ISTC BIG DATA

Polystore, Julia, and productivity in a Big Data world

Tim Mattson, Intel labs

timothy.g.mattson@intel.com

Intel-PI for the Big Data "Intel Science and Technology Center"

People I stole content from for this talk: Stavros Papadopoulos, Jake Bolewski, Mike Stonebraker, Vijay Gadepally, Bill Howe, Magda Balazinska, Jennie Duggan, Aaron Elmore, Sam Madden, Jeremy Kepner, Kiran Pamnany, and Tatiana Shpeisman.

intel MIT BROWN Carnegie Mellon University W U Portland State UNIVERSITY

Legal Disclaimer & Optimization Notice

- INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
-

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Disclaimer

- The views expressed in this talk are those of the speaker and not his employer.

- If I say something “smart” or worthwhile:
  - Credit goes to the many smart people I work with.

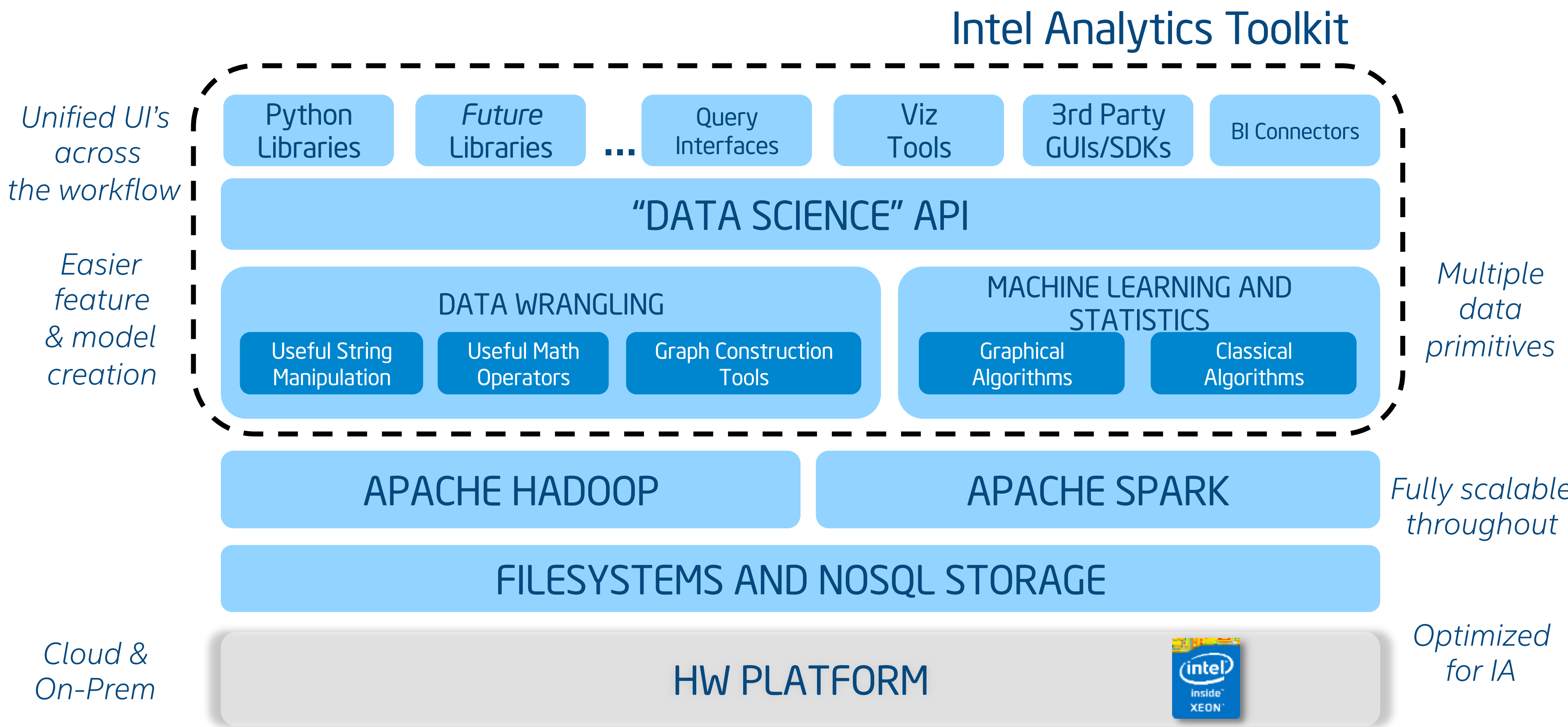
- If I say something stupid…
  - It’s my own fault

I work in Intel’s research labs. I don’t build products. Instead, I get to poke into dark corners and think silly thoughts… just to make sure we don’t miss any great ideas.

Hence, my views are by design <u>far</u> “off the roadmap”.

Big Data Today

common assumption … Big Data = Hadoop/Spark



Third party names are the property of their owners

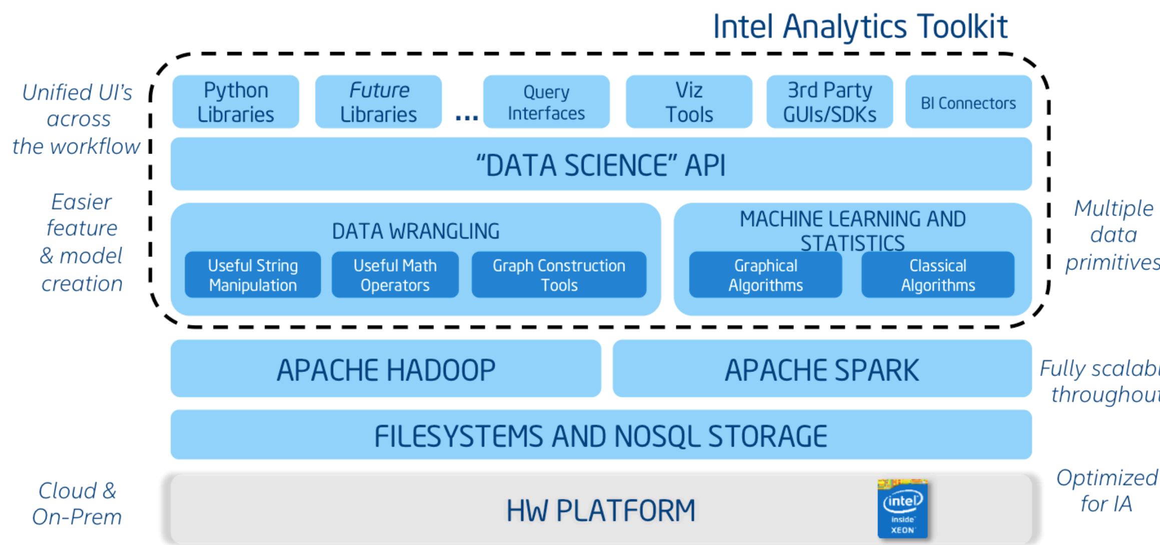
What's next for Big Data

- Hadoop/SPARK is great … it helped put Big Data on the map
- Comes from a time when we were just thrilled to "do" big data.

- Hadoop/Spark design did not emphasize:
  - Performance for complex analytics.
  - Efficient utilization of the hardware
  - Programmability for anything beyond "embarrassingly parallel" applications.

Big Data Today

common assumption … Big Data = Hadoop/Spark

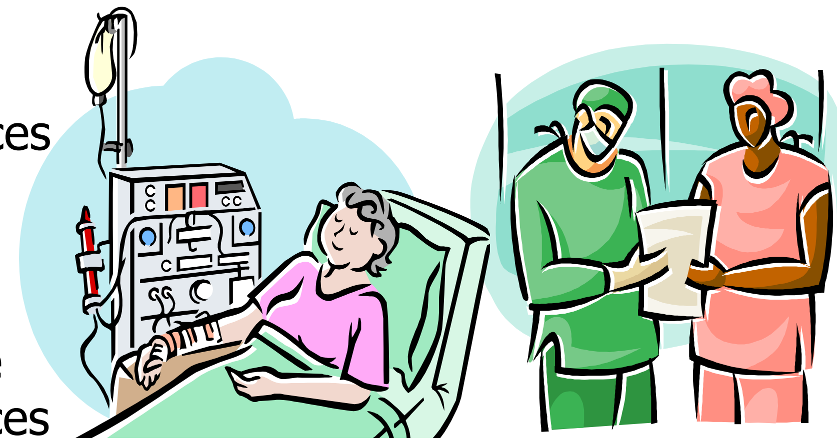


Challenge: What happens when Hadoop/Spark runs out of steam? What comes next?

