X-PROD: EFFICIENT AND SCALABLE CARTESIAN PRODUCT DISTRIBUTION

{gopalv,zhiyuany,hitesh}@apache.org

WHY?

Cartesian products show up without being invited

Theta Joins – Non-equality, Inequality, Complex joins, Skew Joins

- select count(1) from txns t join fraud f on (t.buyer = f.user or t.seller = f.user)

Geo-spatial queries

- select borough, count(crime) from reports, nyc where ST_contains(region, location) group by borough

Similarity queries

- select count(distinct sample_id) from samples s, bases b where distance(s.bits, b.bits) <= 5

Bi-Temporal Overlaps

- select r.period, r.value from rpm r, temp t where OVERLAPS(r.period, t.period) and t.value > 515

SHUFFLE

Map –shuffle- Reduce

Shuffle operates based on partitions

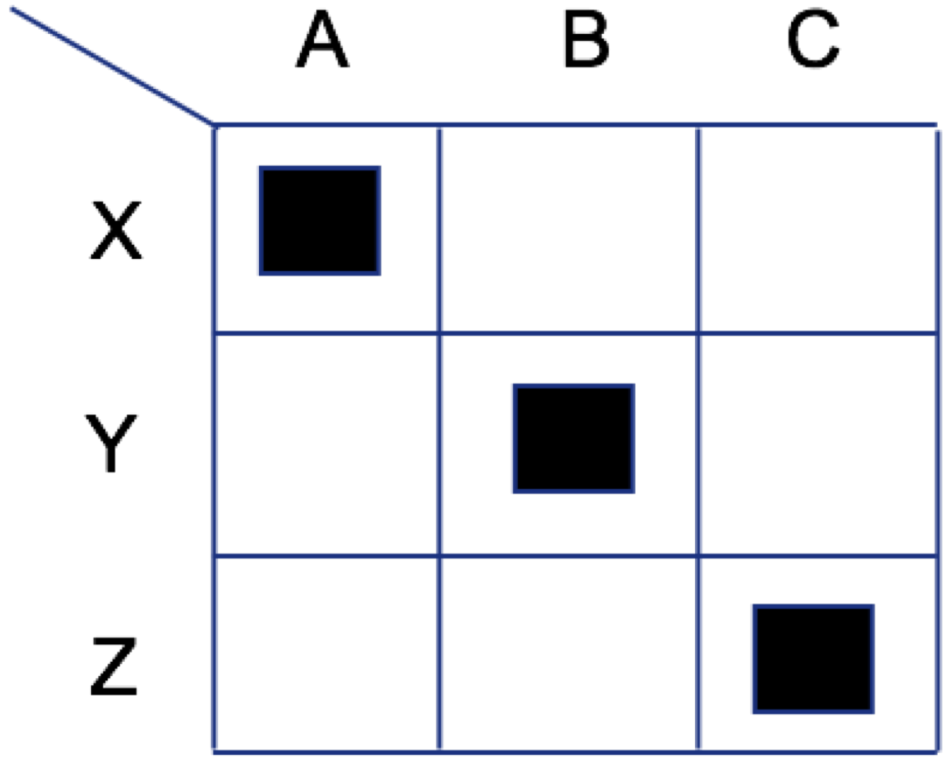
Every mapper writes out N partitions

There are N reducers downstream

Each of N reducer fetches the i^{th} partition

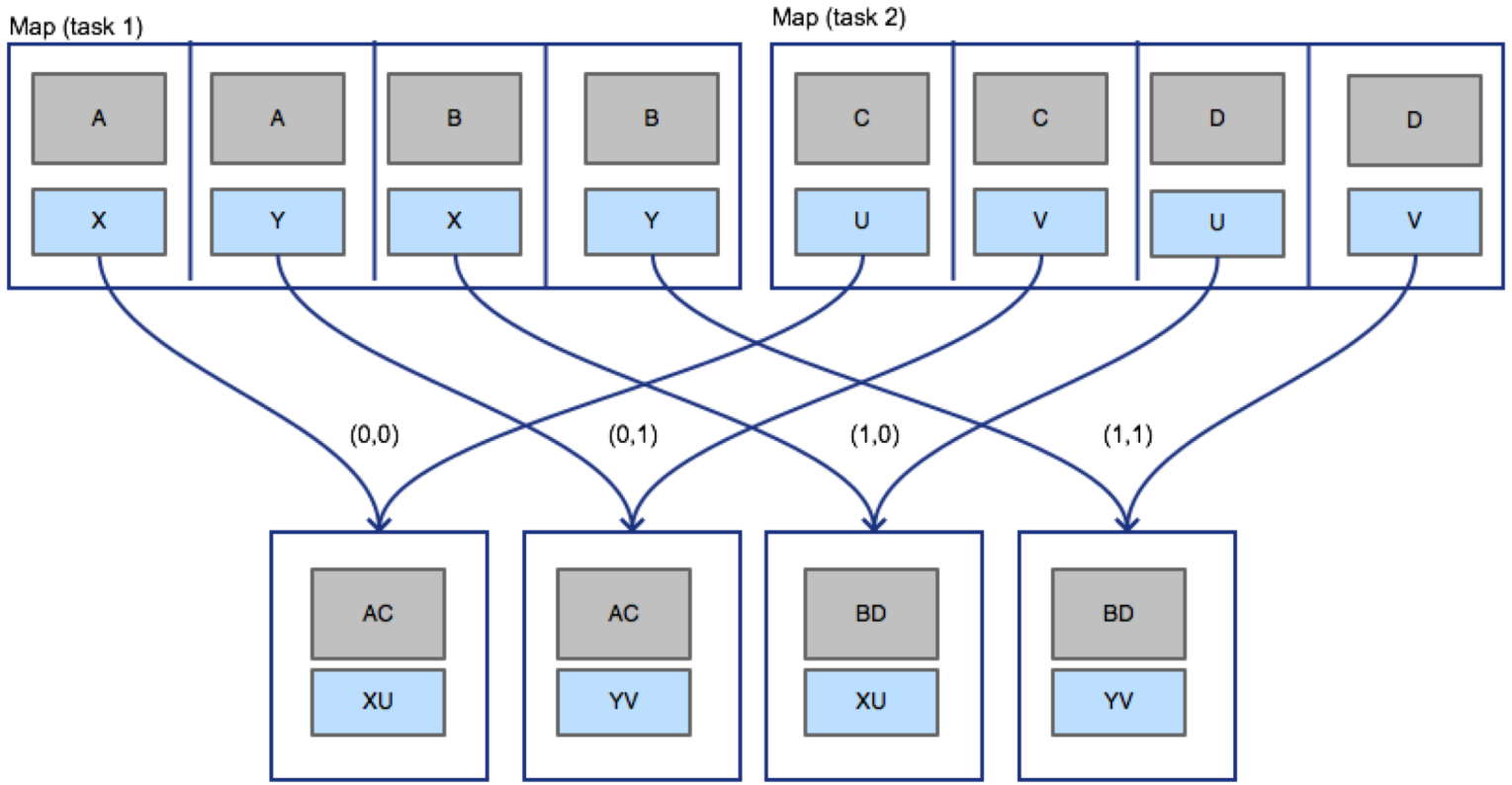
To produce a join

- Each table is assigned a tag
- Mapper writes out N partitions by partition(key)
- Each partition ordered by key,tag
- The Reducer fetches i^{th} partition
- Performs a sort-merge & feeds a join



