BigDAWG Polystore: programmer productivity for complex, heterogeneous big data applications

Tim Mattson, Intel labs

timothy.g.mattson@intel.com

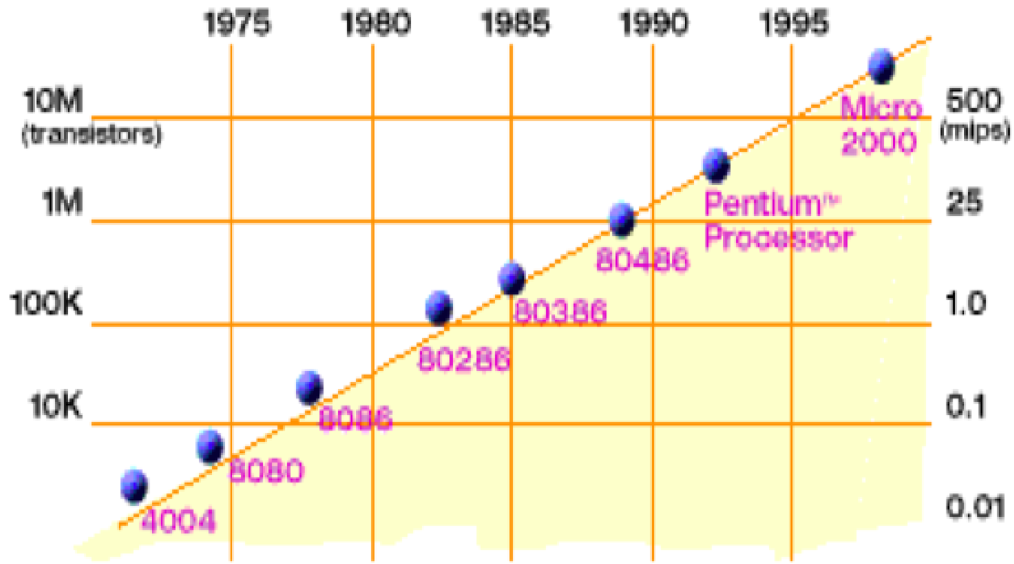
Intel-PI for the Big Data “Intel Science and Technology Center”

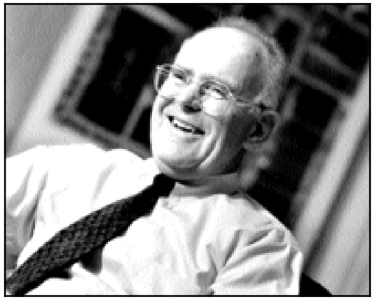
http://istc-bigdata.org/

With help from Vijay Gadepally (MIT LL), Zuohao She (Northwestern), & Adam Dziedzic (U Chicago)

I work at Intel …

Moore's Law





- In 1965, Intel co-founder Gordon Moore predicted (from just 3 data points!) that semiconductor density would double every 18 months.
  - ***He was right!*** Over the last 50 years, transistor densities have increased as he predicted.

Slide source: UCB CS 194 Fall'2010

Every Intel talk is required to have a Moore's law slide and a …

Legal Disclaimer & Optimization Notice

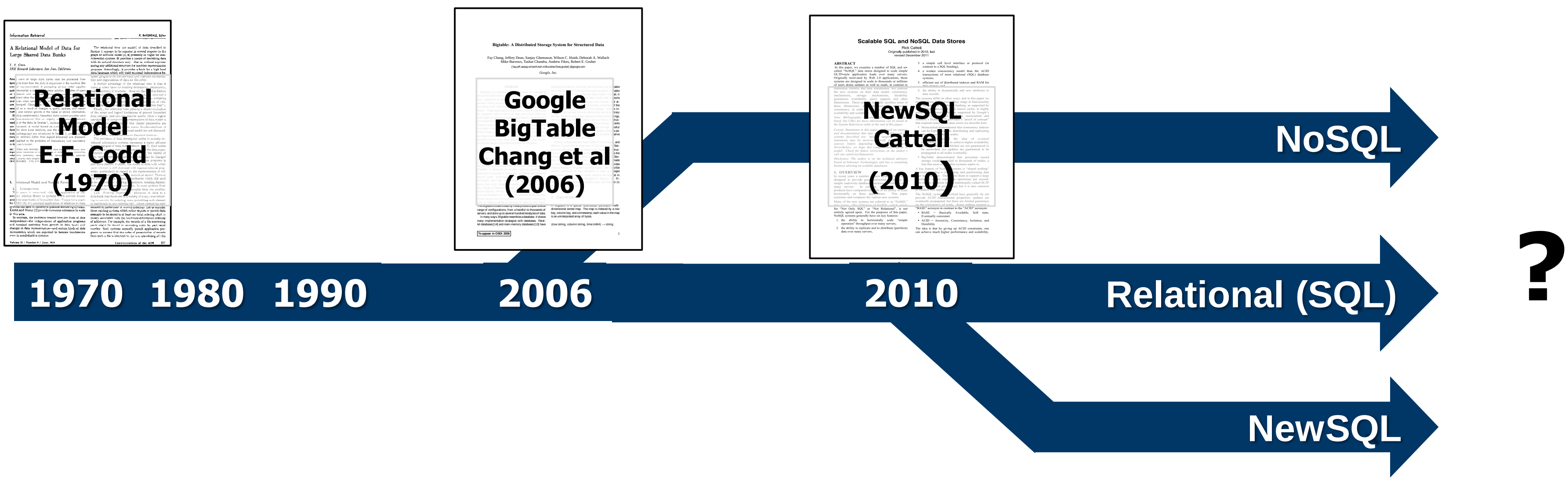
Acknowledgements: The BigDAWG Teams

- Overall BigDAWG vision and leadership
  - Mike Stonebraker[2], Sam Madden[2], and Tim Mattson[1]
- System integration and Implementation leadership
  - **Vijay Gadepally[2]**, Jennie Duggan[5] and Aaron Elmore[6]
- BigDAWG Monitoring Framework
  - Peinan Chen[2]
- BigDAWG Data Migration
  - **Adam Dziedzic**[6], Aaron Elmore[6]
- BigDAWG Executor
  - Ankush Gupta[2]
- BigDAWG Query Optimization
  - **Zuohao She[5]**, Surabhi Ravishankar[5], and Jennie Duggan[5]
- S-Store
  - John Meehan[3], S. Zdonik[3], Shaobo Tian[3], Yulong Tian[3], Nesime Tatbul[1], A. Elmore[6], **Adam Dziedzic[6]**
- Myria
  - Magdalena Balazinska[4] and Bill Howe[4]

With special thanks for slides and generous support from:
- Vijay Gadepally
- Zuohao She
- Adam Dziedzic

Three Eras of Database Technology

SQL Era	NoSQL Era	NewSQL Era	Future
Common interface	**Rapid ingest for internet search**	**Fast analytics inside databases**	



ORACLE PostgreSQL accumulo SciDB GRAPHULO

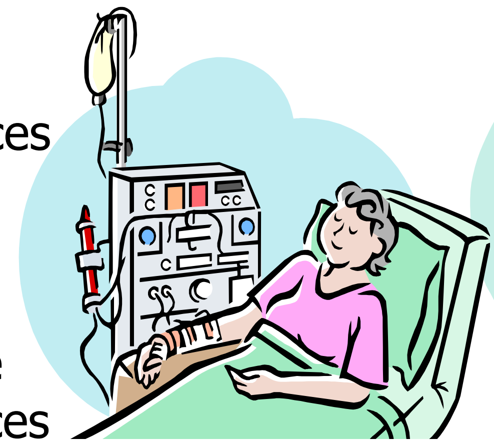
SQL = Structured Query Language
NoSQL = Not only SQL

Source: The BigDAWG Polystore System and Architecture, HPEC'2016, Vijay Gadepally

Big Data in the Real World

- Consider patient data in an Intensive Care Unit (e.g. MIMIC II data set*)

- EKG traces
- Blood oxygen
- Blood pressure
- EEG traces



- Demographic
- Caregiver notes
- Medical charts
- Lab test results
- Xray, MRI, etc.

The challenge … apply predictive analytics across all data … so we can show up to restart a heart before it stops beating!!!

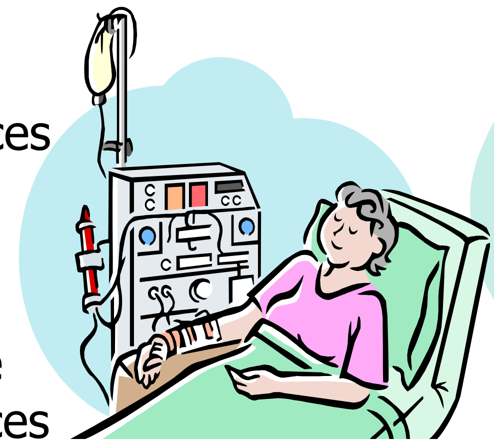
* MIMIC: Multiparameter Intelligent Monitoring in Intensive Care, http://www.physionet.org/mimic2/

Big Data in the Real World

Messy, heterogeneous, complex, streaming …

- Consider patient data in an Intensive Care Unit (e.g. MIMIC II data set*)

Arrays	EKG traces
Time Series	Blood oxygen
Time Series	Blood pressure
Arrays	EEG traces



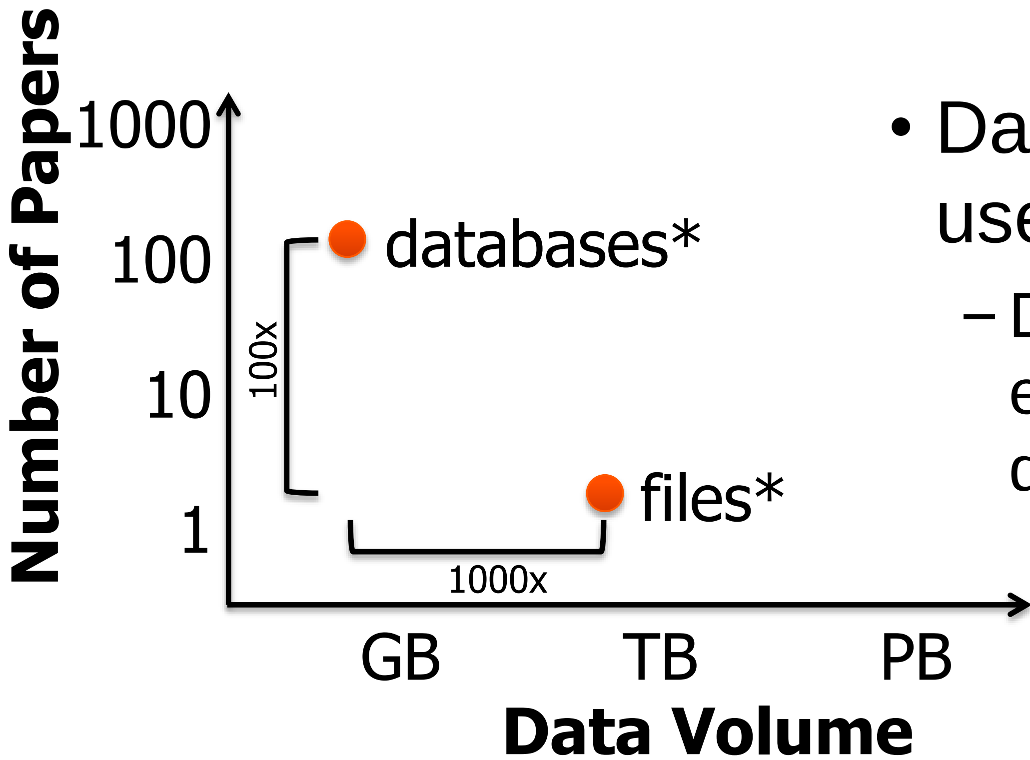
Demographic	tables
Caregiver notes	documents
Medical charts	tables
Lab test results	tables
Xray, MRI, etc.	#images

Time series and tabular data are stored in a DBMS.
Other data? Flat files

* MIMIC: Multiparameter Intelligent Monitoring in Intensive Care, http://www.physionet.org/mimic2/

MIMC doesn't include images. We are talking to several groups to add an image database to our project

Analysis of published MIMICII papers



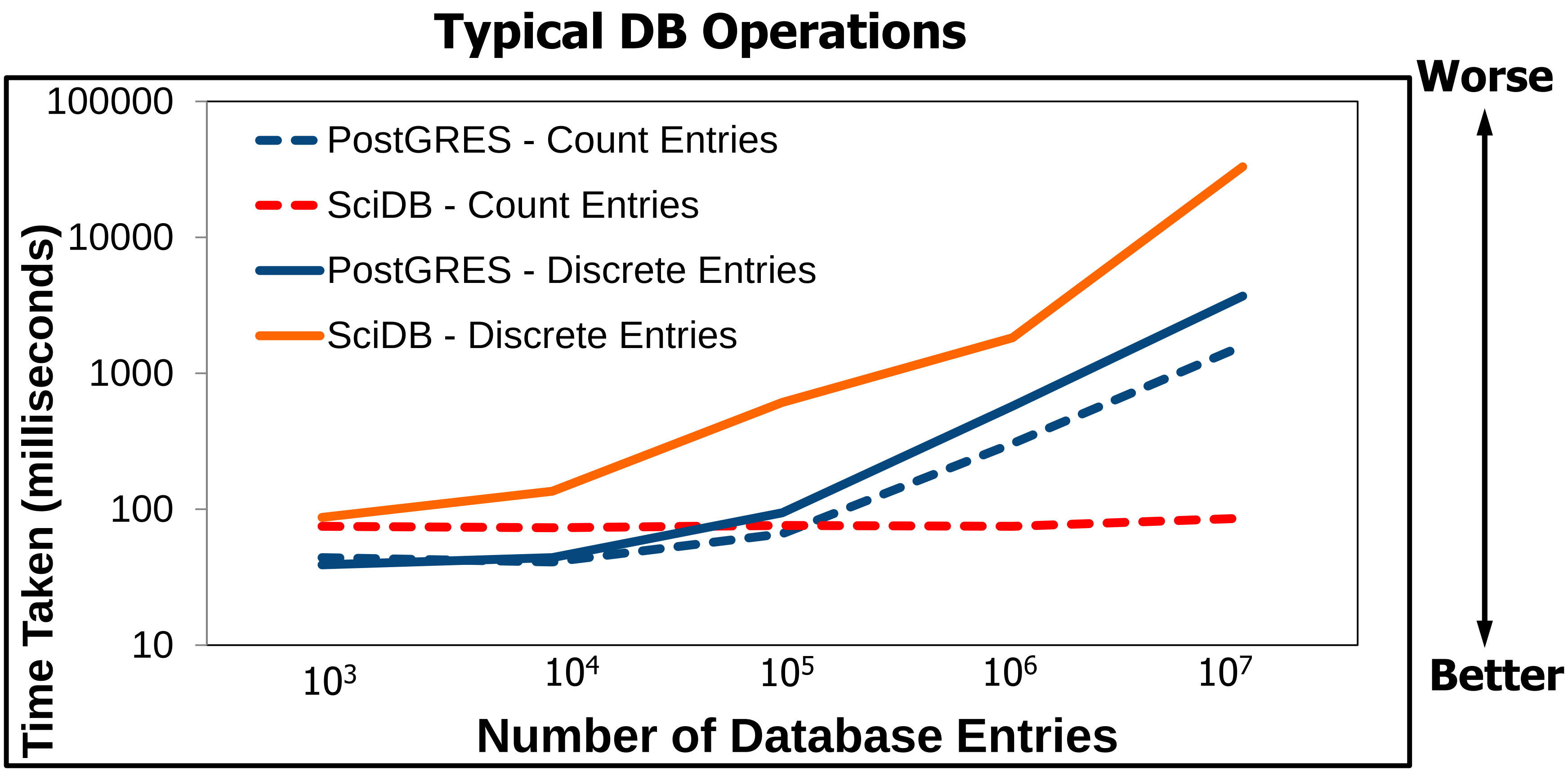
- Data in databases is used; data in files is not
  - Data in files is nearly equivalent to deleting the data

We must bring the power of data bases to all data

*Based on PhysioNet MIMIC2 ICU data

Source: Vijay Gadepally of MIT Lincoln labs

So we should cram all the data into one DBMS? NO!!! One Size Does Not Fit All*



Count and Find Operations

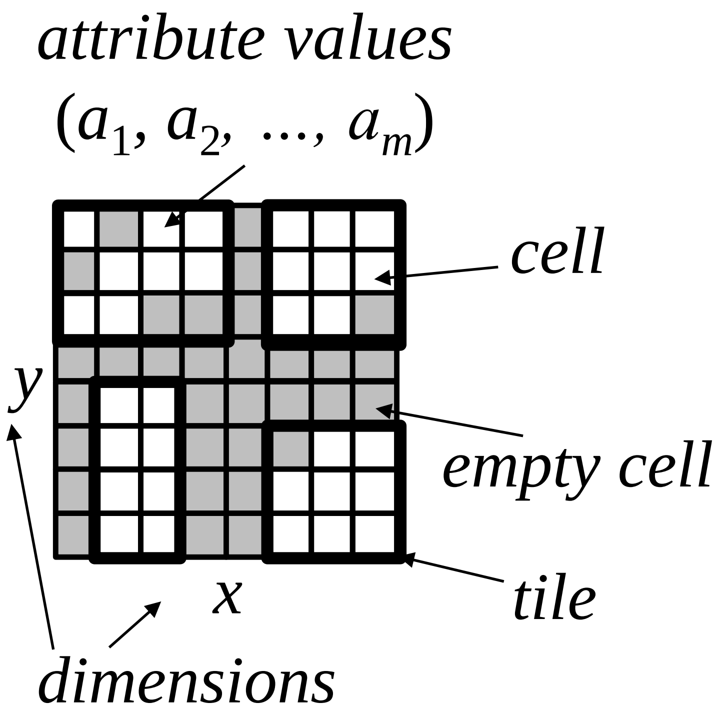
- **SQL database (PostgreSQL) better for some operations than Array database (SciDB)**

*Stonebraker, Michael, and Ugur Cetintemel. "" One size fits all": an idea whose time has come and gone." *21st International Conference on Data Engineering (ICDE'05)*. IEEE, 2005.