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 …

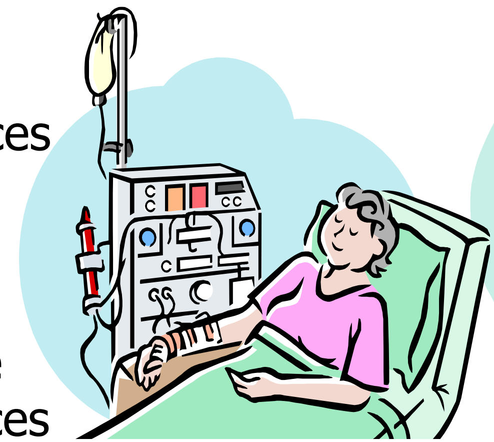
- Consider patient data in an Intensive Care Unit (e.g. MIMIC II data set*)

Arrays
- EKG traces

Time Series
- Blood oxygen

Time Series
- Blood pressure

Arrays
- EEG traces



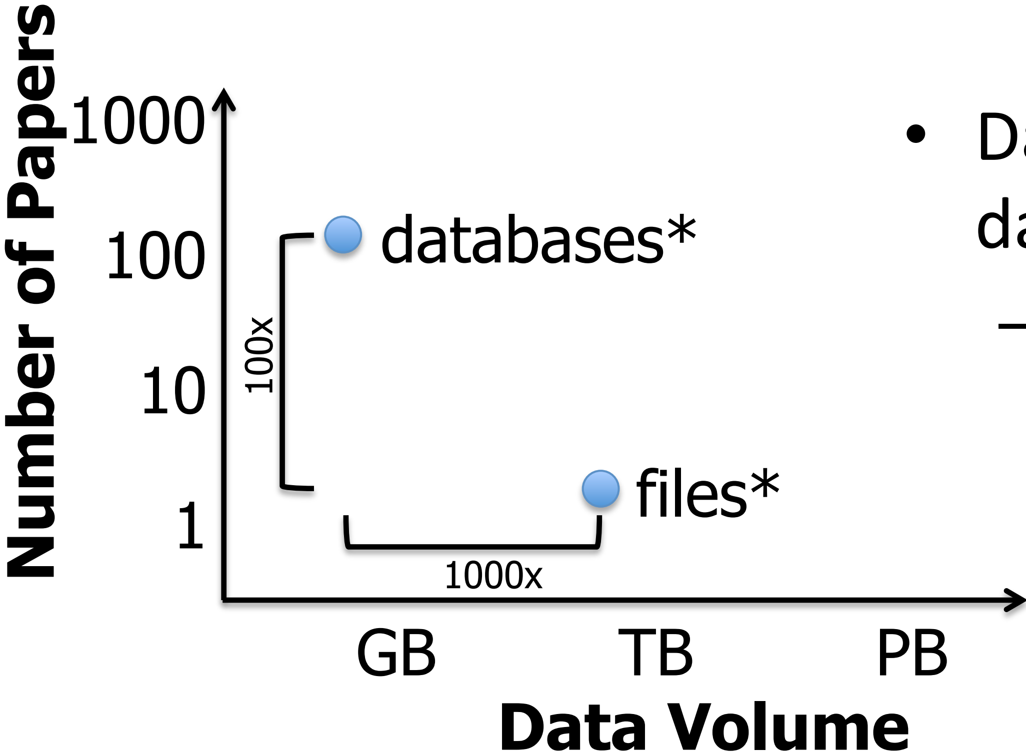
- Demographic — tables
- Caregiver notes — documents
- Medical charts — tables
- Lab test results — tables
- Xray, MRI, etc. — #images

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

A disruptive idea: Match data to the data-store technology but present as a single Data Base Management system to the end-users … A disruptive idea we call *Polystore*.

Source: Vijay Gadepally of MIT Lincoln labs

*Based on PhysioNet MIMIC2 ICU data

BigDawg: An integrated ***polystore*** system

Applications
e.g., Medical data, astronomy, twitter, urban sensing, IoT

Visualization & presentation
e.g., ScalaR, imMens, SeeDB, Prefetching

SW Development
e.g, APIs for traditional languages, Julia, GraphMat, ML Base

BigDAWG Query Language and Data Federation layer

“Narrow Waist” Provides Portability

Stream

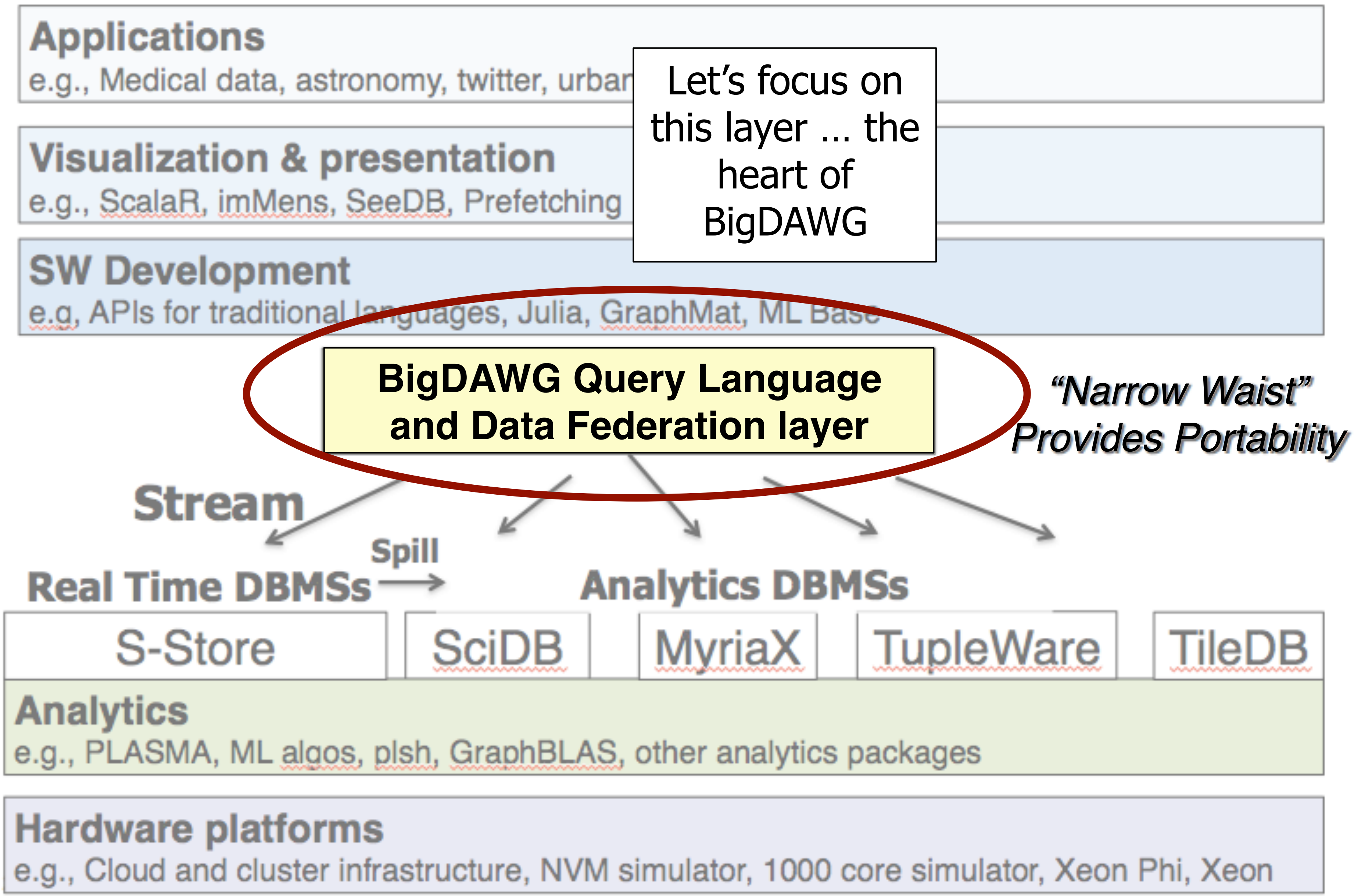
Real Time DBMSs $\xrightarrow{\textbf{Spill}}$ **Analytics DBMSs**

S-Store | SciDB | MyriaX | TupleWare | TileDB

Analytics
e.g., PLASMA, ML algos, plsh, GraphBLAS, other analytics packages

Hardware platforms
e.g., Cloud and cluster infrastructure, NVM simulator, 1000 core simulator, Xeon Phi, Xeon

BigDawg: An integrated ***polystore*** system

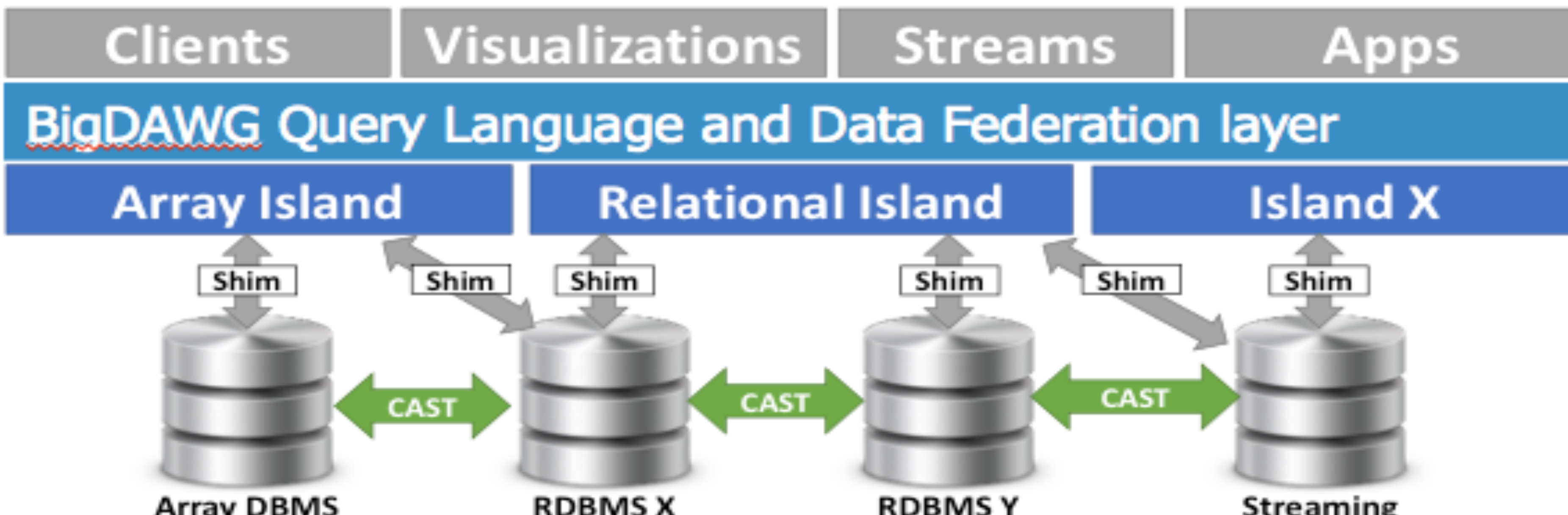


BigDAWG Data Federation

- Two Key Components:
  - BigDAWG Query Language or BQL:
    - the quest for “one query language to rule them all”
  - BigDAWG Data Federation API:
    - **Islands**: a collection of data stores that share a data model and query language
    - **Shims**: to translate queries between islands
    - **Casts**: to move data from one island to another

High risk transformative research … many people think this is impossible.

