【ICDE2015勉強会】

Session 16：Stream (1, 2番目)
Session 18：System-level Techniques (1番目)

担当：渡辺陽介（名古屋大学）

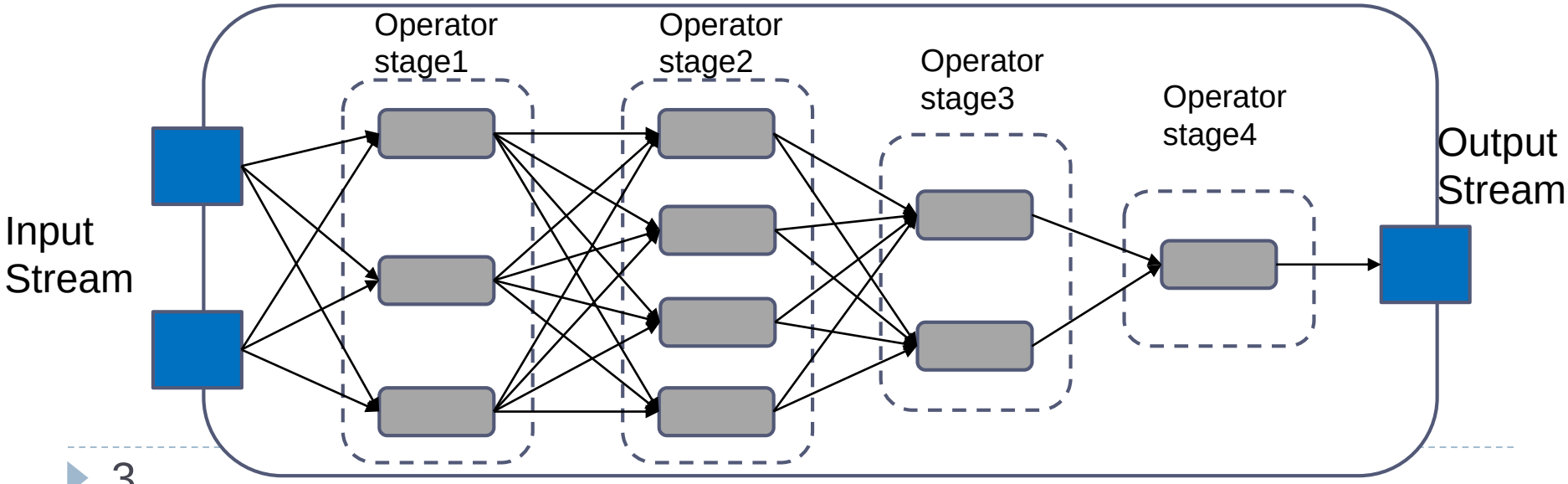
本発表の担当範囲

- Research 16: Stream
  1. ChronoStream: Elastic Stateful Stream Computation in the Cloud
  2. Conflict-Aware Event-Participant Arrangement
     - Streamじゃない．．．
- Research 18: System-level Techniques
  1. Configurable Hardware-based Streaming Architecture using Online Programmable-Blocks

[1本目]

ChronoStream: Elastic Stateful Stream Computation in the Cloud

- ChronoStream：クラウド環境で動く分散ストリーム処理システム
  - ストリーム処理を行うオペレータを複数のインスタンスに分けて複数ノードに配置可能
- クラウド環境では実行中にノード数の増減が起こる
  - スケールアウト，負荷分散，障害回復などのため
  - 各ノード上にあるオペレータの内部状態の維持・共有が重要
    - 内部状態をKey-valueストア（複製あり）で管理



内部状態の管理

- Key-Valueベースの内部状態の管理API
  - Set(Key, Event), Get(Key, Event), Delete(Key)
  - 対象とする内部状態
    - オペレータ内の変数，ウインドウ，データのルーティング表等
- 内部状態の複製管理方式（Chained backup）
  - 内部状態の複製を近隣のノードへ配置
  - チェックポイントの度に内部状態を定期的にコピー

