ARESDB

A GPU-Powered Real-time Analytics Engine

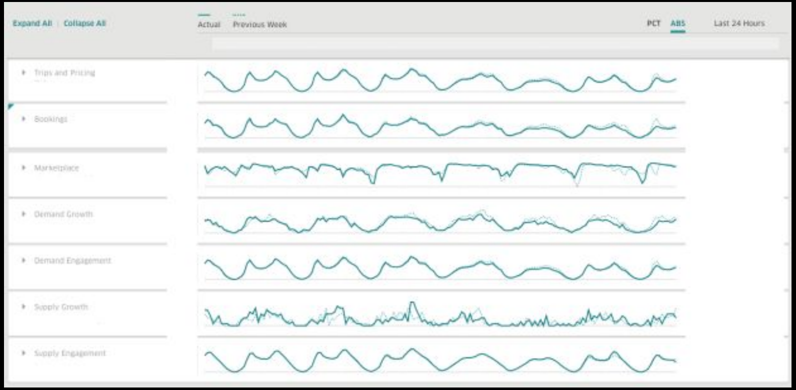
Uber

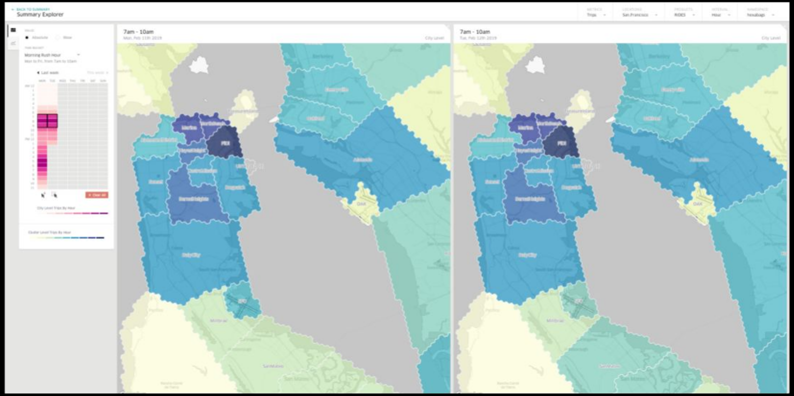
Agenda

- Real-time analytics at Uber
- Leveraging GPU for real-time analytics
- AresDB Architecture and Features
- Learnings From GPU Programming
- Future Directions

Real-time analytics at Uber

Real-time Analytics Use Cases at Uber





Rider eyeballs

Open car information

Dynamic Pricing

Real-time Analytics Use Cases Categorization

	Dashboards	Decision Systems	Ad hoc Queries
Dataset	Subset	Subset	All data
Ingestion Latency	Seconds to Minutes	Seconds to Minutes	Minutes
Query Pattern	Well known	Well known	Arbitrary
Query QPS	Medium	High	Low
Query Latency	Sub seconds	Sub seconds	Minutes
Target Users	City OPS, Executives	Engineers (application developers)	Data Scientists, Analytics, City OPS

Mission of AresDB

- ❑ sub-sec level query latency
- ❑ second to min level ingestion latency

- ❑ High availability (4 9s)
- ❑ High data accuracy (3 9s)

- ❑ Uber scale and beyond

Build a fast, reliable and scalable analytics platform solution to power Uber's Real-Time business intelligence

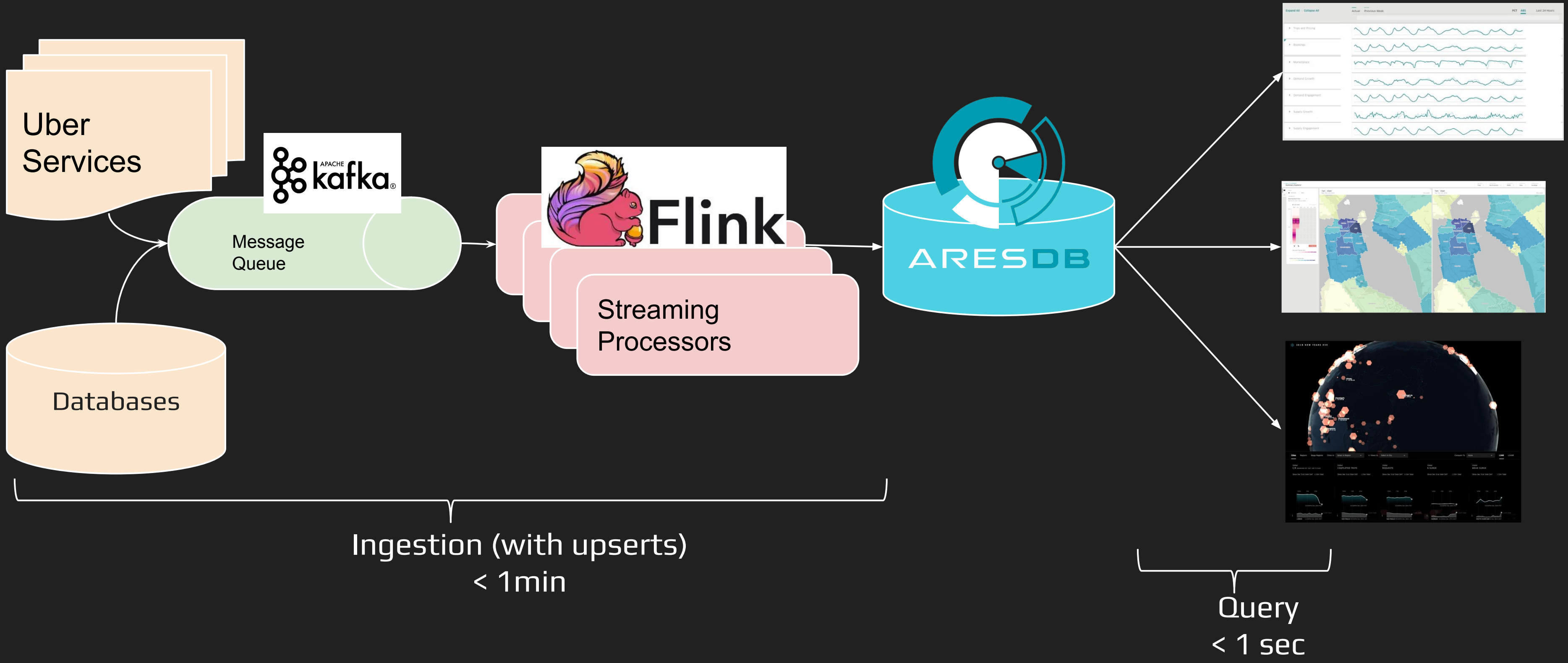
Leveraging GPU for Real-Time Analytics

AresDB: A GPU-Powered Real-time Analytics Engine

ARESDB

- High-efficiency storage
- Low-latency ingestion
- Sub-second query response time
- Feature set for real-time analytics

How is AresDB used at Uber



The Problem: Time-series Analytics

Computing measures by dimensions on time series data

request_at	city_id	fare
2017-04-13 10:25	1	15.3
2017-04-13 11:10	1	7.5
2017-04-14 10:35	1	20.1
2017-04-14 11:40	5	12.1
2017-04-14 15:45	5	5.6

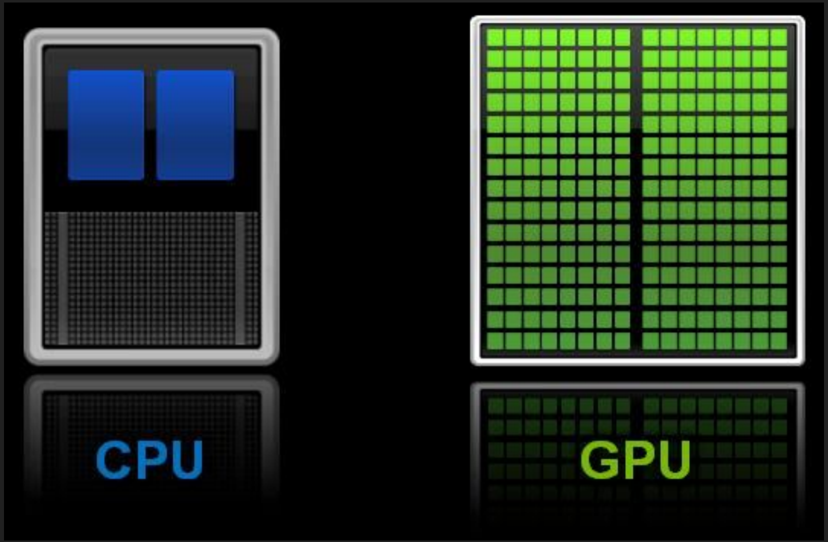
computes

day(request_at)	city_id	sum(fare)
2017-04-13	1	22.8
2017-04-14	1	20.1
2017-04-14	5	17.7