Based on ISTC research over the last 3 years, we think we know how to do this



RDBMS = relational DBMS

Our VLDB’2015 Demo

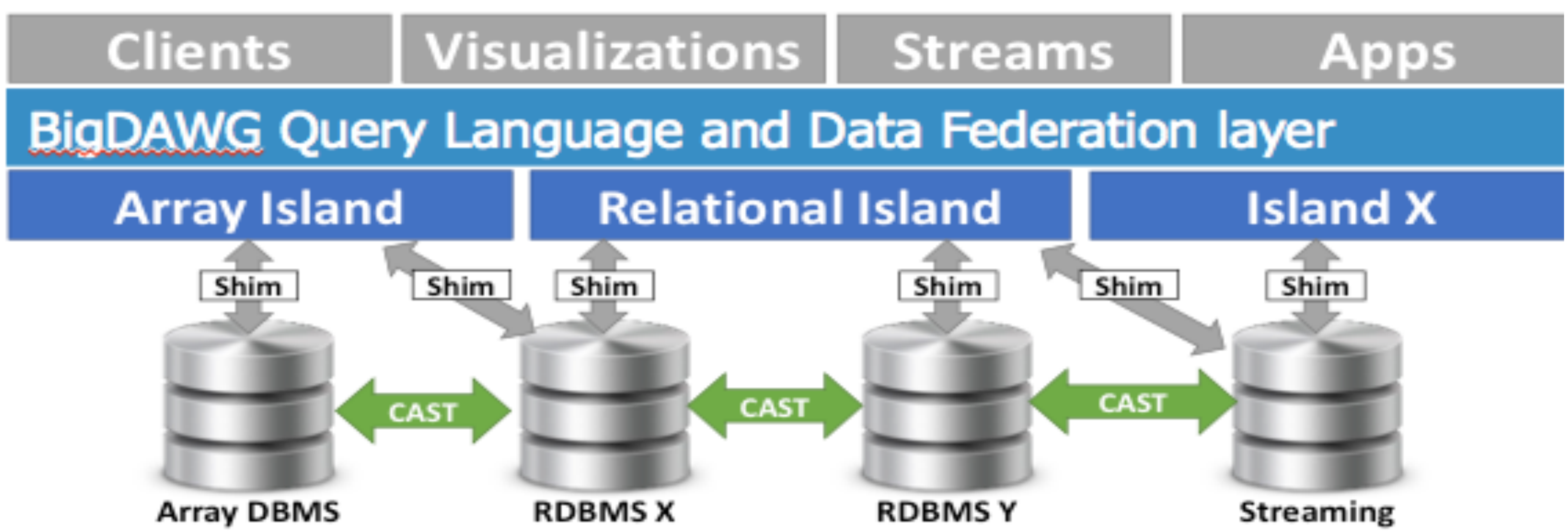
A Demonstration of the BigDAWG Multi-Database System

A. Elmore, Univ. of Chicago
J. Duggan, Northwestern
M. Stonebraker, MIT
M. Balazinska, Univ. of Wash.
U. Cetintemel, Brown
V. Gadepally, MIT-LL
J. Heer, Univ. of Wash.
B. Howe, Univ. of Wash.
J. Kepner, MIT-LL
T. Kraska, Brown
S. Madden, MIT
D. Maier, Portland St U.
T. Mattson, Intel
S. Papadopoulis, Intel / MIT
J. Parkhurst, Intel
N. Tatbul, Intel / MIT
M. Vartek, MIT
S. Zdonik, Brown

ABSTRACT

This paper presents BigDAWG, a reference implementation of a new architecture for “Big Data” applications. Such applications

BigDAWG stores MIMIC II in a mixture of backends, including Postgres, which stores the patient metadata, SciDB [4], which stores the historical waveform data in a time-series (array) database,



Our VLDB’2015 Demo

A Demonstration of the BigDAWG Multi-Database System

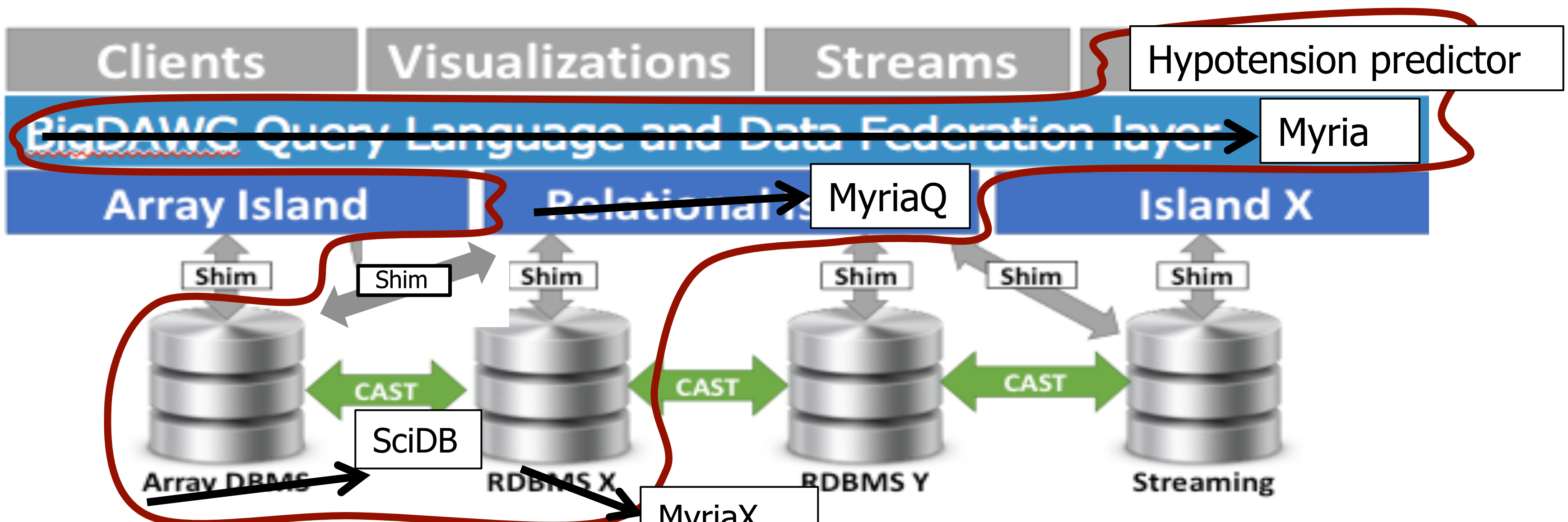
A. Elmore	J. Duggan	M. Stonebraker	M. Balazinska	U. Cetintemel	V. Gadepally	J. Heer	B. Howe	J. Kepner
Univ. of Chicago	Northwestern	MIT	Univ. of Wash.	Brown	MIT-LL	Univ. of Wash.	Univ. of Wash.	MIT-LL
T. Kraska	S. Madden	D. Maier	T. Mattson	S. Papadopoulis	J. Parkhurst	N. Tatbul	M. Vartek	S. Zdonik
Brown	MIT	Portland St.U.	Intel	Intel / MIT	Intel	Intel / MIT	MIT	Brown

We have preliminary results targeting MyriaX, SciDB, S-store, D4M, graphulo and several visualization packages.

AB

This paper presents BigDAWG, a reference implementation of a new architecture for “Big Data” applications. Such applications

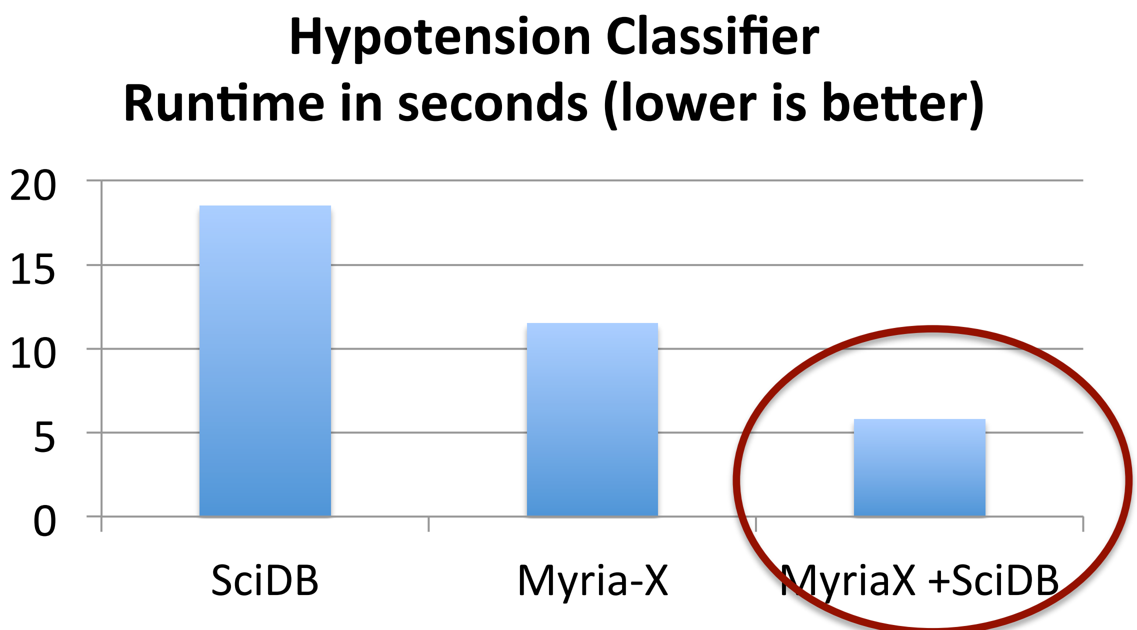
includ- ... which stores the historical waveform data in a time-series (array) database,



Third party names are the property of their owners

The App: Hypotension Predictor

- Problem: blood pressure drops (hypotension) →shock → death. Early intervention is key for survival.
- Solution: Machine learning over heterogeneous data (from MIMIC II) to identify patients about to suffer from a severe drop in blood pressure?
- Algorithm (from Saeed and Mark*) build a classifier .. Haar transforms over MIMICII time series data, summarize as histograms, and performs a K nearest neighbor search. Correlate with patient data.