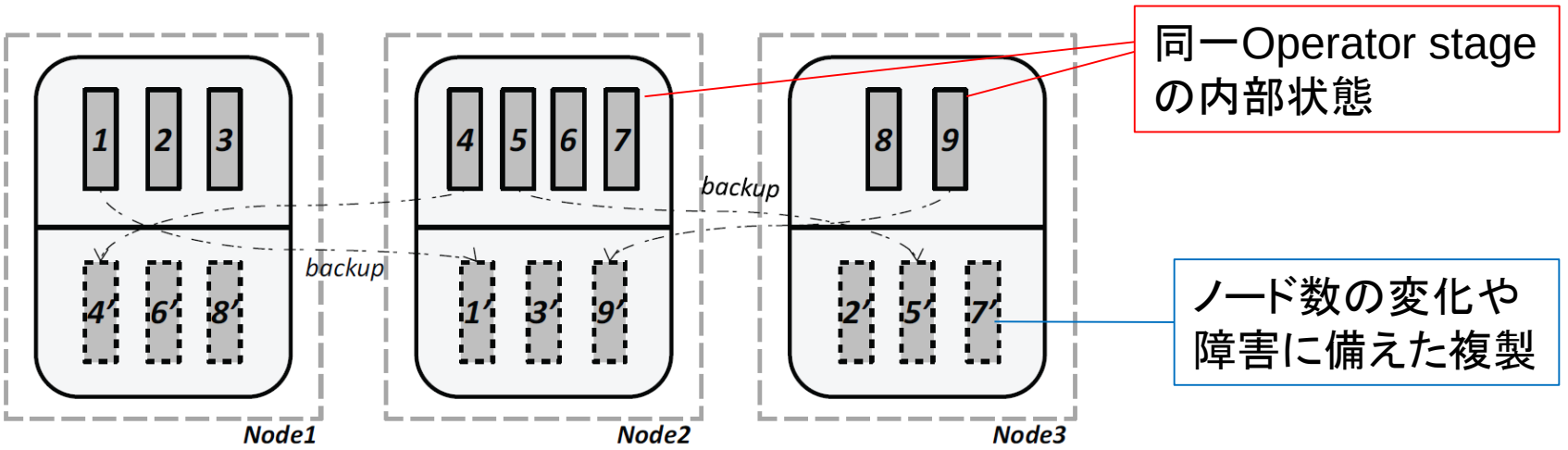