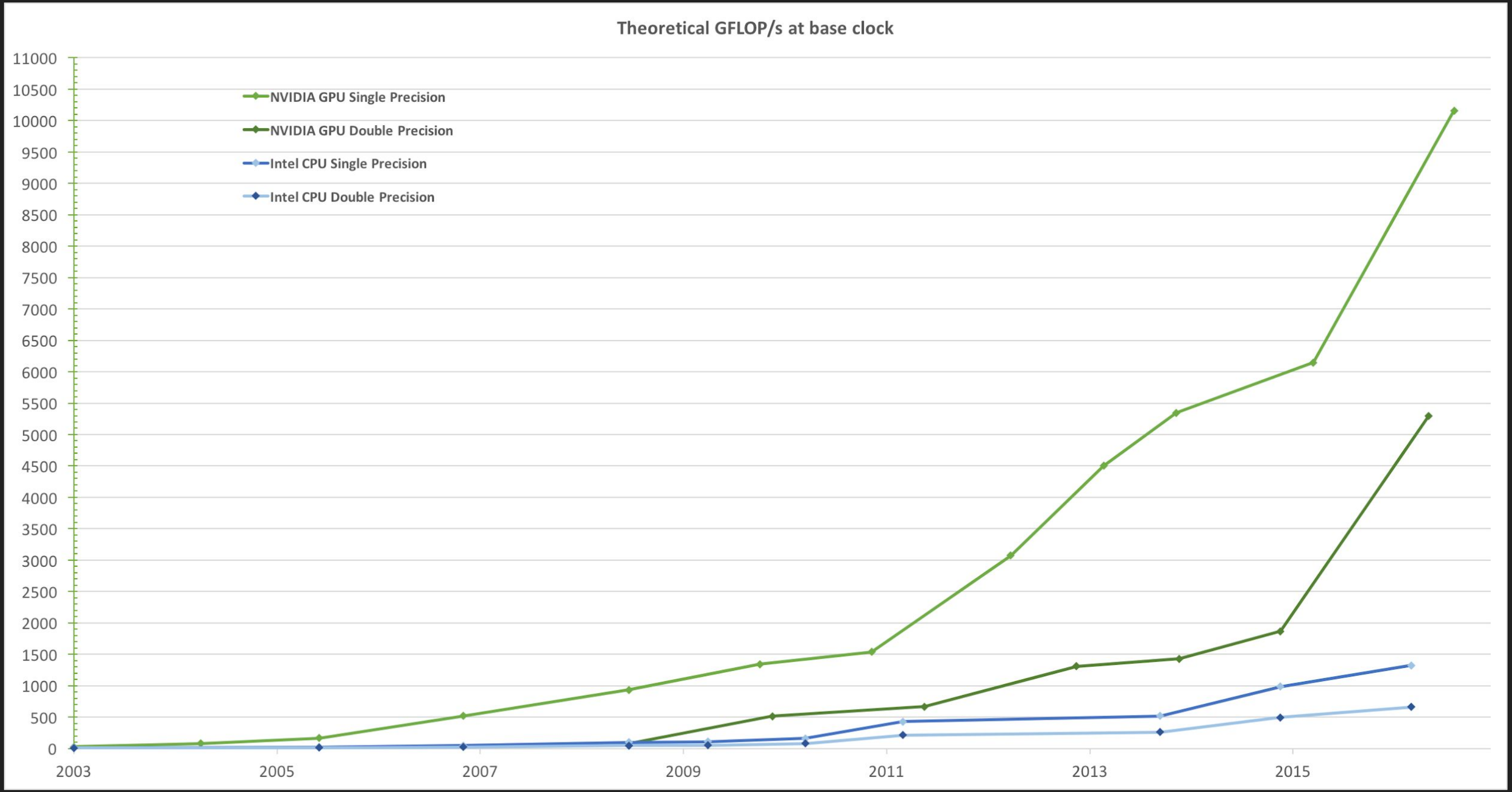
Why are GPUs well-suited ?

GPU vs CPU

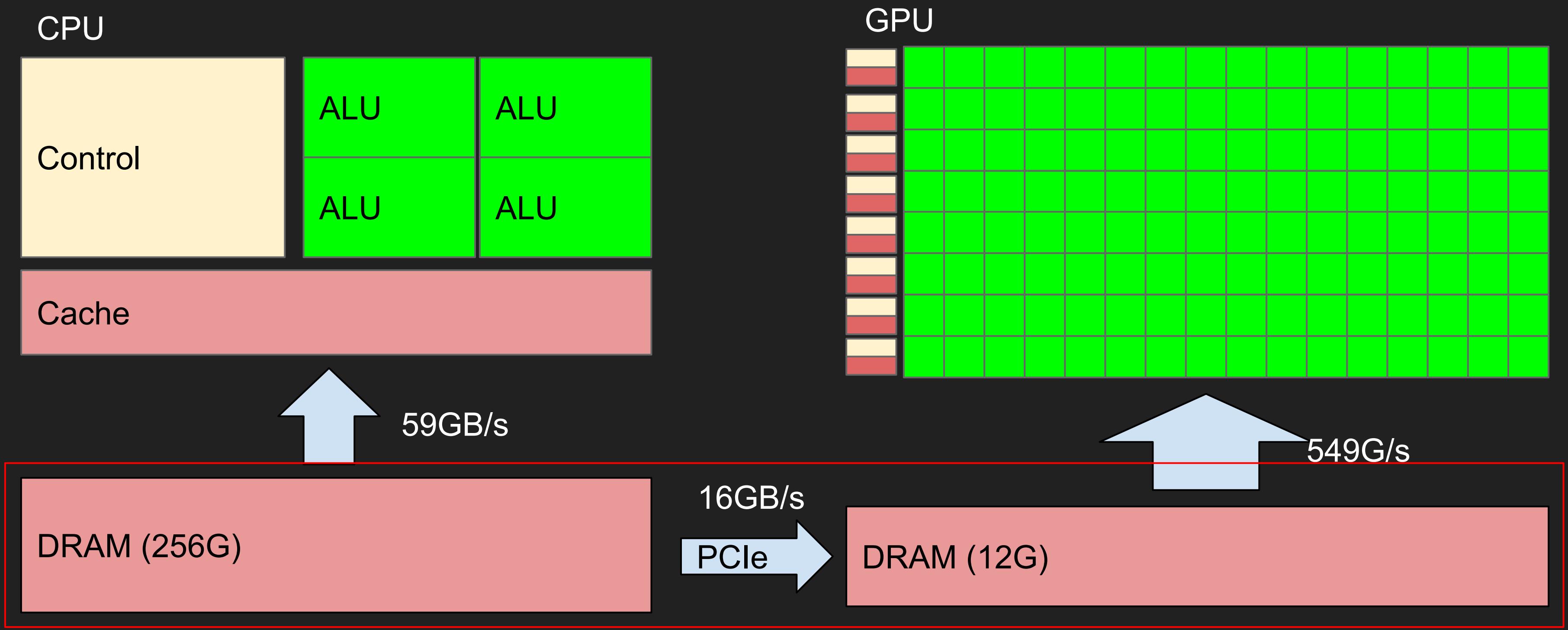
Intel® Xeon® Processor E5-2620 v3	Tesla P100
12 Threads	3584 CUDA cores
59GB/s	549G/s
500 GFIOPS	9.3 TFIOPS



GPU vs CPU


Theoretical GFLOP/s at base clock
NVIDIA GPU Single Precision
NVIDIA GPU Double Precision
Intel CPU Single Precision
Intel CPU Double Precision
11000
10500
10000
9500
9000
8500
8000
7500
7000
6500
6000
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
2003
2005
2007
2009
2011
2013
2015

Hardware Storage Choices

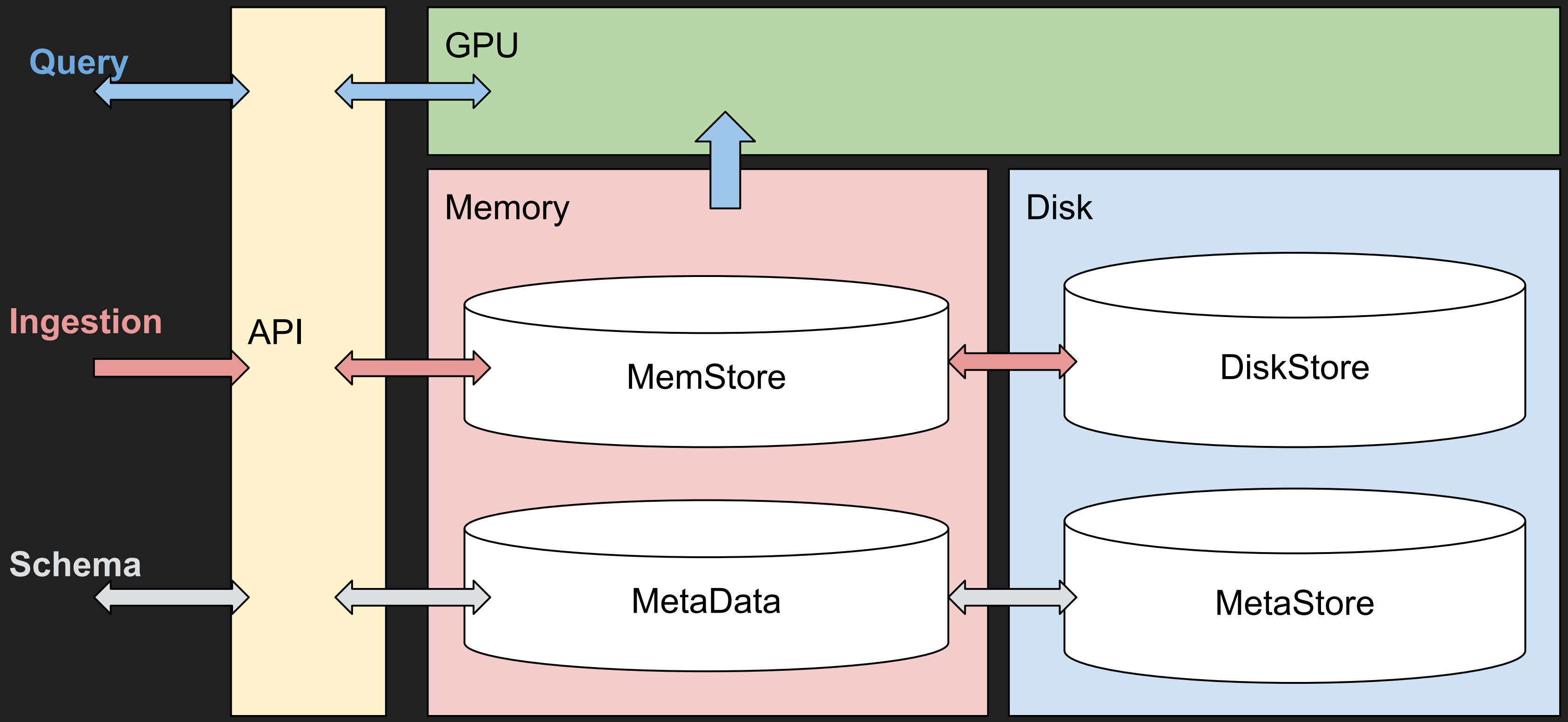


Hardware Storage Choices

	Capacity	Bandwidth to GPUs	Ingestion Method	Comment
HDD	**8TB**	**100MB/s**	File writing similar to traditional databases	Unable to feed data fast enough to fully utilize GPUs
SSD	**4TB**	**600MB/s**		
NVMe	**2TB**	**3GB/s**		And also expensive
Host Memory	**256GB**	**15GB/s** per side; **30GB/s** two sides	Memory writing	Limited by PCIe bandwidth
GPU Memory	**12GBx8**	**500GB/s** on the same GPU; **15GB/s** across GPUs	Sharding across multiple GPUs; Complex memory writing	Tight coupling of storage and computation; ingestion is challenging