Source: Magdalena Balazinska and Brandon Haynes, university of Washington.
*A Novel Method for the Efficient Retrieval of Similar Multiparameter Physiologic Time Series Using Wavelet-Based Symbolic Representations. Mohammed Saeed and Roger Mark, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1839671/

BigDawg: An integrated ***polystore*** system

ISTC BIG DATA

Applications
e.g., Medical data, astronomy, twitter, urban sensing, IoT

Visualization & presentation
e.g., ScalaR, imMens, SeeDB, Prefetching

SW Development
e.g, APIs for traditional languages, Julia, GraphMat, ML Base

BigDAWG Query Language and Data Federation layer

"Narrow Waist" Provides Portability

Stream

Spill

Real Time DBMSs → **Analytics DBMSs**

S-Store | SciDB | MyriaX | TupleWare | TileDB

Analytics
e.g., PLASMA, ML algos, plsh, GraphBLAS, other analytics packages

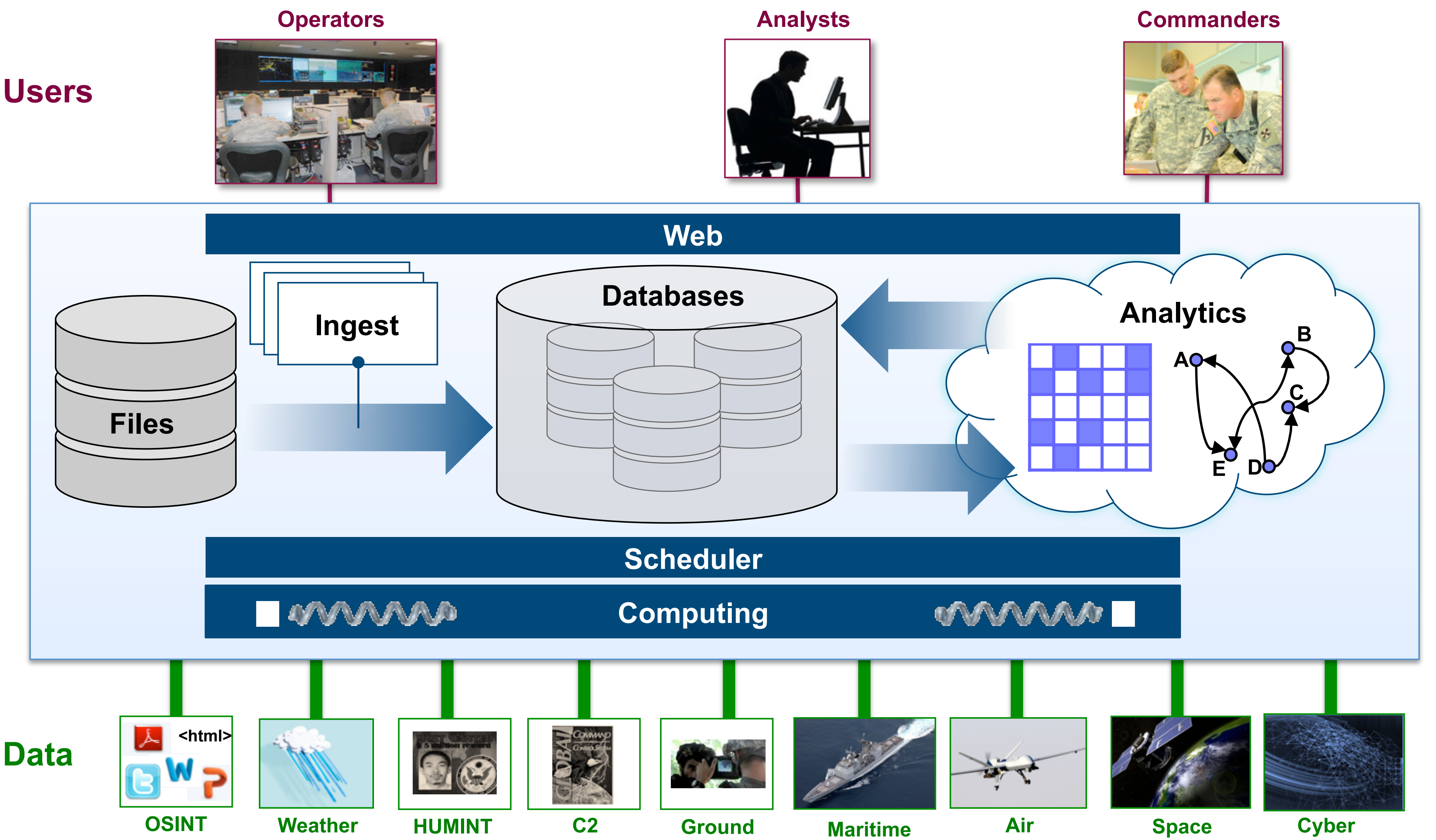
Hardware platforms
e.g., Cloud and cluster infrastructure, NVM simulator, 1000 core simulator, Xeon Phi, Xeon

System Infrastructure for BigDAWG Research

- To explore BigDAWG, we need to:
  - Quickly load a variety of DataBase Management Systems (DBMS)
  - Mix HPC (MPI) jobs with traditional DBMS jobs
  - Manage everything through an end-user driven web interface
- Fortunately, we were able to work with the team behind the MIT SuperCloud.

Enabling on-demand Database computing with MIT SuperCloud Database management system, Andrew Prout, Jeremy Kepner, Peter Michaleas, William Arcand, David Bestor, Bill Bergeron, Chansup Byun, Lauren Edwards, Vijay Gadepally, Matthew Hubbell, Julie Mullen, Antonio Rosa, Charles Yee, Albert Reuther, IEEE High Performance Extreme Computing Conference 2015, 17 September 2015

Common Big Data Architecture



Source: Jeremy Kepner, MIT Lincoln Labs

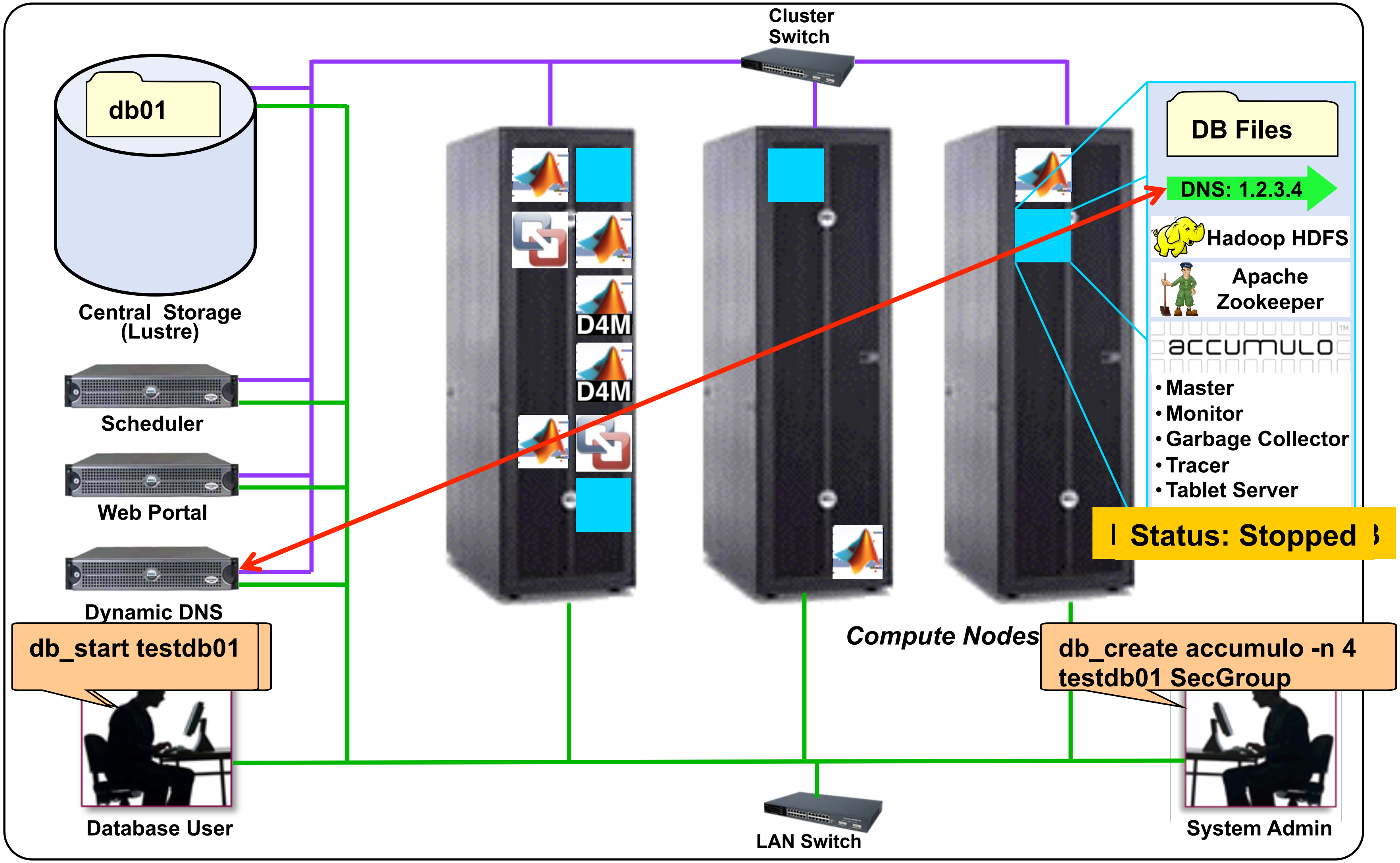
Why a Database Management System?