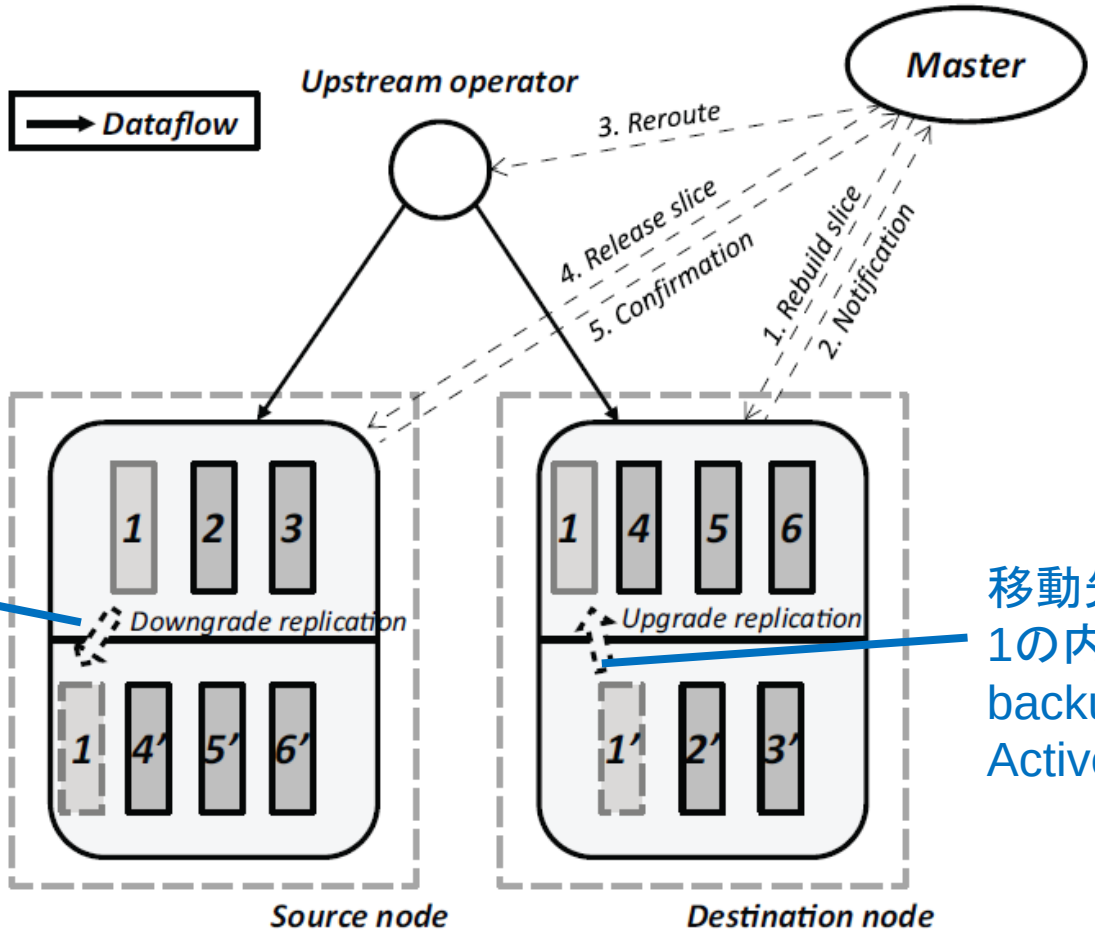


