Polystore Systems for Complex Data Management

Kyle O'Brien

HPEC 2017

Massachusetts Institute of Technology

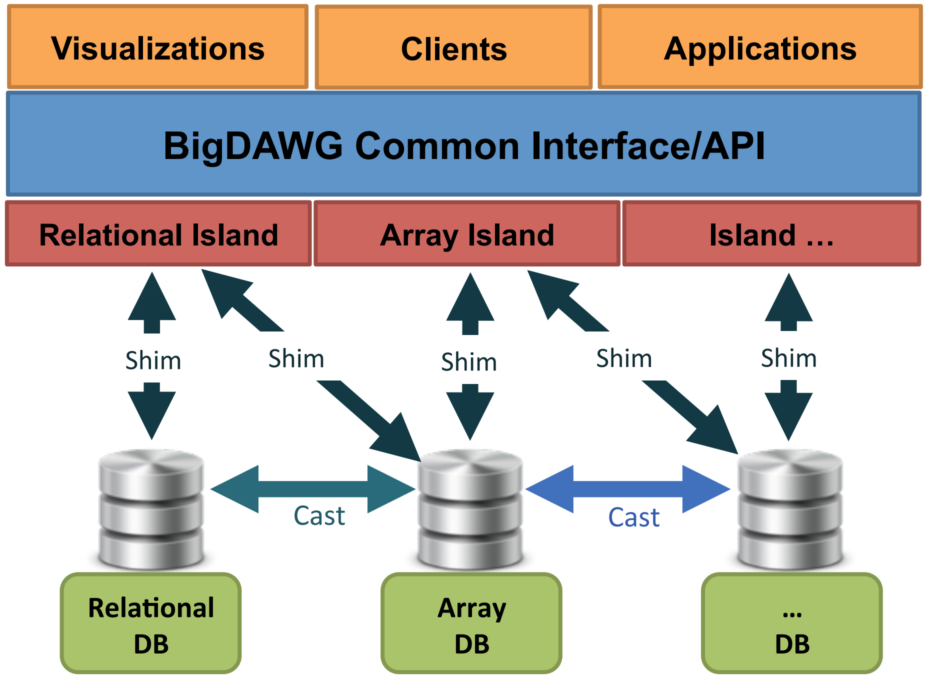
Outline and Tutorial Objectives

- **Part 1: BigDAWG Details**
  - Review basic polystore terminology
  - Common polystore motivations
  - BigDAWG API and query syntax
  - BigDAWG system components in detail
- **Part 2: Hands-on example**
  - Stand up a polystore cluster for Postgres, SciDB, and Accumulo databases using Docker
  - Explore data on each database
  - Execute polystore queries with BigDAWG

Polystore Terminology Review

- **Island:**
  - **An abstraction of database engines having a similar data model and query language.**
  - *Example: Relational Island: Postgres and MySQL have tabular data structures and operate with SQL.*
- **Shim:**
  - **A query transformation operation written in one language but intended for a separate Island. Transparently handles the intended operation on the target system in terms of another language.**
  - *Example: SQL* **`SELECT * FROM table`** *is equivalent to SciDB* **`SCAN(table)`***.*
- **Cast:**
  - **A data transformation operation from one data model to another.**
  - *Example: integers can be cast to floats or to string representations.*

BigDAWG Architecture



Islands provide the intersection of engine capabilities.
The BigDAWG Interface provides the union of Island capabilities.

Common Polystore Motivations

Analytical Optimization

- Data resides on one database engine but desired analytic operations are slow.
- We would like to select data from one database and analyze it on another database.

Location Transparency

- Data is dispersed among several database engines.
- We would like to transparently query for it without having to know the specifics of what data is on what system.

Organizational Legacy

- Data is already distributed among many databases and/or engines because policy or other legacy reasons.
- We would like to query across them while keeping data intact.

Common Polystore Motivations

Analytical Optimization

- Data resides on one database engine but desired analytic operations are slow.
- We would like to select data from one database and analyze it on another database.

Example:

Data exists in Postgres, but desired analytical operations are slow

Analytical operations in SciDB are fast

Required Steps:

1. Select data in Postgres
2. Cast data to SciDB
3. Perform filtering in SciDB

BigDAWG Query Syntax Overview

- **Currently, there is support for 3 Islands, each with its own functional token:**
  - **`bdrel()` – the query targets the relational island and uses SQL.**
  - **`bdarray()` – the query targets the array island and uses SciDB's AFL query language.**
  - **`bdtext()` – the query targets the text island and uses either SQL or D4M.**

- **Data can be explicitly cast from one Island to another with `bdcast()`:**

```
bdcast( BIGDAWG_RETRIEVAL_SYNTAX,
name_of_intermediate_result, {
    {, POSTGRES_SCHEMA_DEFINITION, relational}
    | {, SCIDB_SCHEMA_DEFINITION, array}
    | {, TEXT_SCHEMA_DEFINITION, text}} )
```

Example Polystore Query

```
bdarray(
  filter(
    bdcast(
      bdrel(
        SELECT val FROM table
      ),
      postgres_results,
      '<val:double> [i=0:*,1000,0]',
      array
    ),
    val < 35
  )
)
```

Example Polystore Query

```
bdarray(
  filter(
    bdcast(
      bdrel(
        SELECT val FROM table
      ),
      postgres_results,
      '<val:double> [i=0:*,1000,0]',
      array
    ),
    val < 35
  )
)
```

Step 1: Select data from RELATIONAL Island

Example Polystore Query

```
bdarray(
  filter(
    bdcast(
      bdrel(
        SELECT val FROM table
      ),
      postgres_results,                    # Intermediate result name
      '<val:double> [i=0:*,1000,0]',       # Destination schema
      array                                # Cast destination island
    ),
    val < 35
  )
)
```

Step 2: Cast relational data to ARRAY Island

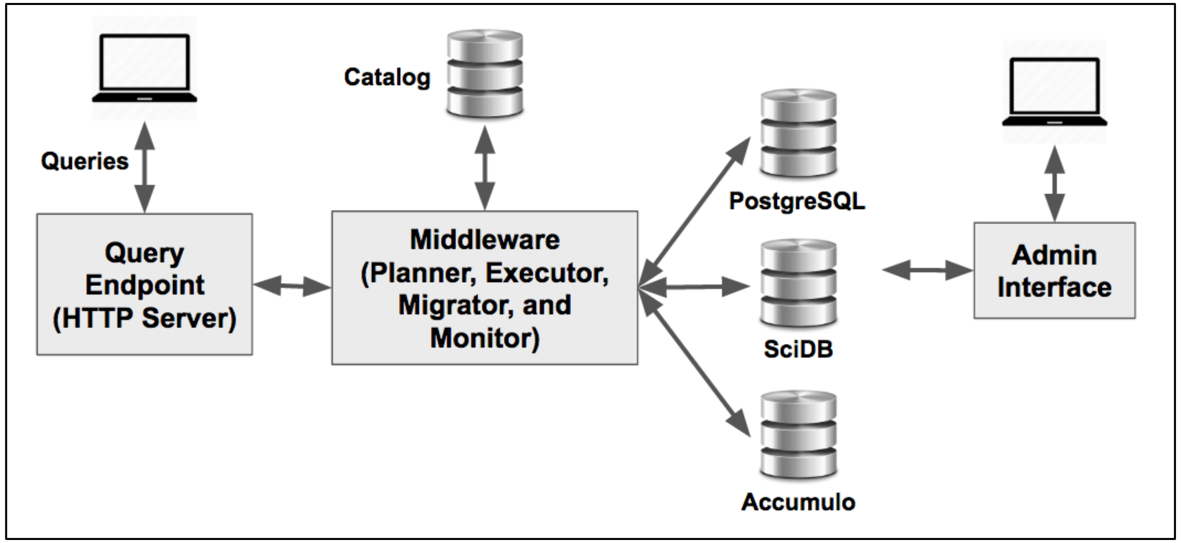
Example Polystore Query

```
bdarray(
  filter(
    bdcast(
      bdrel(
        SELECT val FROM table
      ),
      postgres_results,
      '<val:double> [i=0:*,1000,0]',
      array
    ),
    val < 35
  )
)
```