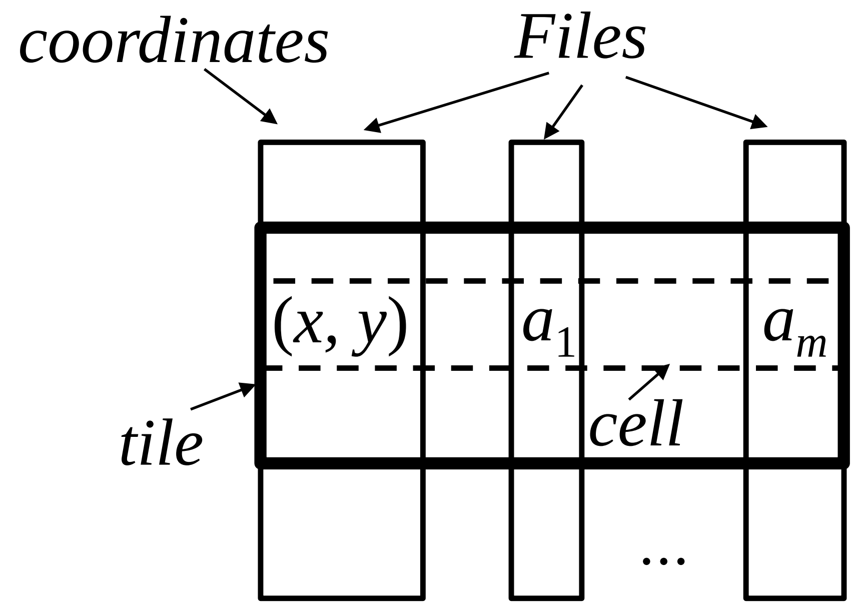
A more extreme "one size does not fit all" example:

TileDB a new array data storage manager optimized for Sparse Arrays

Logical representation



Physical representation



Tile: *Atomic unit of processing*

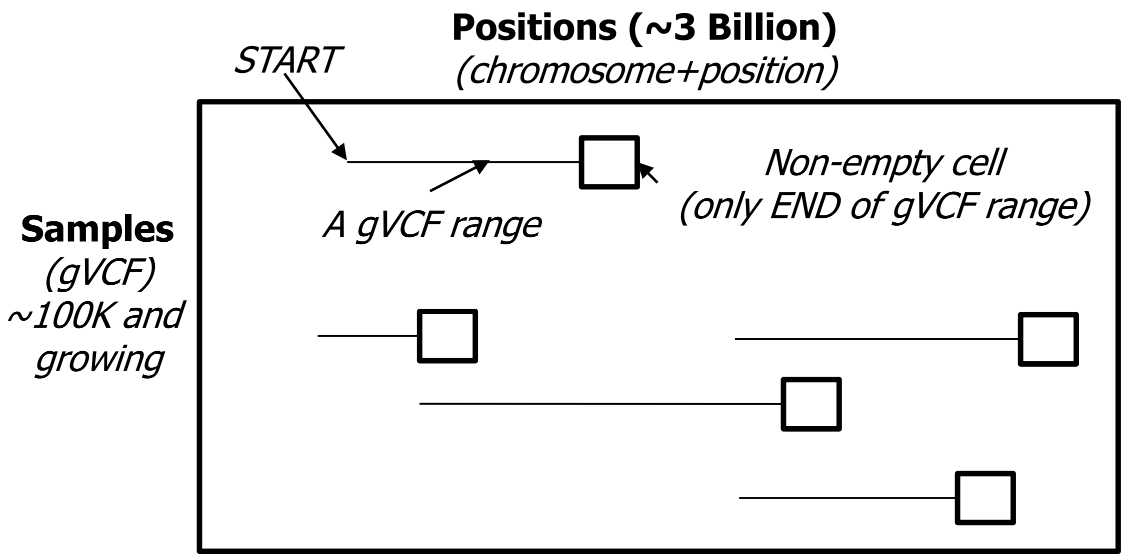
Manage array storage as tiles of different shape/size in the index space, but with ~equal number of non-empty cells

Stavros Papadopoulos of Intel created TileDB

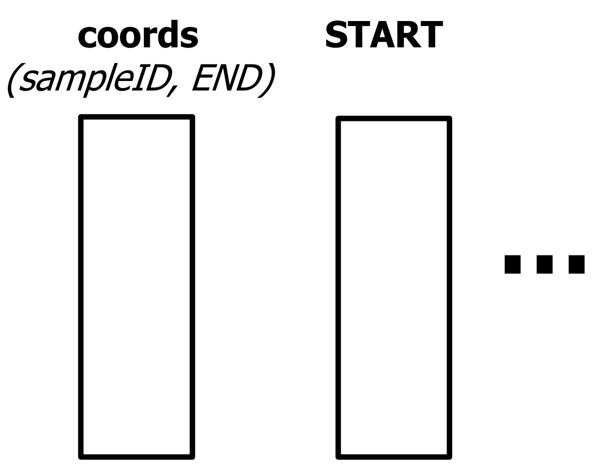
Open Source release: https://github.com/Intel-HLS/GenomicsDB

TileDB is ideal for storing Genomics Data

- Represent variation of a sample from a reference Genome (Genome Variant Call format or gVCF)
- Store as a sparse 2D array in TileDB … store a non-empty cell for every <u>END</u> endpoint of the gVCF ranges



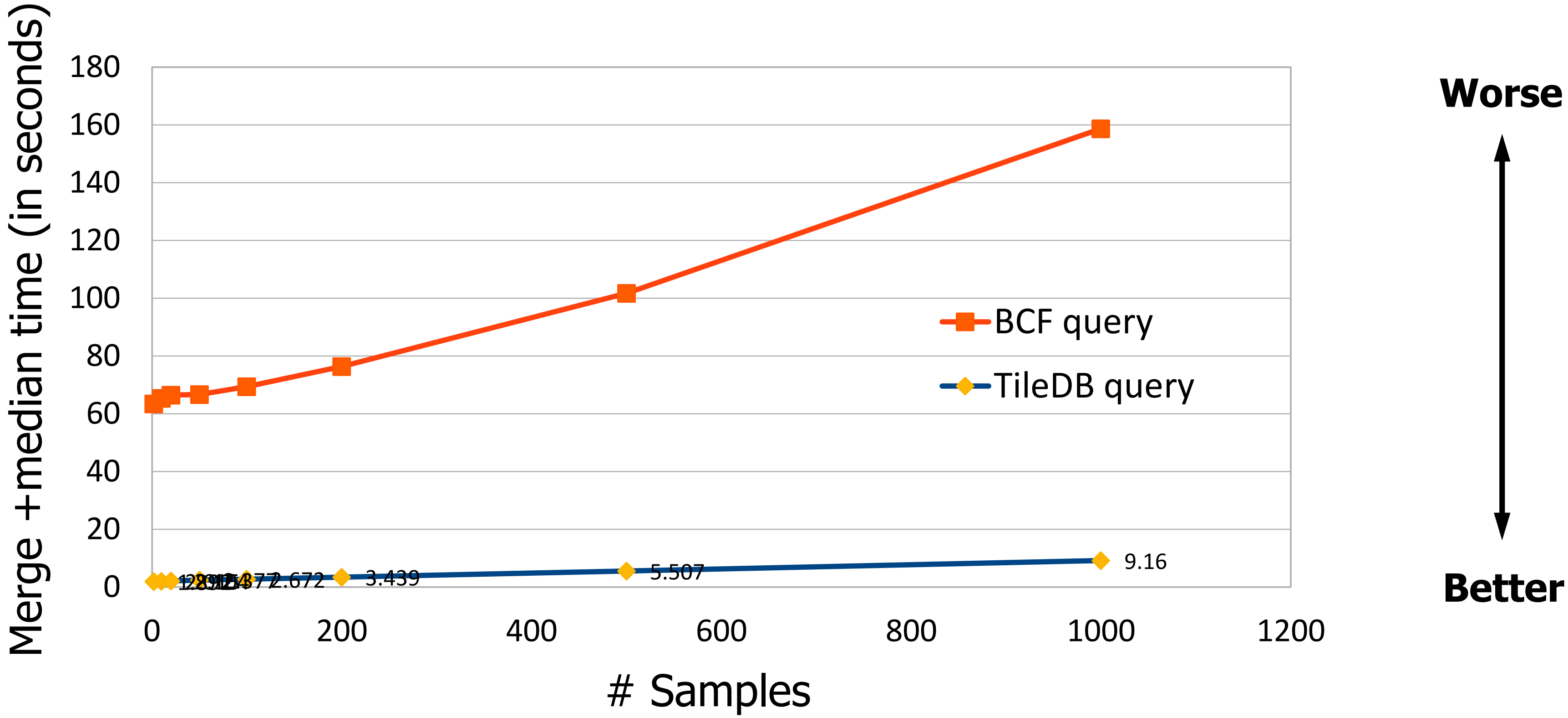
<u>Binary files (one per attribute)</u>



- The cells are sorted in <u>column-major</u> order, and <u>compressed</u>

One Size Does Not Fit All

GenomicsDB/TileDB combine gVCF operation + median (5K random positions)



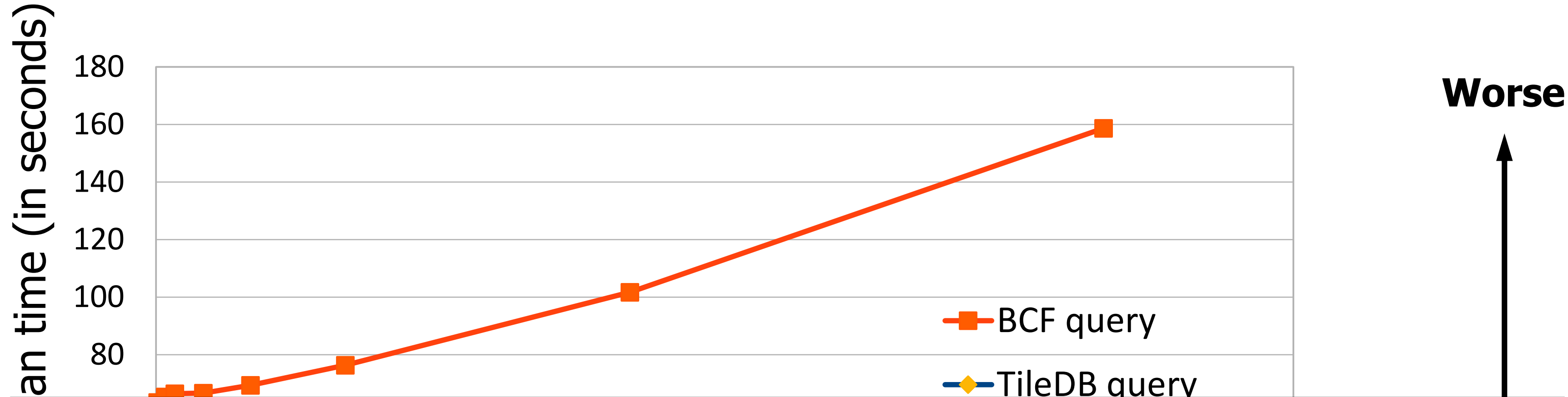
BCF refers to the Broad processing pipeline highly optimized by Intel.

This is what happens when the data-store matches the data

Intel® Xeon® E5 2697 v2 CPU, 12 cores, dual socket, 128 GB RAM, CentOS6.6, Western Digital 4 TB WD4000F9YZ-0 as a ZFS RAID0 pool.

One Size Does Not Fit All

GenomicsDB/TileDB combine gVCF operation + median (5K random positions)



From Eric Banks ot the Broad (April 2016) speaking of TileDB….

"The time it now takes to perform the variant discovery process went from eight days to 18 hours," Banks said. "However, that's with 100 whole genomes. We routinely process projects with thousands of samples, so that speedup itself is truly transformative. …

http://genomicinfo.broadinstitute.org/acton/media/13431/broad-intel-collaboration

This is what happens when the data store matches the data

Intel® Xeon® E5 2697 v2 CPU, 12 cores, dual socket, 128 GB RAM, CentOS6.6, Western Digital 4 TB WD4000F9YZ-0 as a ZFS RAID0 pool.

Three Eras of database technology

SQL Era	NoSQL Era	NewSQL Era	Future
Common interface	**Rapid ingest for internet search**	**Fast analytics inside databases**	**Applications using Multiple storage engines that match the needs of the data.**

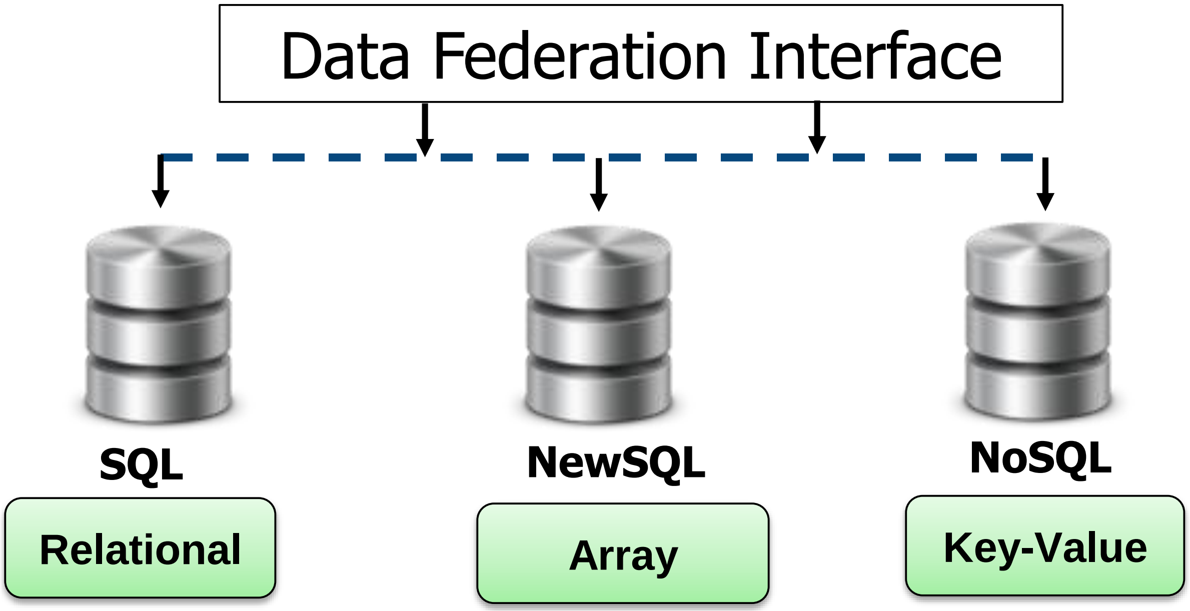
SQL = Structured Query Language
NoSQL = Not only SQL

Source: The BigDAWG Polystore System and Architecture, HPEC'2016, Vijay Gadepally

Third party names are the property of their owners

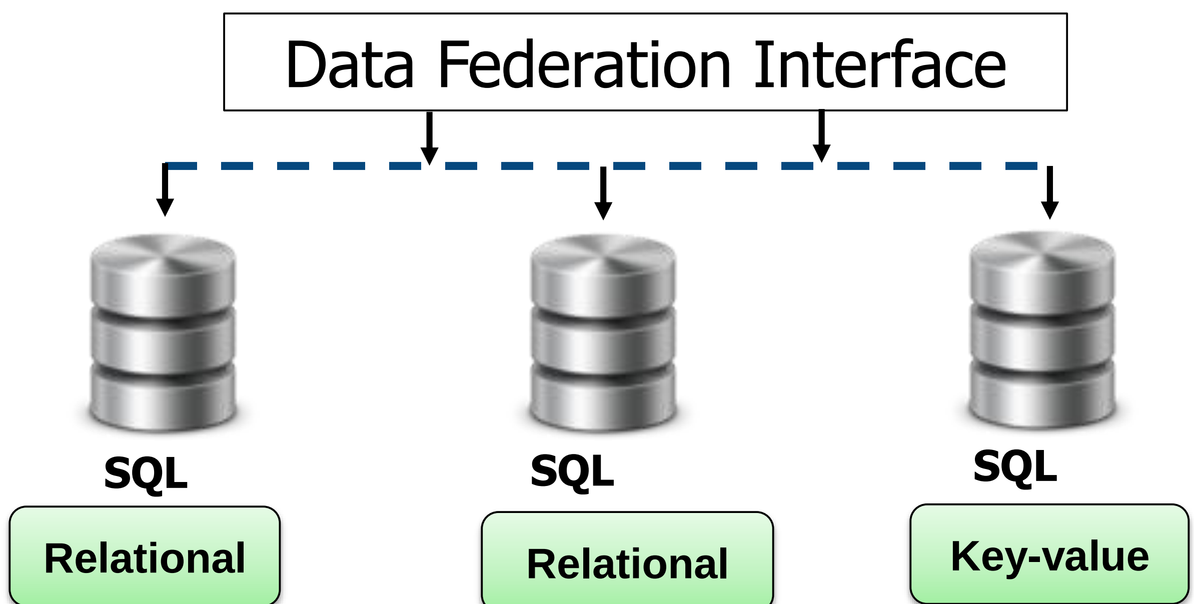
How do we deal with multiple data bases?

- Programmer productivity requires **Data Virtualization**.
  - A data access interface that hides the technical details of stored data, such as location, storage structure, API, access language, and storage technology.
- Typical mechanism for Data Virtualization? … **Data Federation**
  - A form of data virtualization where the data stored in a **heterogeneous** set of **autonomous** data stores is made accessible to data consumers as one integrated data store using on-demand data integration.



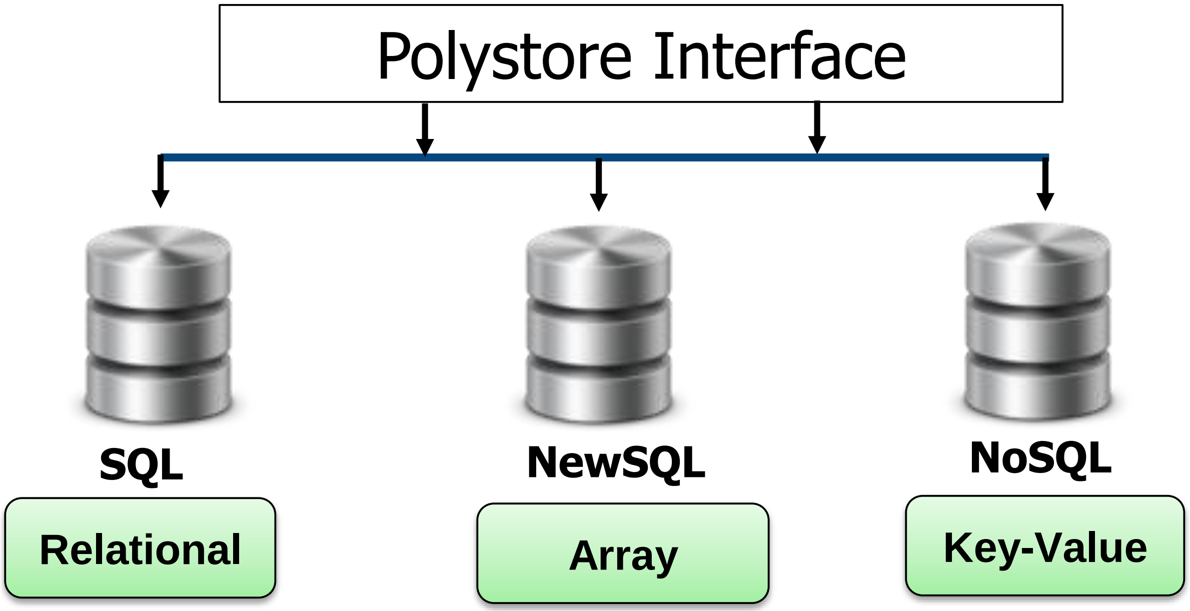
How do we deal with multiple data bases?