Figure4より引用

Workload migration

- 配置された複製を利用した処理の移動
  - （例）1の処理を別ノードへ移す場合



移動元ノード：
1の内部状態を
activeから
backupへ降格

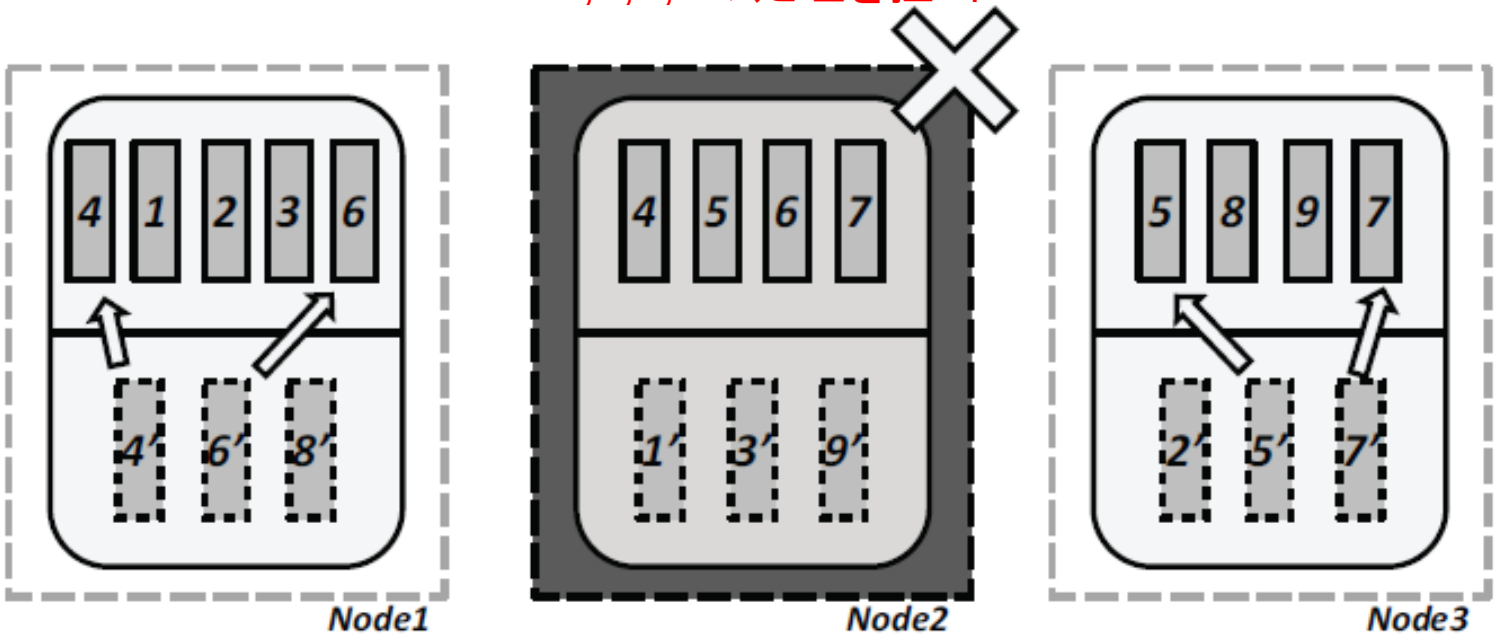
移動先ノード：
1の内部状態を
backupから
Activeへ昇格

Figure6より引用

Fault Tolerance

- 配置された複製を利用した障害回復
  - （例）Node2で障害が起きた場合

障害ノード:
4,5,6,7の処理を担当



4,6の内部状態をbackupから
activeに昇格させ，処理を引き継ぐ

5,7の内部状態をbackupから
activeに昇格させ，処理を引き継ぐ

Figure8より引用

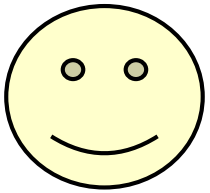
[2本目]

Conflict-Aware Event-Participant Arrangement

- SNSベースのイベント案内システム(Meetup，Whovaなど)の問題点

土曜日の夜

日曜日の
イベントは？



ユーザプロファイル
・スポーツ好き

ハイキング
8時から12時まで

バドミントン
9時から11時まで

バスケットボール
11時半から13時半まで
(バドミントンのコートから
1時間)

競合した候補が提示される

Event-based
social network
(EBSN)

- イベント単体でしかユーザプロファイルとのマッチングをしていない
- 全体的な視点で競合や参加限度を扱っていない

Global Event-participant Arrangement with Conflict and Capacity (GEACC) problem

- 問題設定
  - ユーザ: u1, u2, u3, u4, u5, …
    - 各イベントへの興味を数値化するためのユーザプロファイル
    - 各人が参加できるイベント数の上限
  - イベント: v1, v2, v3, …
    - イベント間の競合関係
    - イベントごとの参加人数の上限
  - 競合せずに上限を満たし，全体の興味を最大化するイベントの割り当てを求める

ユーザ（括弧内はイベント数上限）　　イベント競合関係

イベント（括弧内は参加人数上限）

	u_1 (3)	u_2 (1)	u_3 (1)	u_4 (2)	u_5 (3)	Conflicts
v_1 (5)	**0.93**	0.43	**0.84**	0.64	**0.65**	v_3
v_2 (3)	0	0.35	0.19	**0.21**	**0.4**	NA
v_3 (2)	0.86	**0.57**	0.78	**0.79**	0.68	v_1

Table1より引用

- GEACC問題はNP-hard (定理1)
  - 厳密解を求めるアルゴリズム，近似解を求めるアルゴリズム2種提案

[3本目] Configurable Hardware-based Streaming Architecture using Online Programmable-Blocks