SYNTHETIC JOIN

Cross-products don't have keys



SYNTHETIC SHUFFLE

Tez Edge Plugins

Tez changed shuffle into a routing layer

Every task sends out N-events

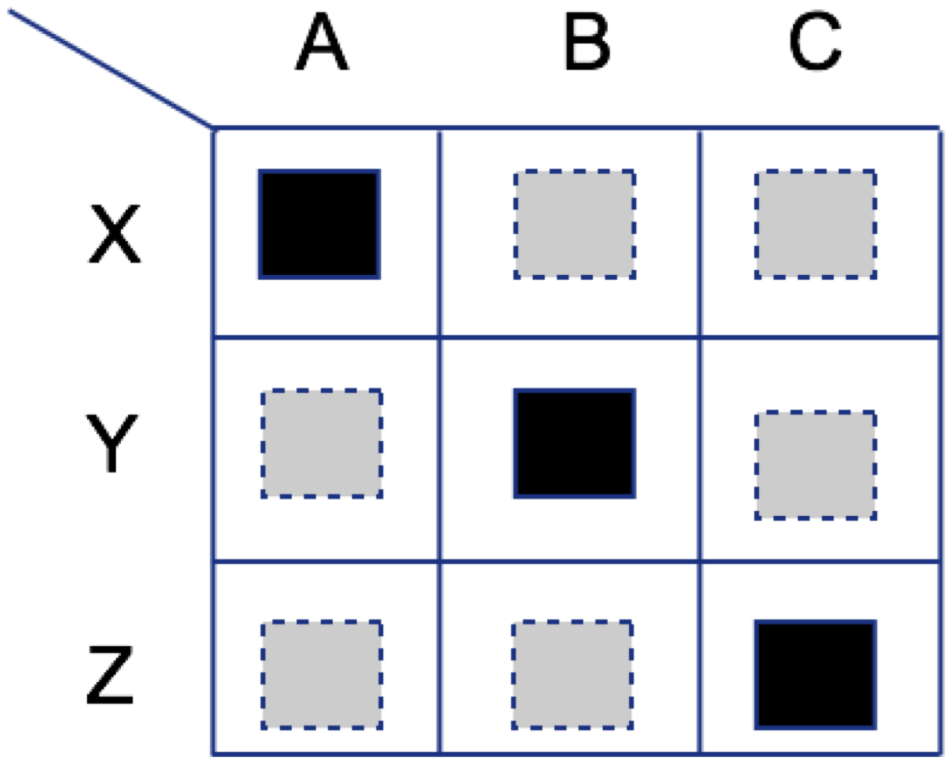
Every reducer gets N-events

A regular Shuffle runs N reducers

A naive X-prod runs N^2 reducers

Tez can also merge adjacent partitions

- Run fewer tasks
- Merge fetches for same task



X-PROD