- Remove requirement for dedicated servers, while avoiding virtual machines (VMs)
  - In addition to performance concerns, VMs historically have caused problems for the timeout-based failure detection features of Accumulo
- Enable rapid creation of new databases
- Reduce waste of resources on idle databases
- Create a viable backup & restore strategy
- Ensure security concerns are appropriately addressed
- Empower the less IT savvy researchers & scientists with self-service commands for common requests
- Integration with HPCC scheduler

Database creation should be closer to a mkdir than a major IT project

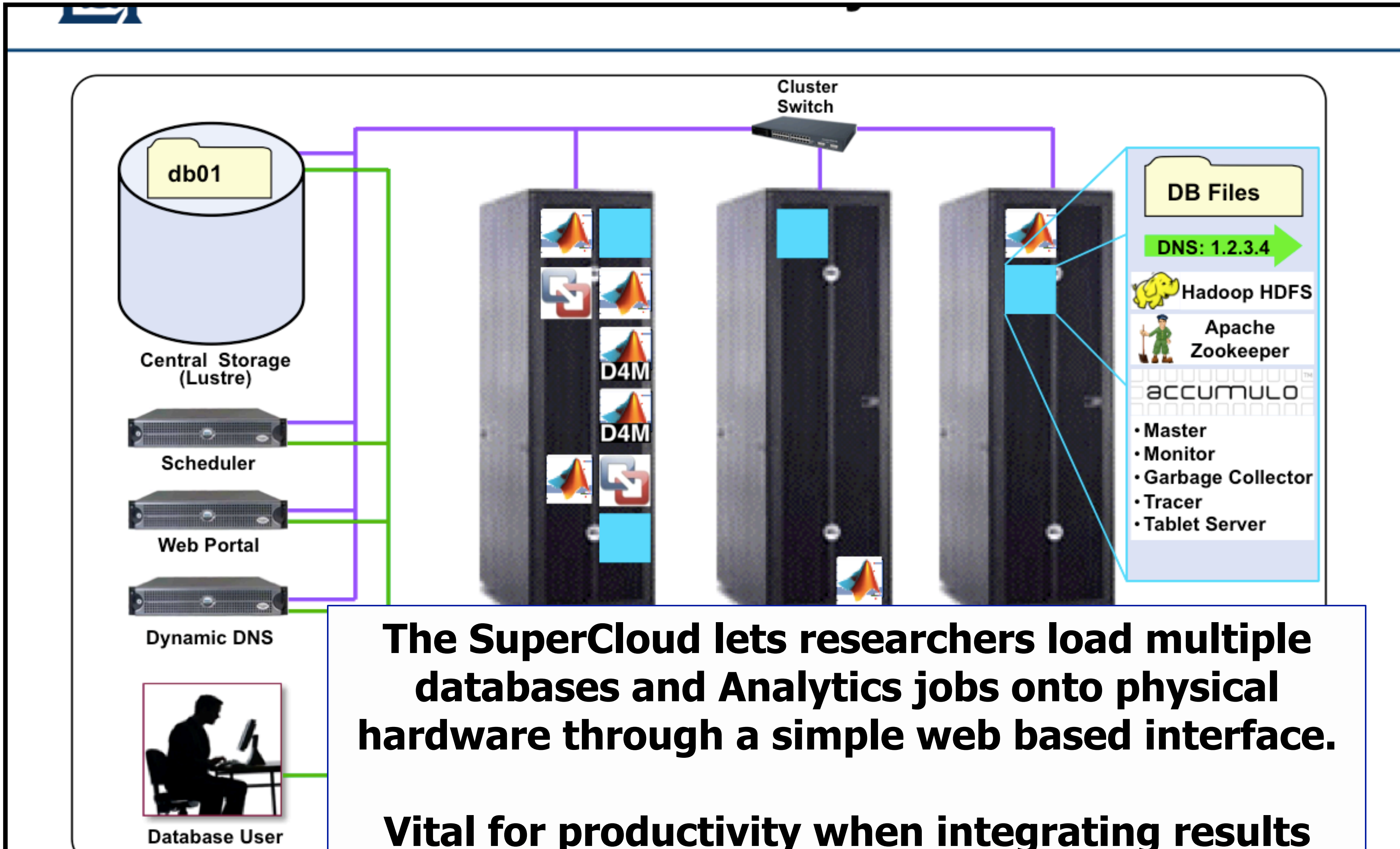
Source: Jeremy Kepner, MIT Lincoln Labs

Database Lifecycle



Source: Jeremy Kepner, MIT Lincoln Labs

MIT SuperCloud and BigDAWG



The SuperCloud lets researchers load multiple databases and Analytics jobs onto physical hardware through a simple web based interface.

Vital for productivity when integrating results into BigDAWG from so many teams!

BigDawg: An integrated ***polystore*** system

Applications
e.g., Medical data, astronomy, twitter, urban sensing, IoT

Visualization & presentation
e.g., ScalaR, imMens, SeeDB, Prefetching

SW Development
e.g, APIs for traditional languages, Julia, GraphMat, ML Base

BigDAWG Query Language and Data Federation layer

"Narrow Waist" Provides Portability

Stream

Spill

Real Time DBMSs

Analytics DBMSs

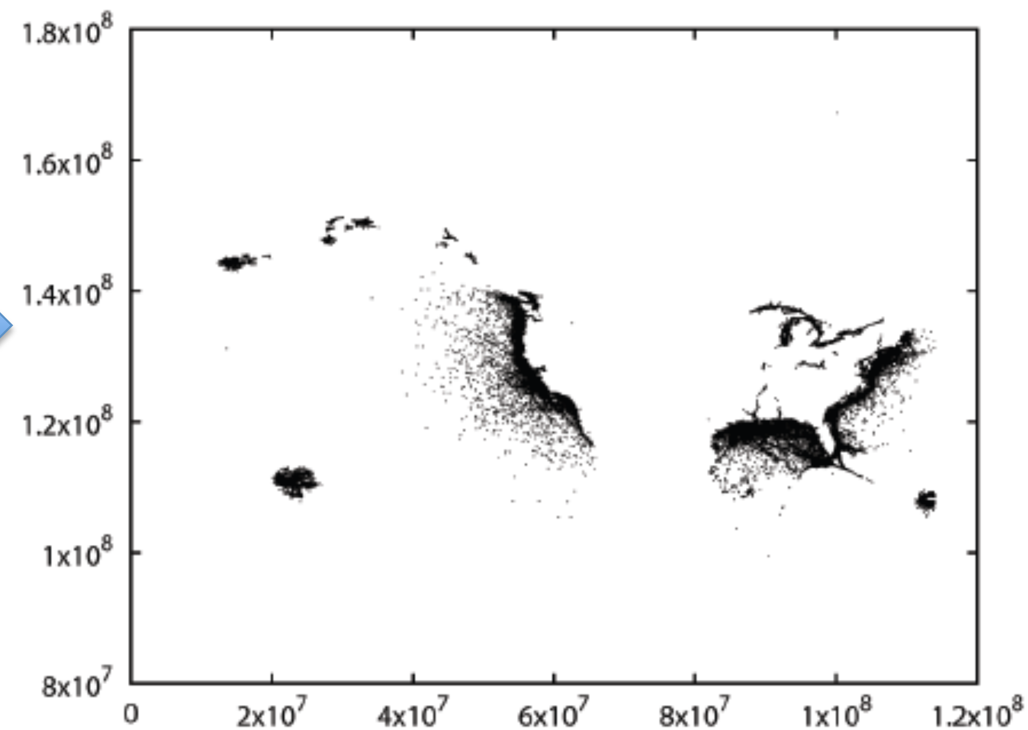
S-Store	SciDB	MyriaX	TupleWare	TileDB

Analytics
e.g., PLASMA, ML algos, plsh, GraphBLAS, other analytics packages

Hardware platforms
e.g., Cloud and cluster infrastructure, NVM simulator, 1000 core simulator, Xeon Phi, Xeon

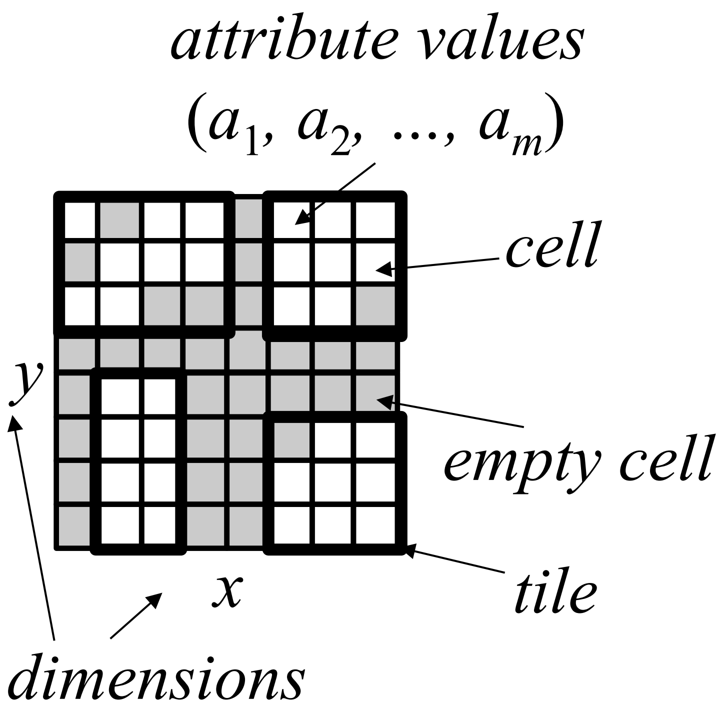
Arrays in Big Data problems

- Data is often naturally considered as an array:
  - An object with multiple dimensions (e.g. 2)
  - The dimensions define a logical coordinate space
  - A cell "exists" at each point in the coordinate space.
  - A cell has one or more attributes which collectively define the "value" at that cell.
- Data is usually sparse
  - E.G. the AIS data set showing ship locations as a function of time in and around U.S. waters