- Data Federation … in practice
  - The single interface imposes a single data model
  - The DBMS are autonomous … not integrated!
- Therefore, disparate data models in the DBMS are hard to support and the federated DBMS are <u>typically</u> based on a single (e.g. SQL) data model ….
  - forces a "One Size Fits All" perspective.



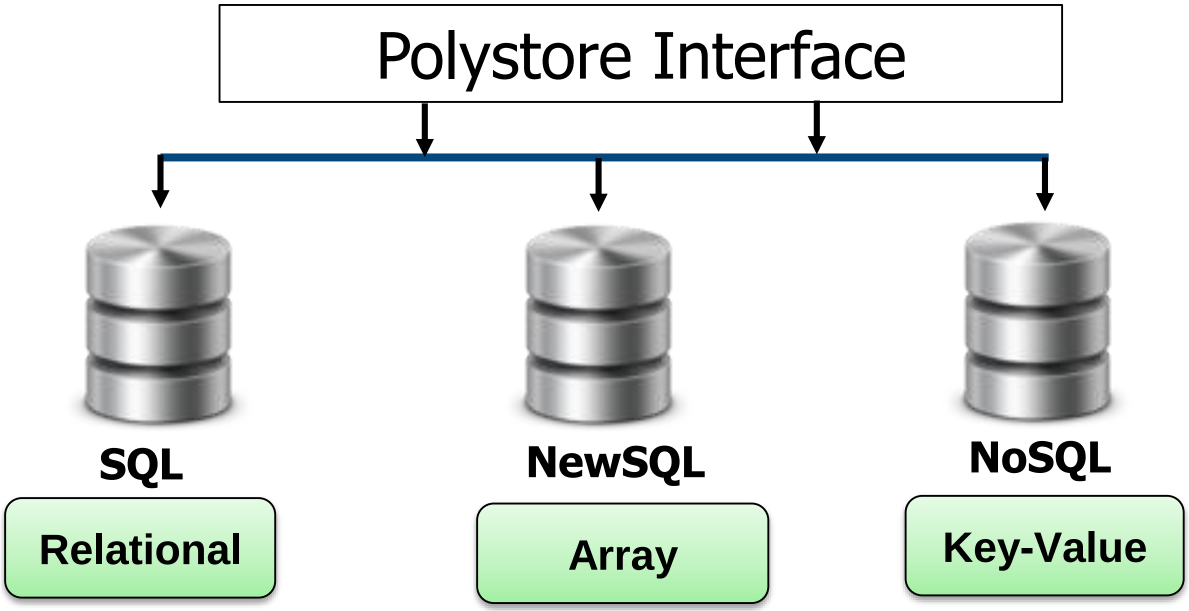
Polystore: a new twist on Data Federation

- Programmer productivity requires *data virtualization*, efficient execution requires benefits of queries that exploit features of a particular data-store
- Polystore:
  - A form of data virtualization where the data stored in a heterogeneous set of **integrated** data stores is exposed through a common interface but the features of the individual data-stores are visible.



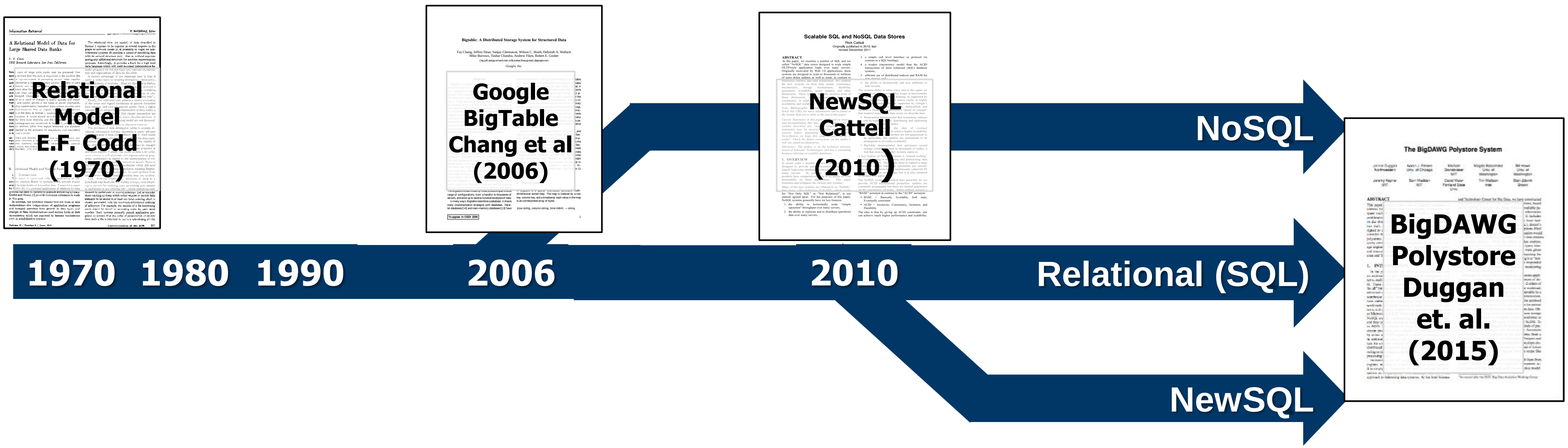
Polystore: a new twist on Data Federation

- Polystore Design forces.
  - **Location independence**: A query does not care which data-store in the polystore system it will target. A huge convenience for programmers.
  - **Semantic Completeness**: Any query natively supported by a data-store in the Polystore system can be expressed.
- The challenge in designing a Polystore system is to balance "location independence" and "Semantic Completeness" without compromising efficient execution.



Three Eras of database technology

SQL Era	NoSQL Era	NewSQL Era	Future
Common interface	**Rapid ingest for internet search**	**Fast analytics inside databases**	**Polystore: matching data to the storage engine**



ORACLE PostgreSQL accumulo SciDB GRAPHULO

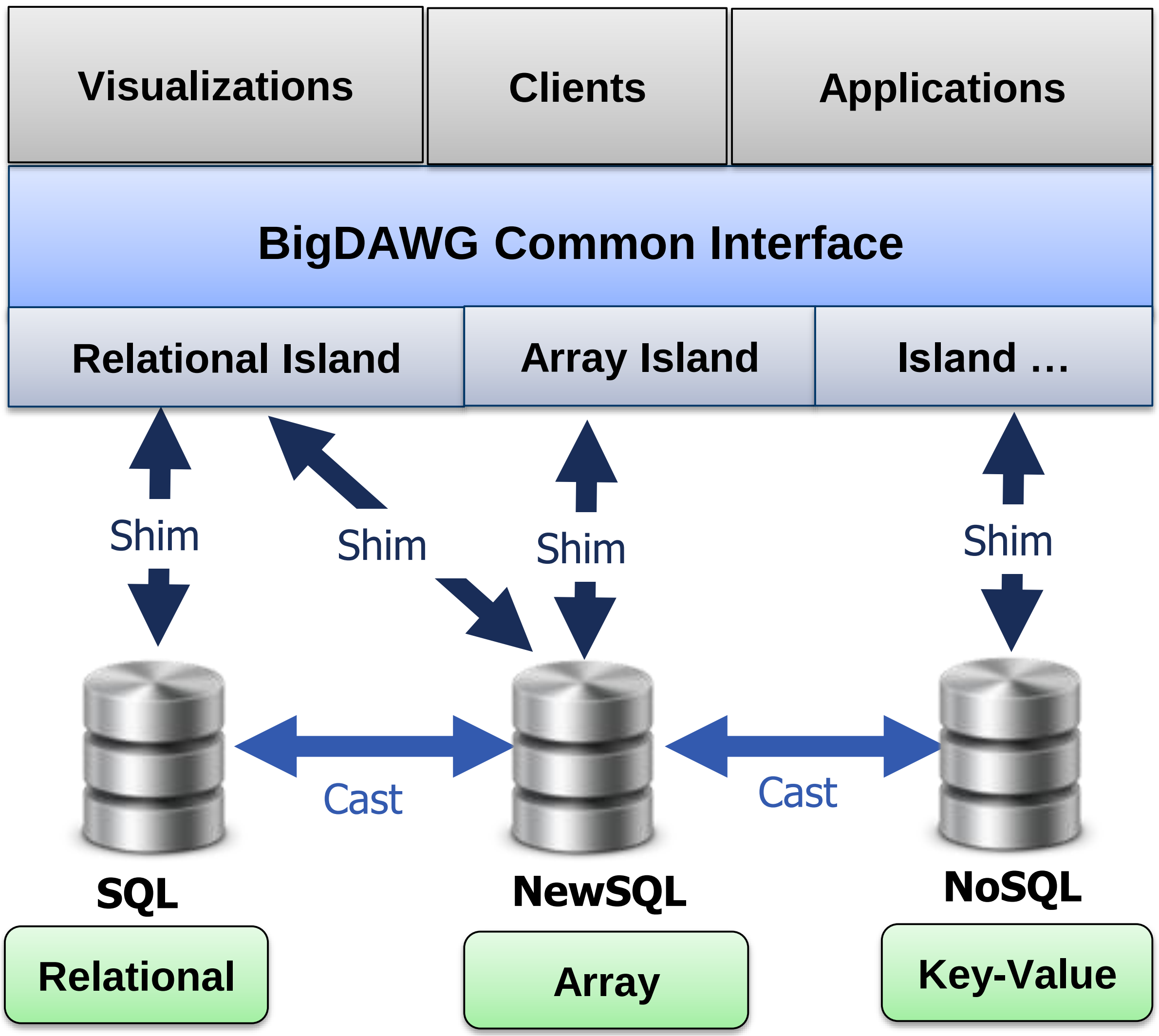
SQL = Structured Query Language
NoSQL = Not only SQL

Source: The BigDAWG Polystore System and Architecture, HPEC'2016, Vijay Gadepally

Third party names are the property of their owners

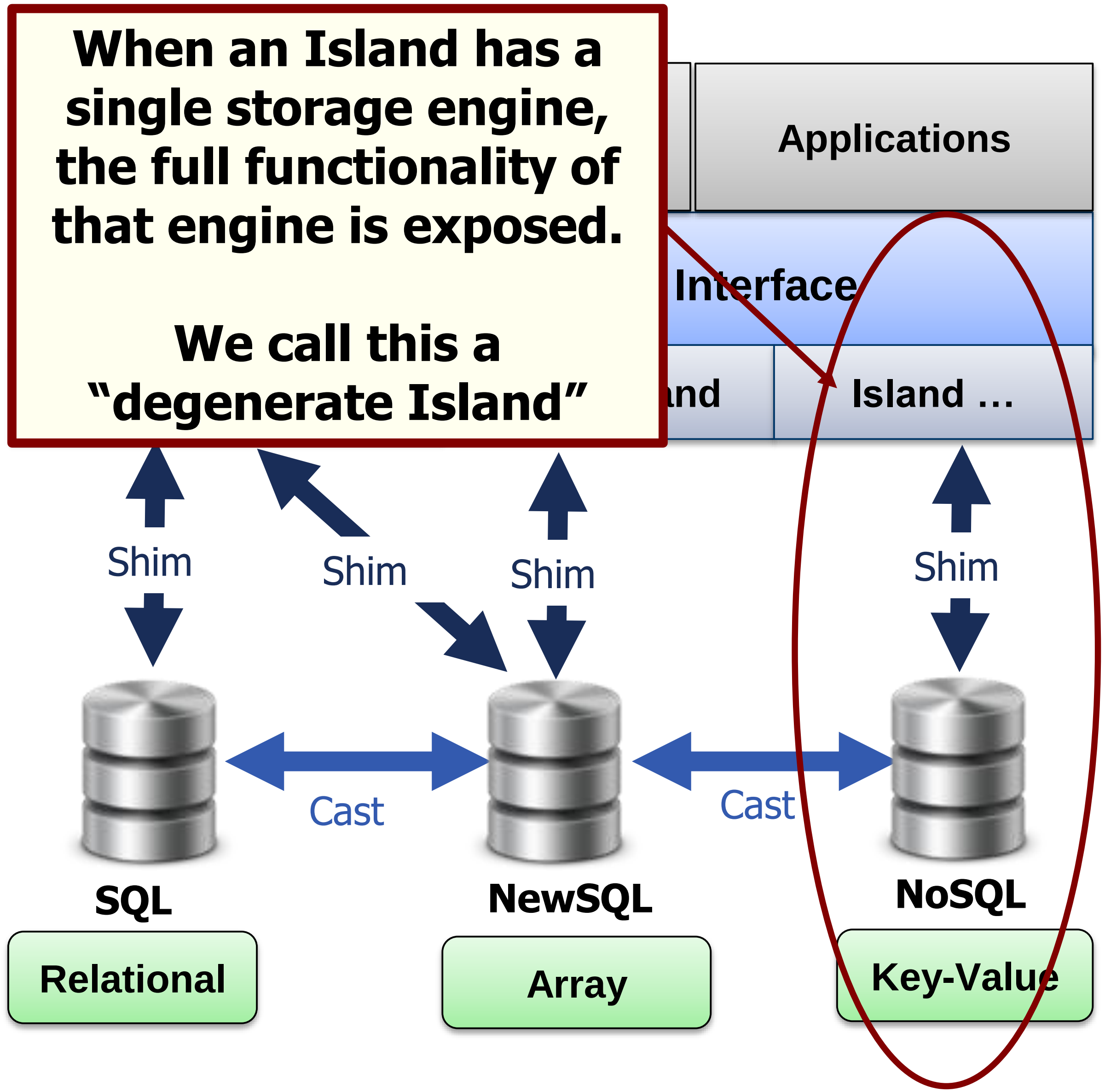
BigDAWG: A Prototype Polystore System

- BigDAWG
  - Polystore: match data to the storage engine
- BigDAWG Islands
  - A data model + query operations
  - One or more storage engines
  - “Shim” connects a BigDAWG query to a data engine
  - “Cast” migrates data from one engine to another

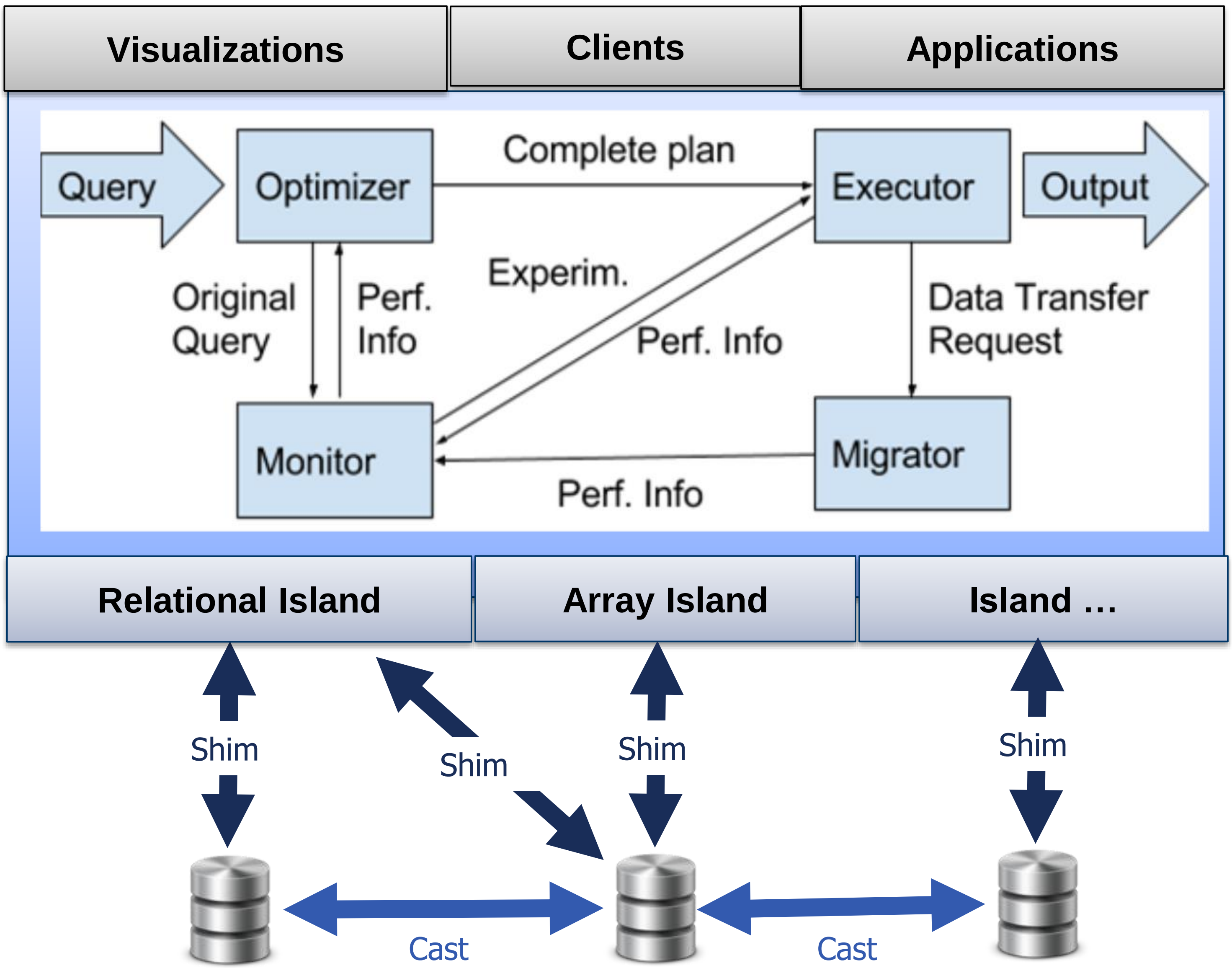


BigDAWG: A Prototype Polystore System

- BigDAWG
  - Polystore: match data to the storage engine
- BigDAWG Islands
  - A data model + query operations
  - One or more storage engines
  - “Shim” connects a BigDAWG query to a data engine
  - “Cast” migrates data from one engine to another