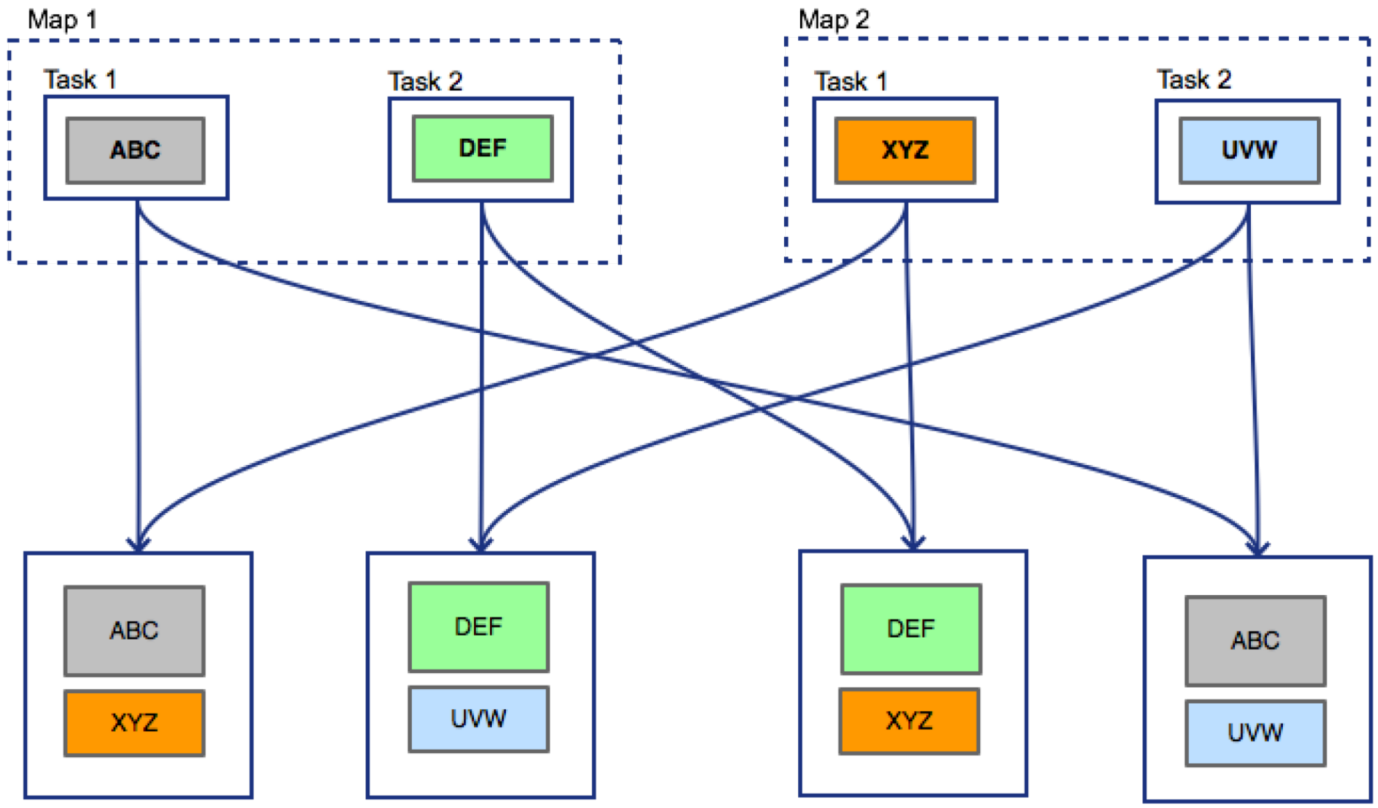
Swapping partition <-> task-id

Join has (J + K) tasks

Every task sends out 1 event

Every reducer gets 2 events

Tez runs J * K reducers



INPUT SKEW X-PROD

Bring back partitions

Join has (J + K) tasks

Every task sends out P events

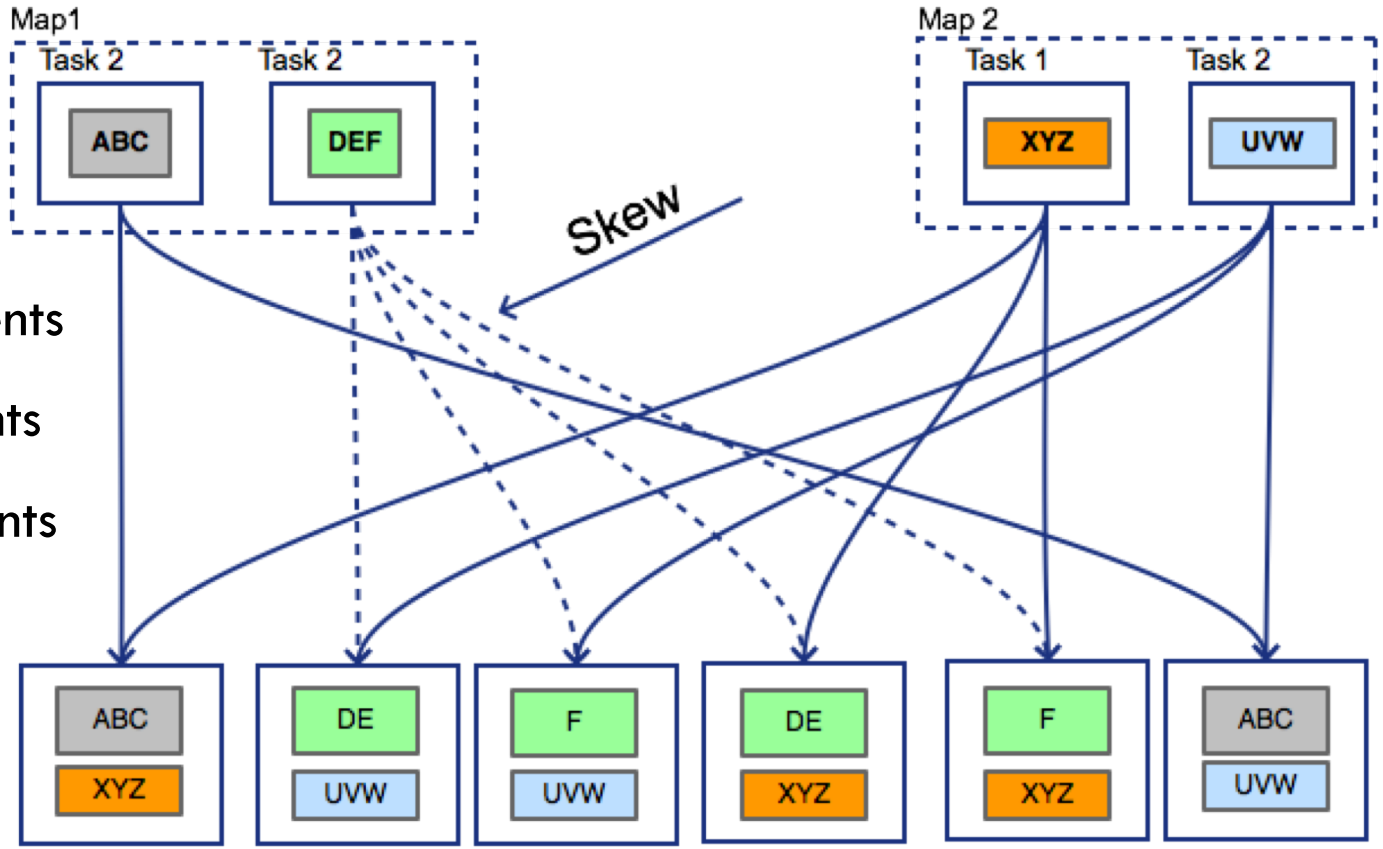
Every reducer gets 2 composite events

Event Group P * J => N chunk events