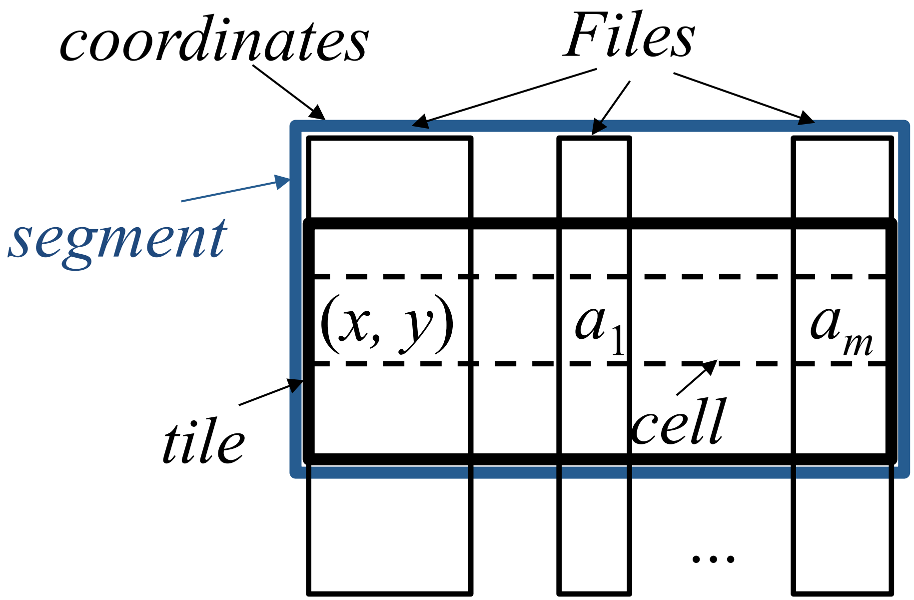


TileDB a new array data storage manager: optimized for Sparse Arrays

Logical representation



Physical representation

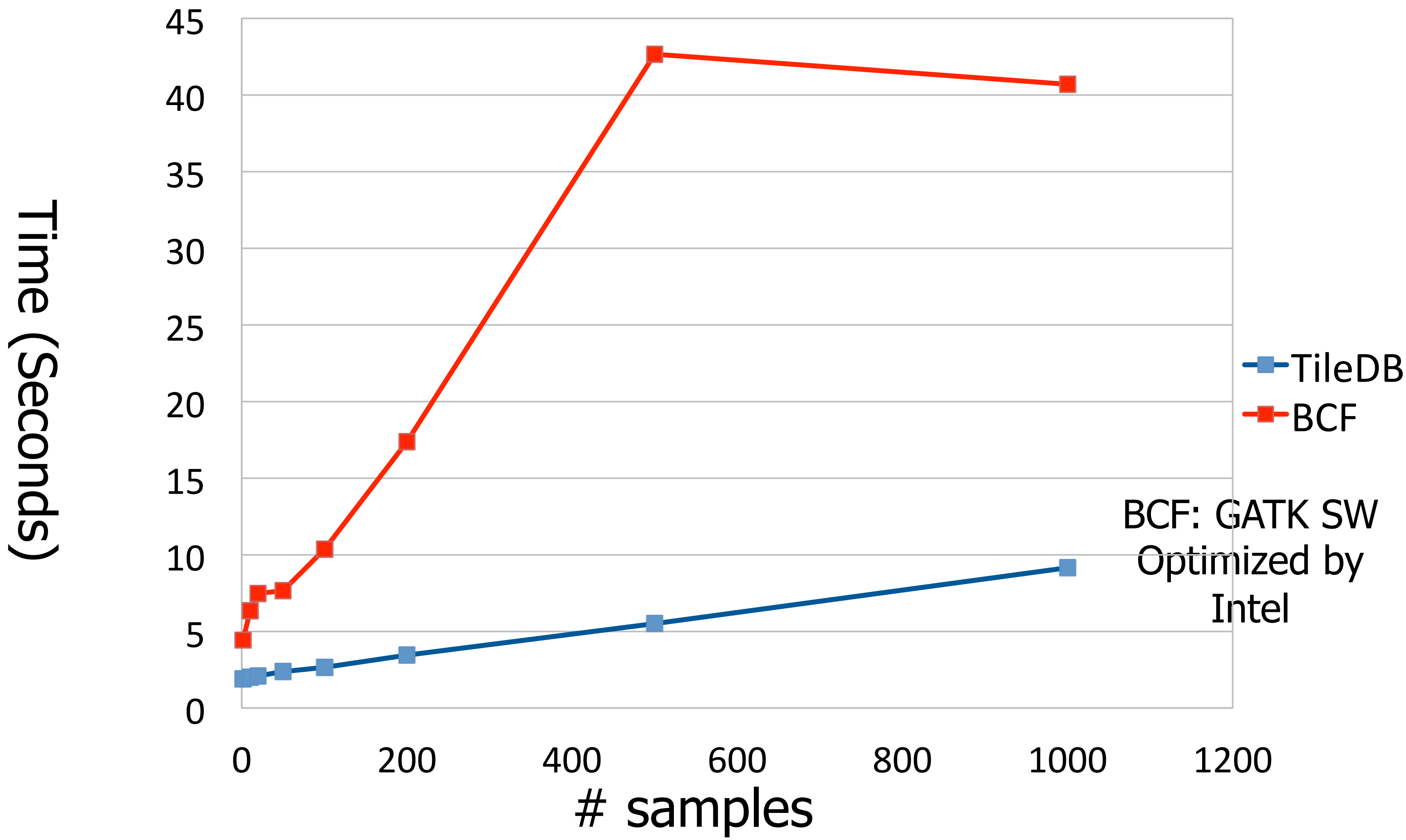


Tile: *Atomic unit of processing*

Segment: *Atomic unit of I/O*

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

Joint Genotyping Benchmark*



*Benchmark jointly developed by Intel and the Broad Genomics Institute. Each sample is 10MB. Compute correlations across samples at ~5000 positions.

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. Single thread/core results.

BigDawg: An integrated ***polystore*** system

ISTC BIG DATA

Applications
e.g., Medical data, astronomy, twitter, urban sensing, IoT

Visualization & presentation
e.g., ScalaR, imMens, SeeDB, Prefetching

SW Development
e.g, APIs for traditional languages, Julia, GraphMat, ML Base

BigDAWG Query Language and Data Federation layer

"Narrow Waist" Provides Portability

Stream

Spill

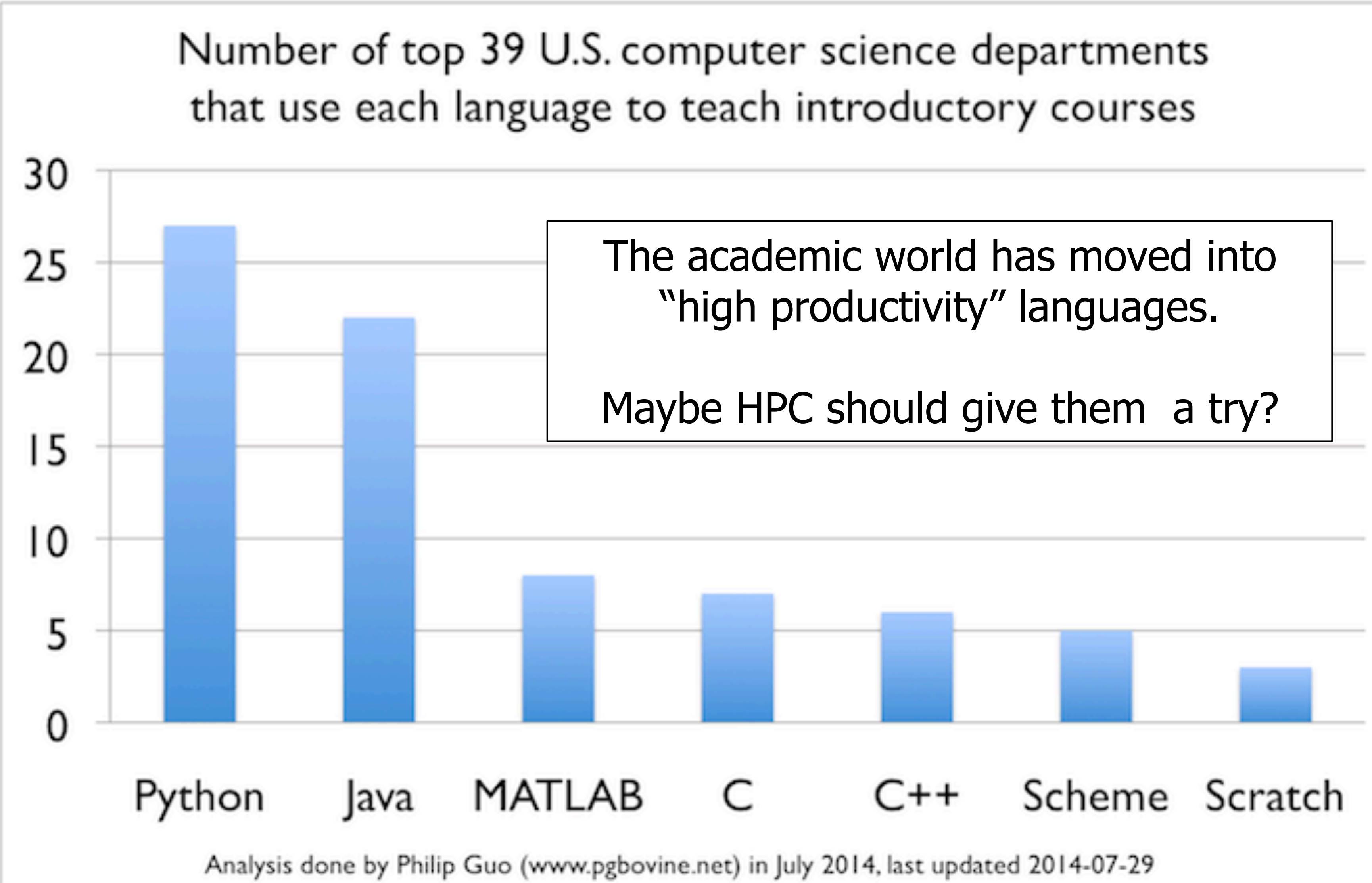
Real Time DBMSs → **Analytics DBMSs**

S-Store | SciDB | MyriaX | TupleWare | TileDB

Analytics
e.g., PLASMA, ML algos, plsh, GraphBLAS, other analytics packages

Hardware platforms
e.g., Cloud and cluster infrastructure, NVM simulator, 1000 core simulator, Xeon Phi, Xeon

Programming languages in Academia



Source: http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext

What is Julia?