- ストリーム処理をハードウェア(FPGA等)で実現する方法は非常に高速
  - 一方で，新規クエリの登録や，既存クエリのワークロード変化への柔軟性が高くない
- 研究の目的
  - ソフトウェアベースのストリーム処理の柔軟性とハードウェアベースのストリーム処理の高性能を両立させたい
  - Online-Programmable Block (OP-Block)
    - FPGAで作られたストリーム処理の構成要素
      - 入力命令に応じてSelection，Projection，Join（入れ子ループ)の処理が可能
  - Flexible Query Processor (FQP)
    - SQLベースのクエリを以下の情報に変換
      - 複数OP-Block間の接続関係
      - 各OP-Blockへの命令

OP-Block Design

Figure1より引用

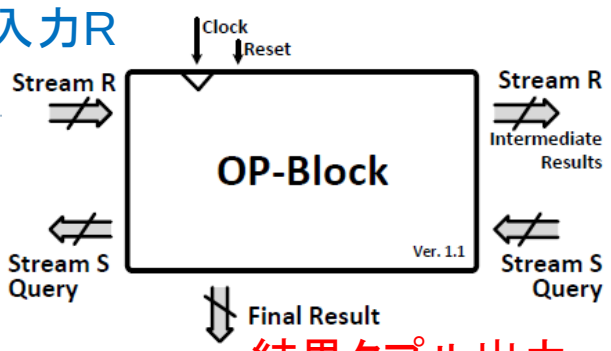
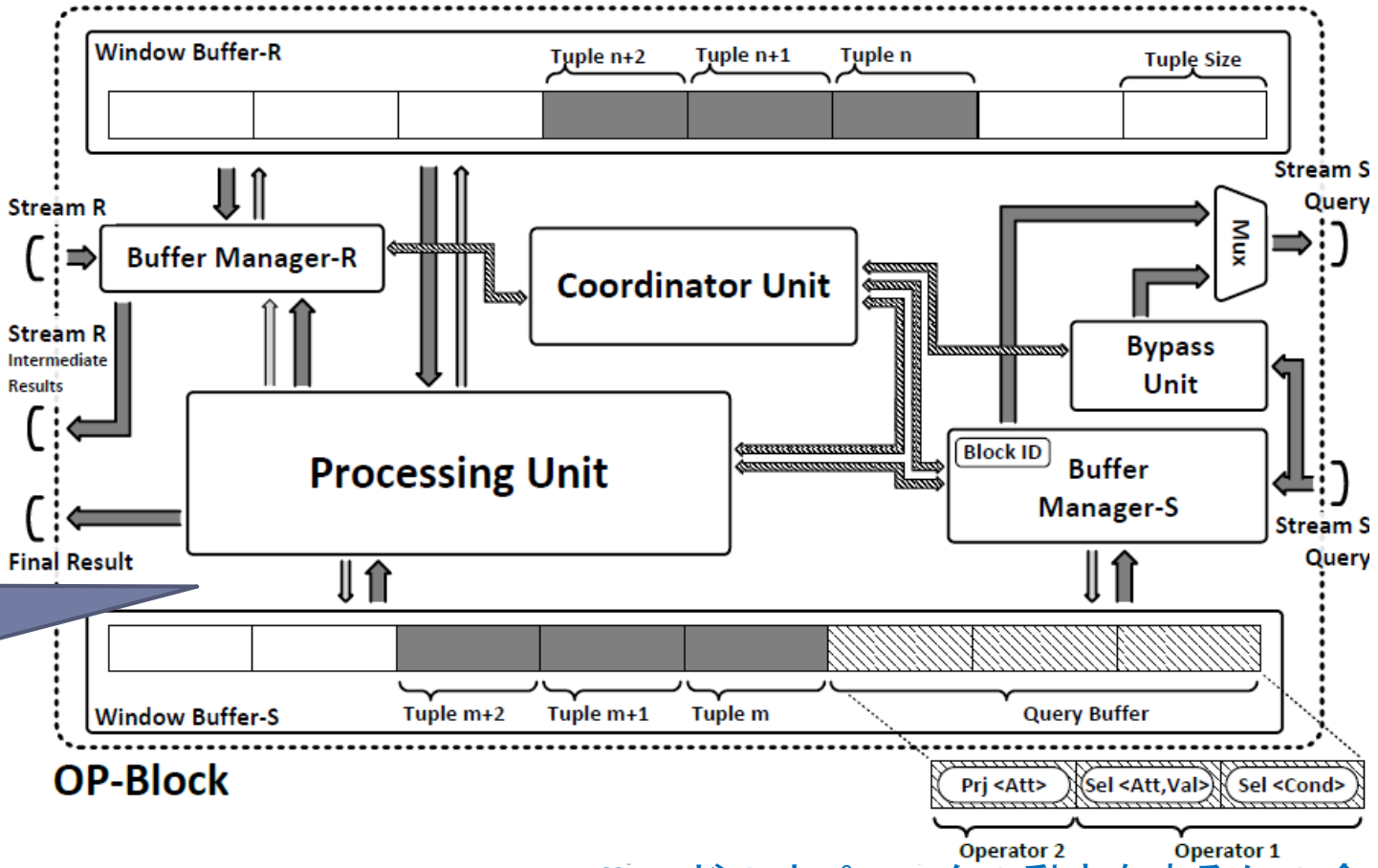