Event Group P * K => M chunk events

cross((P_J),(P_K)) => N * M tuples

Schedule N * M reducers



FAIR GROUPING

Merge partitions if we can

Grouping heuristic with size goals

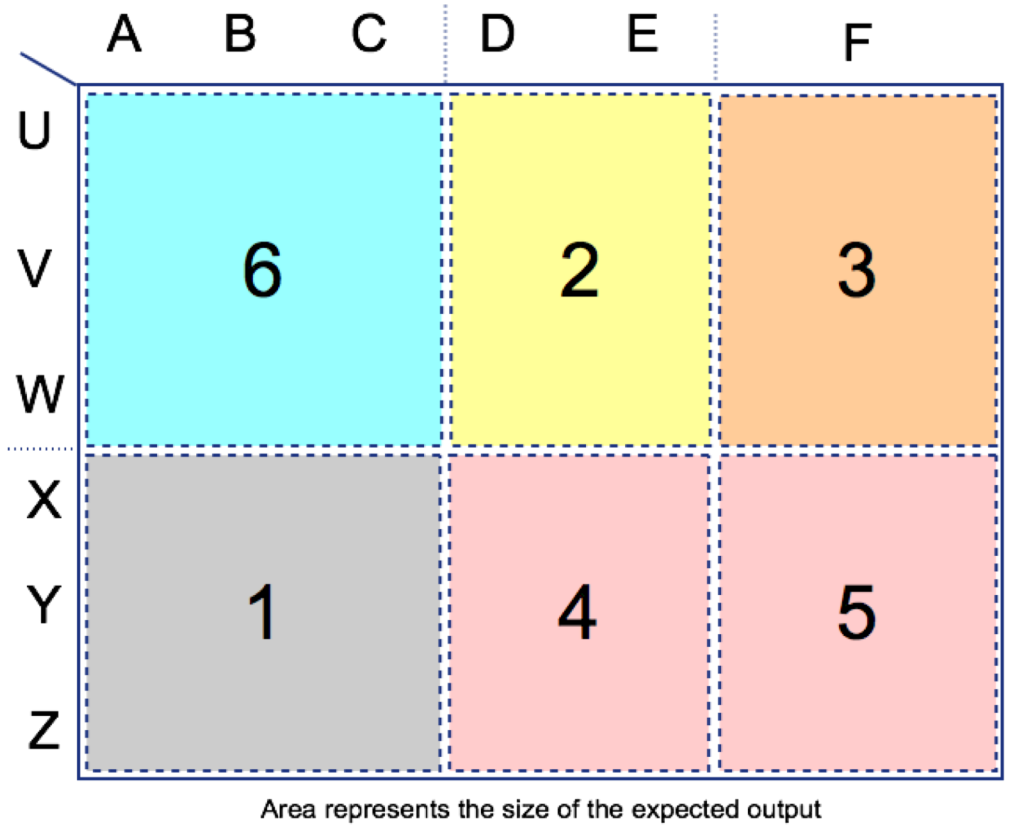
Task output size has a counter

Chunking is done per-task input

- Each chunk only fetches from one node
- Each chunk spans continuous partitions
- Minimum size of a chunk is 1 partition