- Julia is yet another new language!!
- Started in 2009 in Alan Edelman's group at MIT.

- The problem is … do we really need a new language? … computer scientists spend more time creating new languages than making existing languages actually work.

- But people I know and respect have convinced me to taka a close look at Julia
  - It is relatively easy to learn
  - A non-viral open source license (MIT License) so corporate types can play with it.
  - A large and growing community … over 350 contributors since it started in 2009

What is Julia?

- **An excellent foundation for programming research.**
  - Core functionality of Julia is written in Julia.
  - Benefits from large LLVM eco-system.
  - **Introspection**: Exposes transformations from high level code into native assembly code … so you can manipulate them inside Julia

Parse source into AST

↓

Lower AST

```
code_lowered(fib,(Int32))
```

↓

Type Inference

```
code_typed(fib,(Int32))
```

↓

Build LLVM Intermediate Rep.

```
code_llvm(fib,(Int32))
```

↓

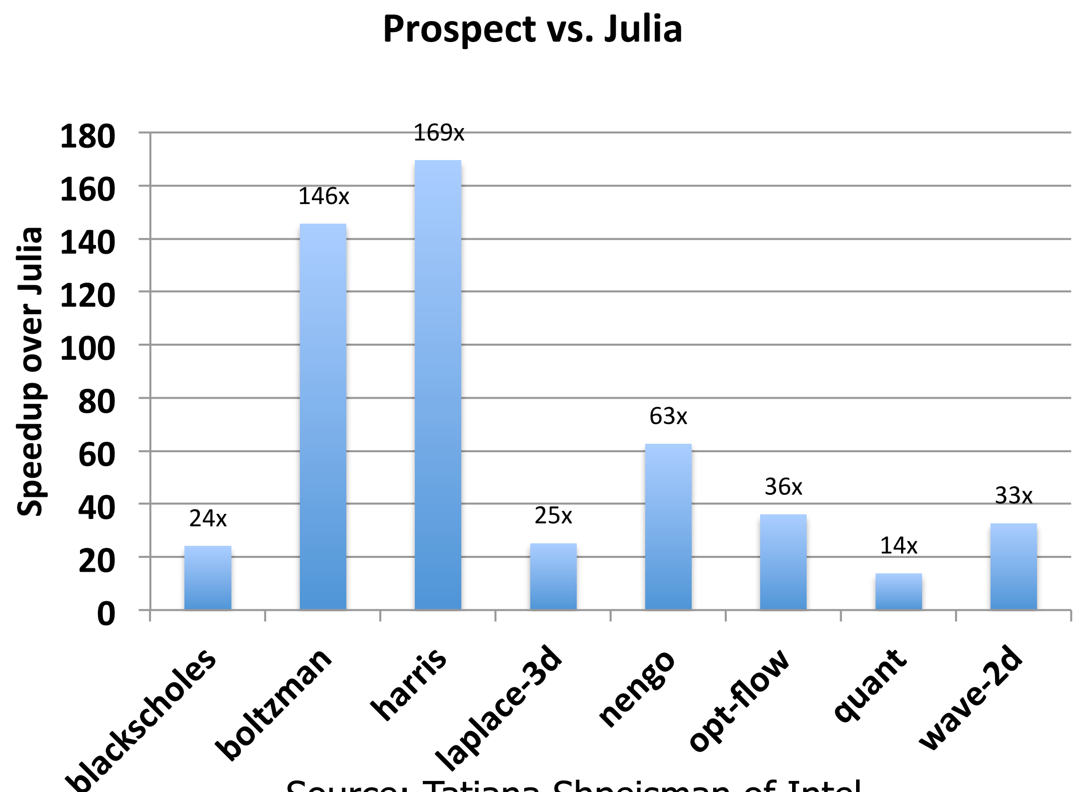
Emit machine code

```
code_native(fib,(Int32))
```

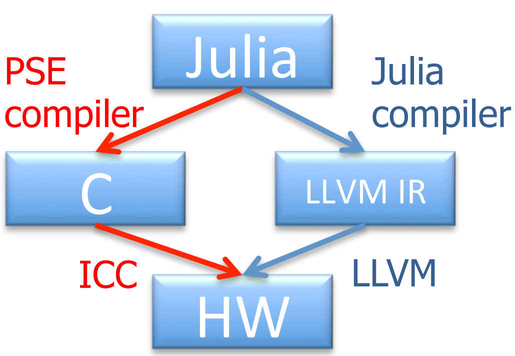
↓

The benefits of introspection

- Using the flexible Julia framework, a group at Intel created C backend integrated with Julia (Prospect … Open Source release Q4'2015)
- Prospect compiler tool generates optimized C code from Julia Source:
- 10-170 speedup over Julia due to:
    - Parallelization
    - Loop fusion
    - Domain specific optimizations
    - vectorization



Source: Tatiana Shpeisman of Intel

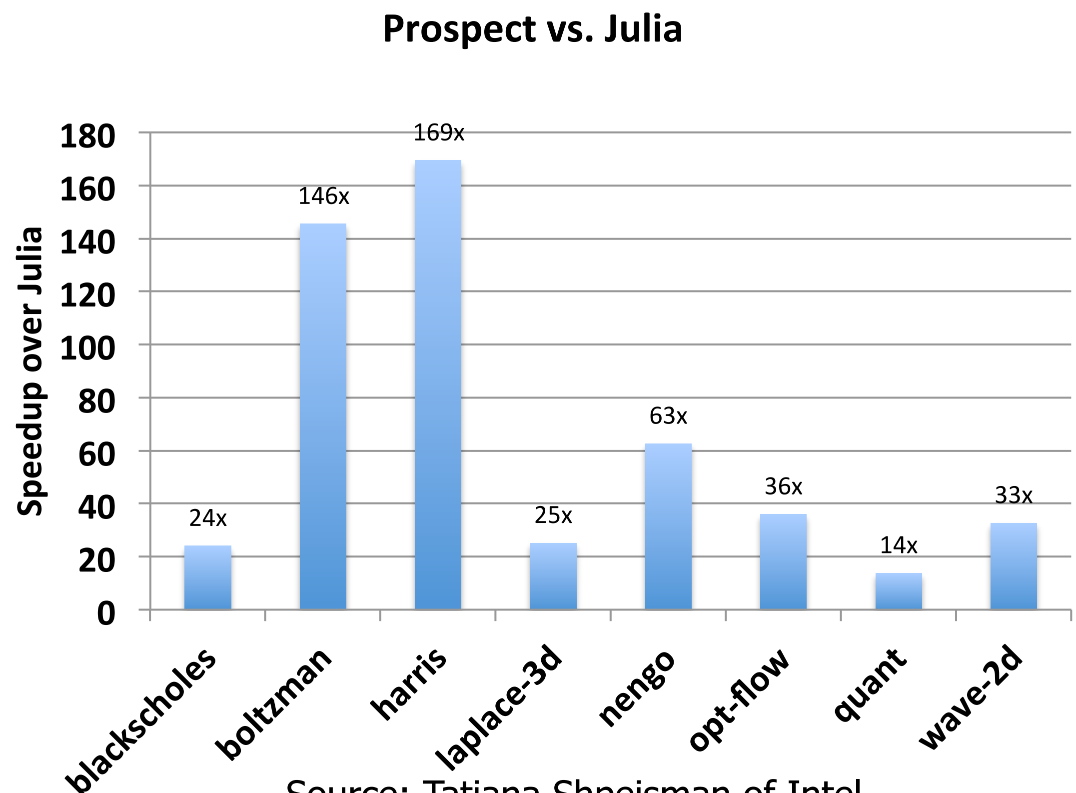


PSE: our Julia based Problem Solving Environment … of which Prospect is a key component.

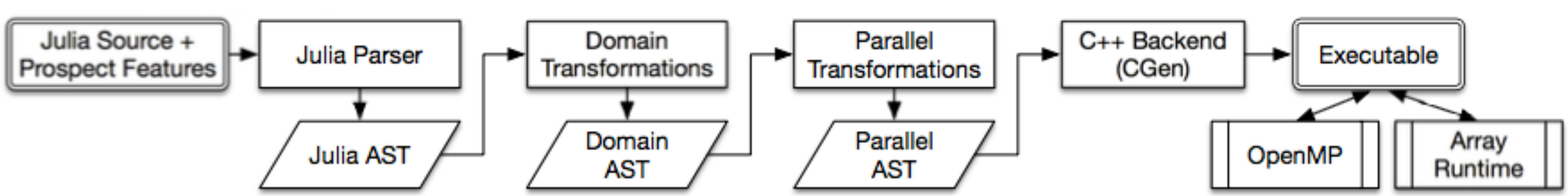
Running on a server with two Intel® Xeon® E5-2690v2 processors at 3 Ghz, 128GB RAM

The benefits of introspection

- Using the flexible Julia framework, a group at Intel created C backend integrated with Julia (Prospect … Open Source release Q4’2015)
- Prospect compiler tool generates optimized C code from Julia Source:
- 10-170 speedup over Julia due to:
  - Parallelization
  - Loop fusion
  - Domain specific optimizations
  - vectorization



Source: Tatiana Shpeisman of Intel



Running on a server with two Intel® Xeon® E5-2690v2 processors at 3 Ghz, 128GB RAM

What is Julia?