AresDB Architecture and Features

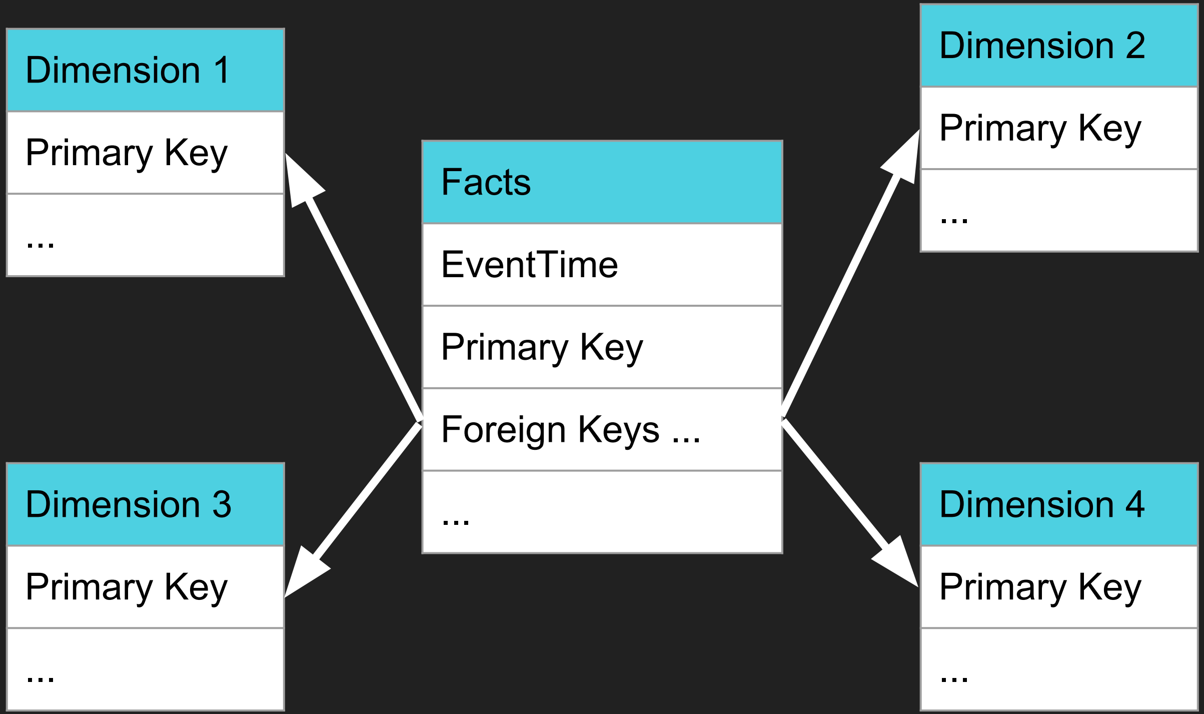
AresDB Architecture: Single Instance



Fact/Dimension Table

- Fact table
  - Facts about a business process
  - Each record associated with an event time (grows with time)
  - E.g. trips, orders, ...
- Dimension table
  - Descriptive attributes/dimensions
  - E.g. product catalogs, cities, ...

Star Schema

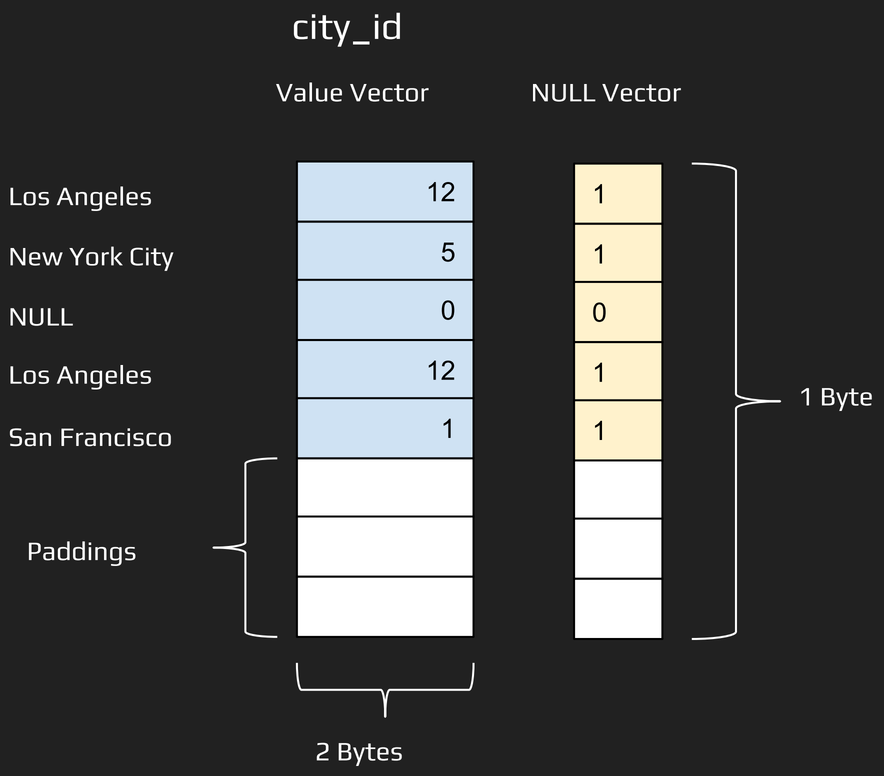


Dimension 1
Primary Key
...
Dimension 2
Primary Key
...
Facts
EventTime
Primary Key
Foreign Keys ...
...
Dimension 3
Primary Key
...
Dimension 4
Primary Key
...

Feature Highlights

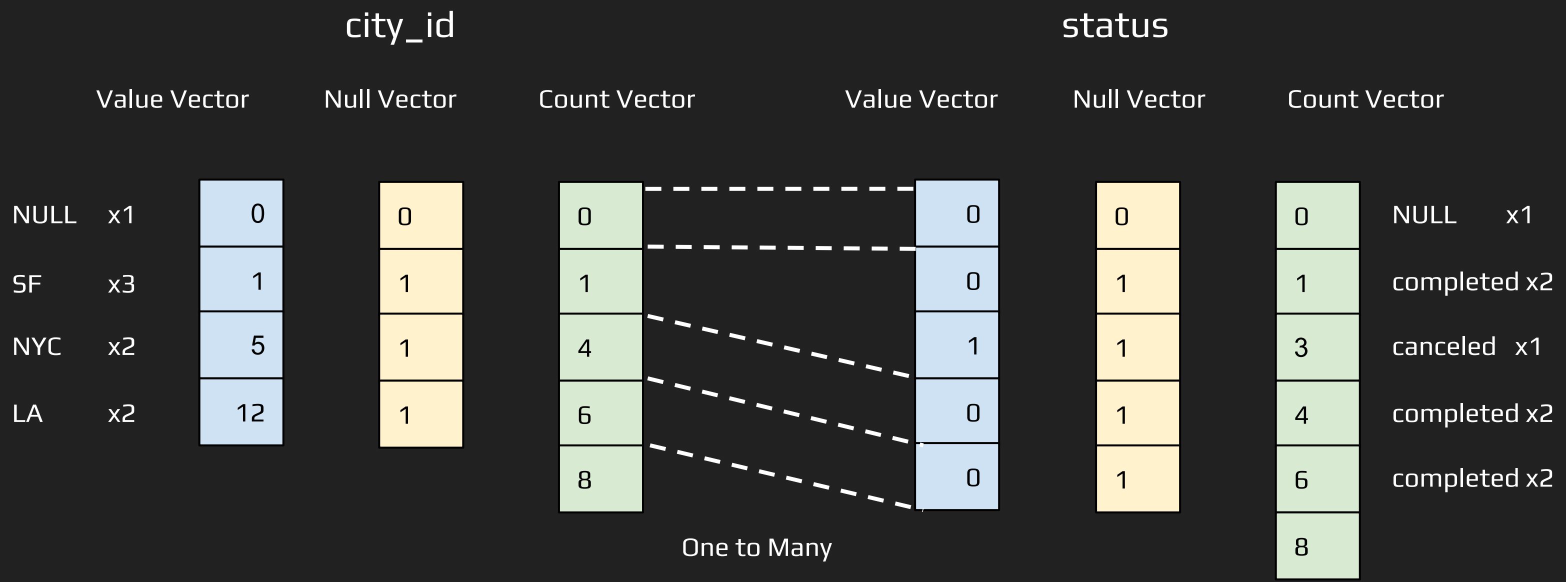
- In-Memory Columnar Storage
- Real-time upserts
- GPU powered query engine
- Analytical Query Feature Set
    - Time zone, Time Filter, Time Bucketization
    - Geospatial analytics
    - Fact/dimension table joins
    - Hyperloglog

