset=536875872 length=268437080 (meta=920, body=268436160) nitems=1303085
RecordBatch[3]: offset=805312952 length=268437080 (meta=920, body=268436160) nitems=1303085
RecordBatch[4]: offset=1073750032 length=268437080 (meta=920, body=268436160) nitems=1303085
RecordBatch[5]: offset=1342187112 length=268437080 (meta=920, body=268436160) nitems=1303085
RecordBatch[6]: offset=1610624192 length=268437080 (meta=920, body=268436160) nitems=1303085
RecordBatch[7]: offset=1879061272 length=268437080 (meta=920, body=268436160) nitems=1303085
RecordBatch[8]: offset=2147498352 length=76206296 (meta=920, body=76205376) nitems=369928
$ ls -lh /tmp/flineorder.arrow
-rw-r--r--. 1 kaigai users 2.1G Oct 12 20:17 /tmp/flineorder.arrow
```

- ✓ -c オプションで指定したクエリの実行結果をArrowファイルとして保存
- ✓ 単純なダンプと異なり、検索条件を指定したり、ソートやJOINなども可能
- ✓ デフォルトで 256MB 毎に RecordBatch を作る（-s オプションで変更可能）
- ✓ --stat オプションはmin/max統計情報の採取を有効にする。（後述）

Arrowファイルの作り方②　FluentdでArrowファイルを生成

Fluentdを介して受け取ったデータを、Apache Arrowファイルとして書き出し



Fluentd

- Treasure Data社の古橋貞之氏によって開発された、ログ収集ツールのデファクト。