- It is a dynamic scripting language (like python or perl)
- Syntax natural to people working with Math plus a rich set of built in, standard operations (Like Matlab or R).

```
tgmattso — julia — 80×32
Last login: Mon Oct 12 15:47:51 on ttys000
rsadhwan-mobl1:~ tgmattso$ Julia
               _
   _       _ _(_)_     |  A fresh approach to technical computing
  (_)     | (_) (_)    |  Documentation: http://docs.julialang.org
   _ _   _| |_  __ _   |  Type "help()" for help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 0.3.9 (2015-05-30 11:24 UTC)
 _/ |\__'_|_|_|\__'_|  |  Official http://julialang.org/ release
|__/                   |  x86_64-apple-darwin13.4.0

julia> A = rand(3,3) + eye(3) # create a random diagogally dominant matrix
3x3 Array{Float64,2}:
 1.06324    0.138392  0.518535
 0.0798805  1.08625   0.0101105
 0.799626   0.680015  1.12775

julia> (Al, Au) = lu(A)
(
3x3 Array{Float64,2}:
 1.0        0.0       0.0
 0.0751295  1.0       0.0
 0.752067   0.535331  1.0,

3x3 Array{Float64,2}:
 1.06324  0.138392   0.518535
 0.0      1.07585   -0.0288468
 0.0      0.0        0.75322   ,

[1,2,3])

julia>
```

Third Party names are the property of their owners

What is Julia?

A general purpose programming language with sophisticated dataflow type discovery

function definition

```
function fib(n)
    if n<2
        n
    else
        fib(n-1)+fib(n-2)
    end
end
```

no need to declare parameter types

implicit return value

With a very compact syntax.

```
fib(n) = n<2 ? n : fib(n-1)+fib(n-2)
```

one line shorthand

time = $O(1.618^n)$

Source: Arch Robison of Intel

What is Julia?

```
function fibfast(n)
    @assert n>=1
    a = zero(n)
    b = one(n)
    for i=2:n
        (a,b) = (b,a+b)
    end
    b
end
```

Macro

Set local variable a to zero of same type as n

Set local variable b to one of same type as n

Loop from 2 to n inclusive

tuple construction and destructuring

implicit return value

time = O(n)

Source: Arch Robison of Intel

What is Julia?

- Multiple dispatch
- Dynamic polymorphism

tgmattso — julia — 54×20

```
julia> *
* (generic function with 115 methods)

julia> *(a::Number, g::Function) = x->a*g(x)
* (generic function with 116 methods)

julia> *(f::Function, t::Number) = x->f(t*x)
* (generic function with 117 methods)

julia> *(f::Function, g::Function) = x->f(g(x))
* (generic function with 118 methods)

julia> x=pi/4
0.7853981633974483

julia> (sin^2)(x)
0.6496369390800624

julia>
```

Since ^ is generically represented in terms of *, this becomes the composition of functions (sin(sin(x)) … which is what Gauss wanted it to be.

Distributed Memory Parallelism

```
function fibpar(n)
    if n<30
        fib(n)
    else
        x = @spawn fibpar(n-1)
        y = fibpar(n-2)
        y+fetch(x)
    end
end
```

Run in parallel

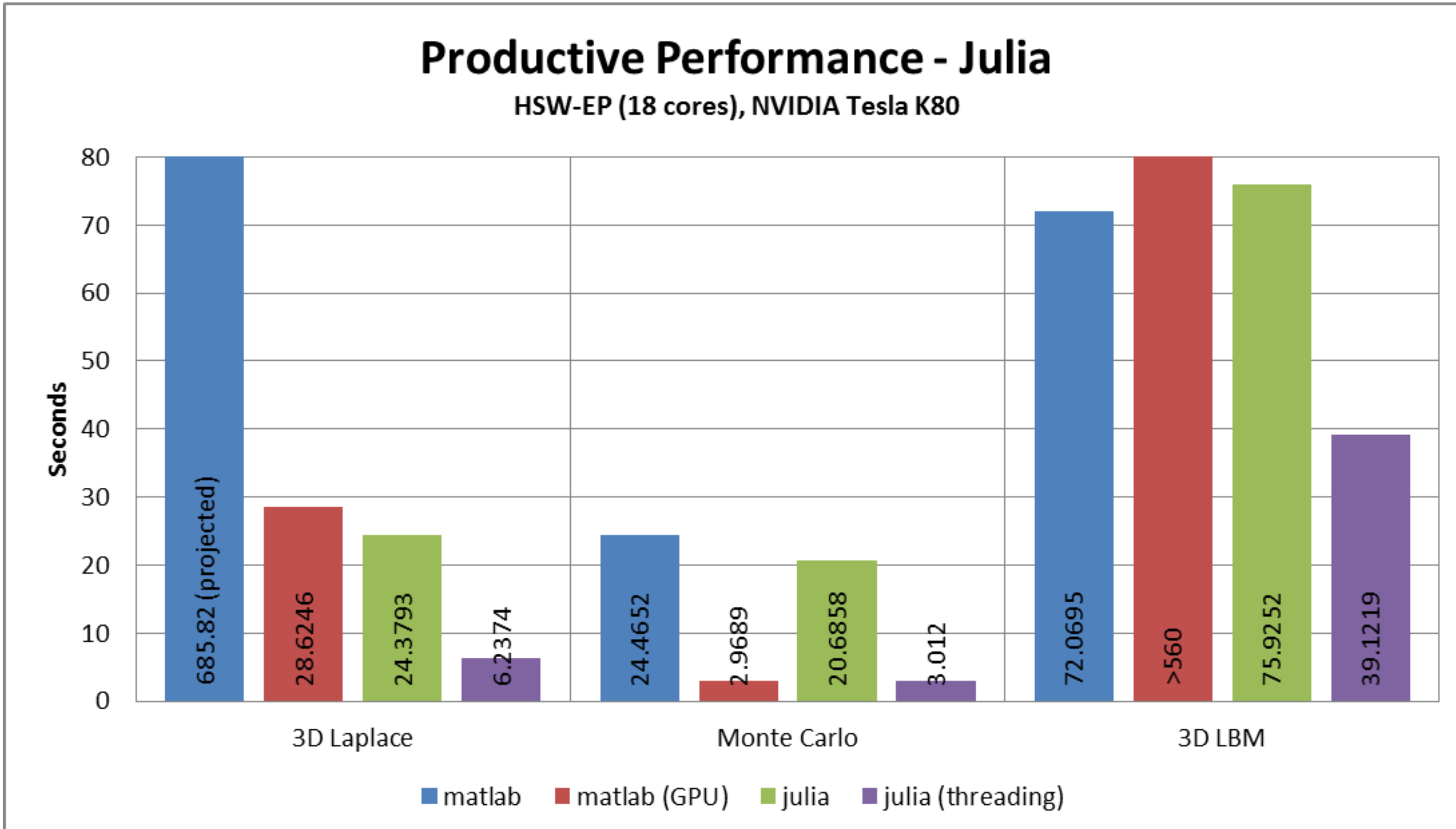
Wait and retrieve result

- Each process is single-threaded.
- Julia coroutine mechanism simplifies writing inter-process synchronization code.
- Also has distributed array objects.

Source: Arch Robison of Intel

Shared Memory Parallelism

Kiran Pamnany of Intel has been experimenting with adding threading to Julia … with some exciting preliminary results



Third Party names are the property of their owners

Julia and TileDB

```julia
# A' when A is an `AbstractMatrix{T}` is lowered to
# `ctranspose{T}(A::AbstractMatrix{T})` in Julia

function ctranspose{T}(TA::TileDB.TiledMatrix{T})
    # Create a new TileDB Matrix with the same schema but transposed dimensions
    dims = size(TA)
    TAᵀ = similar(TA, dims[2], dims[1])

    TA  = open(TA, "r")
    TAᵀ = open(TAᵀ,"w")

    # Add cell to new TiledMatrix with conj value and transposed coordinates
    # TileDB will do a parallel sort update when the cell iterator is finalized
    celliter = TileDB.CellIterator(TA)
    try
        for cell in celliter
            i,j = cell.coords
            push!(TAᵀ, TileDB.Cell{T}((j,i), conj(cell.val)))
        end
    finally
        finalize(celliter)
        close(TAᵀ)
        close(TA)
    end
    return TAᵀ
end
```

TileDB: a persistence layer for out-of-core computations.

Conjugate transpose (A') with TileDB's Arrays, TileDB's generic Array Cell types, and flexible iterator framework.

- Conjugate transpose is one of 5 core methods needed for Julia's generic iterative solvers framework. We have leveraged TileDB and Julia to prototype out-of-core linear algebra operations around these generic abstractions.

Source: Jake Bolewski and Stavros Papadopoulos

Summary

- If "One size does not fit all™" … Then we need Polystore.
- Productivity demands a single interface to multiple stores:
  - BigDAWG API to tie Islands together
  - BQL: the Dream of one Algebra to rule them all

Work in progress

- TileDB: The advantage of a domain specific storage engine.
- Julia: maybe its time to consider a new language?

I'm skeptical, but willing to be convinced

Trademark held by Mike Stonebraker