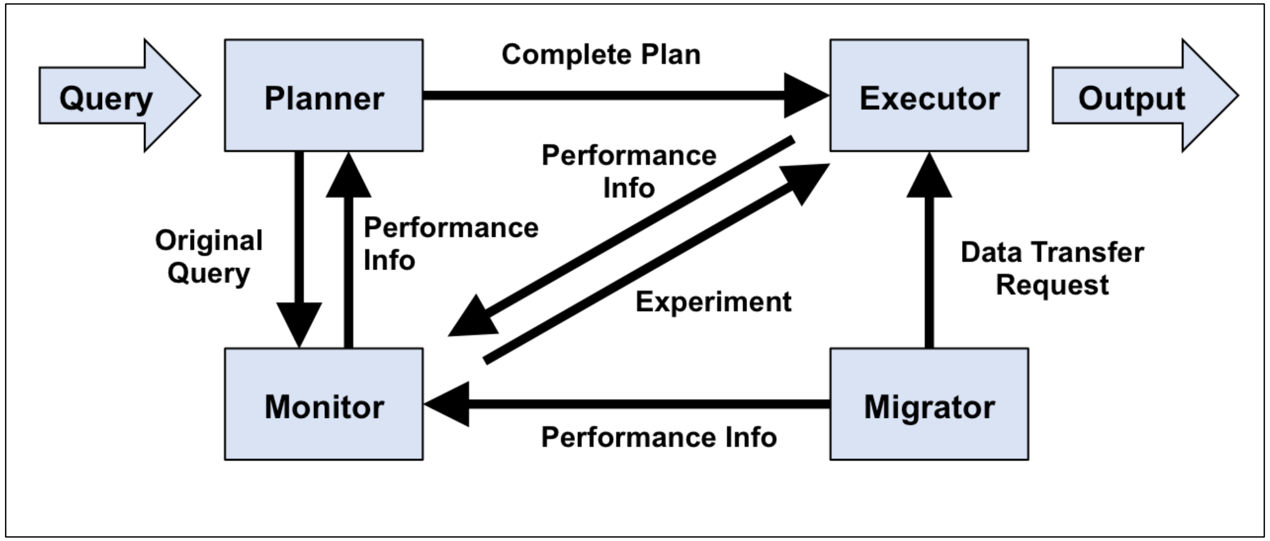
Step 3: Filter data within the ARRAY Island

BigDAWG System Overview

BigDAWG Cluster Overview



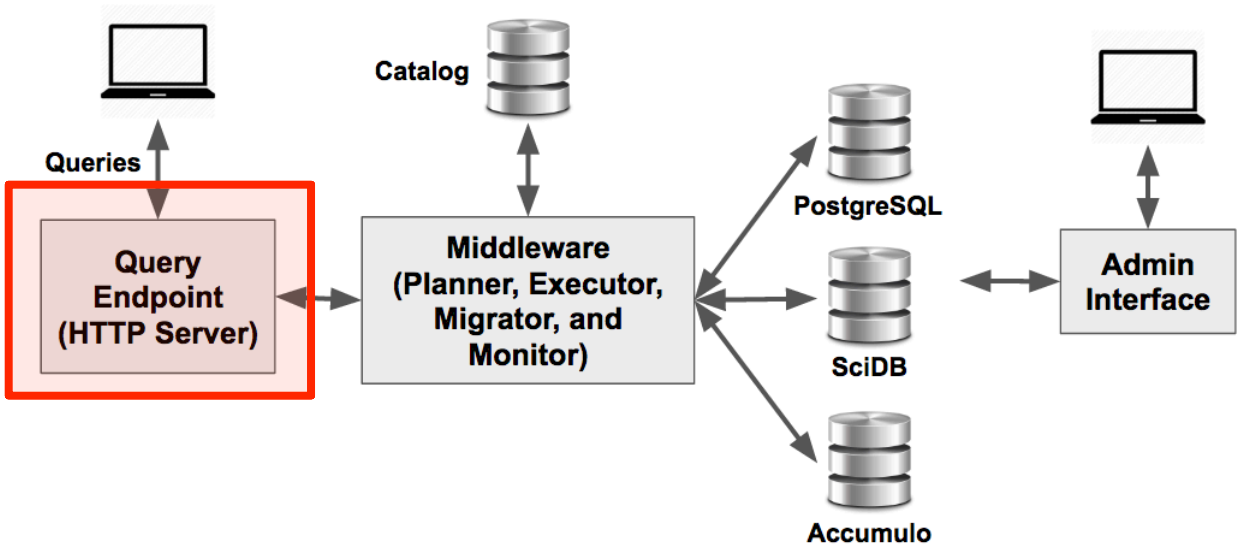
BigDAWG Middleware Components



Query Endpoint

- The Query Endpoint accepts queries from clients over a network and returns results from the Middleware.
- The Query Endpoint is a simple HTTP server that accepts POST requests

BigDAWG Cluster Overview



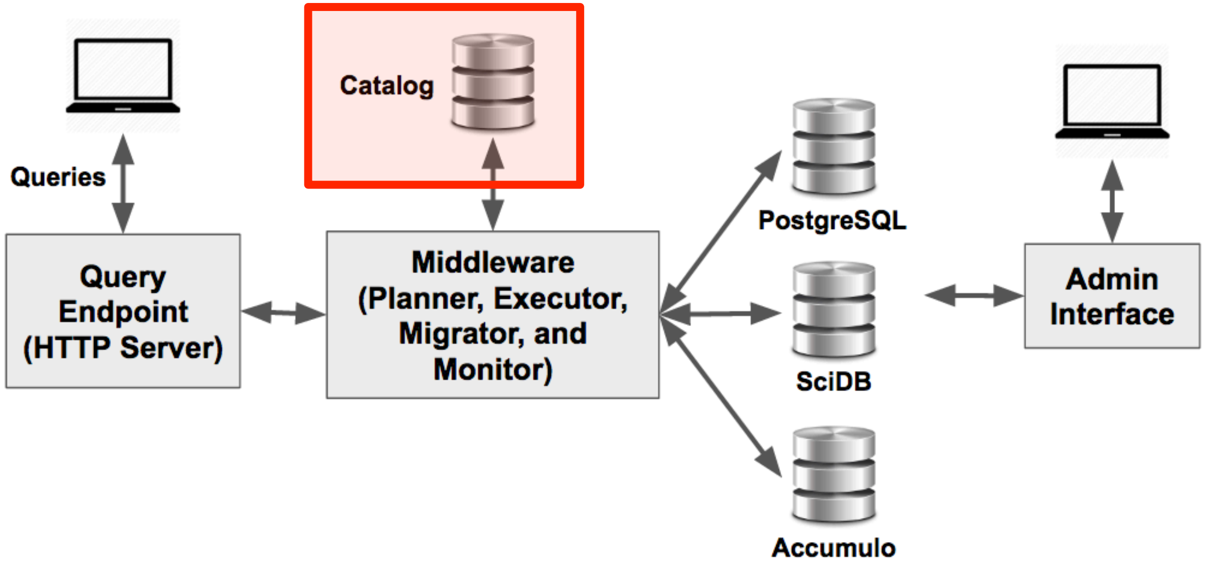
Example Usage:

```
curl -X POST -d <query string> http://192.168.99.100:8080/bigdawg/query/
```