Columnar Storage: uncompressed

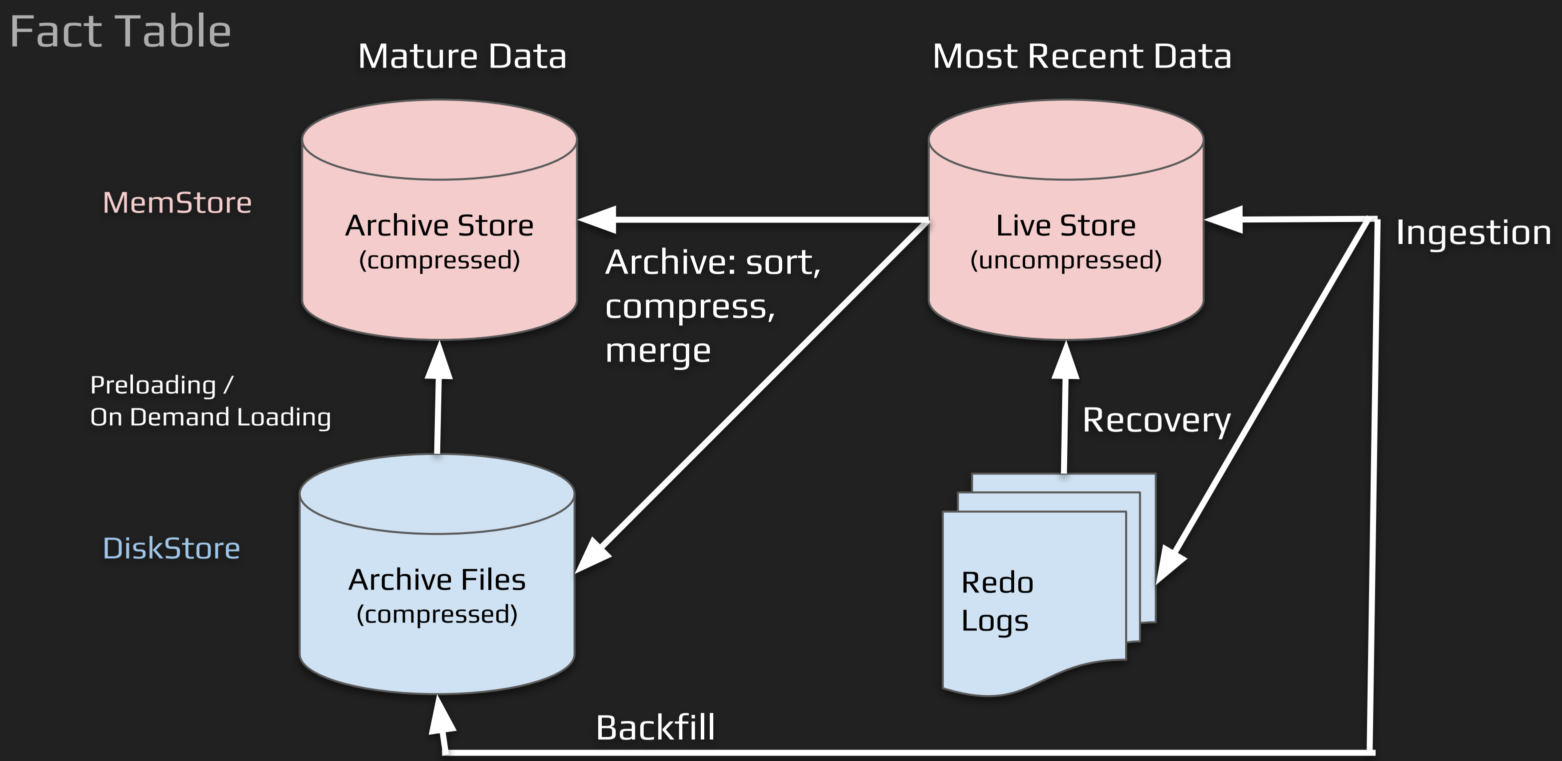


ARROW

Columnar Storage: compressed

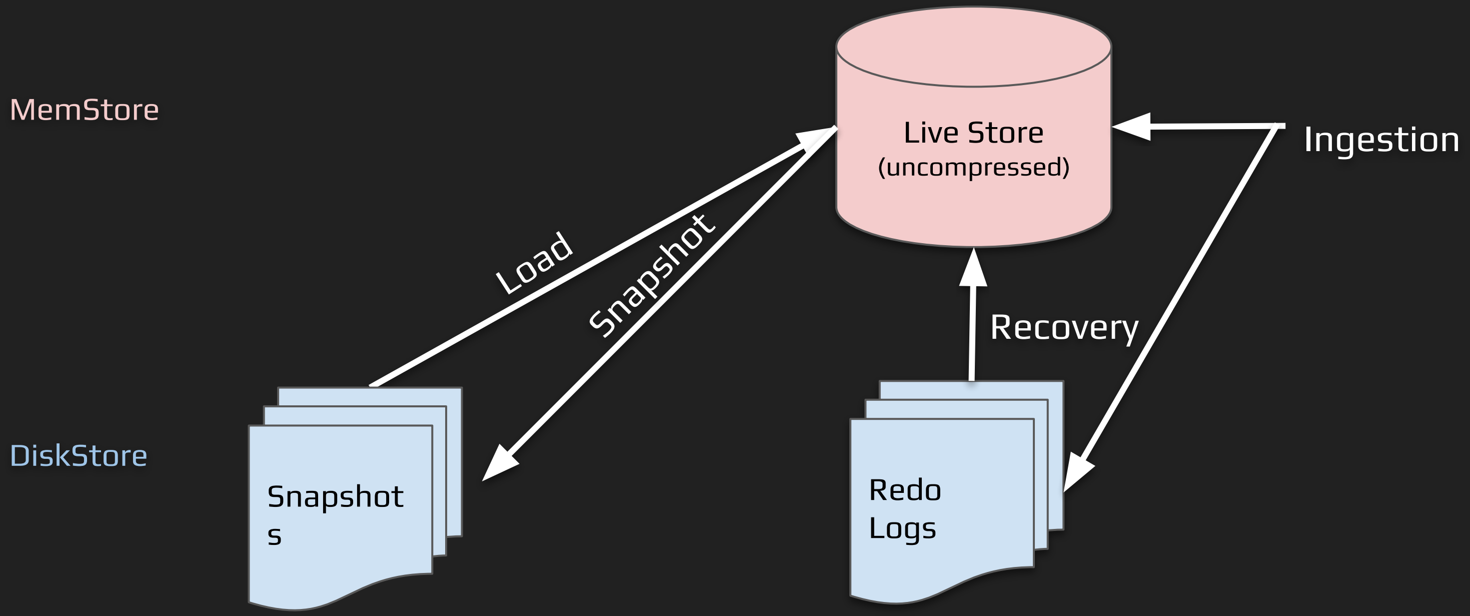


Columnar Storage: fact table



Columnar Storage: dimension table

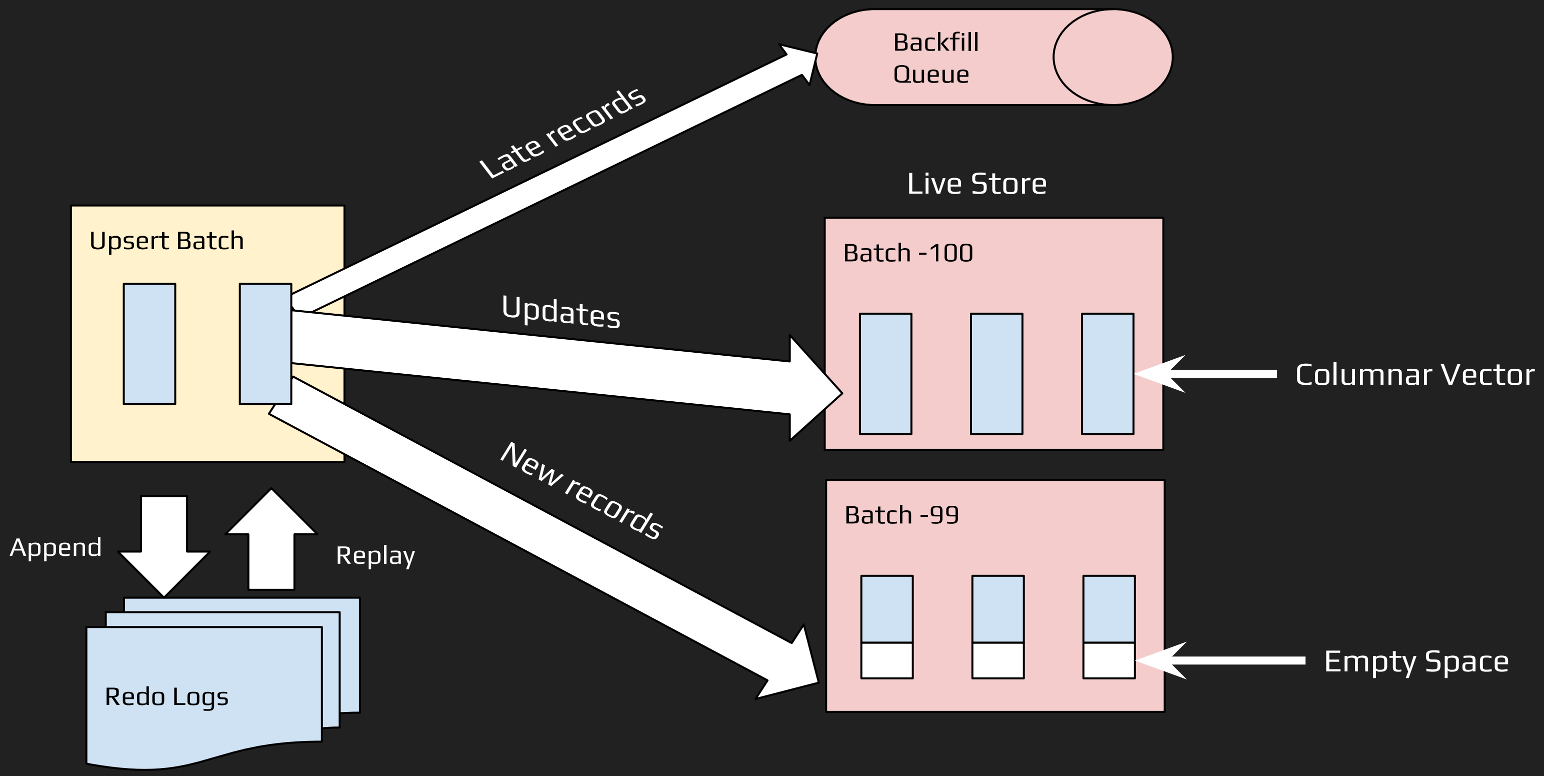
Dimension Table



Feature Highlights

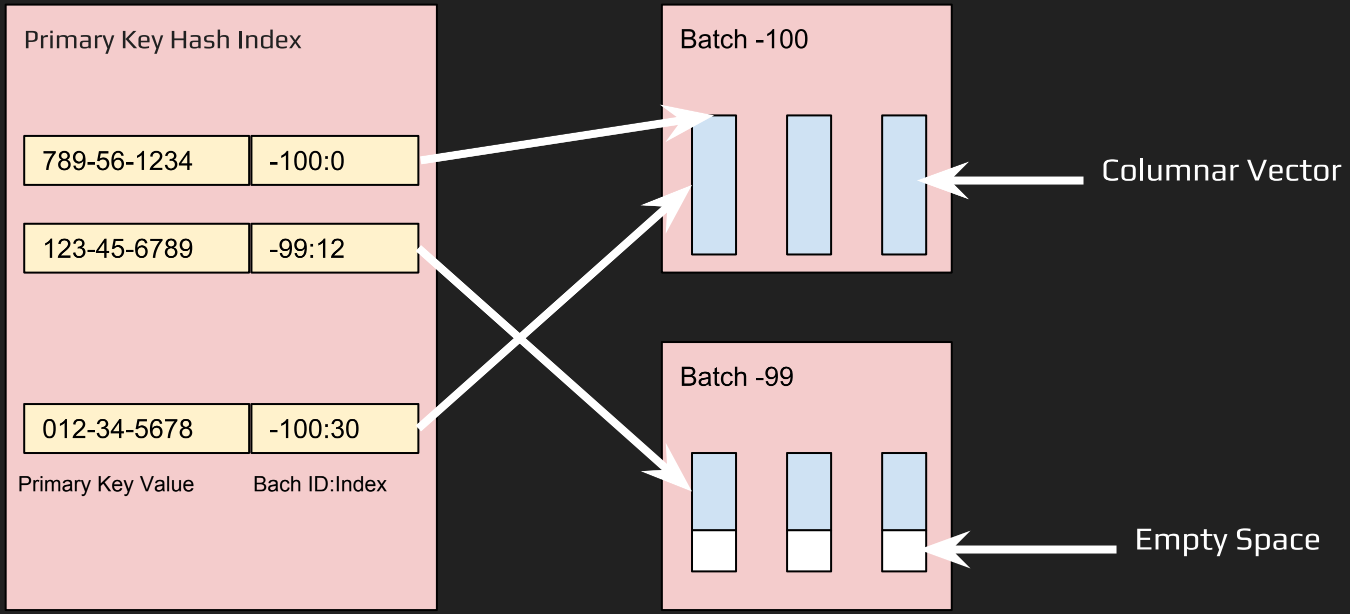
- In-Memory Columnar Storage
- Real-time upserts
- GPU powered query engine
- Analytical Query Feature Set
  - Time zone, Time Filter, Time Bucketization
  - Geospatial analytics
  - Fact/dimension table joins
  - Hyperloglog