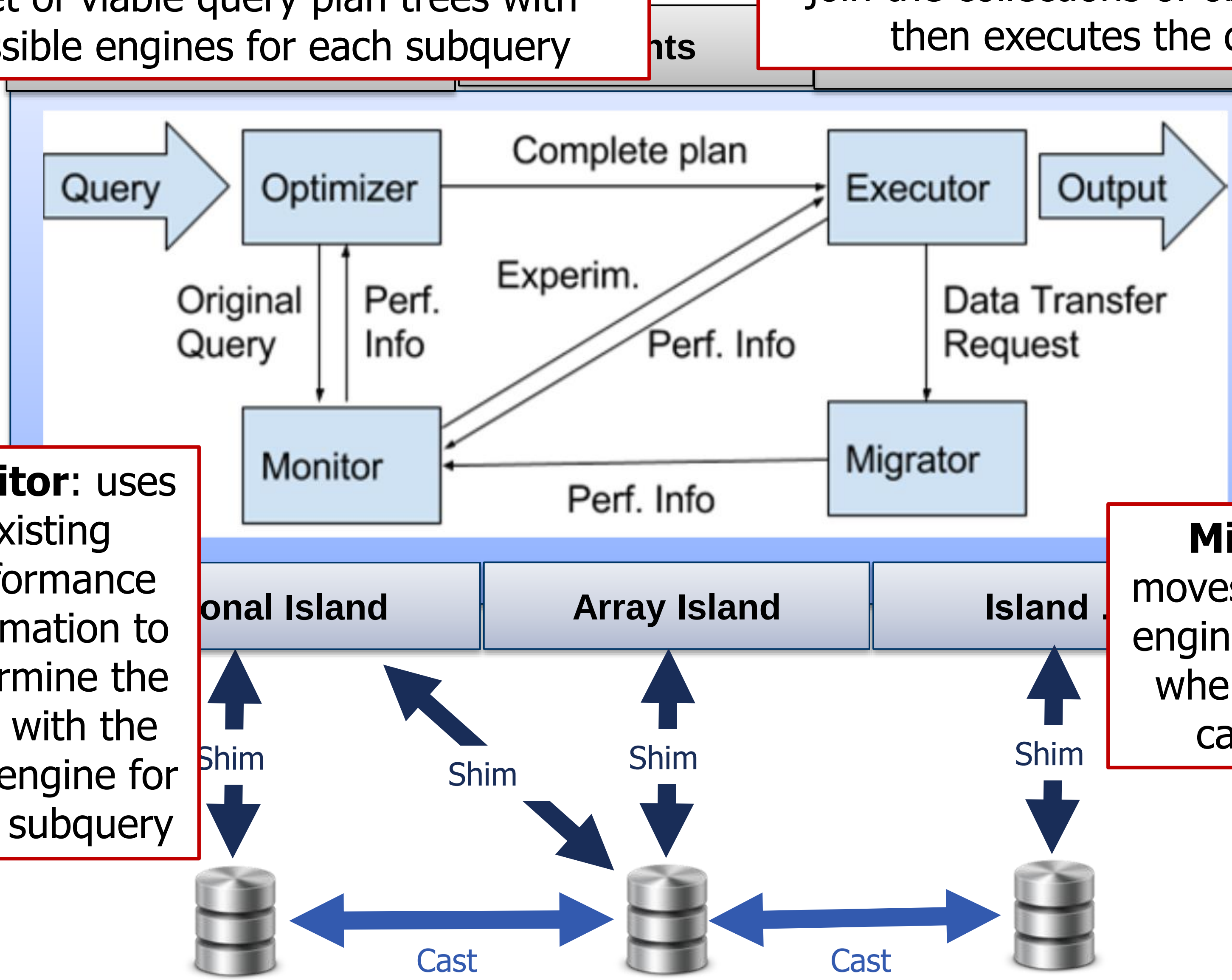
BigDAWG Middleware



BigDAWG Middleware

Optimizer: Parses the query and creates a set of viable query plan trees with possible engines for each subquery

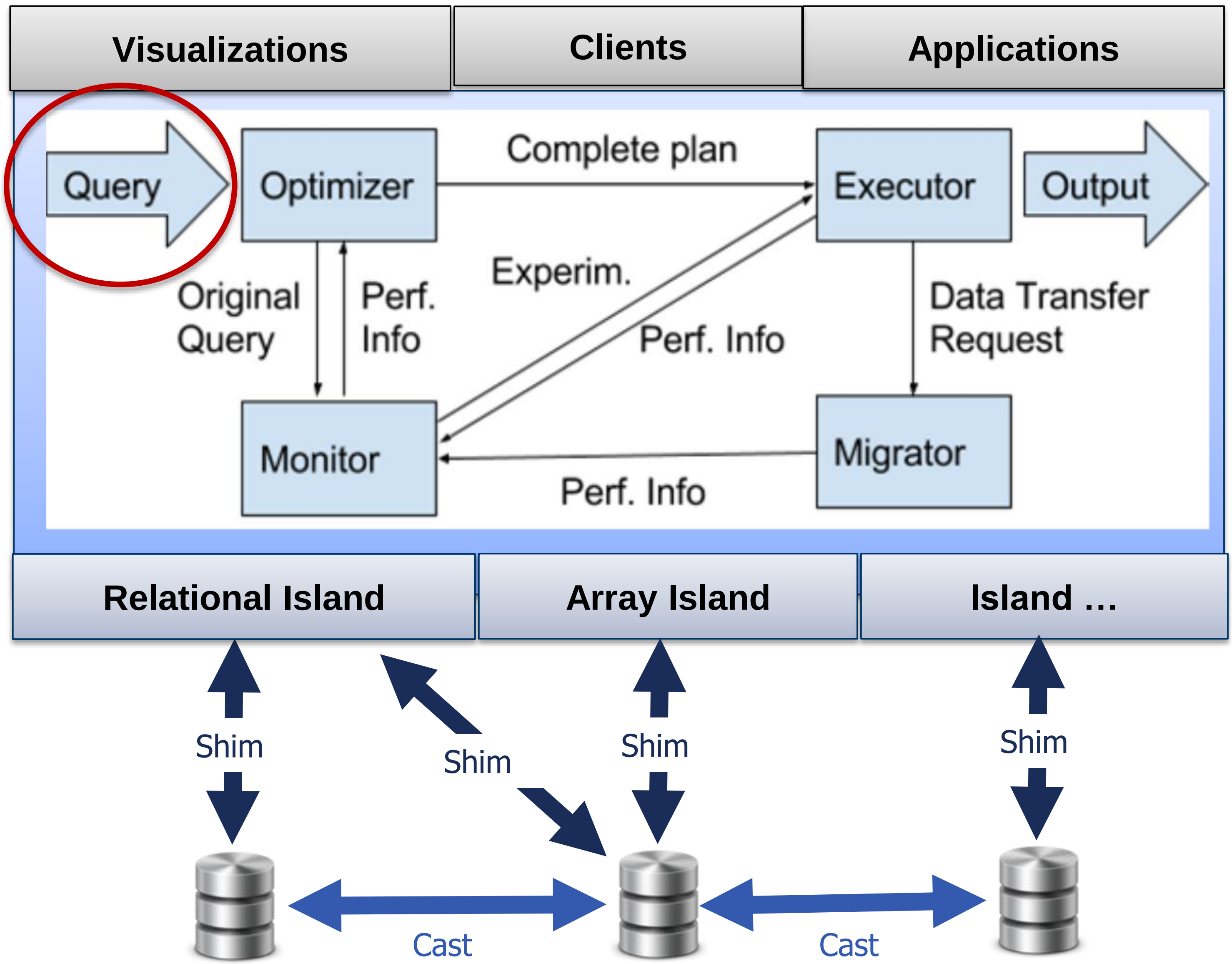
Executor: figures out how to best join the collections of objects and then executes the query



Monitor: uses existing performance information to determine the tree with the best engine for each subquery

Migrator: moves data from engine to engine when the plan calls for it

BigDAWG Middleware



A Big DAWG Query

```
bdarray(
   filter(
      bdcast(
          bdrel( select bodc_sta, time_stp, interp_sal
                   from sampledata.main)
         , intrp_salinity
         , '<bodc_sta:int64, time_stp:datetime, interp_sal:double> [i=0:*,1000,0]‘
         , array)
     , interp_sal < 35))
```

A Big DAWG Query

Using the array island, issue the island’s filter operation

filter([source_array], [logical_expression])

```
bdarray(
  filter(
    bdcast(
      bdrel( select bodc_sta, time_stp, interp_sal
             from sampledata.main)
      , intrp_salinity
      , '<bodc_sta:int64, time_stp:datetime, interp_sal:double> [i=0:*,1000,0]'
      , array)
  , interp_sal < 35))
```

Result is an array with rows for which interp_sal is less than 35

A Big DAWG Query

Create the array for the filter op by casting the table formed by this subquery from the relational island to the array island

```
bdarray(
  filter(
    bdcast(
       bdrel( select bodc_sta, time_stp, interp_sal
              from sampledata.main)
       , intrp_salinity
       , '<bodc_sta:int64, time_stp:datetime, interp_sal:double> [i=0:*,1000,0]‘
       , array)
    , interp_sal < 35))
```

Bdcast ([source_query], name, [Dest_schema_parameters], [target])

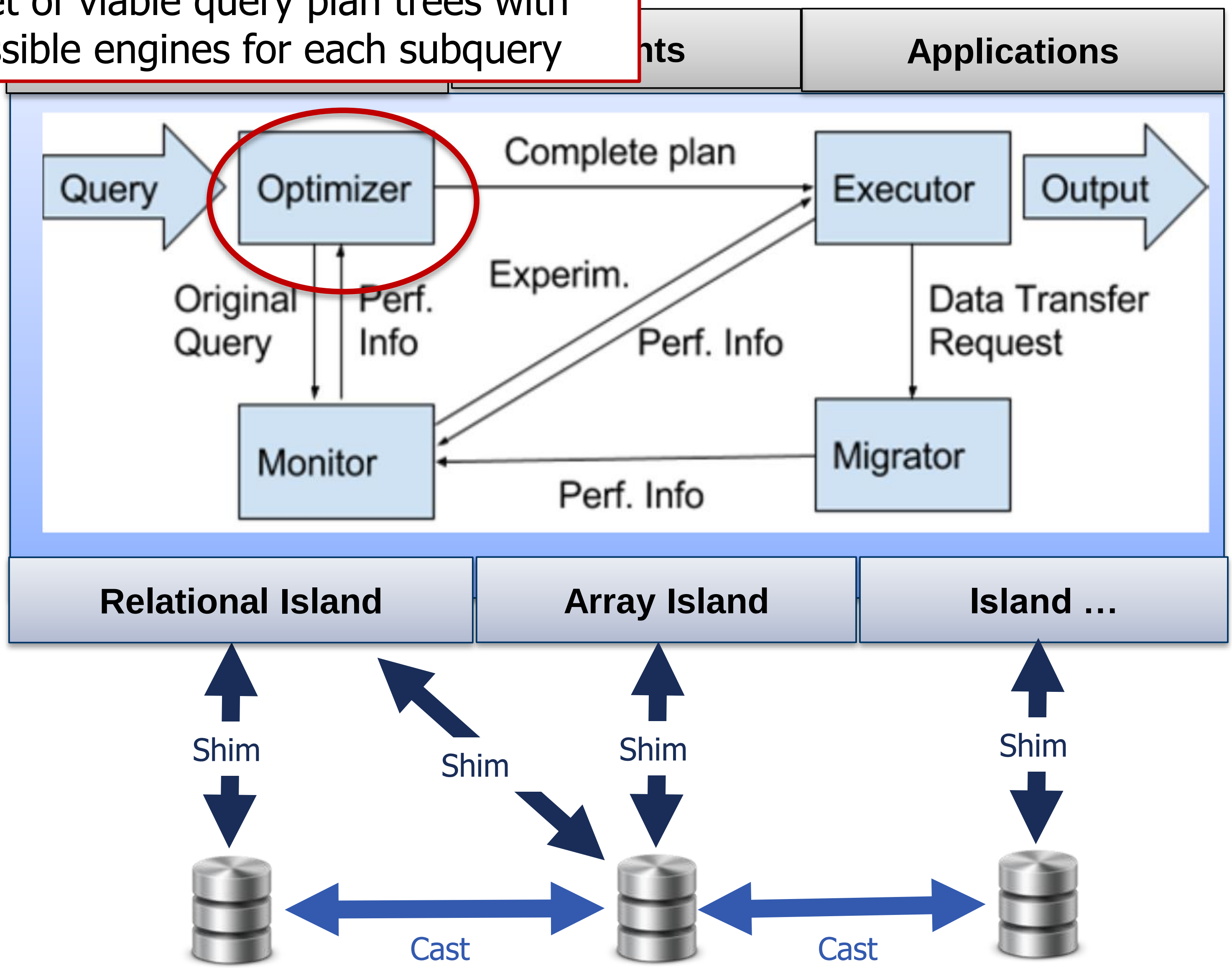
A Big DAWG Query

The array created is named “intrp_salinity”. It has three attributes (bodc_sta, time_stp, and interp_sal) with unbounded number of rows (i=0:*) broken down into chunks of size 1000 with 0 overlap

```
bdarray(
  filter(
    bdcast(
      bdrel( select bodc_sta, time_stp, interp_sal
             from sampledata.main)
      , intrp_salinity
      , '<bodc_sta:int64, time_stp:datetime, interp_sal:double> [i=0:*,1000,0]'
      , array)
  , interp_sal < 35))
```

BigDAWG Middleware

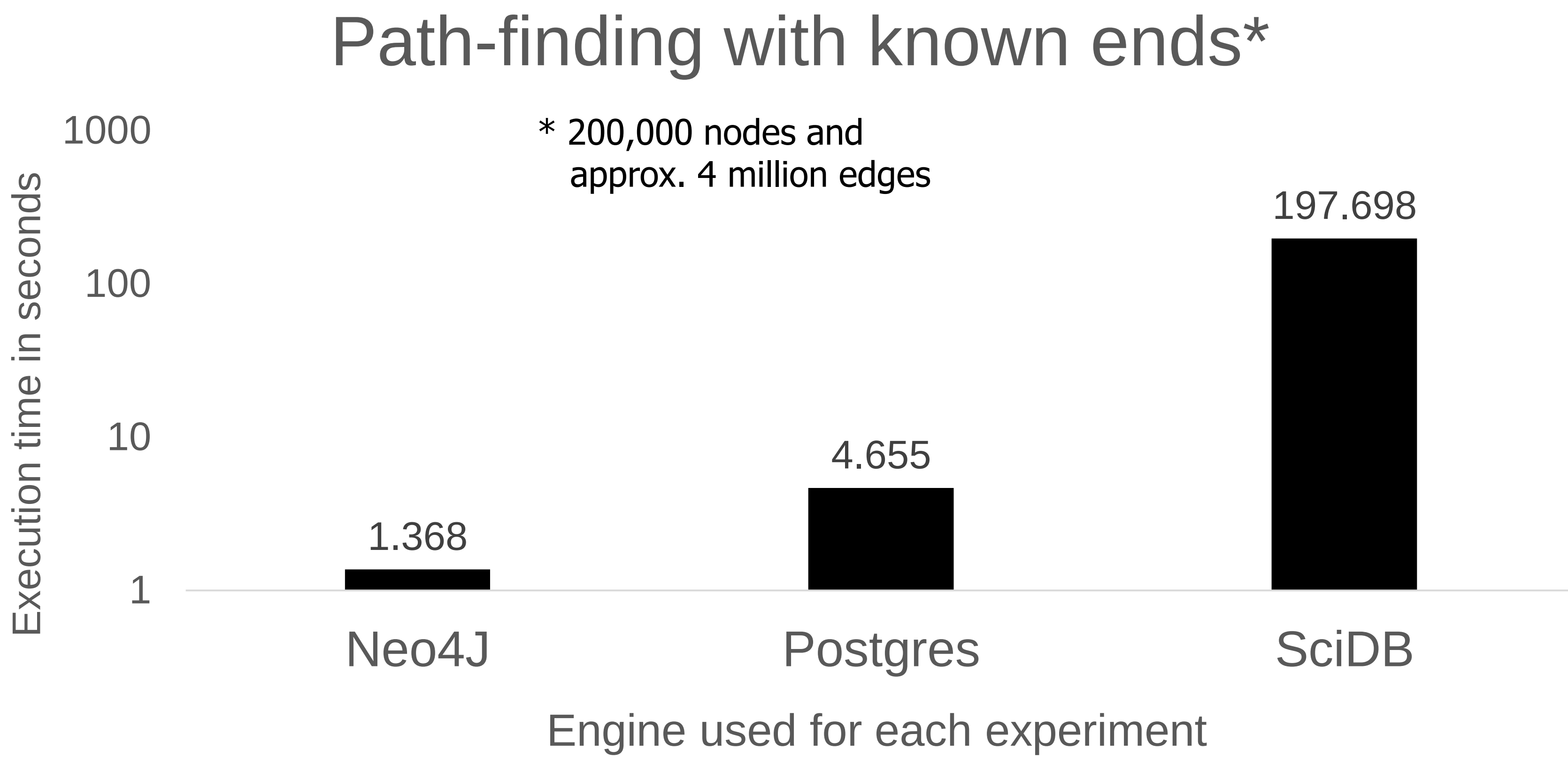
Optimizer: Parses the query and creates a set of viable query plan trees with possible engines for each subquery



Optimizer: turns queries into a logical plan

- The Optimizer generates Logical plans corresponding to the input query.
  - Works with the “monitor” to track historic plans and select the best plan.
- The Optimizer uses planners native to an Island.
- What about cross Island Optimization?
  - We need to build these ourselves

Optimization: Finding the right Island to run a query



Can we translate queries between Islands and then run on the Island that gives us the best performance?

