Federated Analytics & the Need for Automatically-Generated Connectors

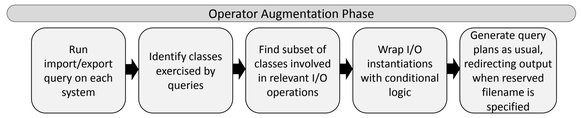
Brandon Haynes, Alvin Cheung, Magdalena Balazinska

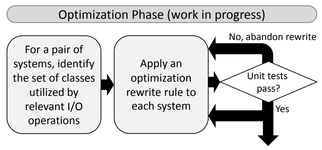
Overview

1. We demonstrate the performance benefit of federated analytics.
2. Federated analytics requires efficient data transfer between analytics systems. We demonstrate a new approach to automatically generate highly-efficient data transfer connectors between arbitrary pairs of single-node or shared-nothing systems.
3. We use a real use-case (Saeed & Mark, 2006) that aims to identify temporally similar patients over the MIMIC-II dataset. The goal is to classify whether a patient is stable or not by finding the k patients with the most similar ECG recordings.

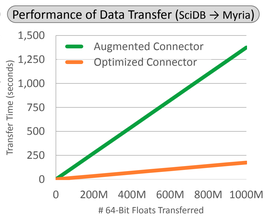
Automatic Connector Generation

Goal: Automatically generate specialized operators that transfer binary data, potentially in parallel, between systems during the execution of federated query plans



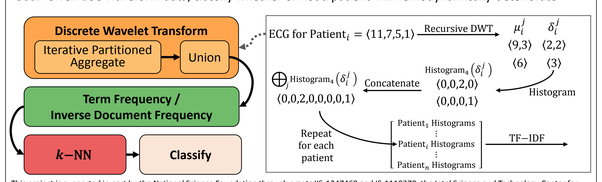


System	LOC Modified	Classes Modified
	Unoptimized (Optimized)	
SciDB	24 (27)	2 (2)
Myria	8 (54)	2 (7)



Analytics Use-Case: Identifying Similar Patients

Goal: Given ECG waveform data, classify whether or not a patient will hemodynamically-deteriorate



This project is supported in part by the National Science Foundation through grants IIS-1247469 and IS-1110370, the Intel Science and Technology Center for Big Data, and a gift from Amazon.

A Demonstration of the BigDAWG Polystore System

Aaron J. Elmore, Jennie Duggan, Michael Stonebraker, Magdalena Balazinska, Ugur Cetintemel, Vijay Gadepally, Jeffrey Heer, Bill Howe, Jeremy Kepner, Tim Kraska, Sam Madden, Dave Maier, Timothy Mattson, Stavros Papadopoulos, Jeff Parkhurst, Nesime Tatbul, Kristin Tufte, Manasi Vartak, and Stan Zdonik

The BigDAWG PolyStore

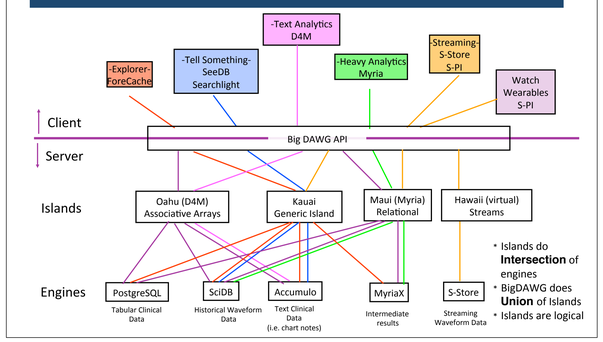
- A **polystore** is a federated db system for multiple, disparate data models
- Polystore requirements:
  - Location transparency
  - Semantic completeness
  - N:N relationship between user semantics and back-ends
- BigDAWG achieves data independence with **islands of information**. Each contains a:
  - Data model, query language, shims to dbs
- Users pose queries by invoking islands with `scope` and `casting` between disparate semantics. Example:

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

Motivation

- Hybrid applications are becoming common
  - Use many back-ends, lots of data models, single dataset
- Example: MIMIC II, intensive care unit dataset
- Workload contains:
  - Streaming for heartrate monitoring
  - Complex analytics for data mining
  - Browsing schemas and patients individually
  - Data Exploration or "tell me something interesting"
- Need a unified interface for diverse users and their workloads

BigDAWG Architecture

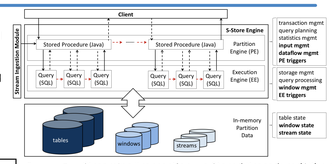


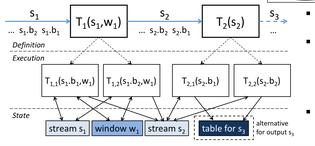
Streaming Analytics with S-Store

Christopher Giossi, David Maier, John Meehan, Jason Muir, Nesime Tatbul, Kristin Tufte, Stan Zdonik

S-Store: A Single System for Transactions and Streams

- A flexible and expressive programming model
- Support for hybrid workloads with streams & shared mutable state
- Strong correctness guarantees (ACID, ordered execution, exactly once)
- Distributed main-memory OLTP system as architectural foundation