- 800種類を越えるプラグインを組み合わせる事で、自在にカスタマイズする事が可能。

arrow-file プラグイン

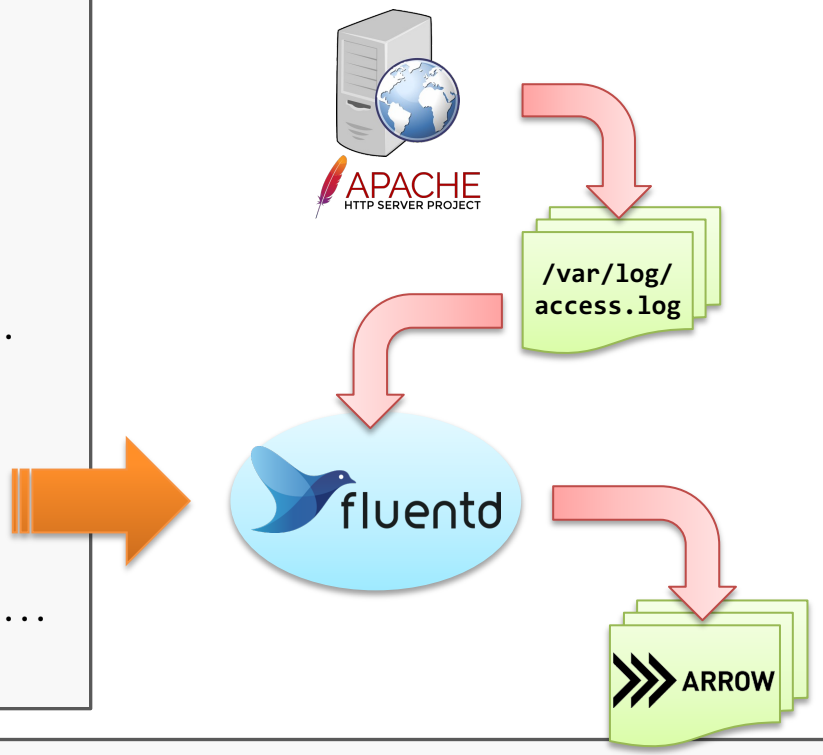
- Fluentd向けOutputプラグインで、受け取ったログを Apache Arrow 形式のファイルとして書き出すことができる。
- 書き出したファイルは、PG-StromのArrow_Fdw機能を用いてそのまま読み出し可能。
- タイムスタンプ等のmin/max統計情報にも対応している。

Arrowファイルの作り方②　FluentdでArrowファイルを生成

設定例）Httpdのログを収集して Arrow形式で出力

/etc/td-agent/td-agent.conf 設定例

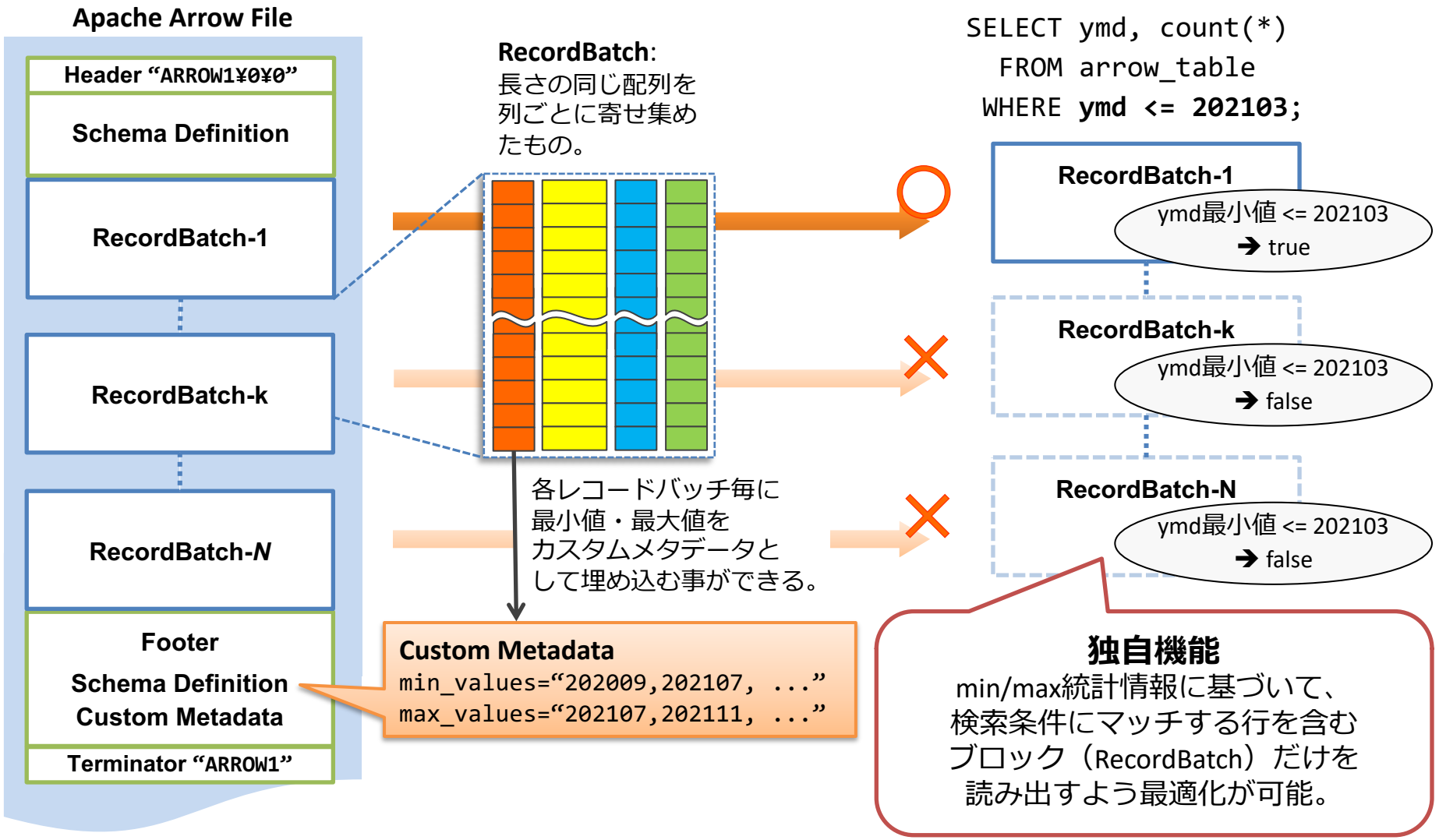
```
<source>
  @type tail
  path /var/log/httpd/access_log
  pos_file /var/log/td-agent/httpd_access.pos
  tag httpd
  format apache2
  <parse>
    @type apache2
    expression /^(?<host>[^ ]*) [^ ]* (?<user>[^ ]*) ...
    time_format %d/%b/%Y:%H:%M:%S %z
  </parse>
</source>

<match httpd>
  @type arrow_file
  path /tmp/mytest%Y%m%d.%p.arrow
  schema_defs "ts=Timestamp[sec],host=Utf8,method=Utf8,...
  ts_column "ts"
</match>
```




```
$ arrow2csv /tmp/mytest20220124.3206341.arrow --head
"ts","host","method","path","code","size","referer","agent"
"2022-01-24 06:13:42","192.168.77.95","GET","/docs/ja/js/theme_extra.js",200,195,"http://b...