SciDB array database from paradigm4 Neo4 graph database fror neo4j

Goal: Translate queries between islands

- Approach:
  - Build a translation framework
  - Equivalence rules mapping between Islands
- Example equivalence rule …

- {name: “SQL to AFL matrix multiplication”,
  source : SQL, destination : AFL,
  {matrix_1 : table,
  original_attribute:
  {row : integer, col : integer, value : double precision}},
  destination_attribute:
  {row : dimension, col : dimension, value : double}
  {matrix_2 : …}
  source_query: “SELECT m1.row, m2.col, ...”,
  destination_query: “spgemm(matrix_1, matrix_2)”}

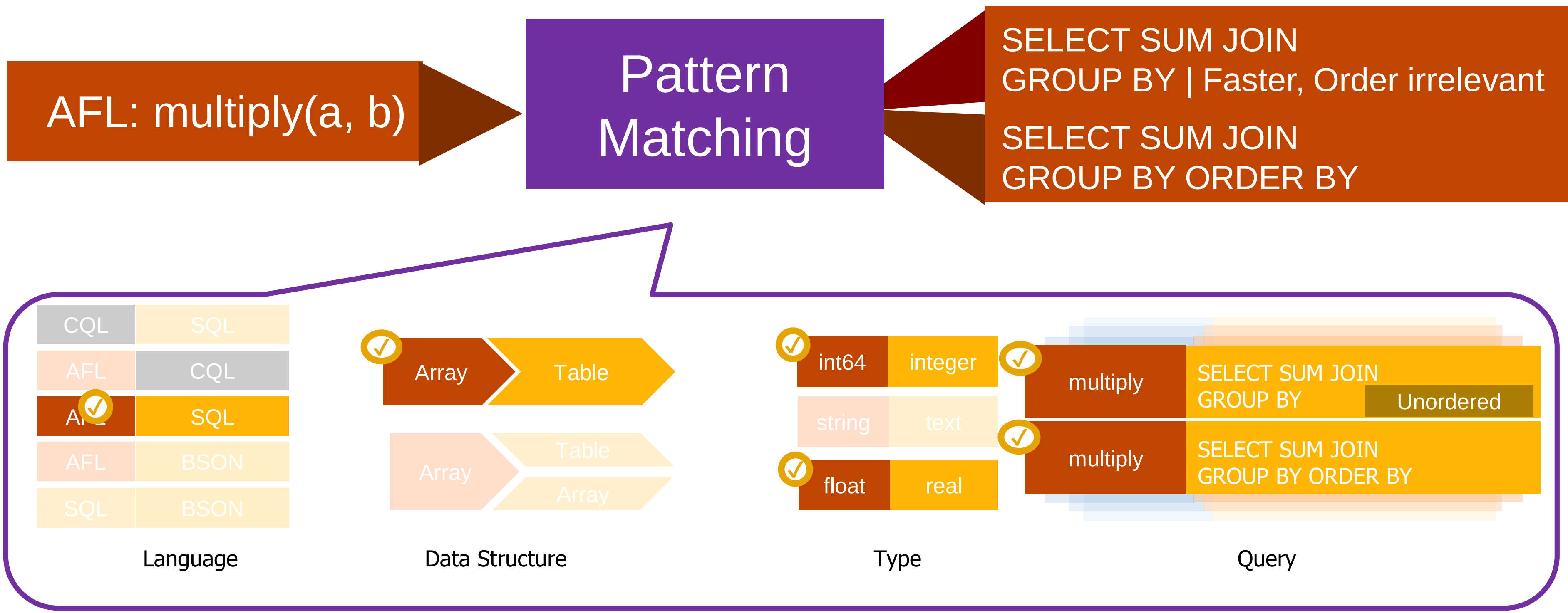
Rule name

Islands

Data Structure mappings

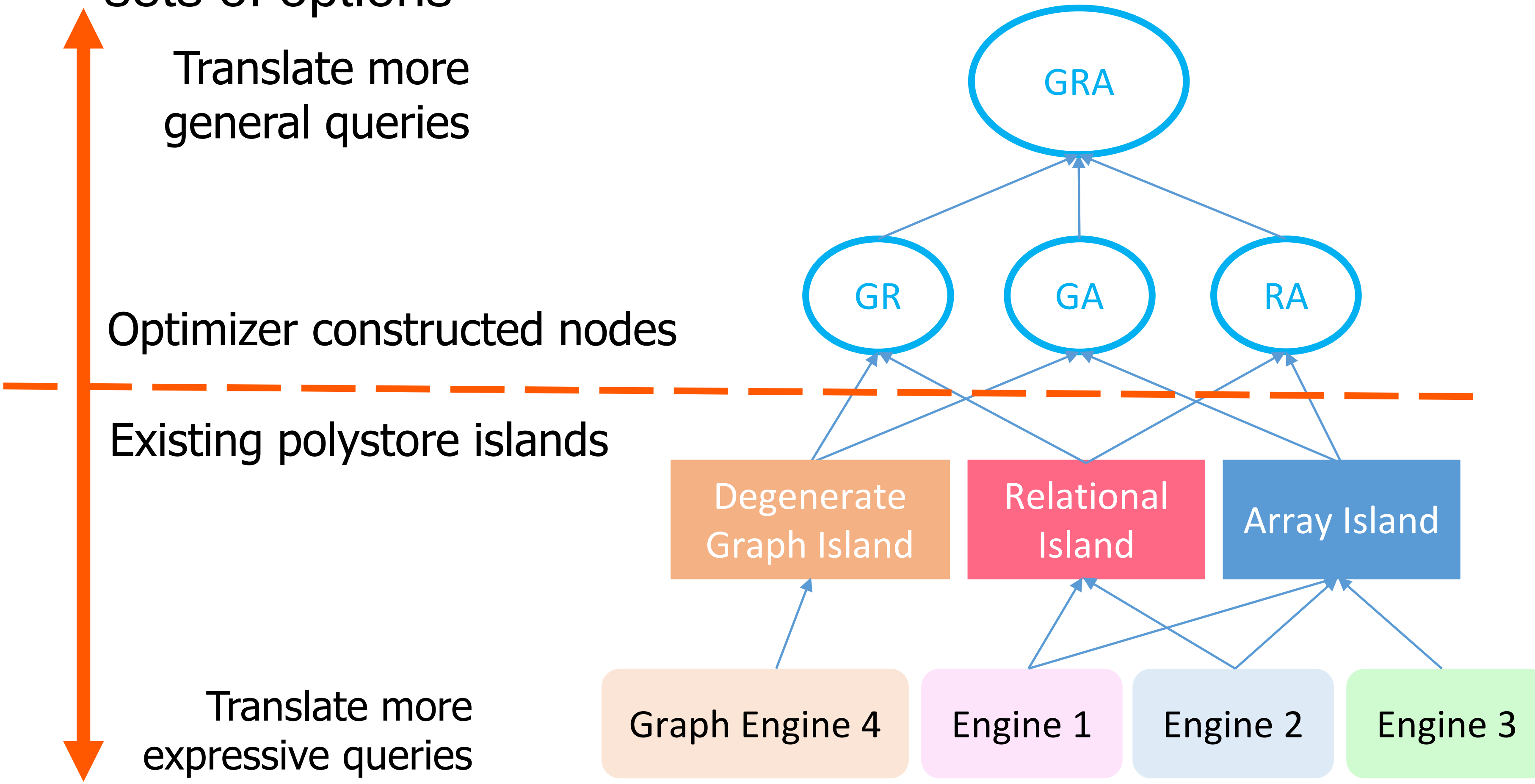
Query mappings

Using Equivalence Rules for translation

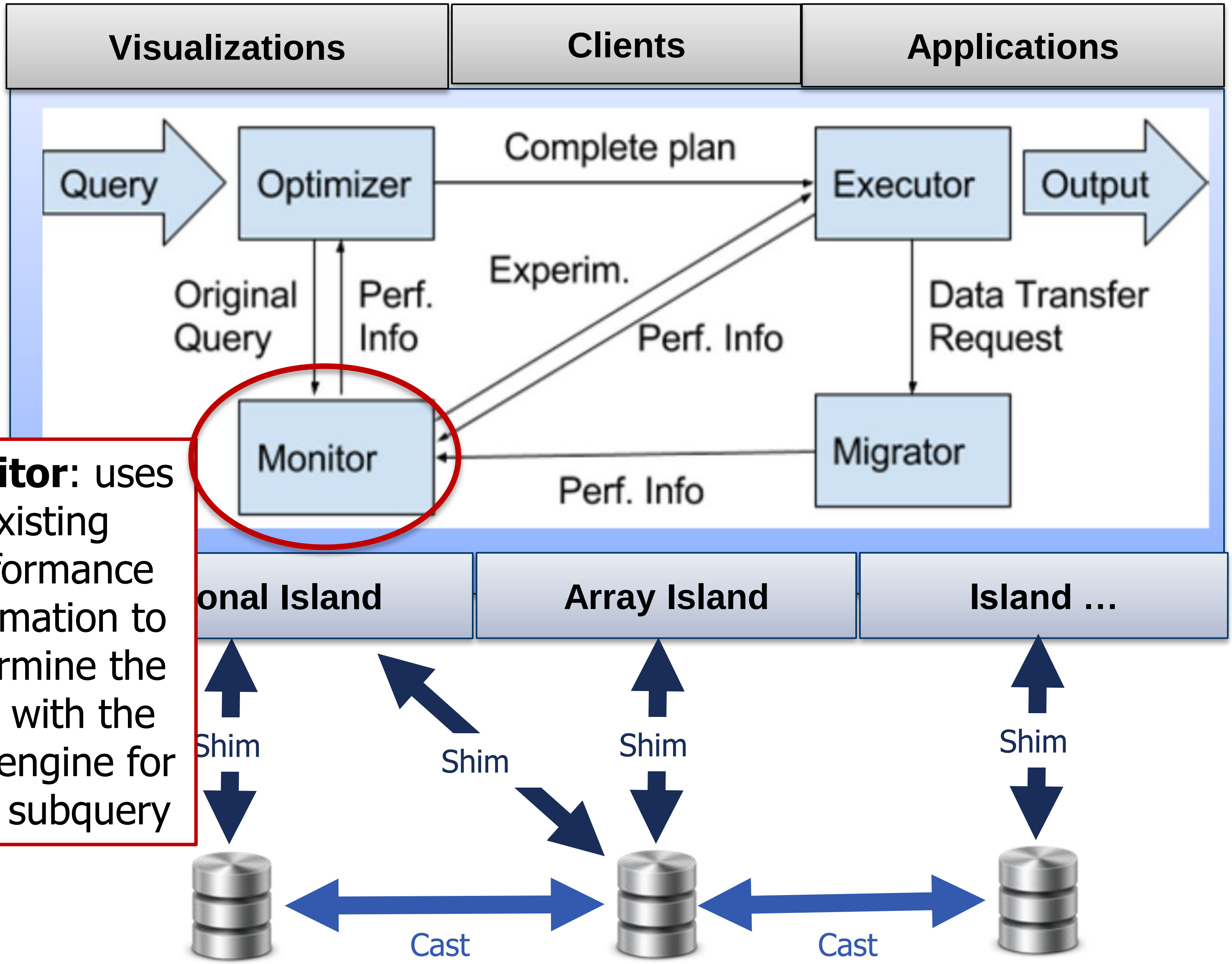


The Semantic Lattice

- Organize collections of equivalence rules (both generated and user provided) into a semantic lattice to reason over sets of options

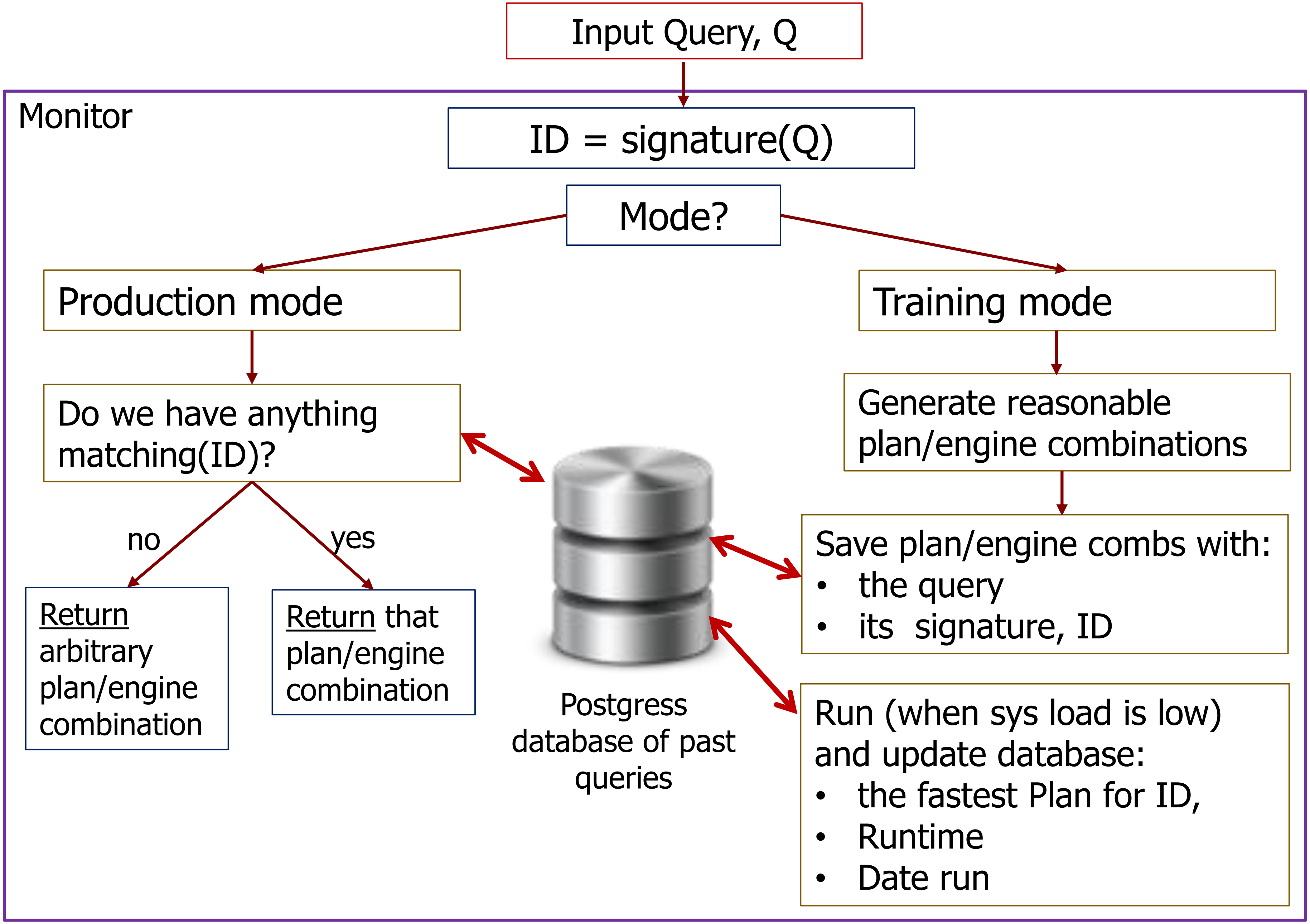


BigDAWG Middleware



Monitor: uses existing performance information to determine the tree with the best engine for each subquery

BigDAWG Monitor: find best execution plan for a Query



Production mode gains

Query No.	Training Mode	Production Mode	Without Monitor
1	826 ms	265 ms	281 ms
2	882 ms	190 ms	346 ms
3	62539 ms	20559 ms	20990 ms
4	491 ms	160 ms	166 ms
5	6592 ms	1977 ms	2308 ms
6	24294 ms	6146 ms	9074 ms
7	28165 ms	7648 ms	10259 ms
8	19073 ms	4496 ms	7289 ms
9	15806 ms	4652 ms	5577 ms
10	78487 ms	23496 ms	27496 ms

- 10 different queries with two possible query trees tested
- Training mode – each query run through two possible query trees
- Production mode – executor runs best query determined by training mode
- Best case: Production mode is ~60% of time running without Monitor (randomly select 1 of the 2 possible query trees)

Production mode gains

Query No.	Training Mode	Production Mode	Without Monitor
1	826 ms	265 ms	281 ms
2	882 ms	190 ms	346 ms
3	62539 ms	20559 ms	20990 ms
4	491 ms	160 ms	166 ms

This is all very preliminary … we know there is much left to explore before we have a production worthy monitor.

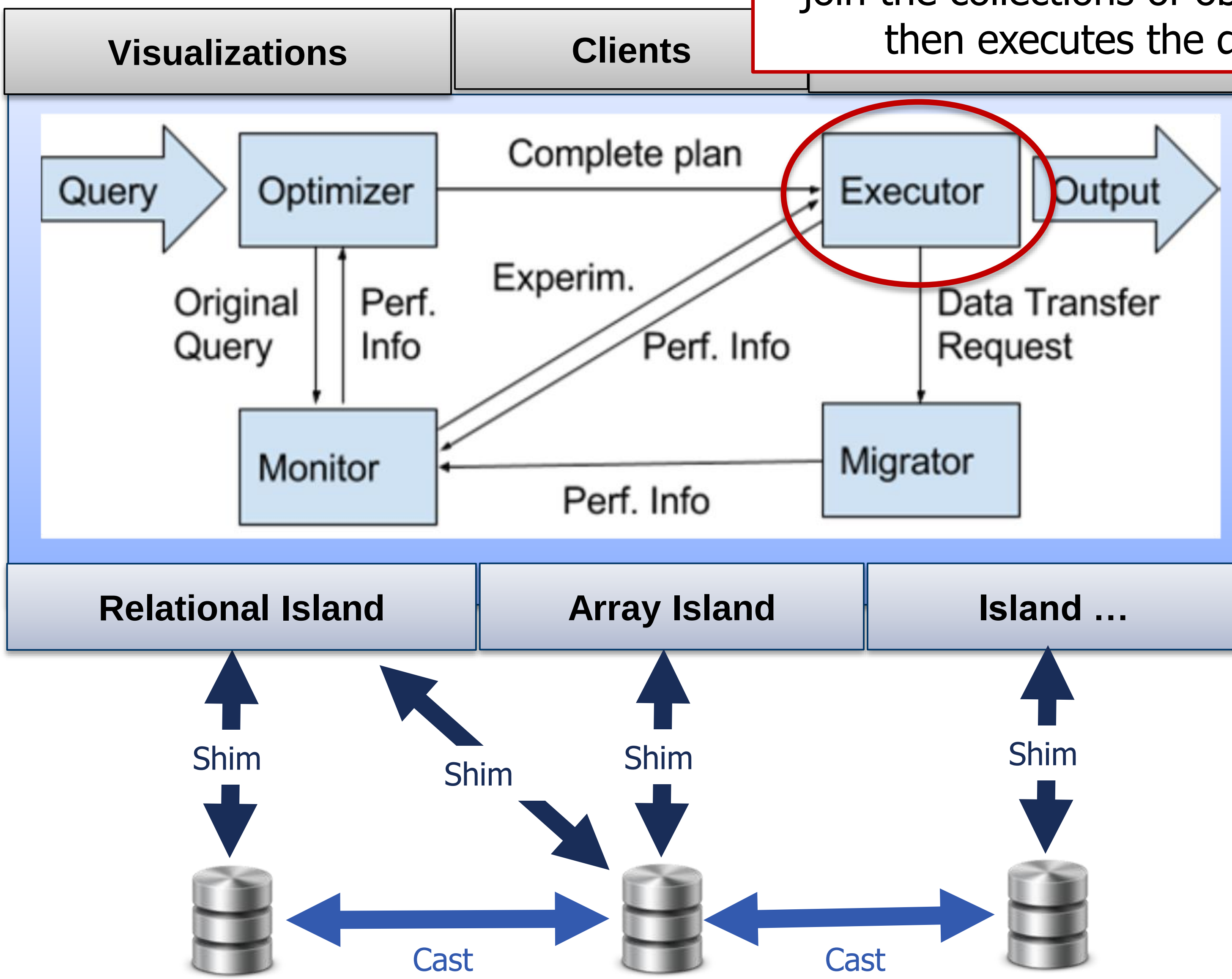
But early results are promising. Future work:

- Different Signature definitions to improve matching and reduce searching times.
- Explore a broader range of queries and engines
- Machine learning to predict "best" plans for new queries even when matching queries aren't available.

- Best case: Production mode is ~60% of time running without Monitor (randomly select 1 of the 2 possible query trees)

BigDAWG Middleware

Executor: figures out how to best join the collections of objects and then executes the query



The BigDAWG Executor

- The executor receives a Logical Query Plan from the optimizer
  - Logical plan: an execution graph … nodes with tasks and dependencies
- Basic nodes that map onto a single island
  - Issue executions on respective islands
  - Execute in parallel with a dataflow pattern
- Complex executions spanning Islands are more involved.

The BigDAWG Executor: complex queries

Consider the Shuffle join: A multi-engine join where query predicates are "shuffled" between Islands

Find distribution of join attributes to use in load balancing

Migrate join units to execution engines as needed

Outputs of engines combined on the single destination engine

Skew Examination → Join-Unit Assignment → Join-Unit Colocation → Tuple Comparison → Join Result Union

Schedule join units for execution

Carry out the join operation of join units

- join-units: small *non-overlapping ranges of tuples (rows in PostgreSQL, cells in SciDB, key-value pairs in Accumulo, etc.) participating in the join.*
- Each join-unit consists of a fraction of the full query predicate, and tuples are mapped to a join-unit based on the value of their join attribute.