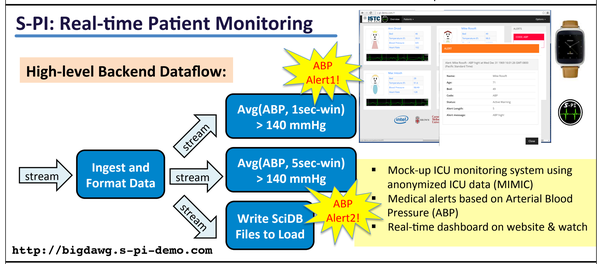




- Streaming Transaction = a Stored Procedure (SP) + a batch of input tuples
- Dataflow graph is a DAG of SP's
  - Transaction Executions (TE's) can't be arbitrarily ordered
  - **Window state** can only be accessed by consecutive TE's of a given SP
- Presence of **shared tables** and OLTP txns require additional isolation rules
  - Nested txns can protect state within dataflows

S-PI: Real-time Patient Monitoring

High-level Backend Dataflow:

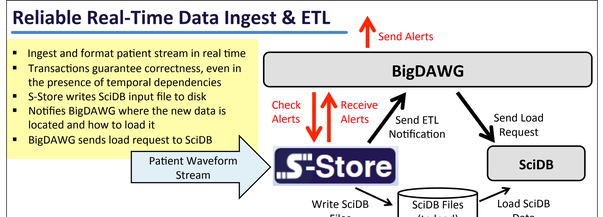


- Mock-up ICU monitoring system using anonymized ICU data (MIMIC)
- Medical alerts based on Arterial Blood Pressure (ABP)
- Real-time dashboard on website & watch

http://bigdawg.s-pi-demo.com

Reliable Real-Time Data Ingest & ETL

- Ingest and format patient stream in real time
- Transactions guarantee correctness, even in the presence of temporal dependencies
- S-Store writes SciDB input file to disk
- Notifies BigDAWG where the new data is located and how to load it
- BigDAWG sends load request to SciDB



http://sstore.cs.brown.edu/

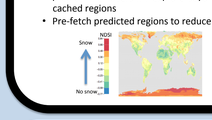
Dynamic Generation and Prefetching of Data Tiles for Exploratory Visualization

Leilani Battle, Remco Chang, Michael Stonebraker

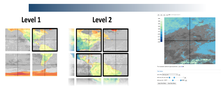
Tufts University

Introduction

- Scientists want to visually explore massive datasets
- Problem: existing visual exploration systems don't scale to larger datasets
- Our approach:
  - Support detail-on-demand interfaces
  - Pre-compute zoom levels offline
  - predict when users will explore beyond pre-cached regions
  - Pre-fetch predicted regions to reduce latency

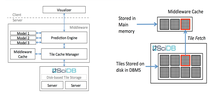


Exploring MODIS Data



- Divide datasets into fixed-sized subsets (i.e. tiles)
- Create layers of tiles to support zoom levels
  - Users explore MODIS data in three phases:
    - Foraging
    - Sensemaking
    - Navigation

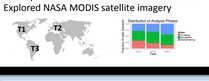
Architecture



- Client retrieves tiles from backend
- Middleware queries SciDB to compute and fetch tiles

User Study

- Participants: 18 researchers in earth sciences
- Explored NASA MODIS satellite imagery

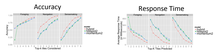


Prediction Engine

- Keep track of user's recently requested tiles
  - Use past requests to predict future requests
- Top level: predict user's current phase
- Bottom level: predict low-level behaviors
- Run multiple models in parallel at bottom level
  - Learn user's most common movement sequences
  - Compare *signatures* for each tile

Performance

- Cache hit = 19.5ms, cache miss = 984ms
- 400% improvement over non-prefetching systems
- 88% improvement over existing prefetching algo's



Future Work

- Supporting both pre-computation and prefetching
- Building a general-purpose signature toolbox
- Extending caching model for multi-user system
- Provide recommendations client-side to help users find new regions to explore

Technologies for Visualization of Big Medical Text Data

Leilani Battle, Lauren Edwards, Vijay Gadepally, Brendan Gavin, Braden Hancock, Dylan Hutchison, Jeremy Kepner, Andrew Moran