Real-time upserts: ingestion flow

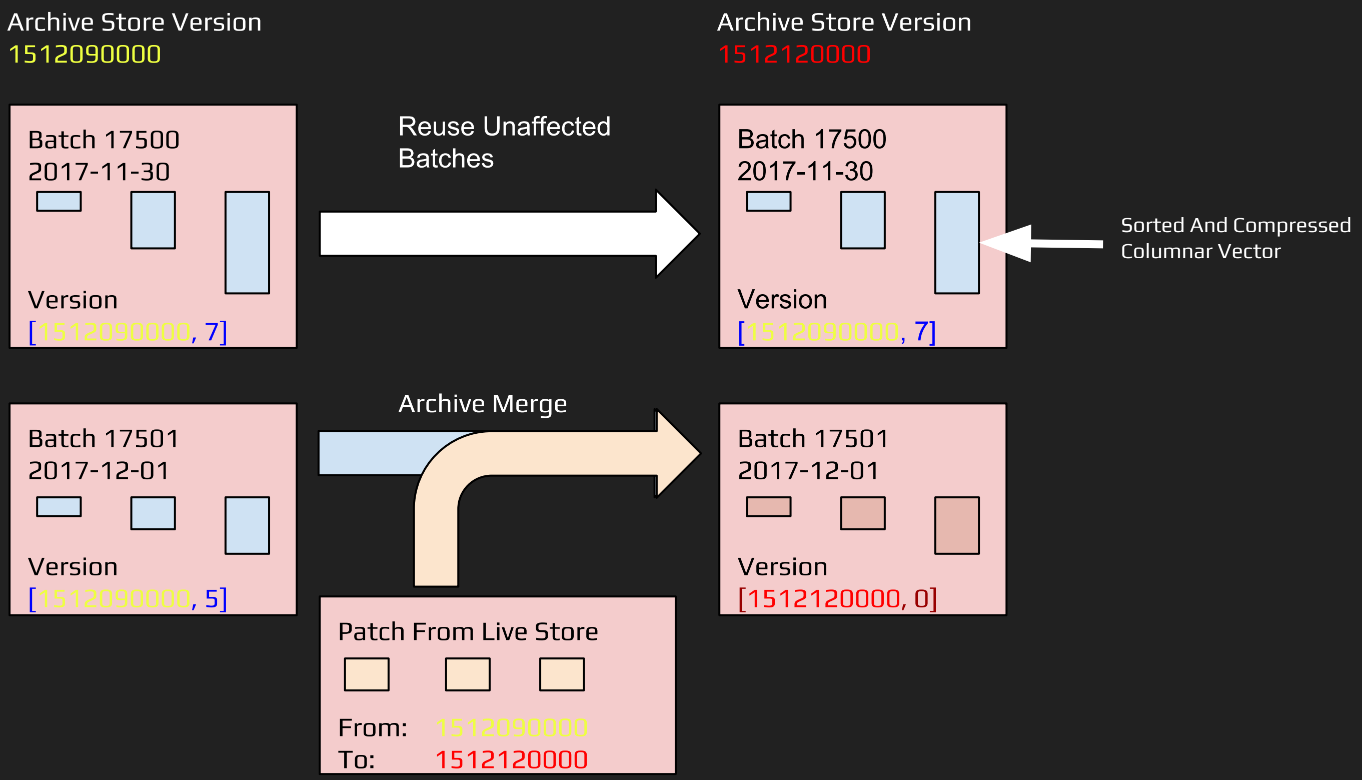


Real-time upserts: deduplication

Live Store



Real-time upserts: archiving



Real-time upserts: event timeline

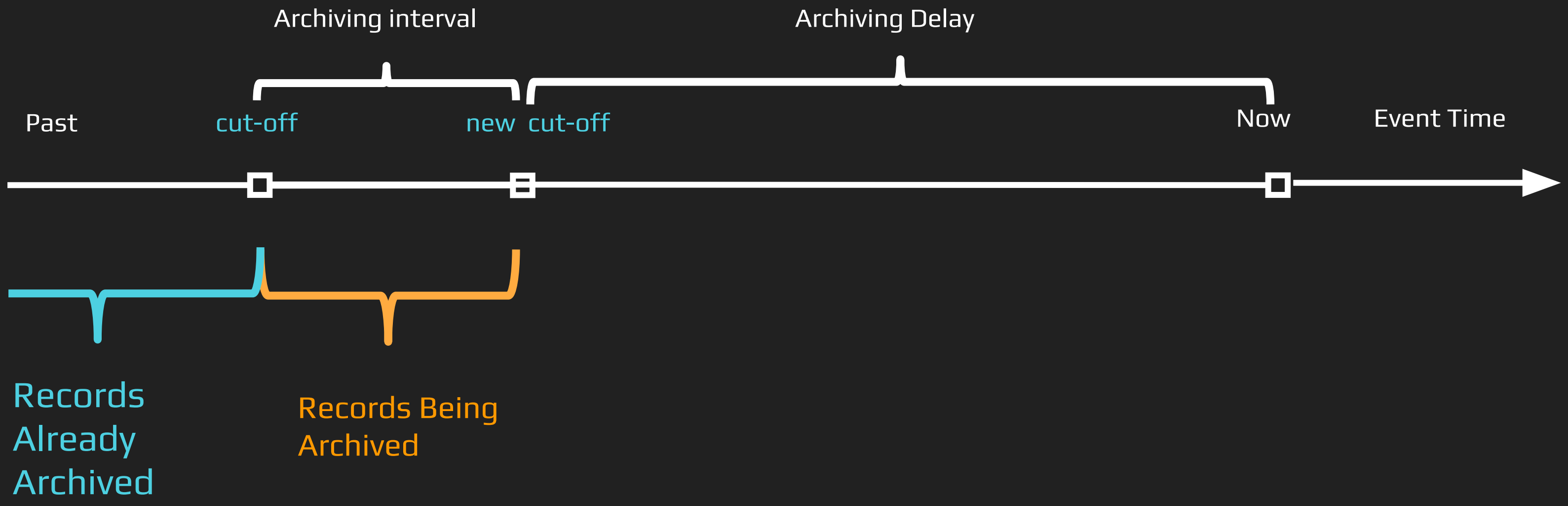
Ingestion

Real-time upserts: event timeline

Query

Real-time upserts: event timeline

Archiving



Real-time upserts: event timeline

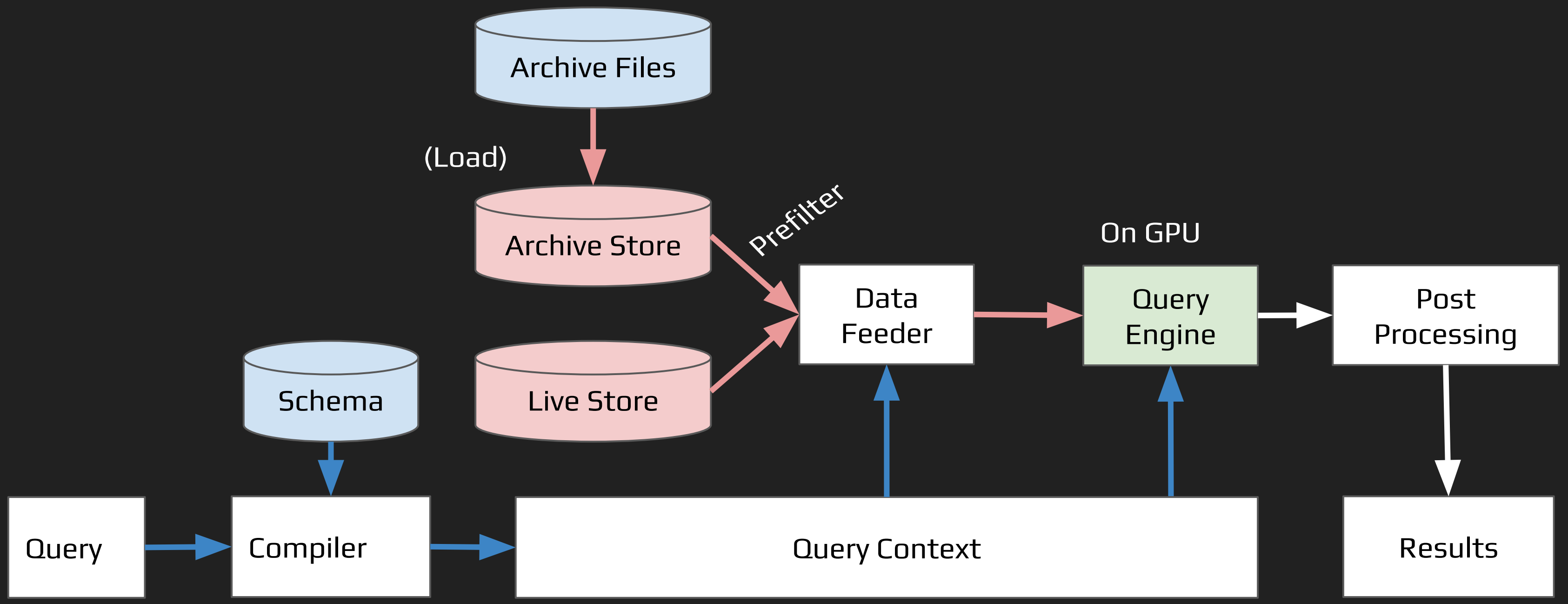
Archiving

Feature Highlights

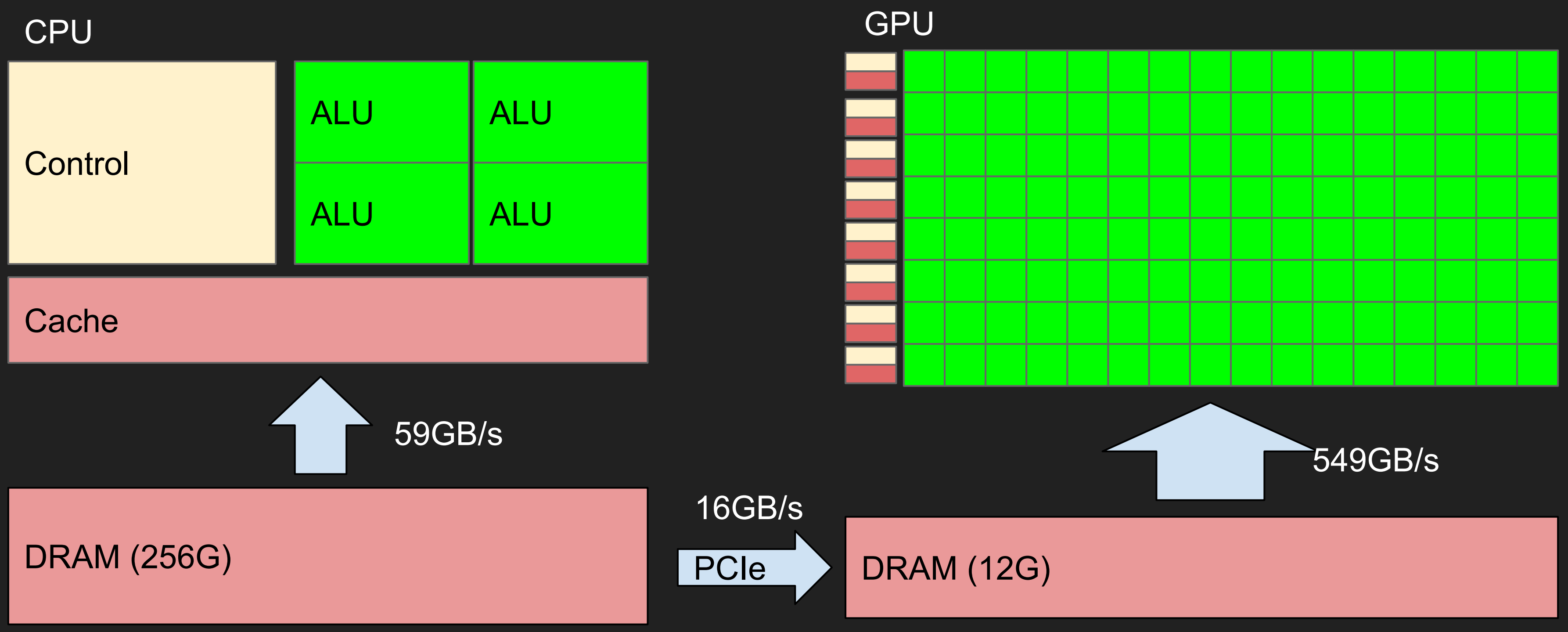
- In-Memory Columnar Storage
- Real-time upserts with deduplication
- GPU powered query engine
- Analytical Query Feature Set
  - Time zone, Time Filter, Time Bucketization
  - Geospatial analytics
  - Fact/dimension table joins
  - Hyperloglog