Catalog

- **Purpose: BigDAWG must be aware of all database and data object components before it can handle queries. This information is all stored in the Catalog.**
- **Implementation: The Catalog itself is a Postgres cluster with two databases: `bigdawg_catalog` and `bigdawg_schemas`.**
- **The `bigdawg_catalog` database contains 5 tables:**
  - `catalog.engines`
  - `catalog.databases`
  - `catalog.objects`
  - `catalog.shims`
  - `catalog.casts`
- **The `bigdawg_schemas` database contains one table per data object. Each data object's schema is encoded with a Postgres table schema.**

BigDAWG Cluster Overview



Catalog: Engines, Databases, and Objects

eid [PK] serial	name character varying(15)	host character varying(40)	port integer	connection_properties character varying(100)
0	postgres0	bigdawg-postgres-catalog	5400	PostgreSQL 9.4.5
1	postgres1	bigdawg-postgres-data1	5401	PostgreSQL 9.4.5
2	postgres2	bigdawg-postgres-data2	5402	PostgreSQL 9.4.5
3	scidb_local	bigdawg-scidb-data	1239	SciDB 14.12
4	saw ZooKeeper	zookeeper.docker.local	2181	Accumulo 1.6

catalog.engines

- Engine connection properties.

dbid [PK] serial	engine_id serial	name character varying(15)	userid character varying(15)	password character varying(15)
0	0	bigdawg_catalog	postgres	test
1	0	bigdawg_schemas	postgres	test
2	1	mimic2	postgres	test
3	2	mimic2_copy	postgres	test
4	0	tpch	postgres	test
5	1	tpch	postgres	test
6	3	scidb_local	scidb	scidb123
7	4	accumulo	bigdawg	bigdawg

catalog.databases

- Database-to-engine mapping.
- Database connection properties.

oid [PK] serial	name character varying(50)	fields character varying(800)	logical_db serial	physical_db serial
0	mimic2v26.a_chartdurations	subject_id,icustay_id,itemid,	2	3
1	mimic2v26.a_iodurations	subject_id,icustay_id,itemid,	2	3
2	mimic2v26.a_meddurations	subject_id,icustay_id,itemid,	2	3
3	mimic2v26.additives	subject_id,icustay_id,itemid,	2	3

catalog.objects

- Object-to-database mapping
- Object field names.

Catalog: Schemas

- Purpose: Casting an object from one database to another requires knowledge of that object's schema
- Object schemas are stored in the `bigdawg_schemas` database.
- Example:

Data object: MIMIC II Admissions table

hadm_id [PK] integer	subject_id integer	admit_dt timestamp without time zone	disch_dt timestamp without time zone
1459	83	3424-03-21 00:00:00	3424-03-28 00:00:00
2075	3	2682-09-07 00:00:00	2682-09-18 00:00:00
5712	61	3353-01-10 00:00:00	3353-02-09 00:00:00
7149	61	3352-06-23 00:00:00	3352-07-26 00:00:00
8743	94	2656-08-18 00:00:00	2656-09-10 00:00:00

BigDAWG schema:

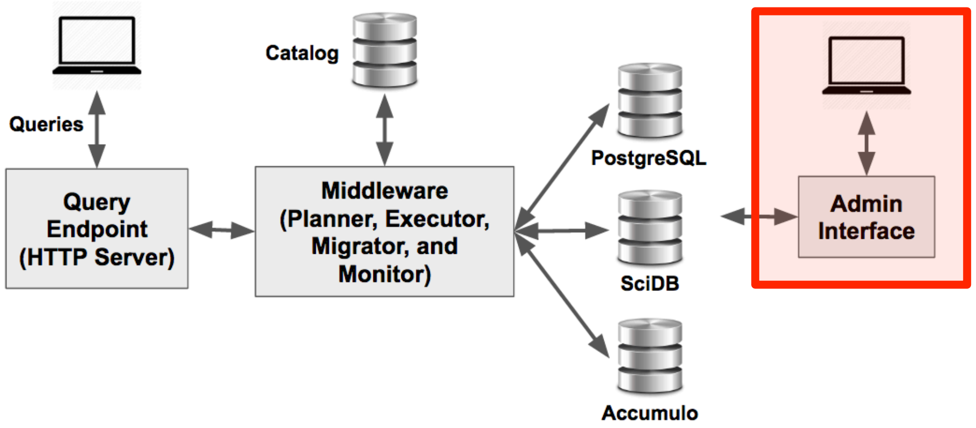
```
CREATE TABLE mimic2v26.admissions
(
  hadm_id integer,
  subject_id integer,
  admit_dt timestamp without time zone,
  disch_dt timestamp without time zone
)
```