"2022-01-24 06:13:42","192.168.77.95","GET","/docs/ja/js/theme.js",200,4401,"http://buri/d...
"2022-01-24 06:13:42","192.168.77.95","GET","/docs/ja/img/fluentd_overview.png",200,121459...
"2022-01-24 06:13:42","192.168.77.95","GET","/docs/ja/search/main.js",200,3027,"http://bur...
"2022-01-24 06:13:42","192.168.77.95","GET","/docs/ja/fonts/Lato/lato-regular.woff2",200,1...
          :                  :         :           :
```

ユニーク技術：Apache Arrowのmin/max統計情報（1/3）

中身を読むまでもなく、明らかに検索条件にマッチしないブロックを読み飛ばす。



ユニーク技術：Apache Arrowのmin/max統計情報（2/3）

明らかに検索条件に合致しない RecordBatch を読み飛ばしているのが分かる

```
postgres=# import foreign schema flineorder
                    from server arrow_fdw
                    into public options (file '/tmp/flineorder.arrow');
IMPORT FOREIGN SCHEMA

postgres=# explain (analyze, costs off)
           select count(*),sum(lo_revenue)
             from flineorder
            where lo_orderdate between 19930101 and 19931231;
                                          QUERY PLAN
-----------------------------------------------------------------------------------------------
 Aggregate (actual time=36.967..36.969 rows=1 loops=1)
   ->  Custom Scan (GpuPreAgg) on flineorder (actual time=36.945..36.950 rows=1 loops=1)
         Reduction: NoGroup