Query Engine

High level architecture

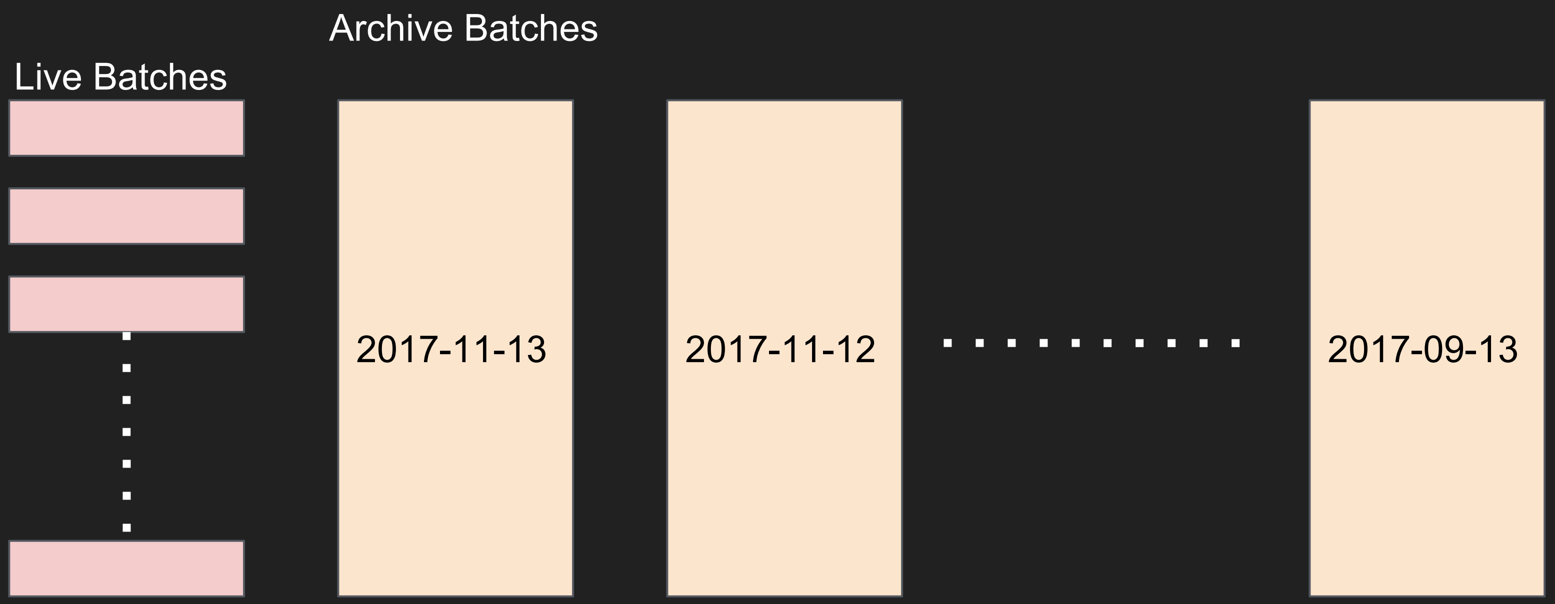


Data Feeding



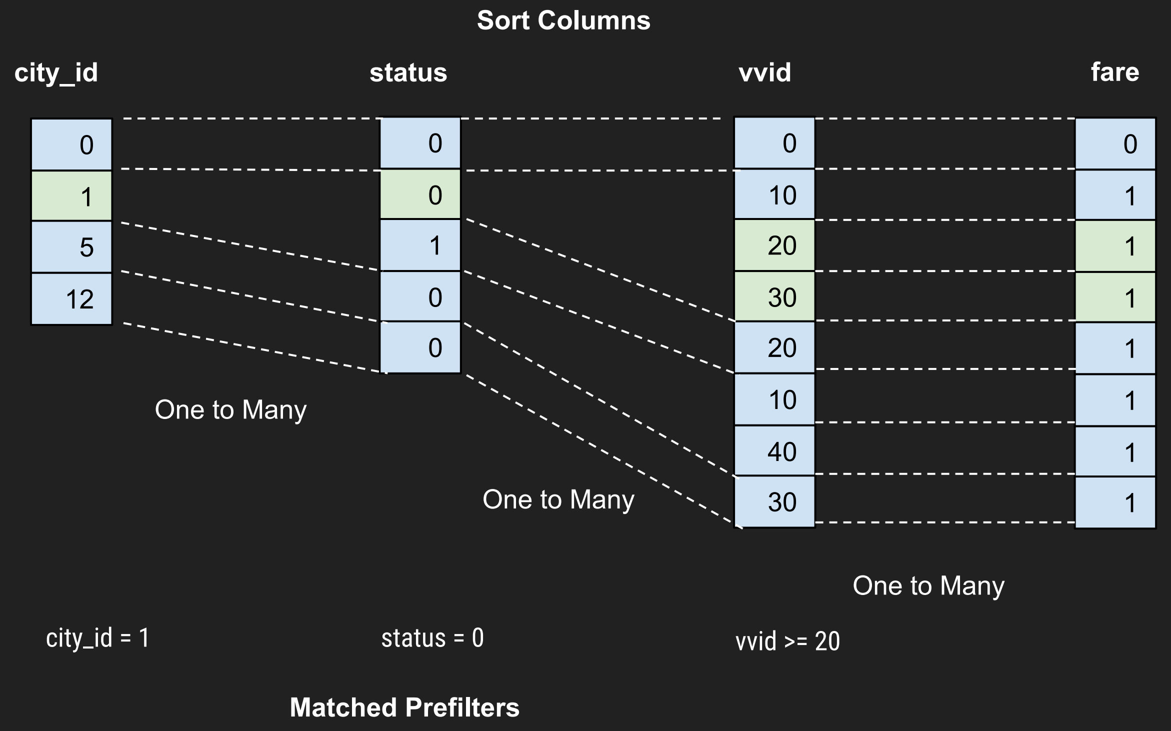
Data Feeding

Partitioned Data



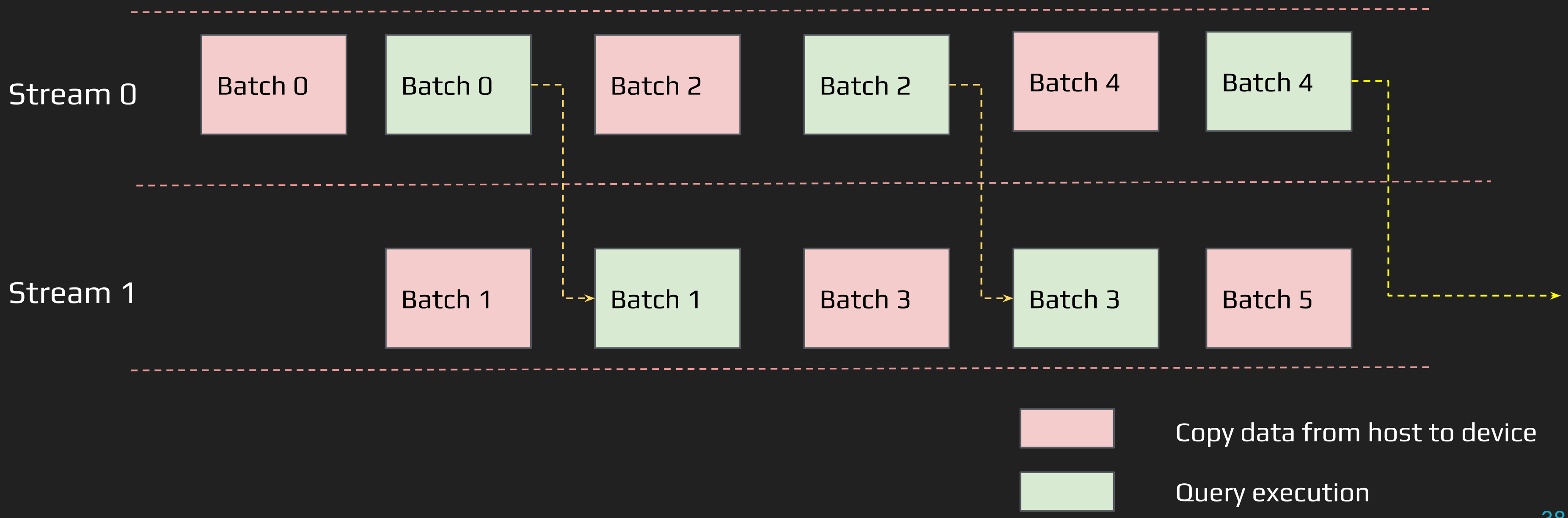
Data Feeding

Prefilter



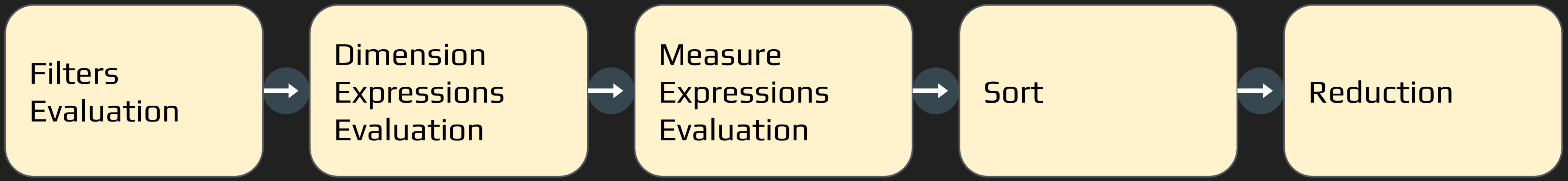
Data Feeding

Pipelining



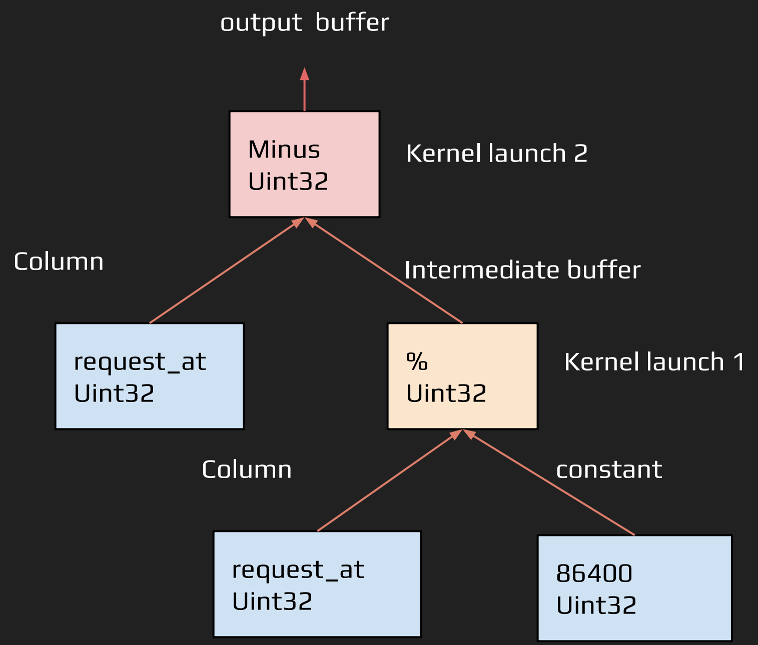
Query Execution

Execution Stages



Query Execution

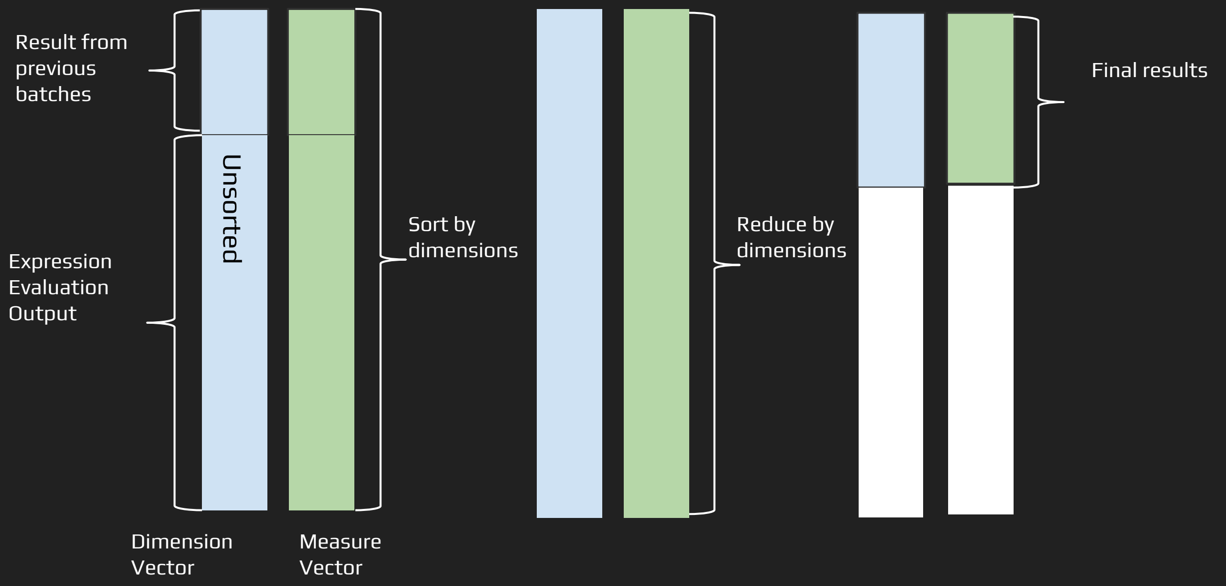
Expression Evaluation



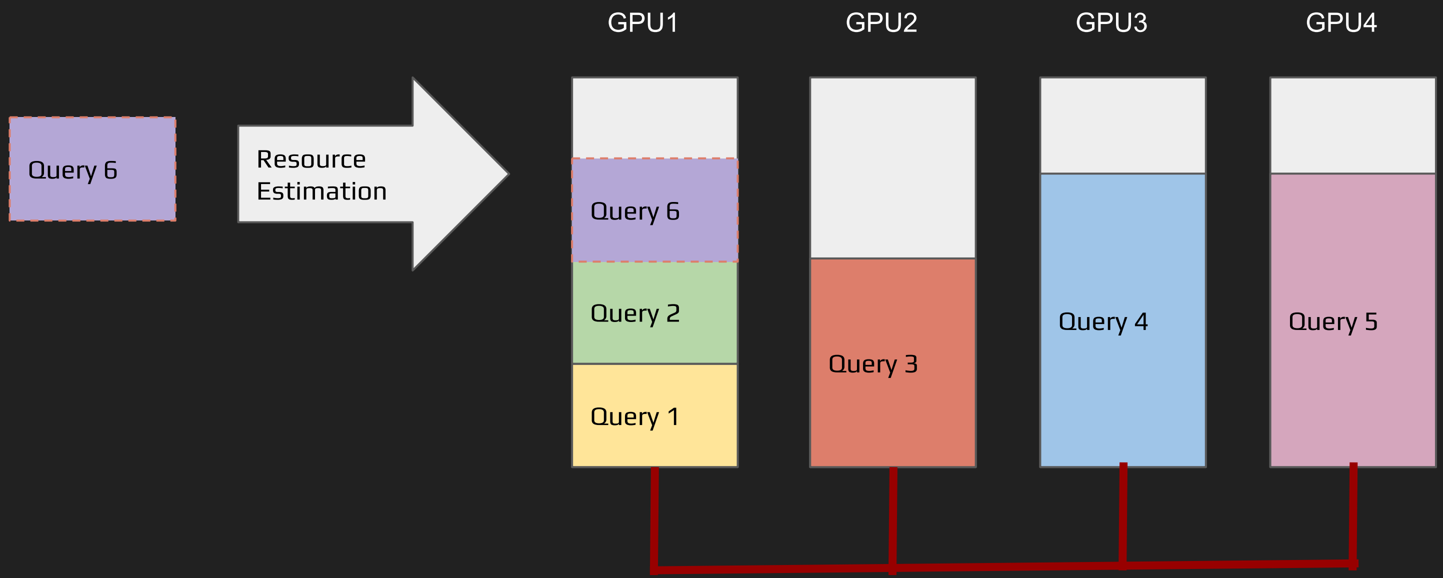
- One operator per kernel on non-leaf nodes
- Each leaf node is one of
  - column/constant
- Non-root, non-leaf node
  - kernel launch
  - output to intermediate buffer
- Root node
  - Kernel launch
  - Write to output buffer
- E.g., request_at - request_at % 86400

Query Execution

Aggregation (Sort and Reduction)



Device Resource Management



Feature Highlights

- In-Memory Columnar Storage
- Real-time upserts with deduplication
- GPU powered query engine
- Feature set for analytical queries
  - Time zone, Time Filter, Time Bucketization
  - Fact/dimension table joins
  - Geospatial analytics
  - Hyperloglog

Timezone, Time Filter, Time Bucketization

```sql
SELECT   Count(*),