Figure2より引用



Selection
Projection
Joinの処理が可能

QueryBuffer：どのオペレータの動きをするかの命令

Configuration Example

```
CREATE STREAM CS_SEL1 AS
SELECT *
  FROM Customer_Stream
 WHERE Age > 25;

CREATE STREAM CS_OUT1 AS
SELECT *
  FROM CS_SEL1 [Rows 1536], Product_Streams [Rows 1536]
 WHERE Order ID = Product ID;

CREATE STREAM CS_SEL2 AS
SELECT *
  FROM CS_SEL1
 WHERE Gender = female;

CREATE STREAM CS_OUT2 AS
SELECT *
  FROM CS_SEL2 [Rows 2048], Product_Streams [Rows 2048]
 WHERE Order ID = Product ID;
```

Figure8より引用

OP-Block一つでタプル512行分のウインドウが保持可能

ウインドウのためのバッファ
[Rows 1536]には3個必要
[Rows 2048]には4個必要

Block間のトポロジー情報と 各BlockのQueryBufferへの命令に変換

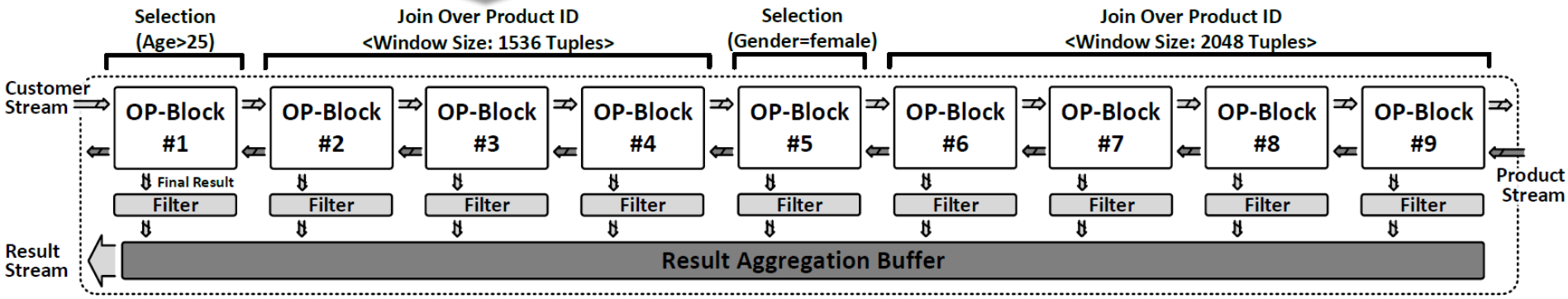


Figure9より引用