         Outer Scan: flineorder (actual time=10.693..20.367 rows=2606170 loops=1)
         Outer Scan Filter: ((lo_orderdate >= 19930101) AND (lo_orderdate <= 19931231))
         Rows Removed by Outer Scan Filter: 966184
         referenced: lo_orderdate, lo_revenue
         Stats-Hint: (lo_orderdate >= 19930101), (lo_orderdate <= 19931231) [loaded: 2, skipped: 7]
         files0: /tmp/flineorder.arrow (read: 206.00MB, size: 2120.69MB)
 Planning Time: 0.175 ms
 Execution Time: 42.146 ms
(11 rows)
```

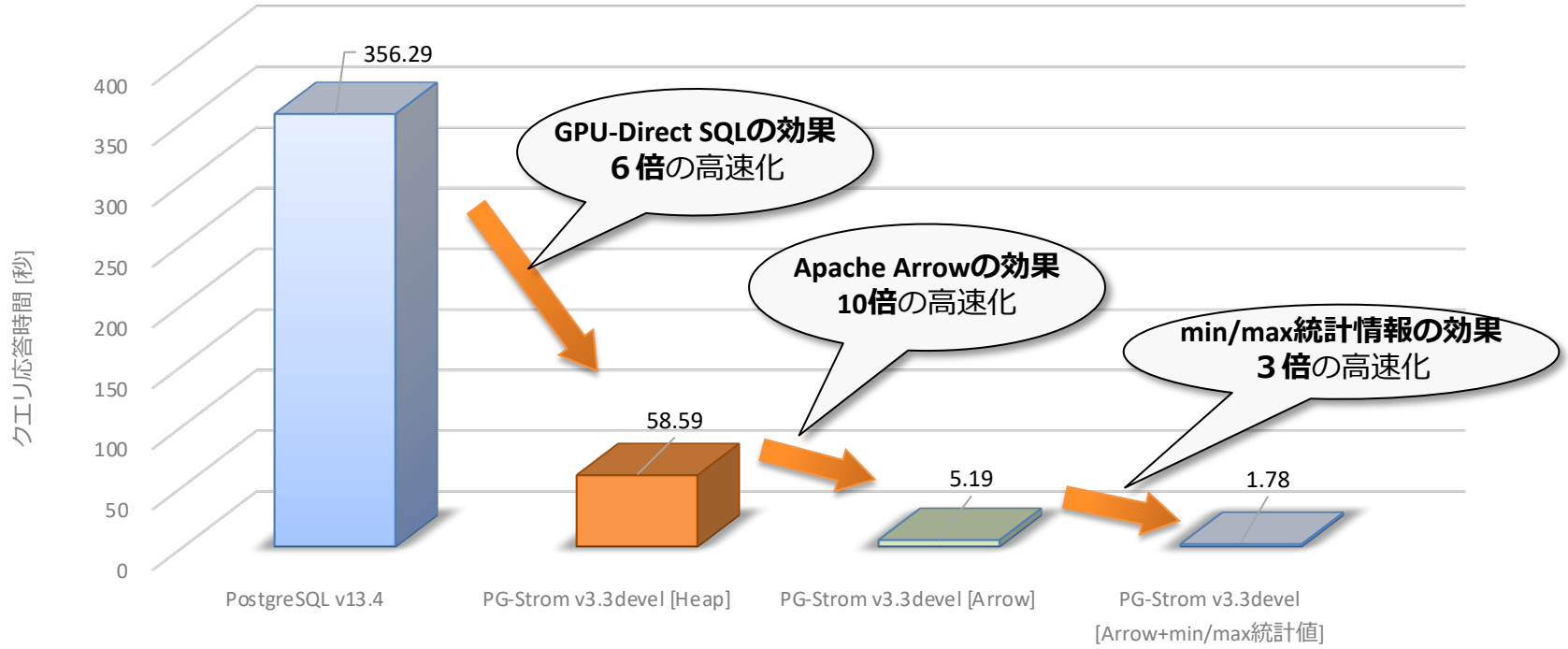
検索条件にマッチしない事が明らかなら、そもそも読み出す必要もない。

ユニーク技術：Apache Arrowのmin/max統計情報（3/3）

統計情報を活用するよう、SSBMクエリQ1_1を微修正

```
select sum(lo_extendedprice*lo_discount)
from lineorder,date1
where lo_orderdate = d_datekey
and d_year = 1993
and lo_discount between 1 and 3
and lo_quantity < 25;
```

```
select sum(lo_extendedprice*lo_discount)
from lineorder
where lo_orderdate between 19930101
                       and 19931231
and lo_discount between 1 and 3
and lo_quantity < 25;
```

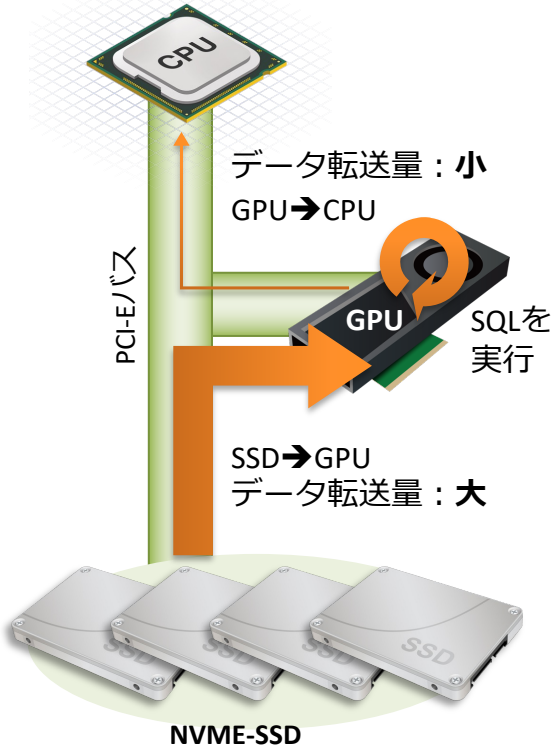


今後の技術ロードマップ

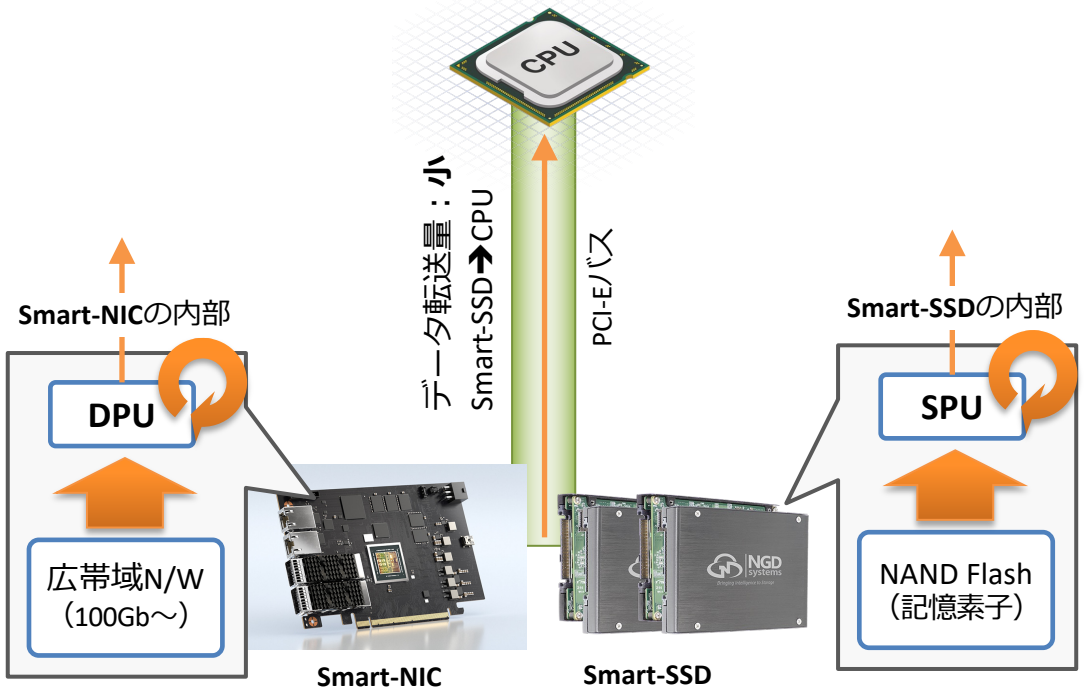
～Smart-NIC/Smart-SSDへの対応～

Smart-SSD / Smart-NICへの対応に向けて

現行技術：**GPU-Direct SQL**



新技術：**Smart-SSD / Smart-NIC対応**