Unix_timestamp(Convert_tz(Concat(Date_format(Convert_tz(From_unixtime(((driver_info.first_active_at) -
(driver_info.first_active_at)  % 900000) /
1000),'GMT', 'America/Los_Angeles'),'%Y-%m-%d
%H:'),
Lpad(15*Floor(Minute(Convert_tz(From_unixtime(((driver_info.first_active_at) -(driver_info.first_active_at)
%900000)/1000),'GMT','America/Los_Angeles'))/15),
2,'0')),'America/Los_Angeles','UTC')) AS
time_dimension, driver_info.flow_type
FROM driver_info
WHERE driver_info.first_active_at >=
1534810500000
AND    driver_info.first_active_at <
1534813200000
GROUP BY 2,3
```

V.S.

```json
{
  "table": "driver_info",
  "measures": [
    {
      "sqlExpression": "count(*)"
    }
  ],
  "dimensions": [
    {
      "alias": "ts",
      "sqlExpression": "first_active_at",
      "timeBucketizer": "day"
    },
    {
      "sqlExpression": "flow_type"
    }
  ],
  "timeFilter": {
    "column": "first_active_at",
    "from": "7 days ago"
  },
  "timezone": "America/Los_Angeles"
}
```

Analytical Query Features

- Fact/Dimension Table Join
  - E.g. trips.city_id = cities.id
- Hyperloglog Cardinality Estimation
  - countDistinctHLL(driver_id)
  - Dedicated hll column
- Geospatial analytics
  - GeoPoint, GeoShape
  - GeoIntersect(point, shape)