Postgres table definitions are used to encode all object schemas

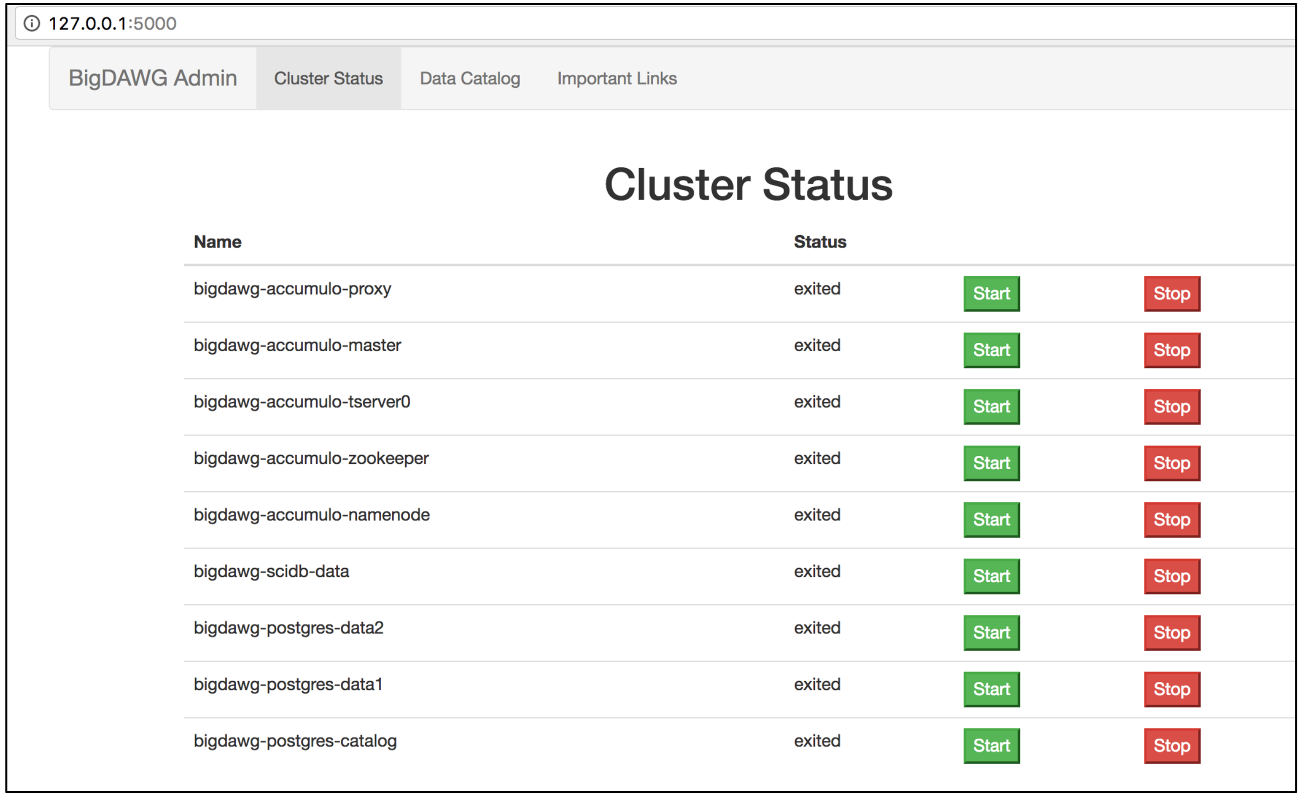
Administrative Interface

The Administrative Interface is a browser-based UI to view the Catalog and start/stop the cluster

BigDAWG Cluster Overview



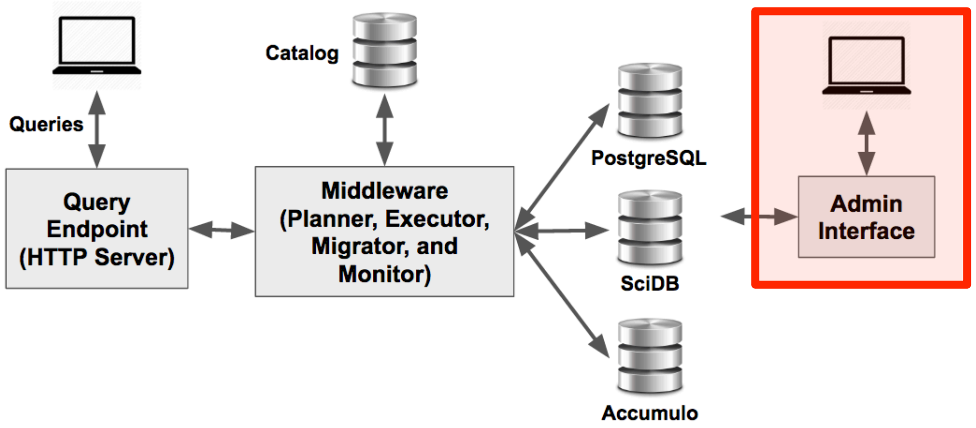
Cluster Startup and Shutdown Interface



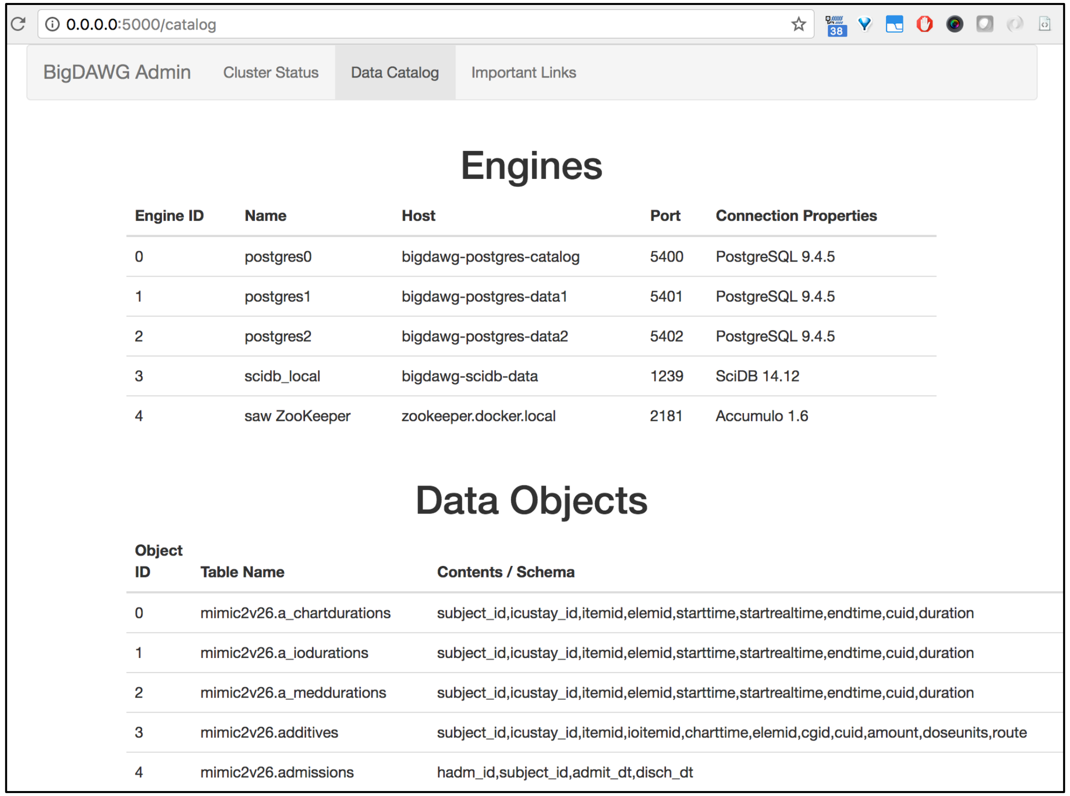
Administrative Interface

The Administrative Interface is a browser-based UI to view the Catalog and start/stop the cluster

BigDAWG Cluster Overview



Catalog Viewer



Engines

Engine ID	Name	Host	Port	Connection Properties
0	postgres0	bigdawg-postgres-catalog	5400	PostgreSQL 9.4.5
1	postgres1	bigdawg-postgres-data1	5401	PostgreSQL 9.4.5
2	postgres2	bigdawg-postgres-data2	5402	PostgreSQL 9.4.5
3	scidb_local	bigdawg-scidb-data	1239	SciDB 14.12
4	saw ZooKeeper	zookeeper.docker.local	2181	Accumulo 1.6