ABSTRACT

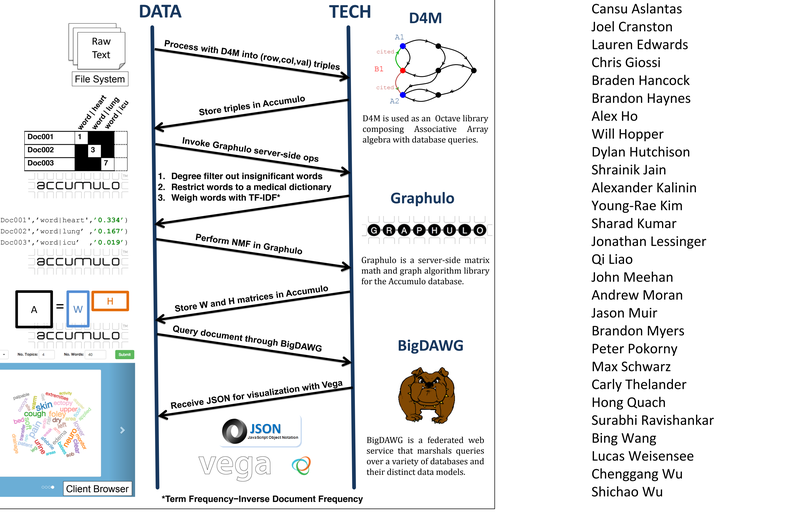
The ISTC Big Data Analytics Working Group (BigDAWG) project allows users to explore and analyze heterogeneous data sets that vary in size, schema, and organization. One use case for BigDawg is the MIMIC II public data set, which contains multiple years of ICU reports for thousands of patients at Boston's Beth Israel Deaconess Hospital.

In this poster, we highlight the technologies that enable topic modeling over the corpus of MIMIC documents: D4M, Graphulo, and Vega. The result is a prototype system that visualizes the content of a given medical report using the BigDAWG API.

SYSTEM PROTOTYPE

The diagram to the right depicts the steps for processing data into query-ready form, and then steps to answer a visualization query from a client through the BigDAWG web service. We show a sample of input data, the technology used for processing, and a sample of resulting data for each step. Data originates from raw text files, Accumulo stores intermediate and processed data, and a client web browser presents a query and visualization interface.

All operations except for the Non-negative matrix factorization (NMF) step do not require holding an entire dataset in the memory of a single node. D4M's ingest step leverages pMATLAB to ingest triples into Accumulo from multiple processes; Graphulo runs within Accumulo's tablet servers, gaining parallelism and minimizing data movement; and Vega visualizes only the topics relevant to the client.



FUTURE WORK

- Use original ICU data directly from PostgreSQL by creating a BigDAWG "shim," instead of exporting from PostgreSQL to text files for pre-processing and loading into Accumulo.
- Accelerate an NMF variant that does not require holding data in memory, which enables distributed execution in an Accumulo instance via Graphulo.
- Explore alternative visualization tools for processed medical data, as well as techniques that consider word position beyond bag-of-words approaches.

ISTC BIG DATA

Intel Science and Technology Center for Big Data

Additional Contributors

Adam Dziedzic
Krishna Chaitanya Aluru
Cansu Aslantas
Joel Cranston
Lauren Edwards
Chris Giossi
Braden Hancock
Brandon Haynes
Alex Ho
Will Hopper
Dylan Hutchison
Shrainik Jain
Alexander Kalinin
Young-Rae Kim
Sharad Kumar
Jonathan Lessinger
Qi Liao
John Meehan
Andrew Moran
Jason Muir
Brandon Myers
Peter Pokorny
Max Schwarz
Carly Thelander
Hong Quach
Surabhi Ravishankar
Bing Wang
Lucas Weisensee
Chenggang Wu
Shichao Wu

SeeDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics

(*VLDB vol 8, issue 13)

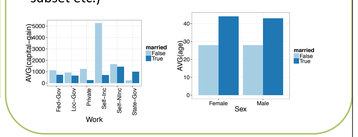
Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, Neoklis Polyzotis

Motivation

- Data visualization first step in analysis
- High-dimensional data → 100s of visualizations
- Manual specification hampers rapid analysis
- **SeeDB: visualization recommendation tool**
- **Automatically identify and recommend "interesting" visualizations**

Approach

- Interesting-ness defined via utility metric
- Visualization is interesting if it shows deviations from reference (e.g. entire dataset, comparable subset etc.)



System Architecture

Mixed-initiative interface
manual visualization + recommendations.

Challenges
- 100+ visualizations (>10s min)
- Require interactive latency
- Resources wasted on low-utility visualizations

Optimizations
- Classical DBMS optimizations: *Shared scans, bin-packing, ||*
- Pruning optimizations

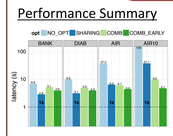
Pruning optimizations
- Data processed in chunks
- Partial results → prune

OPT 1:
Build worst-case CI; discard intervals not in top-k

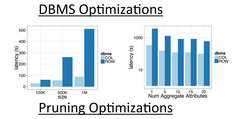
OPT 2:
Model problem as resource allocation via bandits

Evaluation

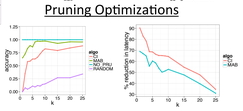
Performance Summary



DBMS Optimizations



Pruning Optimizations



Optimizations provide 50X – 300X speedup in ROW, 10X – 30X in COL
Near-interactive latency

User Study:
Comparison of manual only interface and SeeDB
- SeeDB finds insights faster
- 2-factor ANOVA shows significant effect on **number of bookmarks** ($F(1,1) = 18.609$, $p < 0.001$) and **bookmarking rate** ($F(1,1) = 10.034$, $p < 0.01$)