Join-Unit assignments

- The challenge is to distribute join-units (i.e. “computations”) among engines to maximize performance.
- Several different strategies were considered
  - DFB: Move all tables to the final destination engine in the plan.
  - MFB: Pick the engine that requires movement of smallest tables
  - Hash: Randomly assigns tuples to participating engines
  - MBH: Assign-join unit to engine that minimizes tuple movement
  - Tabu: A local optimization algorithm that improves on the MBH result

Skew agnostic

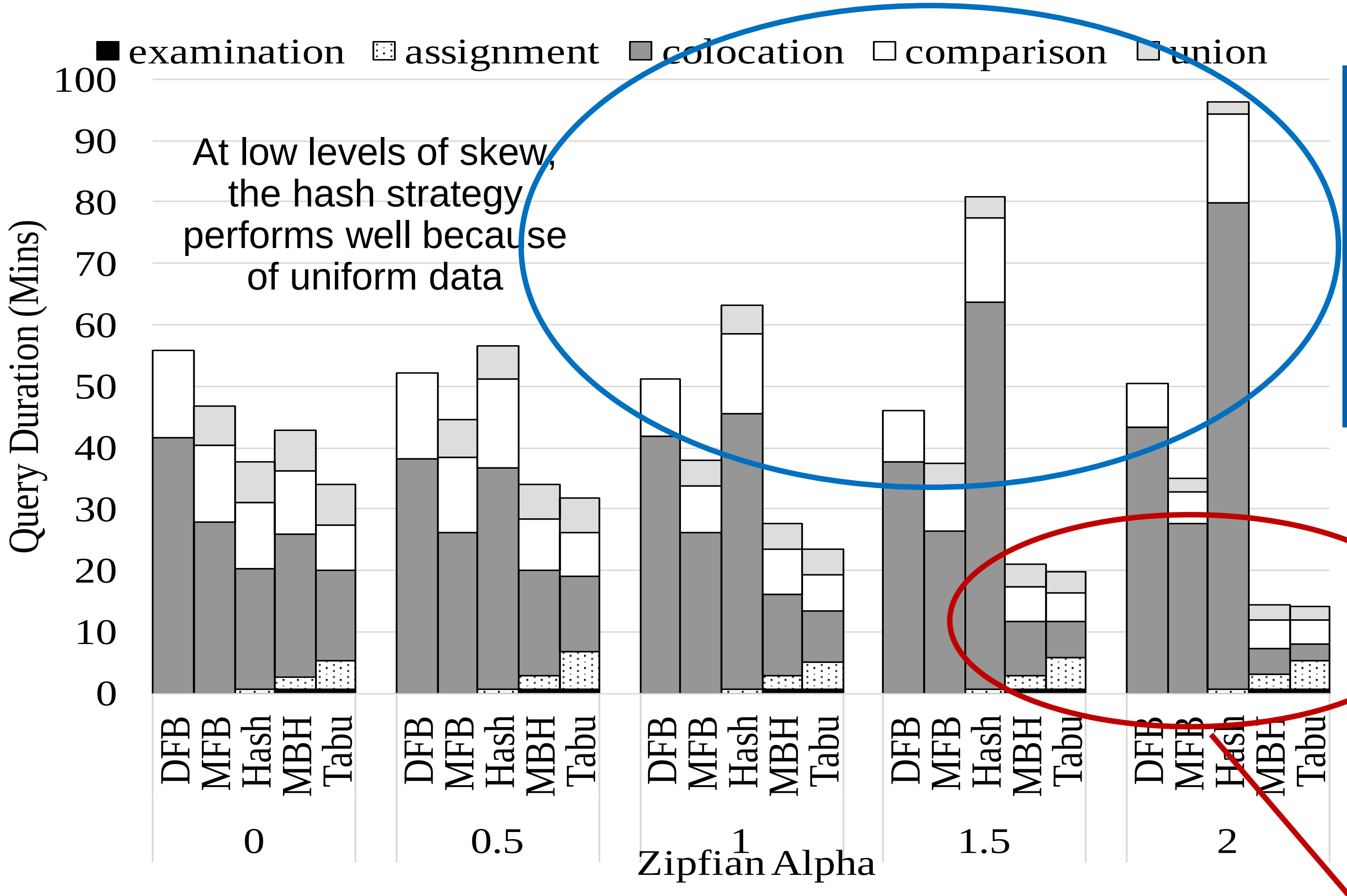
- Experiment
  - Generate data sets with known skew from a Zipf (power law) distribution ranging from uniform (α=0) to heavily skewed (α=2)
  - Considered full table scan vs sampling for understanding skew .. sampling was less expensive and resulted in good distributions.

Skew: a measure of how uneven the distribution of data is in a Data Base.

DFB: Destination Full Broadcast, MFB: Minimal Full Broadcast, MBH: Minimum bandwidth Heuristic

Shuffle Join results:

Different load balancing algorithms and data-skews



As skew increases, skew-agnostic assignments (DFB, MFB, Hash) perform much worse

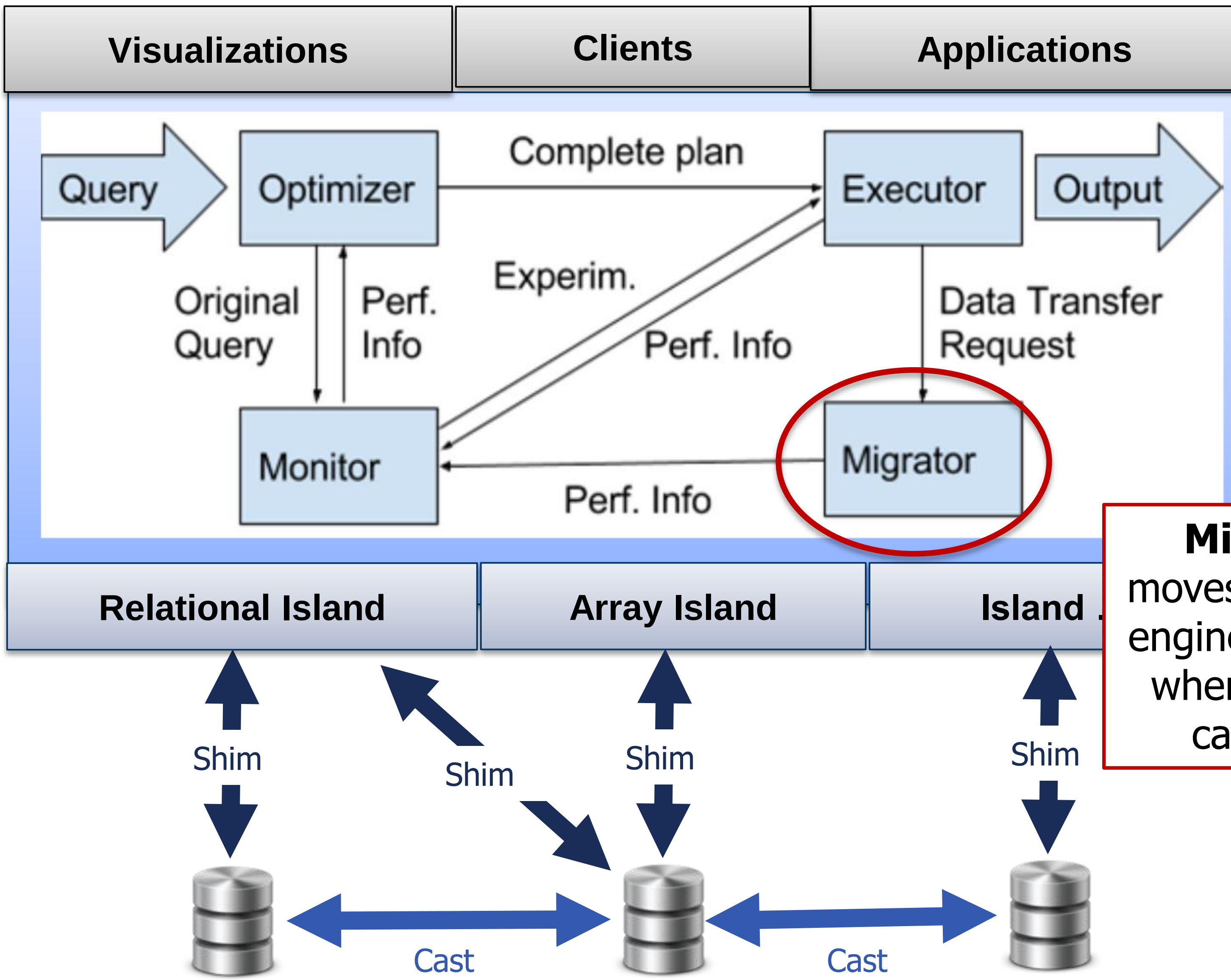
MBH and Tabu consider skew to produce a more balanced load … hence outperform other methods

2 Postgress instances running:

```
SELECT * FROM A,B WHERE A.id=B.id
```

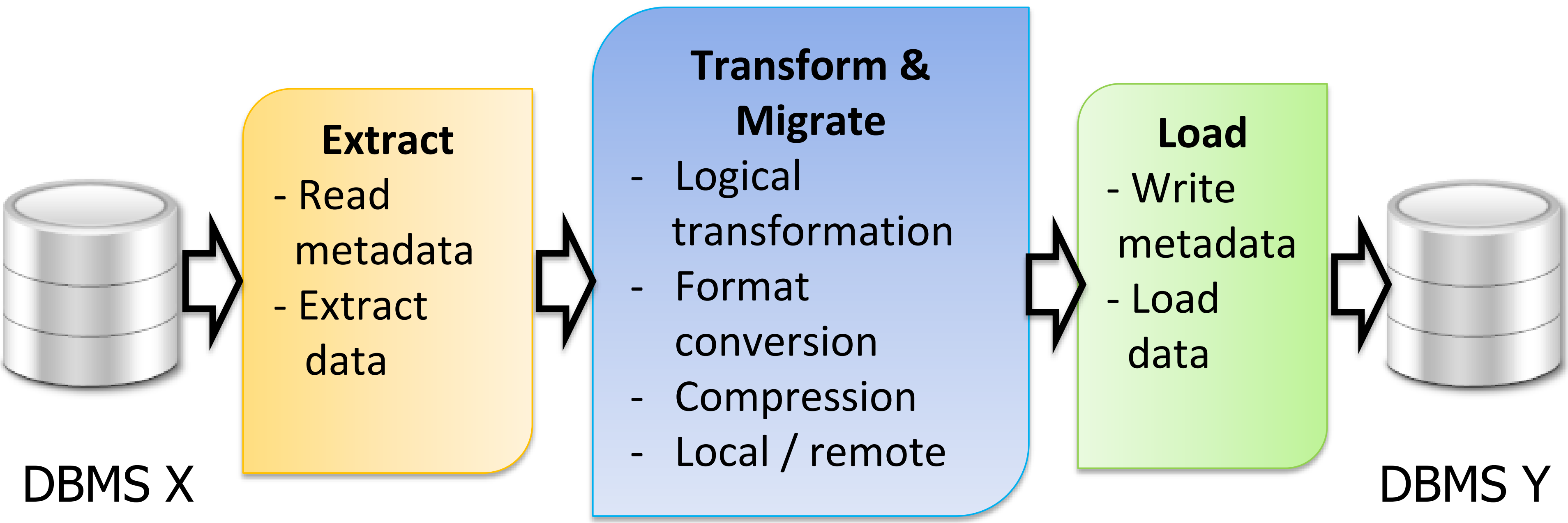
System 1: Four 3.5 GHz Intel® Xeon® cores, 8 GB, ~150 GB Data.
System 2: One 3.5 GHz Intel® Xeon® cores, 8 GB, ~75GB Data

BigDAWG Middleware



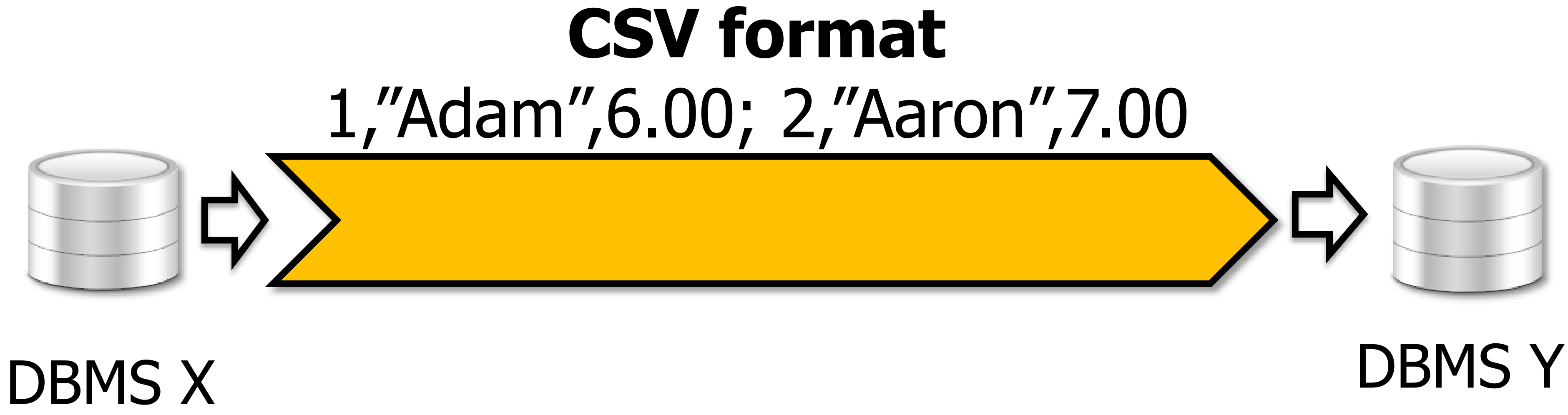
Migrator: moves data from engine to engine when the plan calls for it