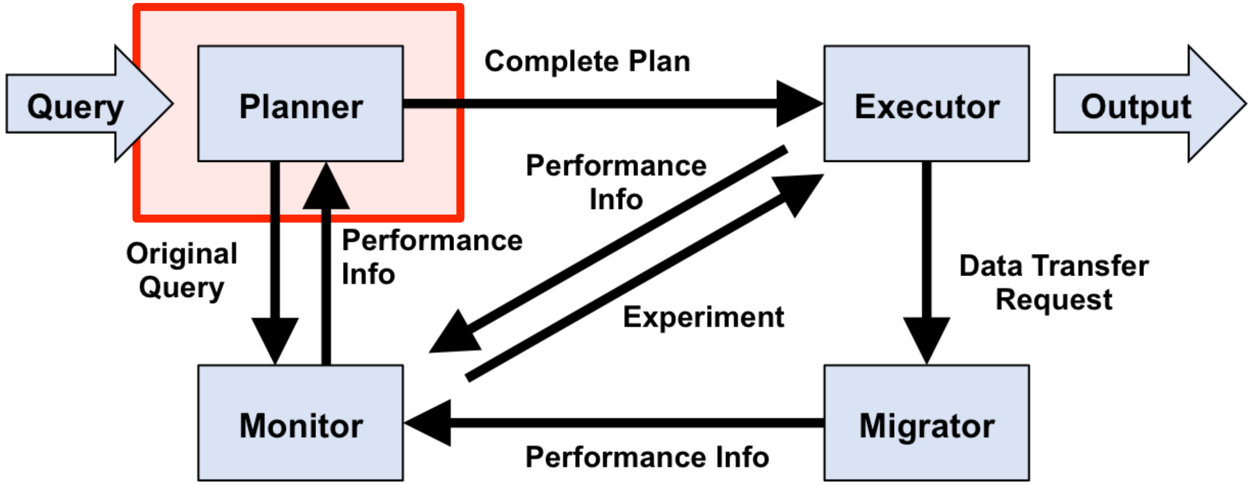
Data Objects

Object ID	Table Name	Contents / Schema
0	mimic2v26.a_chartdurations	subject_id,icustay_id,itemid,elemid,starttime,startrealtime,endtime,cuid,duration
1	mimic2v26.a_iodurations	subject_id,icustay_id,itemid,elemid,starttime,startrealtime,endtime,cuid,duration
2	mimic2v26.a_meddurations	subject_id,icustay_id,itemid,elemid,starttime,startrealtime,endtime,cuid,duration
3	mimic2v26.additives	subject_id,icustay_id,itemid,ioitemid,charttime,elemid,cgid,cuid,amount,doseunits,route
4	mimic2v26.admissions	hadm_id,subject_id,admit_dt,disch_dt

Planner

- The Planner coordinates all query execution:
  - Query parsing, planning, and optimization
  - Invokes the Executor with a final query execution plan to obtain results

Middleware Components

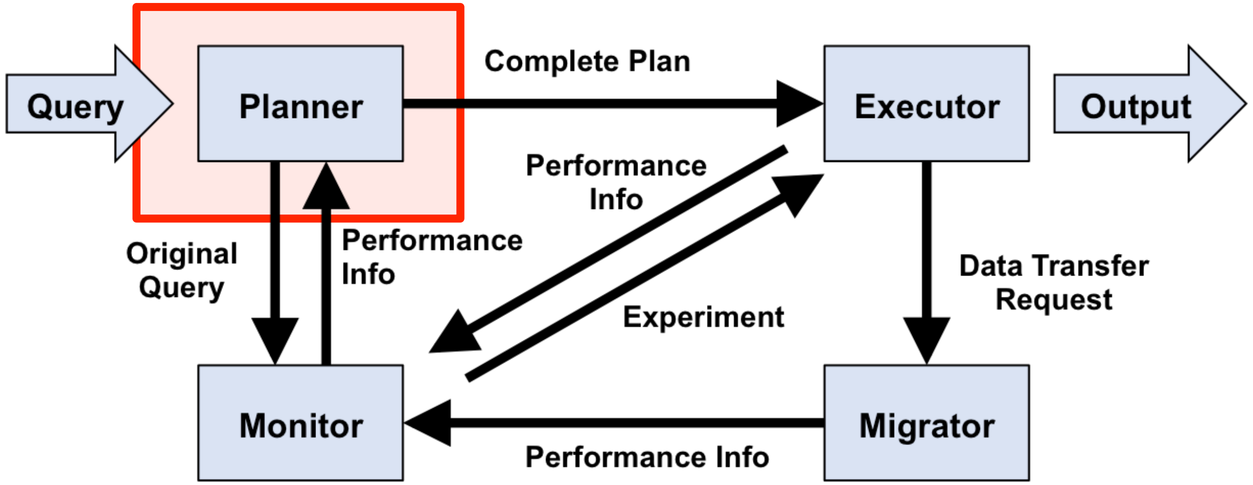


Planner Optimization

The Planner works in two modes:

- **Training Mode (pre-production learning):**
  - **Enumerate all orderings of execution steps that result in identical results (semantic equivalence).**
  - **Send plans to Monitor to gather execution metrics**
  - **Choose fastest plan**
- **Non-Training Mode (production-ready):**
  - **Extract features from the query and consult the Monitor for the best execution plan**

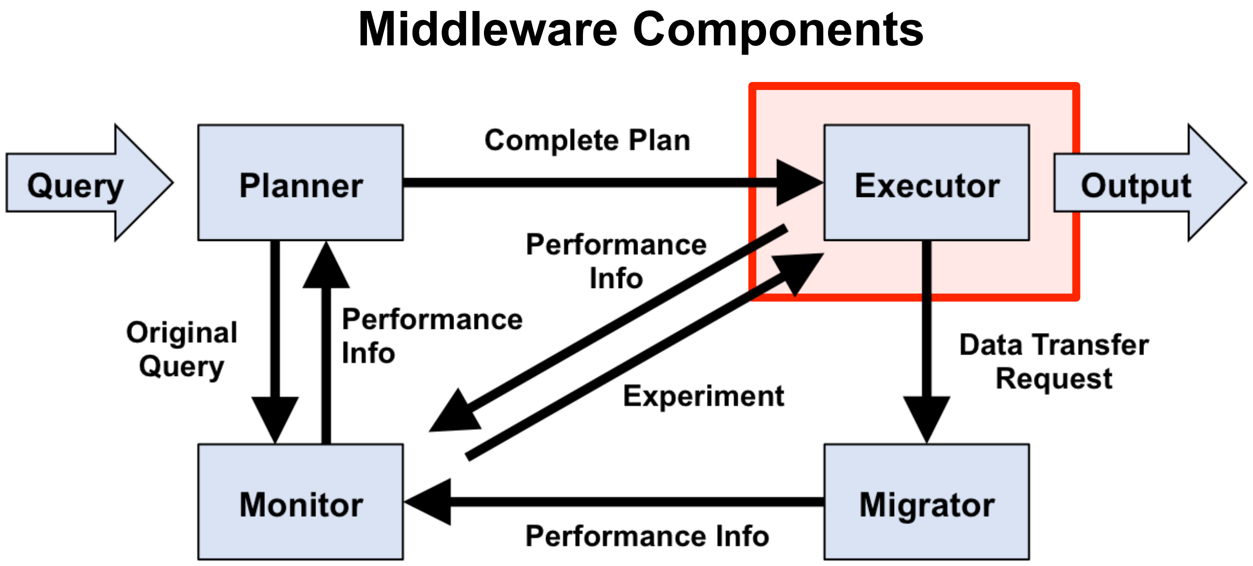
Middleware Components



Z. She, S. Ravishankar, and J. Duggan, “Bigdawg polystore query optimization through semantic equivalences,” in High Performance Extreme Computing Conference (HPEC), 2016 IEEE. IEEE, 2016, pp. 1–6