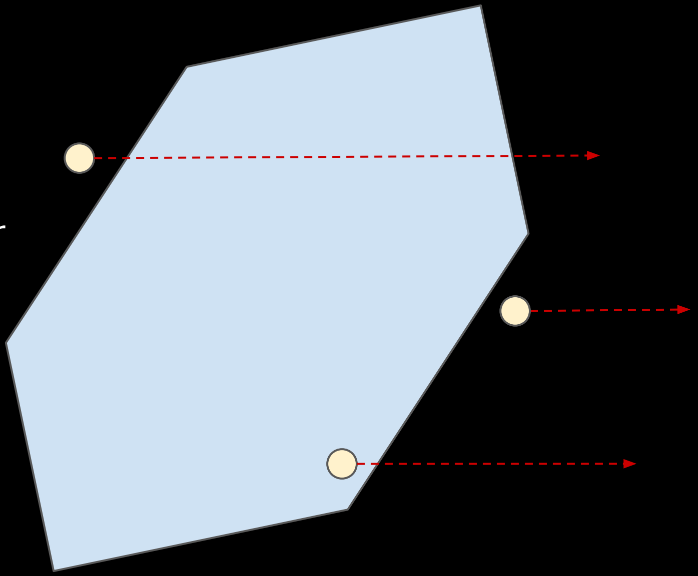
Learnings from GPU Programming

Learnings from GPU Programming

- Maximize parallelism
- Optimize memory access
- Maximize arithmetic intensity
- Reduce data transfer between GPU/CPU
- Profile, profile, profile

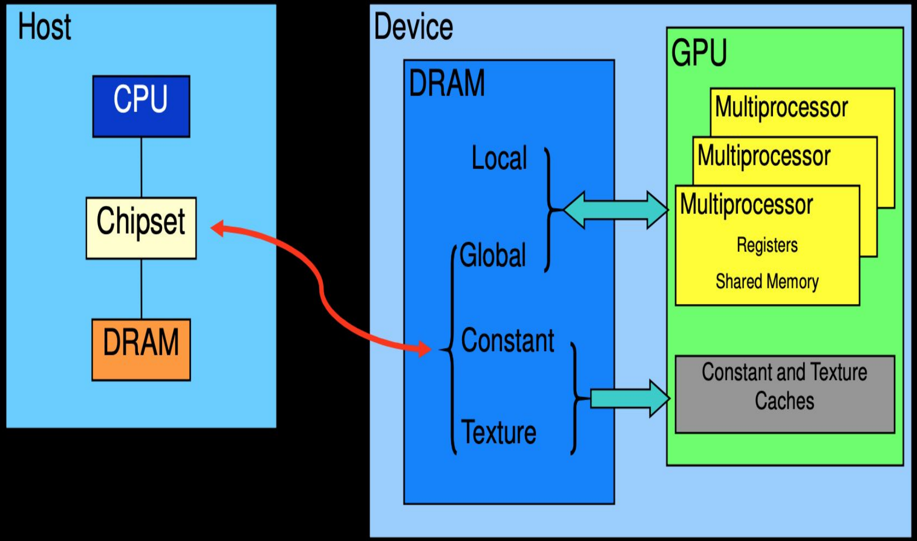
Maximize Parallelism

- Partition your computation to keep the GPU multiprocessors equally busy
  - Many threads, many thread blocks
  - E.g. Inclusion test for a point and polygons
    - One shape per thread vs one edge per thread
- Keep resource usage low enough to maximize occupancy
  - Register, shared memory
  - Careful design of data structure
    - Use less wide data type
      - Int64 -> uint32
    - Reuse memory space
      - Union
    - Passing offsets instead of pointers



Optimize Memory Access

- Coalesced vs. non-coalesced = order of magnitude
  Global/Local device memory
- Shared memory
- Constant memory



http://developer.download.nvidia.com/CUDA/training/NVIDIA_GPU_Computing_Webinars_CUDA_Optimization_April-2009.pdf
retrived date: 01/10/2019

Maximize Arithmetic Intensity

- GPU spends its transistors on ALUs, not memory
- Sometimes it's better to recompute than to cache
- Do more computation on GPU instead of transferring back to CPU

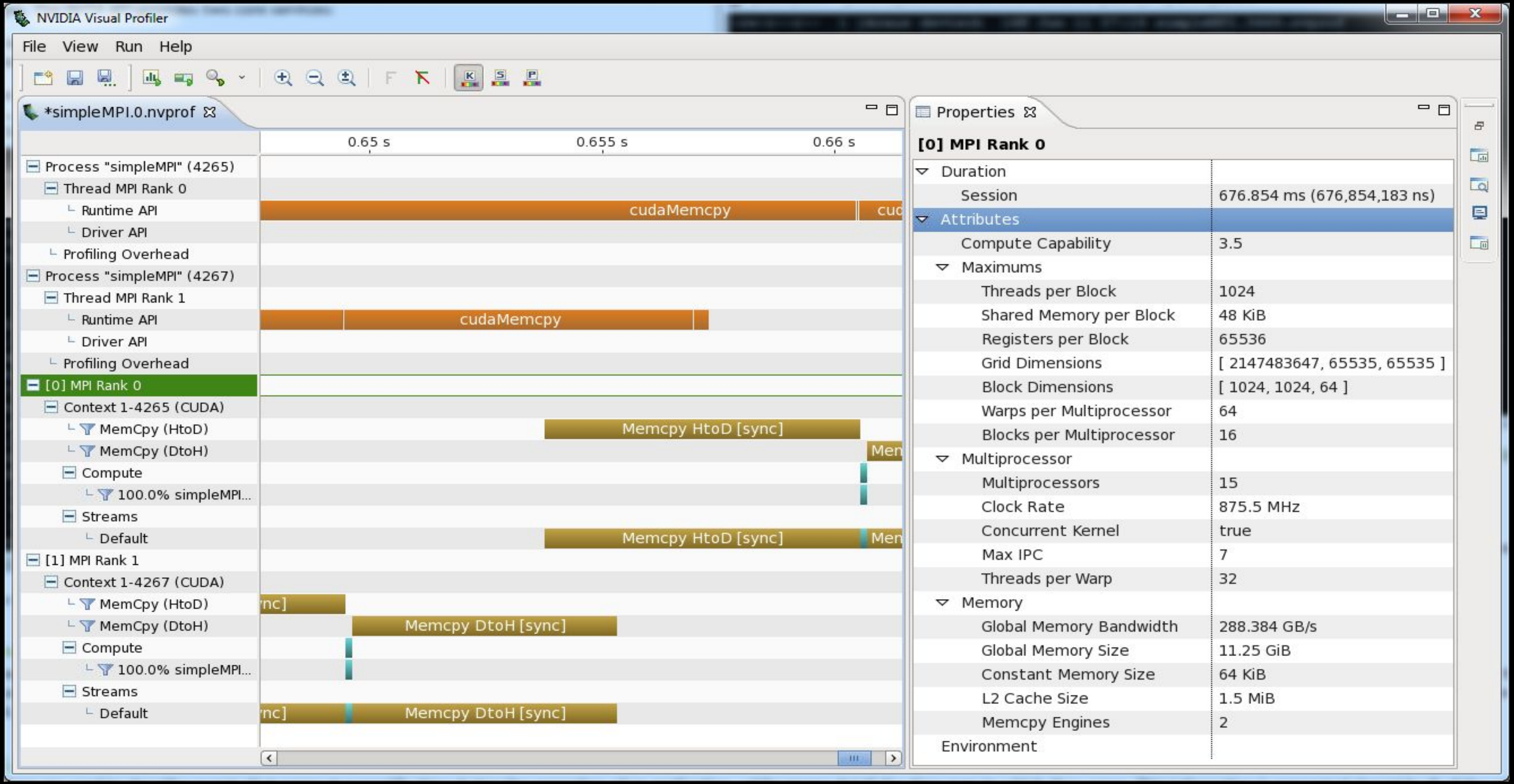
Minimize CPU/GPU Transfers

- Group transfers
- Overlapping data transfers and computation
  - Async and stream api
  - Stream = sequence of operations that execute in order on GPU
  - Pipeline execution
- Pinned memory vs. pageable memory

Profiling GPU Program NVVP

- Nvidia visual profiler
- Unified CPU/GPU timeline
- Automated performance analysis
- Guided application analysis

NVVP cont'd



NVIDIA Visual Profiler
File View Run Help
*simpleMPI.0.nvprof
0.65 s
0.655 s
0.66 s
Process "simpleMPI" (4265)
Thread MPI Rank 0
Runtime API
Driver API
Profiling Overhead
Process "simpleMPI" (4267)
Thread MPI Rank 1
Runtime API
Driver API
Profiling Overhead
[0] MPI Rank 0
Context 1-4265 (CUDA)
MemCpy (HtoD)
MemCpy (DtoH)
Compute
100.0% simpleMPI...
Streams
Default
[1] MPI Rank 1
Context 1-4267 (CUDA)
MemCpy (HtoD)
MemCpy (DtoH)
Compute
100.0% simpleMPI...
Streams
Default
cudaMemcpy
cudaMemcpy
Memcpy HtoD [sync]
Memcpy HtoD [sync]
Memcpy DtoH [sync]
Memcpy DtoH [sync]
Properties
[0] MPI Rank 0
Duration
Session 676.854 ms (676,854,183 ns)
Attributes
Compute Capability 3.5
Maximums
Threads per Block 1024
Shared Memory per Block 48 KiB
Registers per Block 65536
Grid Dimensions [2147483647, 65535, 65535]
Block Dimensions [1024, 1024, 64]
Warps per Multiprocessor 64
Blocks per Multiprocessor 16
Multiprocessor
Multiprocessors 15
Clock Rate 875.5 MHz
Concurrent Kernel true
Max IPC 7
Threads per Warp 32
Memory
Global Memory Bandwidth 288.384 GB/s
Global Memory Size 11.25 GiB
Constant Memory Size 64 KiB
L2 Cache Size 1.5 MiB
Memcpy Engines 2
Environment

[1]

Future Directions

- Beyond single instance
  - Sharding
  - Replication
- Ease of adoption
  - SQL interface
  - Native Kafka support
- More query features (eg. fact to fact table joins)
- Query engine optimizations (eg. GPU memory caching)
- Grow AresDB together with the community

Questions ?

Tech blog: https://eng.uber.com/aresdb/

Git repo: https://github.com/uber/aresdb

Questions: email uberopen@uber.com

Follow our Facebook page:

www.facebook.com/uberopensource

Thank you

Questions: email uberopen@uber.com

Follow our Facebook page: www.facebook.com/uberopensource