Data Migrator Pipeline

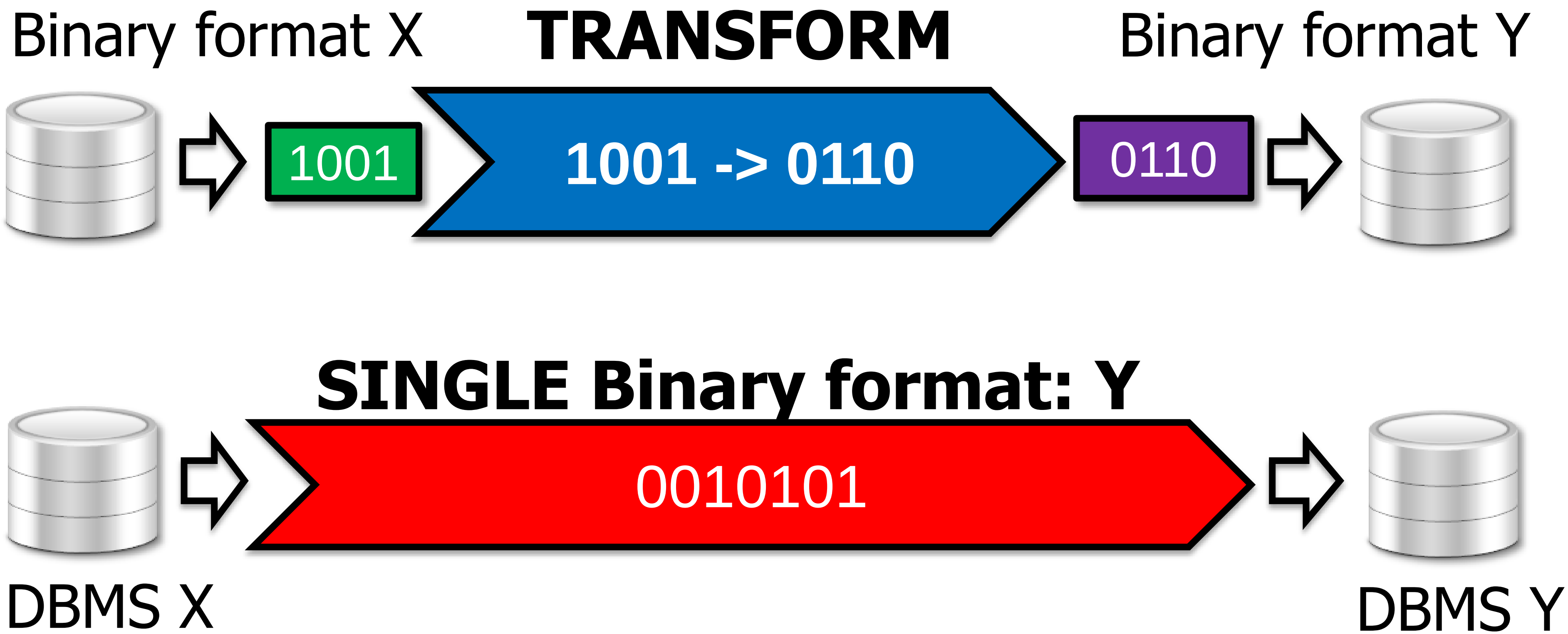


No disk materialization

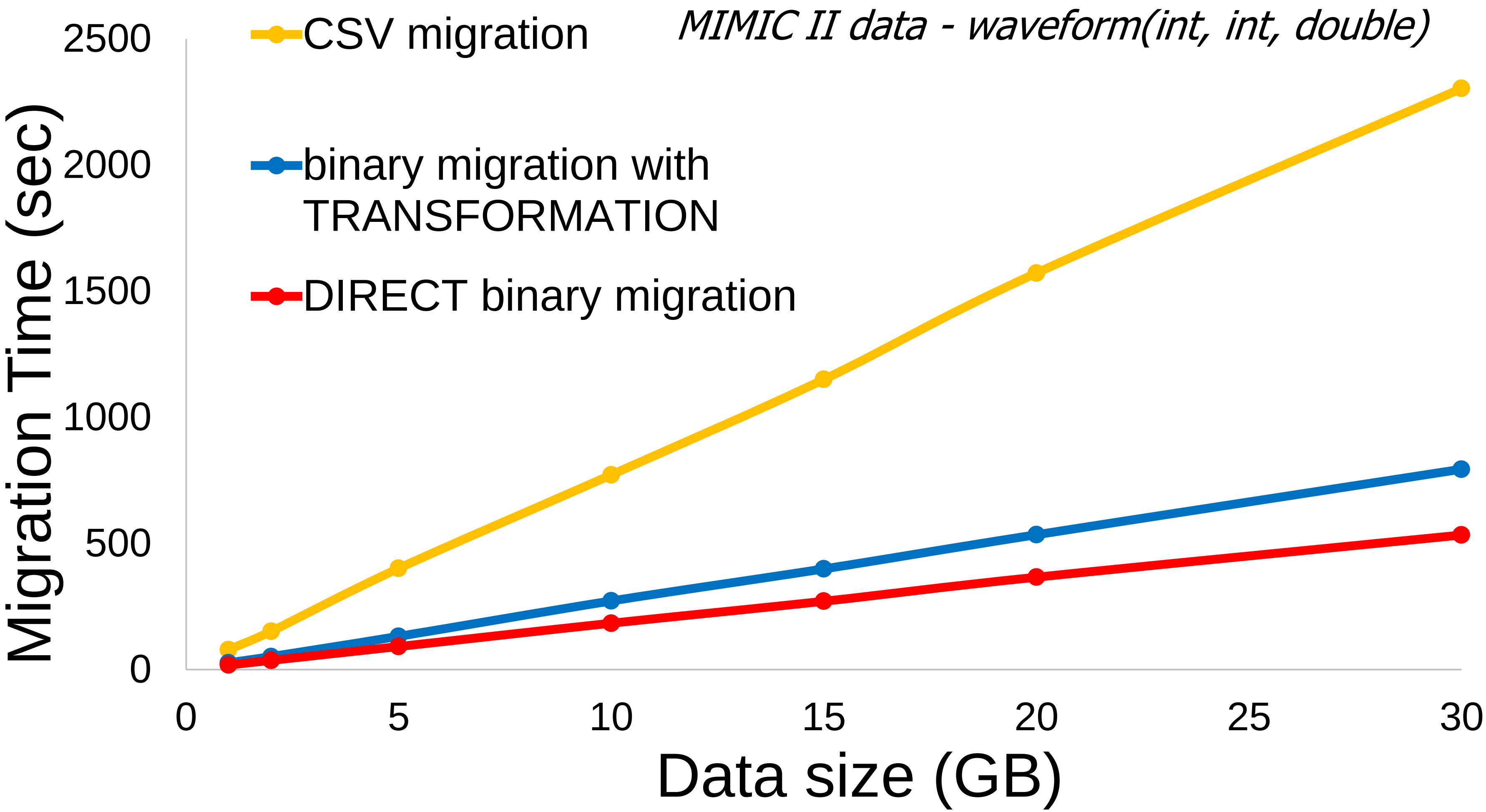
Current approach: CSV migration



Our approach: binary migration

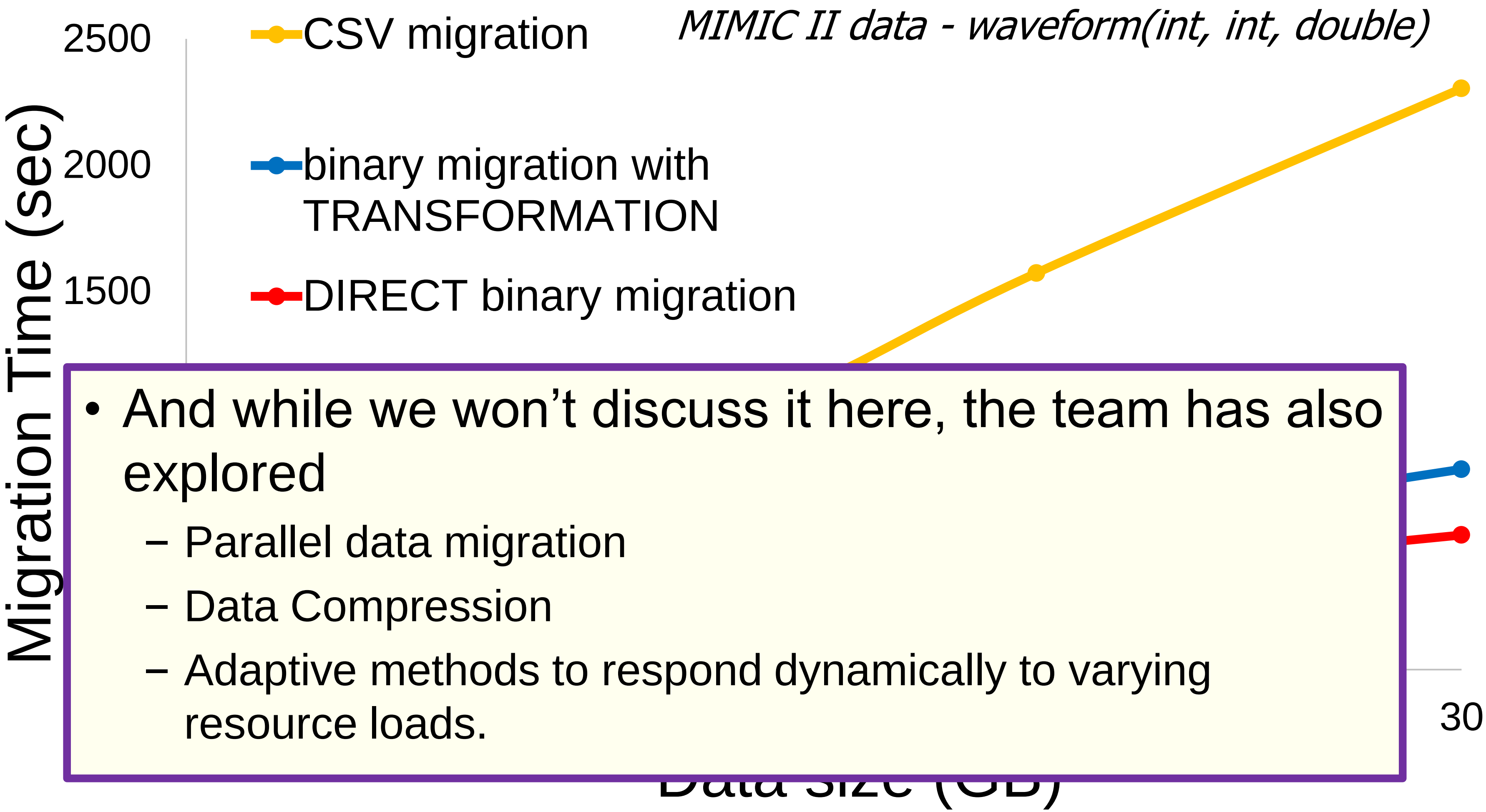


Data Migration from PostgreSQL to SciDB



TRANSFORMATION is 3X, DIRECT is 4X faster than CSV migration

Data Migration from PostgreSQL to SciDB

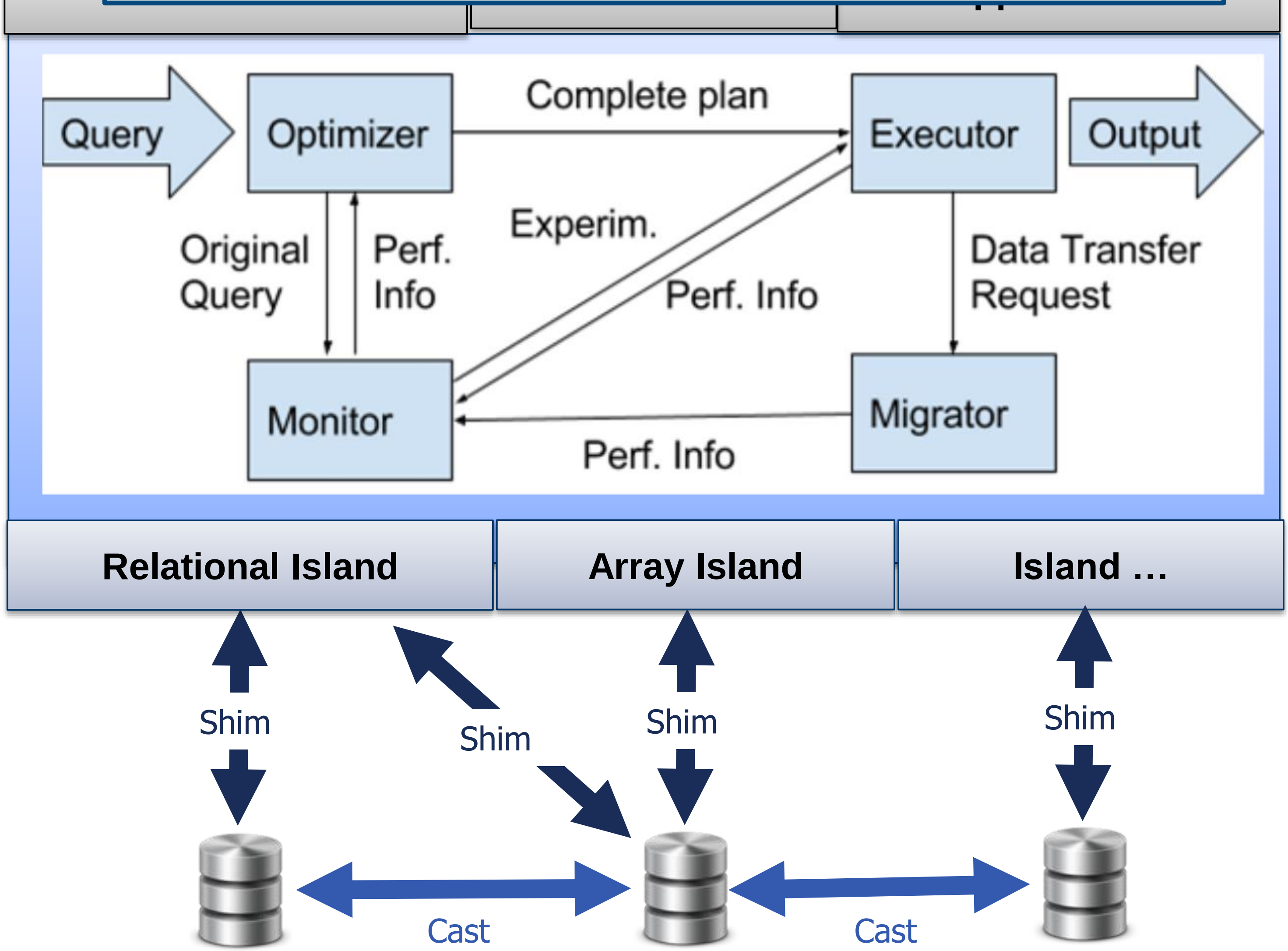


- And while we won’t discuss it here, the team has also explored
  - Parallel data migration
  - Data Compression
  - Adaptive methods to respond dynamically to varying resource loads.

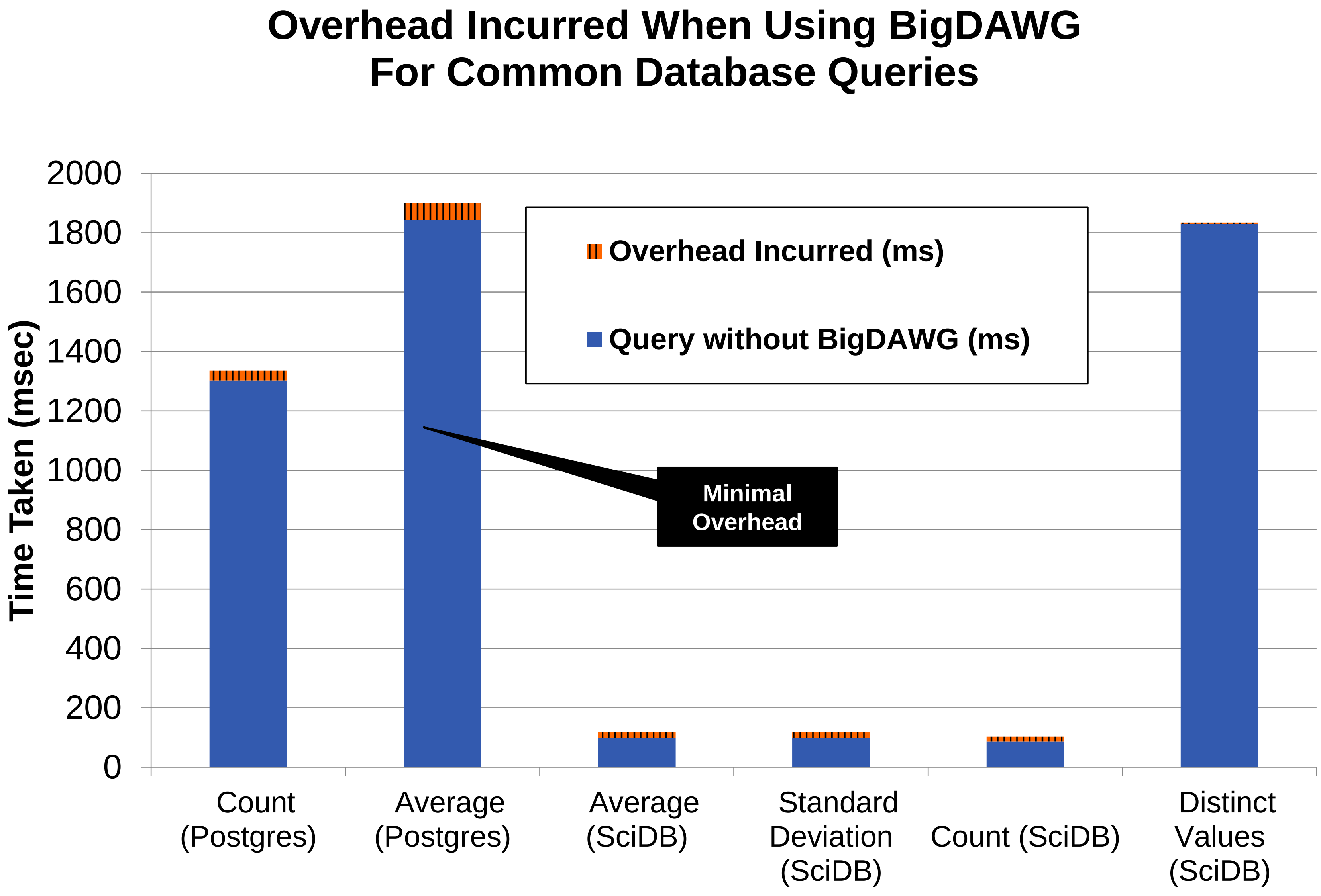
TRANSFORMATION is 3X, DIRECT is 4X faster than CSV migration

BigDAWG Middleware

How well does this work when we pull everything together into one system?



Prototype BigDAWG Overhead

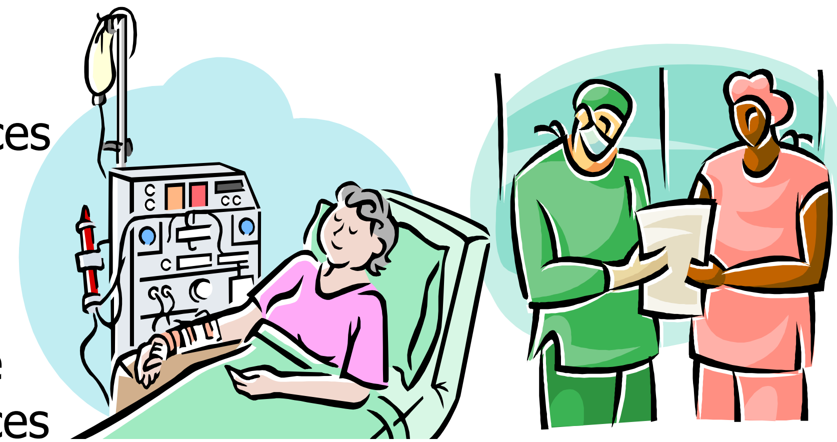


Big Data in the real world

Messy, heterogeneous, complex, streaming …

- Consider patient data in an Intensive Care Unit (e.g. MIMIC II data set*)

Arrays	EKG traces
Time Series	Blood oxygen
Time Series	Blood pressure
Arrays	EEG traces



Demographic	tables
Caregiver notes	documents
Medical charts	tables
Lab test results	tables
Xray, MRI, etc.	#images

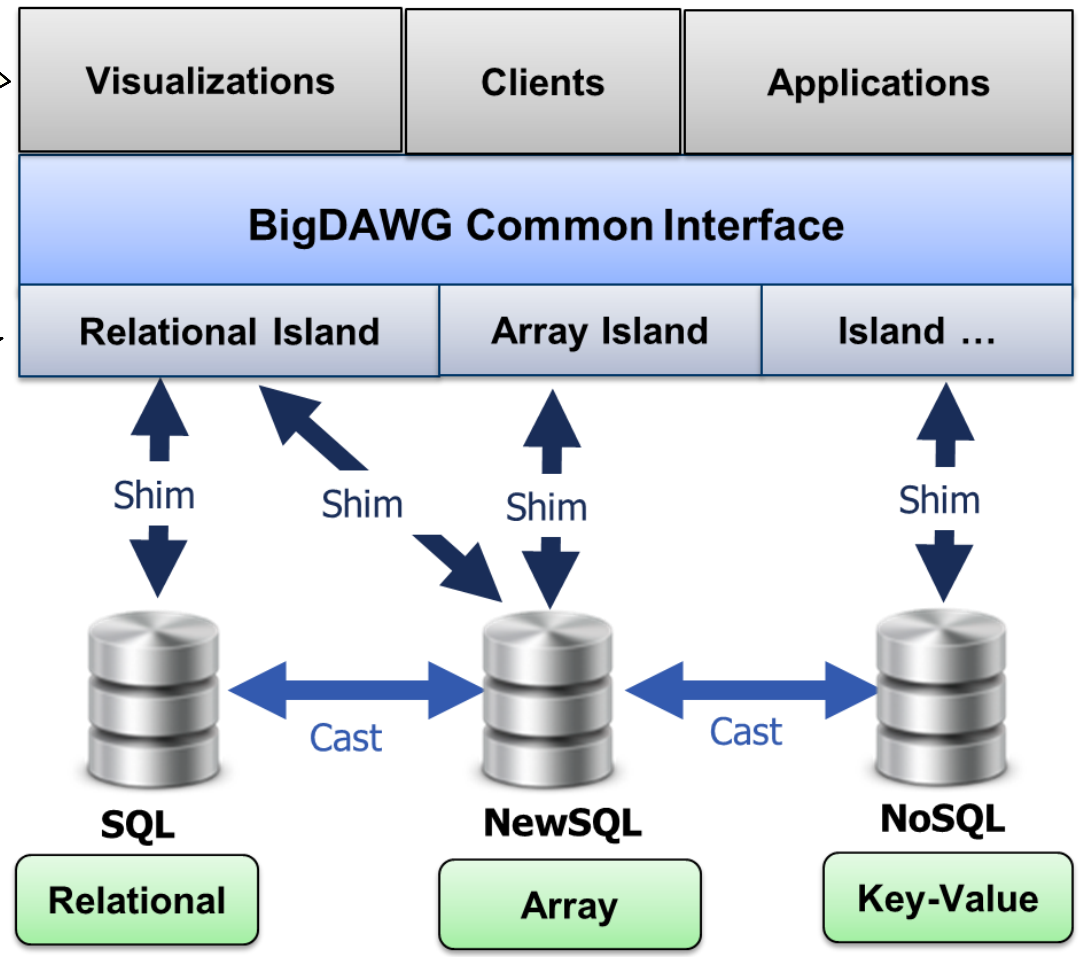
* MIMIC: Multiparameter Intelligent Monitoring in Intensive Care, http://www.physionet.org/mimic2/

MIMC doesn't include images. We are talking to several groups to add an image database to our project

BigDAWG: A Prototype Polystore System

Application level tools
- **Data Exploration**
- **Data Visualization**
- **Deep Analytics**
- **Streaming Analytics**
- **Extract-Transform-Load**