Executor

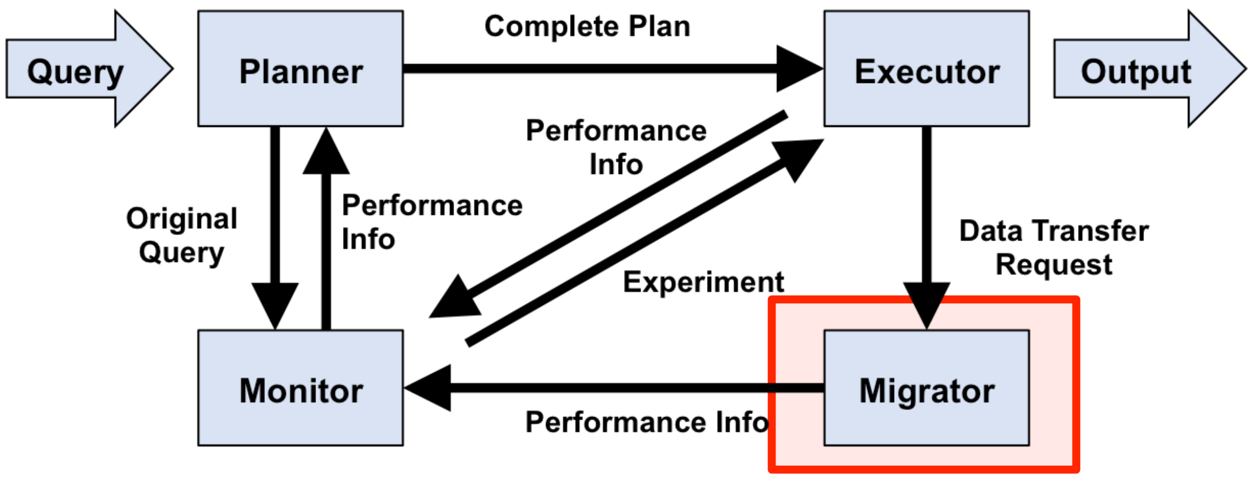
- The Executor performs query execution plans (QEPs) from either the Planner or Monitor.
- The Executor traverses a QEP and issues sub-queries to databases for intra-island tasks and invokes the Migrator when data movement is necessary.
- Performance information is sent back to the Monitor in training mode.



Migrator

- The Migrator abstractly handles data movement between two databases.
  – Called by the Executor when needed to complete a query plan.
  – Consults the Catalog to infer inter-island cast parameters

Middleware Components

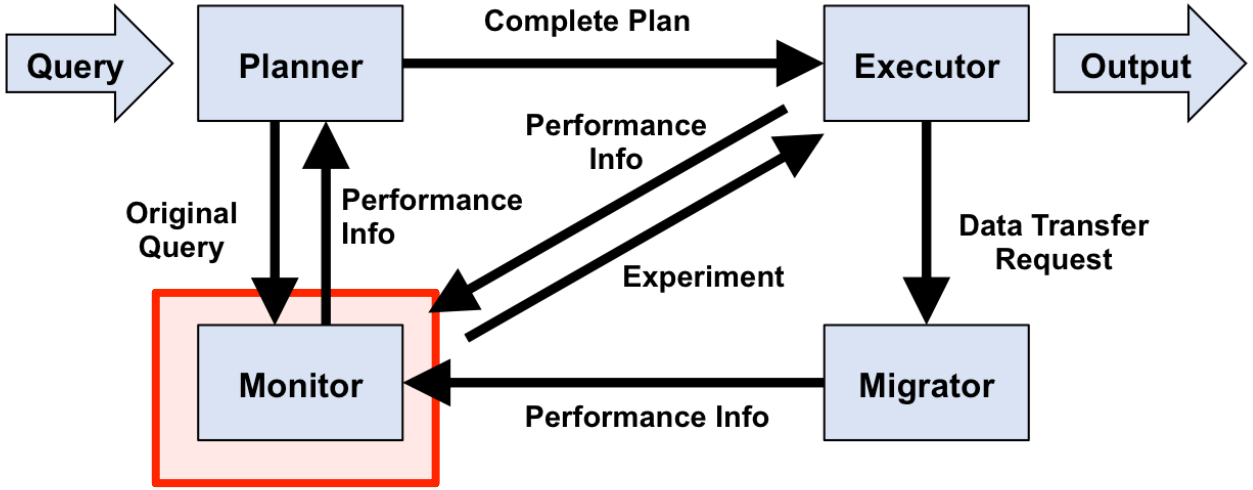


A. Dziedzic, A. J. Elmore, and M. Stonebraker, “Data transformation and migration in polystores,” in High Performance Extreme Computing Conference (HPEC), 2016 IEEE. IEEE, 2016, pp. 1–6.

Monitor

- **The Monitor tracks the execution times of query plans as they are performed by the Executor.**
- **The Monitor runs training executions to learn execution metrics, and extracts query signatures from a query plan to infer optimal execution plans for new queries.**

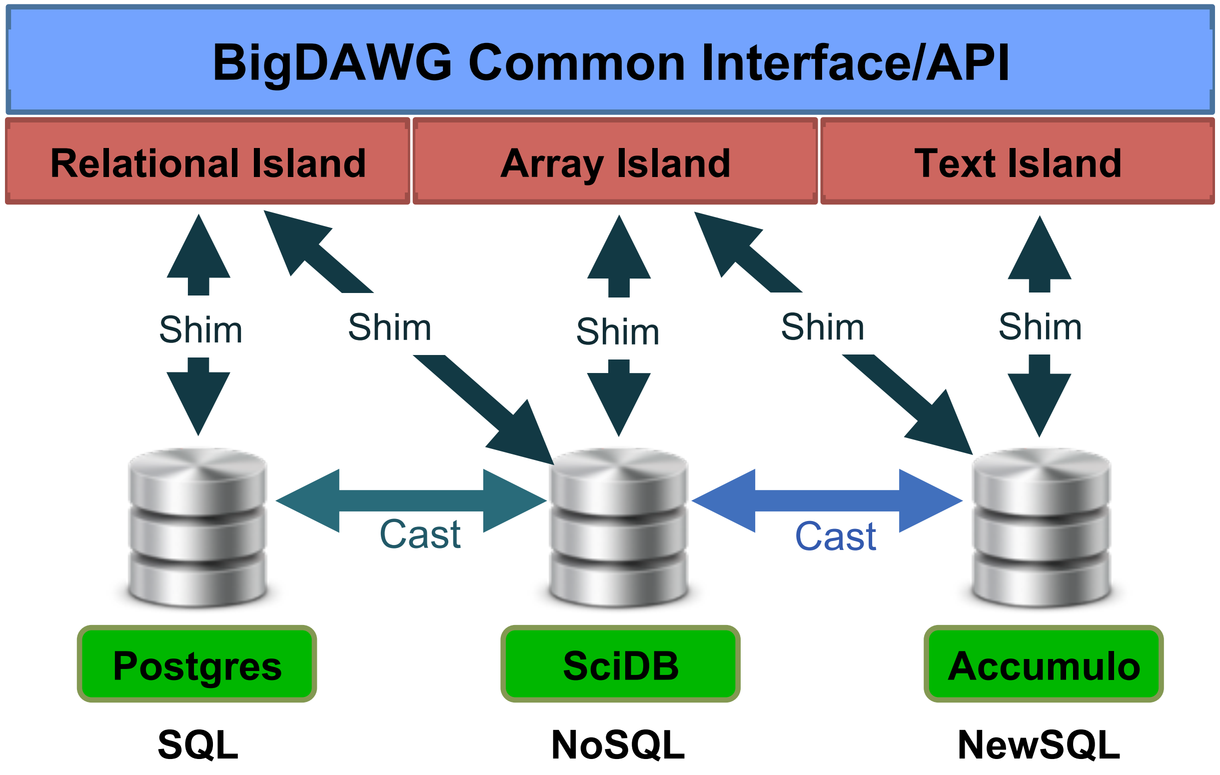
Middleware Components



P. Chen, V. Gadepally, and M. Stonebraker, "The bigdawg monitoring framework," in High Performance Extreme Computing Conference (HPEC), 2016 IEEE. IEEE, 2016, pp. 1–6.