Grouping tries to merge chunks

- Merge by host to reduce network hops



LOCALITY SCHEDULING

Reducers with locality

Grouping by host merges only one side

Composite events produced have locality

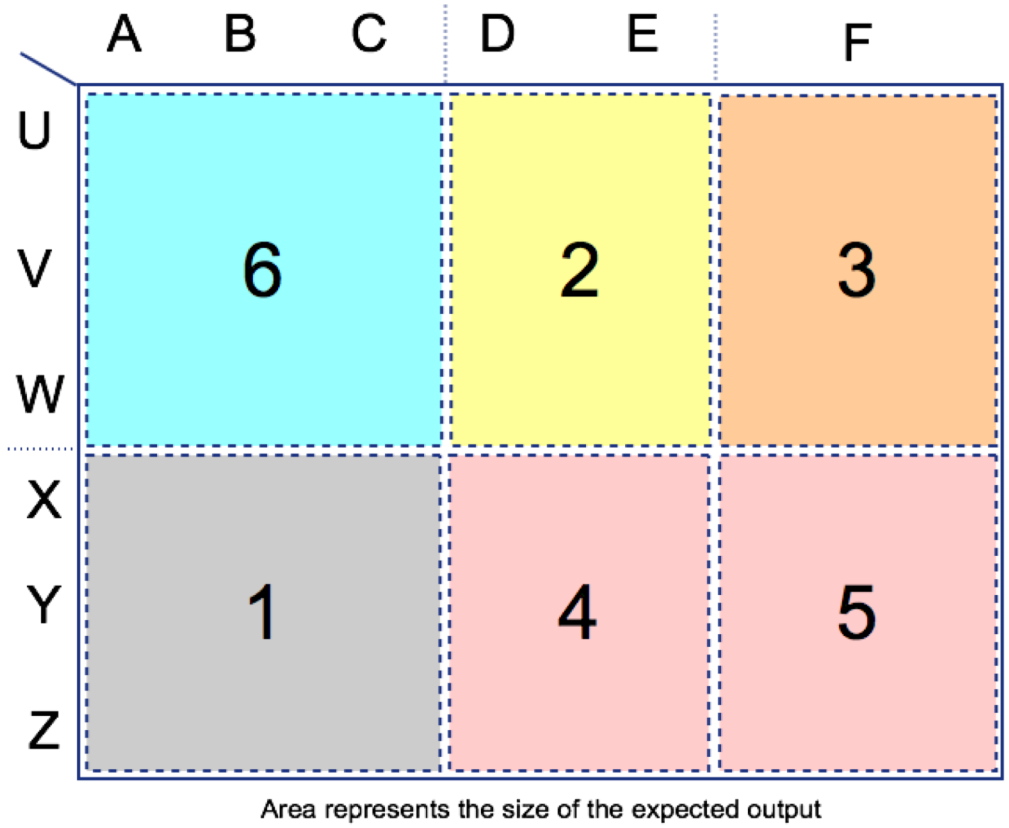
- Both events in the tuple have locality

Scheduling can pick heavy-side locality

- Both host and rack locality

Tez Shuffle has a local short-circuit path

- File permissions within the same DAG



THANKS TO THE APACHE TEZ & APACHE HIVE COMMUNITIES

BENCHMARKS

1 hour to 1 minute

```
CREATE TEMPORARY TABLE
Distance_H_H as

SELECT Postal1, Postal2,

6371 * ( 2 * asin(if(1<sqrt( pow(sin(
(latRad2 - latRad1)/2 ),2) + cos(latRad1) *
cos(latRad1) * pow(sin((longRad2-
longRad1)/2),2)),1, sqrt( pow(sin( (latRad2 -
latRad1)/2 ),2) + cos(latRad1) *
cos(latRad1) * pow(sin((longRad2-
longRad1)/2),2)))) ) as Distance

FROM v_postal_H_H

WHERE TblID1<TblID2;
```