- 現行技術：SSD ➔ GPUへデータを転送する段階で「ゴミデータ」がPCI-Eバスを通らざるを得ないため、『GPUのデータ処理能力が PCI-Eバスの帯域に縛られている』
- Smart-NIC / Smart-SSD対応： NIC上のプロセッサ(DPU)や、SSD上のプロセッサ(SPU)でSQLを処理し、不要なデータがPCI-Eバスに漏出する前に削ぎ落す。

➔ データ処理能力がPCI-Eバスの帯域に縛られなくなる事を期待。

直感的なアーキテクチャの理解

GPU-Direct SQL

（動力集中方式）

現行技術

Smart-SSD / Smart-NIC

（動力分散方式）

開発中技術

2023年に登場予定

- ✓ 現在入手可能で最も高性能なプロセッサである **GPU** を数枚使用し、SQLワークロードを数千スレッドの並列処理で実行する。
- ✓ GPUの計算能力に利点があり、PostGISや類似度計算など、**複雑な条件句の処理においては依然として有利**と考えられる。
- ✓ データ転送能力が PCI-E バス帯域によって律速されてしまう点が現在の課題。

- ✓ 今後、2023年にかけて Smart-NIC や Smart-SSD 製品の市場投入が本格的に進む。
- ✓ **GPUやCPUほど高性能ではない**ものの、DPUやSPUはデータに近いところでSQLを処理できるため、PCI-Eバスに漏出するデータ量自体を減らす事ができる。
- ✓ PCI-Eバスはもはや律速要因とはならない。
- ✓ **低消費電力**はもう一つの訴求ポイント

まとめ

GPU-Direct SQL

- P2P-DMA技術を用いて、SSDからGPUへ「直接データを供給」する。
- 2021年以降、Linux kernelドライバ機能をNVIDIA GPUDirect Storageが代替し、リモートのストレージや、NFS区画からの直接読み出しも可能に。
- PCI-E 4.0のEPYCサーバでは、概ね20GB/sに迫るパフォーマンスを実測。

Apache Arrow

- ビッグデータ処理でよく利用される、列指向の構造化データ形式。
- 追記のみ可能で、時系列のログを記録し続ける IoT/M2M データに適する。
- IoT/M2Mデータ収集によく利用される Fluentd のプラグインを提供しており、収拾したログを直接 Apache Arrow 形式で書き出す事ができる。
- ➔ PG-Stromはこれをそのまま読み込む事ができるため、データを改めてDBへインポートする手間がかからない。
- 独自機能の min/max 統計情報を用いる事で、より効率的な絞り込みが可能

リソース

- GitHub: https://github.com/heterodb/pg-strom
- Manual: http://heterodb.github.io/pg-strom/ja/

オモシロ技術を、カタチにする。