Islands
- **D4M (Associative Arrays)**
- **Myria (SQL+ Iteration)**
- **Streams**
- ***Degenerate* Islands**



SciDB

accumulo

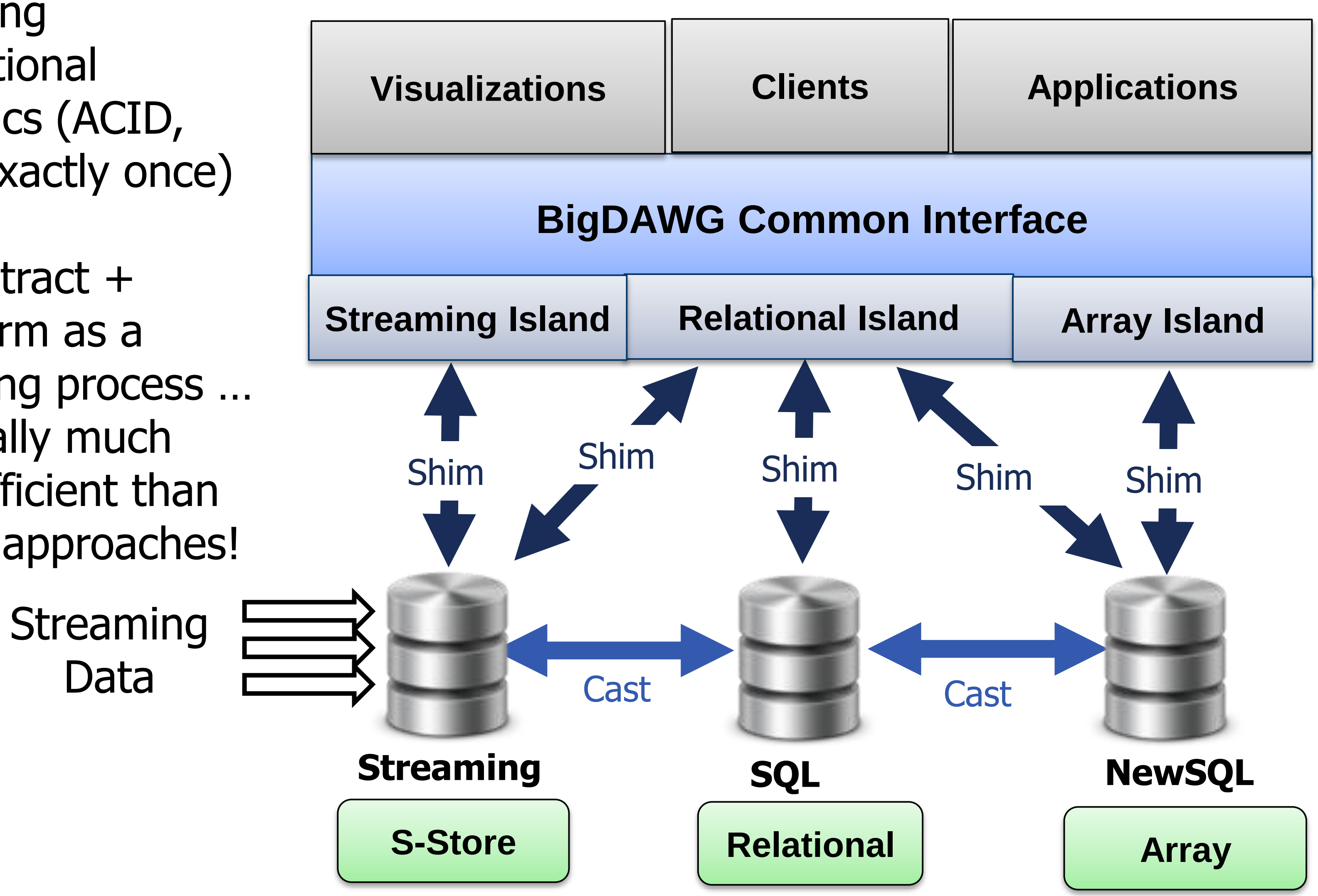
Myria

Third Party Names are the Property of their Owners.

BigDAWG Streaming Island: S-Store

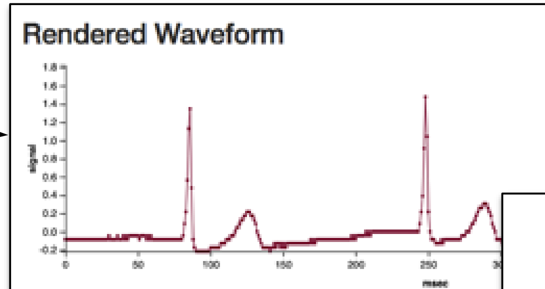
S-Store:

- A system for Streaming transactional semantics (ACID, order, exactly once)
- ETL: **E**xtract + **T**ransform as a streaming process … potentially much more efficient than current approaches!

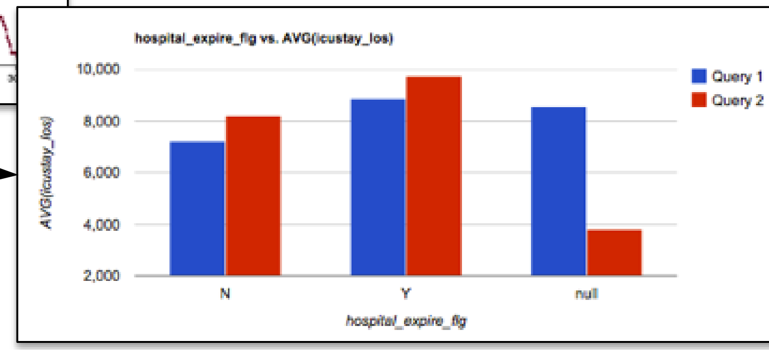


Polystore Case Study: MIMIC II Dataset

Data Explorer

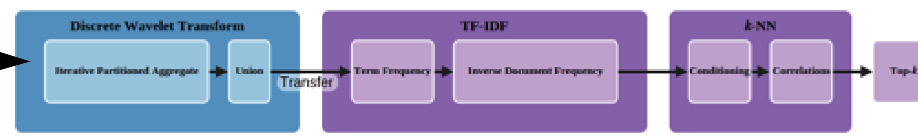


Tell Me Something Interesting

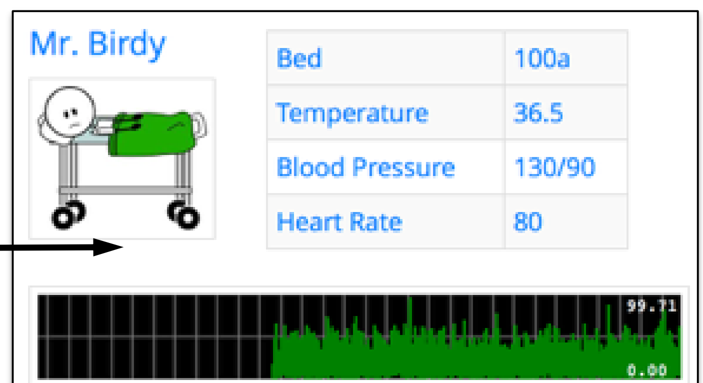


Text Analytics

Waveform Analytics

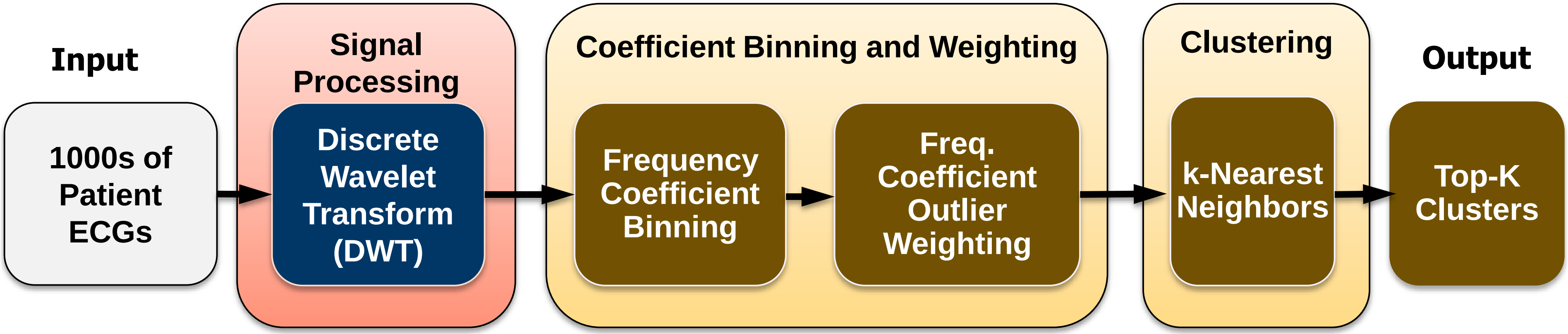


Streaming Analytics (S-Store)



Video Link

BigDAWG Polystore Waveform Analytics



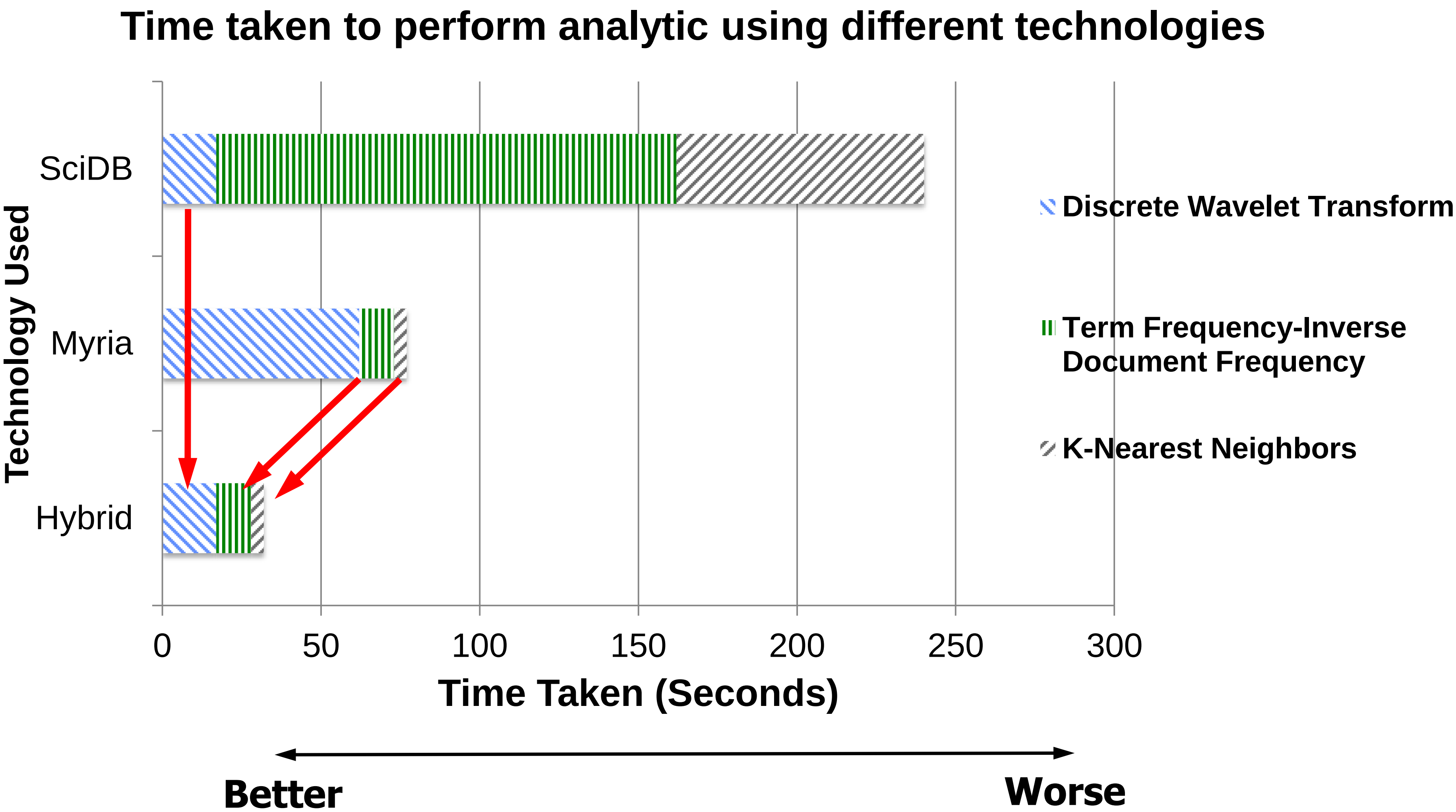
- Goal:
  - Find patients with similar ECG time-series*
- Procedure
  - Perform Discrete Wavelet Transform of ECG
  - Generate wavelet coefficient histogram
  - TF-IDF waveform coefficients (weight rare changes higher)
  - Cluster and correlate against other ECGs

- Show timings for individual components in two different DBMS scenarios
  - Option 1: Do everything in one DB
  - Option 2: Use the DB most suited for each component
    - Tough without coordinator SW
    - Incur inter-database cast operation overhead

* *A novel method for the efficient retrieval of similar multiparameter physiologic time series using wavelet-based symbolic representations*, Saeed & Mark, AMIA 2006

TF-IDF=Term Frequency-Inverse Document Frequency

Polystore Analytics Performance



Third Party Names are the Property of their Owners.

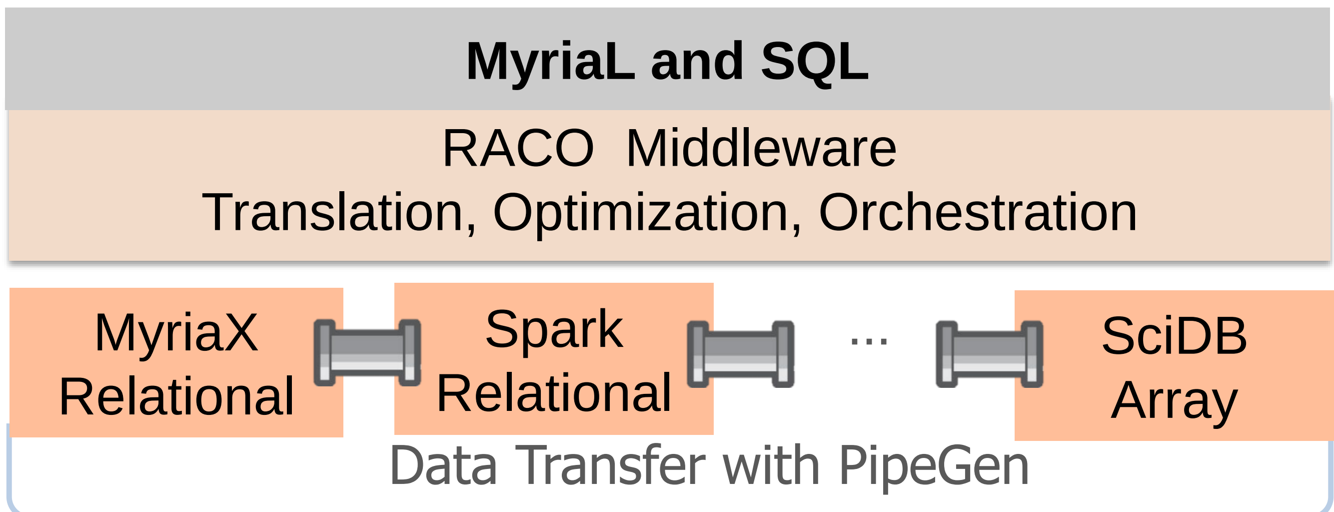
Future work

- Open source release of BigDAWG Q1’2017
- Explore Features in Myria that can help BigDAWG

Myria: A stack for Big Data Analytics

Myria is a Polystore system … emphasizes location independence

- Supports operations across multiple data stores
- Includes its own query execution engine MyriaX
- Occupies a hybrid relational-array Island in BigDAWG



Myria is a Cloud Service

- Deployed in the Amazon cloud
- Focus on efficiency and productivity
- Tested on applications from multiple scientific domains

http://myria.cs.washington.edu

Future work

- Open source release of BigDAWG Q1’2017
- Explore Features in Myria that can help BigDAWG
  - Cloud infrastructure
  - Web front end
  - Automatically generated casts with Mryia PipeGen
- Explore probabilistic data structures in the executor to further reduce data transfers.
- Explore additional datasets to stress-test the system
  - Ocean Metagenomics work underway
- Add new Islands
  - TileDB, Tuppleware (from Brown)
- Build on S-Store work to support ETL capabilities in BigDAWG.

Conclusion

- The future belongs to polystore systems
  - A single high level data management system that is composed of many individual storage management systems.
    - Storage management matches the data for a better performance.
    - Analytics embedded into the storage managers to keep computing near the data.
- BigDAWG is an effective Prototype to prove the concept.
  - There is a great deal of work needed to turn it into a general purpose tool for data scientists.
  - Early results, however, are encouraging

Research topics included in the workshop:

- New Computational Models for Big Data
- Languages/Models for integrating disparate data (e.g. graphs, arrays, relations)
- Query evaluation and optimization in federated or polystore systems
- High Performance/Parallel Computing Platforms for Big Data
- Integration of HPC and Big Data platforms
- Data Acquisition, Integration, Cleaning, and Best Practices
- Complex Big Data Applications in Science, Engineering, Medicine, Healthcare, Finance, Business, Transportation, Retailing, Telecommunication, Government and Defense applications
- Efficient data movement and scheduling, failures and recovery for analytics

Keynotes

Luna Dong
of Amazon

Fatma Ozcan
of IBM

Details

October 10, 2016:
Full workshop papers submission deadline

November 15, 2016:
Camera-ready of accepted papers

December 5-8, 2016:
Workshops Dates

Contact:
Vijay Gadepally
(vijayg@mit.edu)

References (All in the HPEC'2016 Proceedings)

- **The BigDAWG Polystore System and Architecture** *Vijay Gadepally, Peinan Chen (MIT), Jennie Duggan (Northwestern University), Aaron Elmore (University of Chicago), Brandon Haynes (University of Washington), Jeremy Kepner, Samuel Madden (MIT), Tim Mattson (Intel), Michael Stonebraker (MIT)*

- BigDAWG Polystore Query Optimization Through Semantic Equivalences *Zuohao She, Surabhi Ravishankar, Jennie Duggan (Northwestern University)*

- **The BigDawg Monitoring Framework *Peinan Chen, Vijay Gadepally, Michael Stonebraker (MIT)***

- **Cross-Engine Query Execution in Federated Database Systems *Ankush M. Gupta, Vijay Gadepally, Michael Stonebraker (MIT)***

- **Data Transformation and Migration in Polystores** *Adam Dziedzic, Aaron J. Elmore (University of Chicago), Michael Stonebraker (MIT)*

- **Integrating Real-Time and Batch Processing in a Polystore** *John Meehan, Stan Zdonik Shaobo Tian, Yulong Tian (Brown University), Nesime Tatbul (Intel), Adam Dziedzic, Aaron Elmore (University of Chicago)*