Hands-On Demo: Cluster and Dataset Overview

Cluster Overview

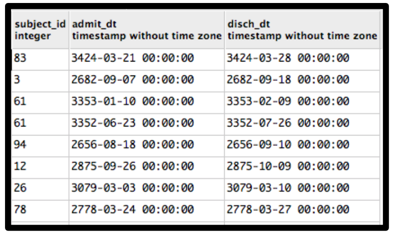


Dataset Overview

MIMIC II dataset*

- **Full dataset: over 3 terabytes (TB) total**
- **1000s of intensive care unit patients from 2001-2008**

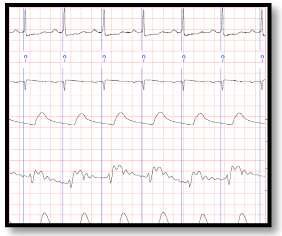
Postgres



subject_id integer	admit_dt timestamp without time zone	disch_dt timestamp without time zone
83	3424-03-21 00:00:00	3424-03-28 00:00:00
3	2682-09-07 00:00:00	2682-09-18 00:00:00
61	3353-01-10 00:00:00	3353-02-09 00:00:00
61	3352-06-23 00:00:00	3352-07-26 00:00:00
94	2656-08-18 00:00:00	2656-09-10 00:00:00
12	2875-09-26 00:00:00	2875-10-09 00:00:00
26	3079-03-03 00:00:00	3079-03-10 00:00:00
78	2778-03-24 00:00:00	2778-03-27 00:00:00

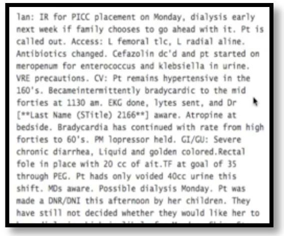
Tabular Data
Demographic information, lab test results, hospital accounting records

SciDB



Physiological Signals
Electrocardiogram (ECG) traces, blood pressure monitoring, pulse oximeter readings

Accumulo



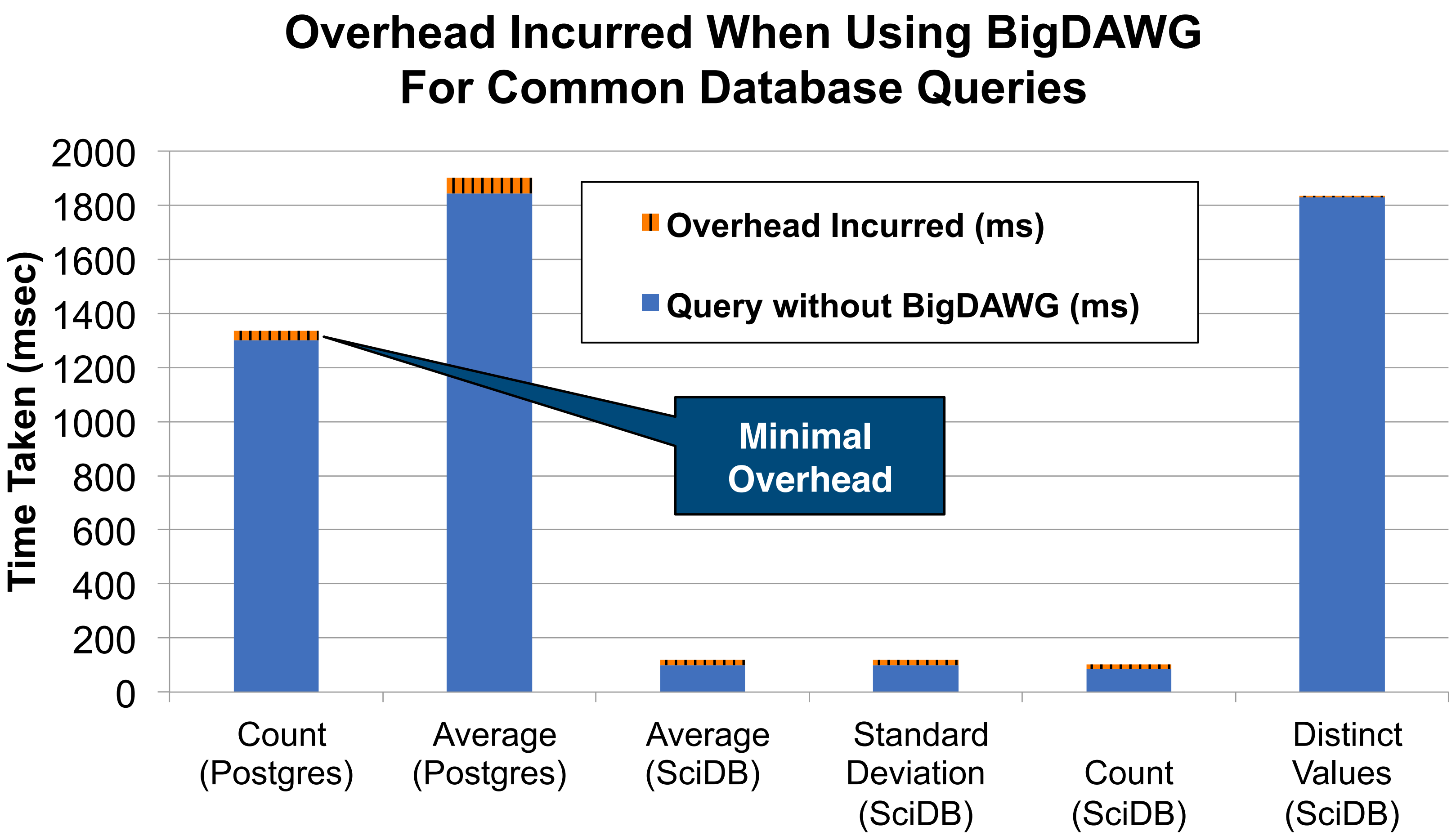
lan: IR for PICC placement on Monday, dialysis early next week if family chooses to go ahead with it. Pt is called out. Access: L femoral tlc, L radial aline. Antibiotics changed. Cefazolin dc'd and pt started on meropenum for enterococcus and klebsiella in urine. VRE precautions. CV: Pt remains hypertensive in the 160's. Becameintermittently bradycardic to the mid forties at 1130 am. EKG done, lytes sent, and Dr [**Last Name (STitle) 2166**] aware. Atropine at bedside. Bradycardia has continued with rate from high forties to 60's. PM lopressor held. GI/GU: Severe chronic diarrhea, Liquid and golden colored.Rectal fole in place with 20 cc of ait.TF at goal of 35 through PEG. Pt hads only voided 40cc urine this shift. MDs aware. Possible dialysis Monday. Pt was made a DNR/DNI this afternoon by her children. They have still not decided whether they would like her to

Freeform Text
Caregiver notes and test reports

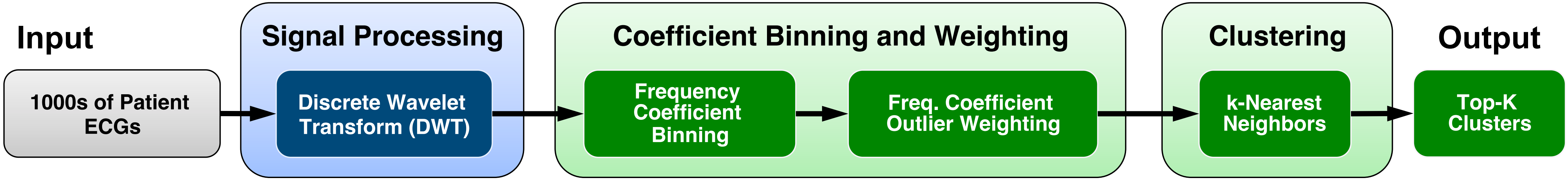
* *Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): A public-access intensive care unit database*, Saeed, et al., Crit Care Med. 2011

BACKUP

Prototype BigDAWG Overhead



BigDAWG Polystore Analytic Example



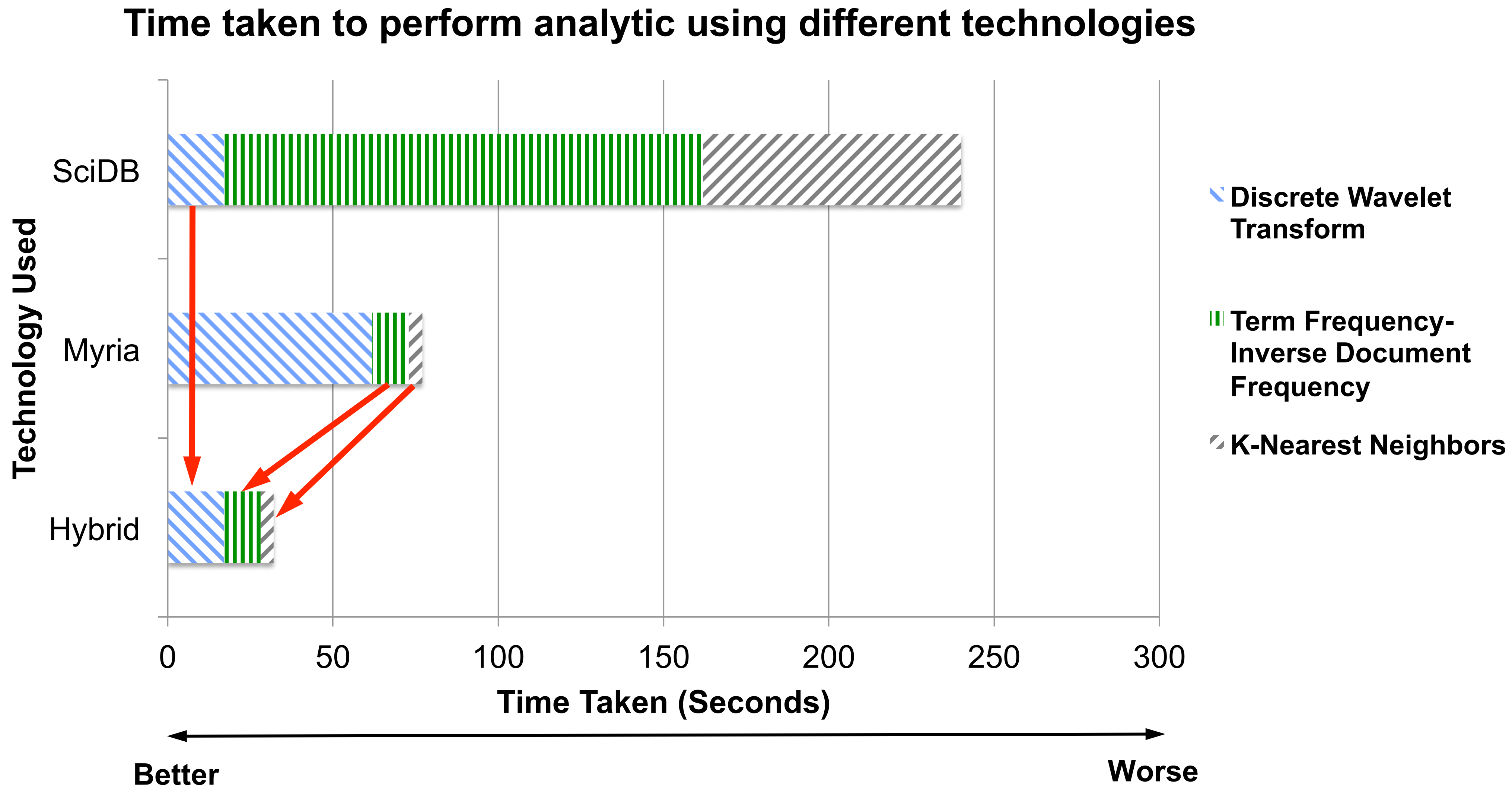
- Goal: Find patients with similar ECG time-series*
- Procedure
  - Perform Discrete Wavelet Transform of ECG
  - Generate wavelet coefficient histogram
  - TF-IDF waveform coefficients (weight rare changes higher)
  - Correlate against all other ECGs

- Show timings for individual pieces in two different types of databases
  - Option 1: Pick a DB and do the whole thing in it
    - This takes time and may not be ideal
  - Option 2: Let the best DB for each piece do their part
    - Tough without some coordinator
    - 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 Analytic Performance (1)



Medical Applications: HPEC'15

Example Polystore Query

```
bdarray(                 # Array island scope
  filter(                # Array island query operation
    bdcast(              # CAST operation
      bdrel(             # Relational island scope
        SELECT val       # Relational island query
      ),
      postgres_results,                 # Intermediate cast result name
      '<val:double> [i=0:*,1000,0]',    # Destination schema
      array              # Cast destination island
    ),
    val < 35             # Finish array island query
  )
)
```

```
RELATIONAL(
        SELECT *
        FROM  R, CAST(A, relation)
        WHERE R.v = A.v
)
```

catalog.objects table

This table lists all data objects managed by BigDAWG.

catalog.objects

oid [PK] serial	name character varying(50)	fields character varying(800)	logical_db serial	physical_db serial
0	mimic2v26.a_chartdurations	subject_id,icustay_id,itemid,	2	3
1	mimic2v26.a_iodurations	subject_id,icustay_id,itemid,	2	3
2	mimic2v26.a_meddurations	subject_id,icustay_id,itemid,	2	3
3	mimic2v26.additives	subject_id,icustay_id,itemid,	2	3

logical_db and physical_db:

In case an object was created on one database and migrated to another, we track the initial "logical" database ID and the current "physical" database ID to know the original schema.

- **Columns:**
  - oid: Object ID. The primary key.
  - name: Name of the data object (i.e., table name)
  - fields: comma-separated list of field names in the data object
  - logical_db: references catalog.databases.dbid, the original database where the object was created.
  - physical_db: references catalog.databases.dbid, the current location of the data object.

catalog.shims table

catalog.shims

shim_id [PK] serial	island_id serial	engine_id serial	access_method character varying(30)
0	0	0	N/A
1	0	1	N/A
2	0	2	N/A
3	1	3	N/A
4	2	4	N/A

- Columns:
  - shim_id: primary key
  - island_id: ID for island
  - engine_id: references catalog.databases.eid to indicate the engine
  - access_method: To account for any version-specific language syntax (not currently used).