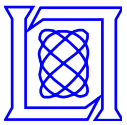


Persistent Surveillance Supercomputing in a Can

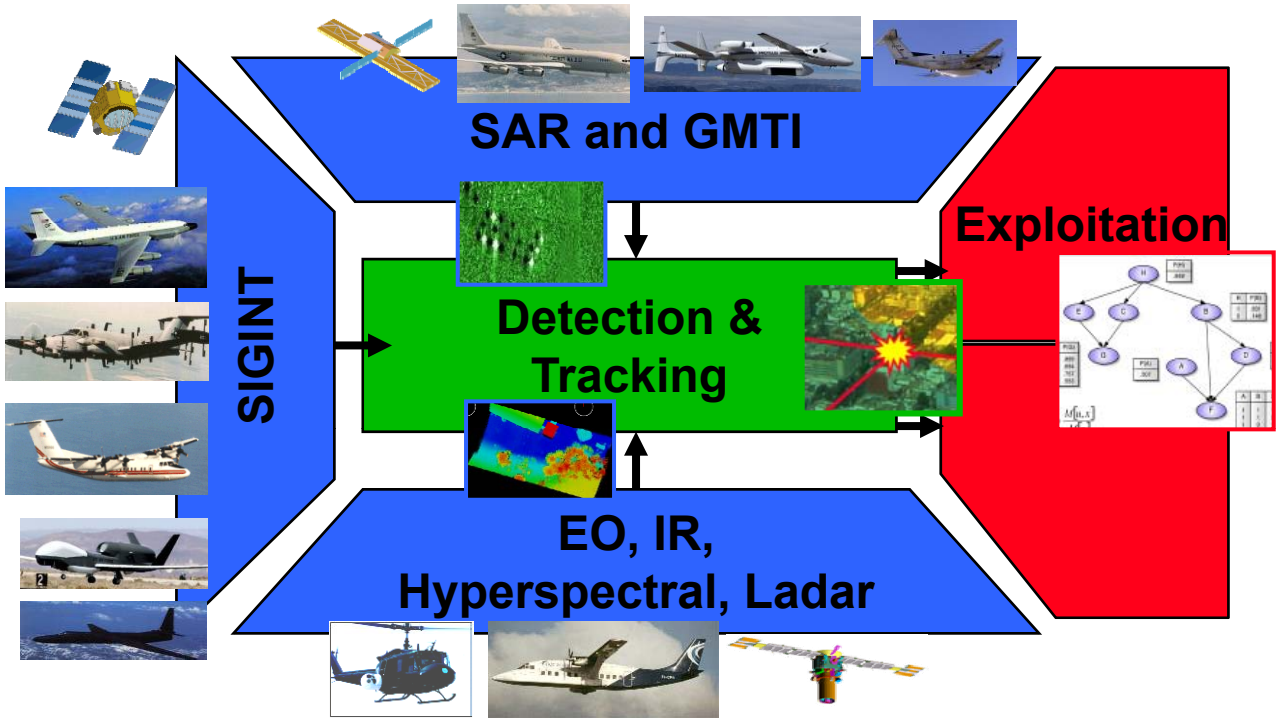
Jeremy Kepner, William Arcand, Chansup Byun, Bill Bergeron, Matthew Hubbell, Andrew McCabe, Peter Michaleas, Julie Mullen & Albert Reuther

HPEC Workshop

September 15, 2010

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Persistent Surveillance Supercomputing Requirements



Stage	Calibration & registration	Detection & tracking	Exploitation
Algorithms	Front end signal processing	Back end signal processing	Graph analysis
Data	Sensor inputs	Dense Arrays	Graphs
Kernels	FFT, FIR, SVD, …	Kalman, MHT, …	BFS, DFS, SSSP, …
Architecture	Embedded	Cloud	Graph processor
Efficiency	25% - 100%	10% - 25%	< 0.1%

Algorithm prototyping
- Front end
- Back end
- Exploitation

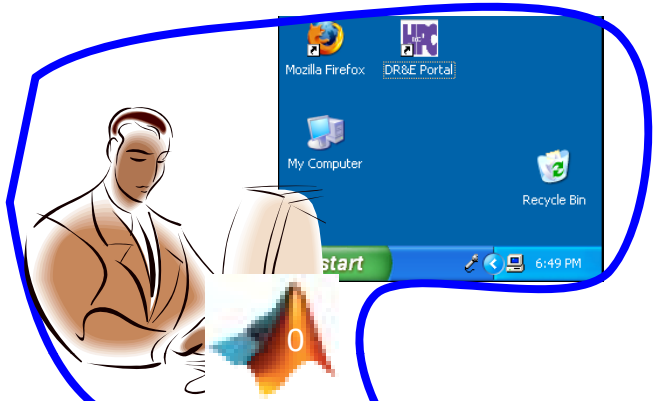
Processor prototyping
- Embedded
- Cloud
- Graph

Outline

- **Introduction**
  - *LLGrid*
  - *Interactive Supercomputing*
  - *Parallel Matlab*
  - *Usage*
- Front End Processing
- Back End Processing
- System Architecture
- Summary

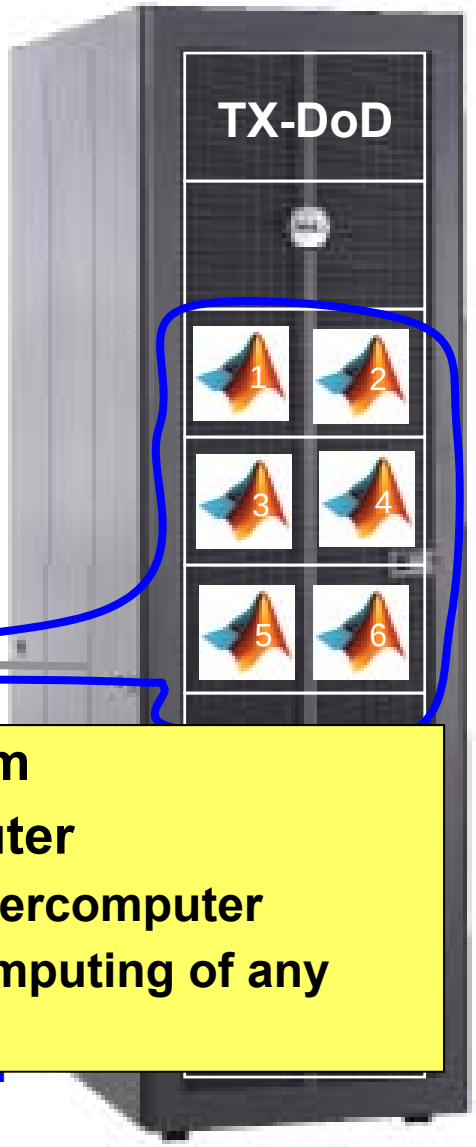
What is LLGrid?

Best of desktop + Best of supercomputing



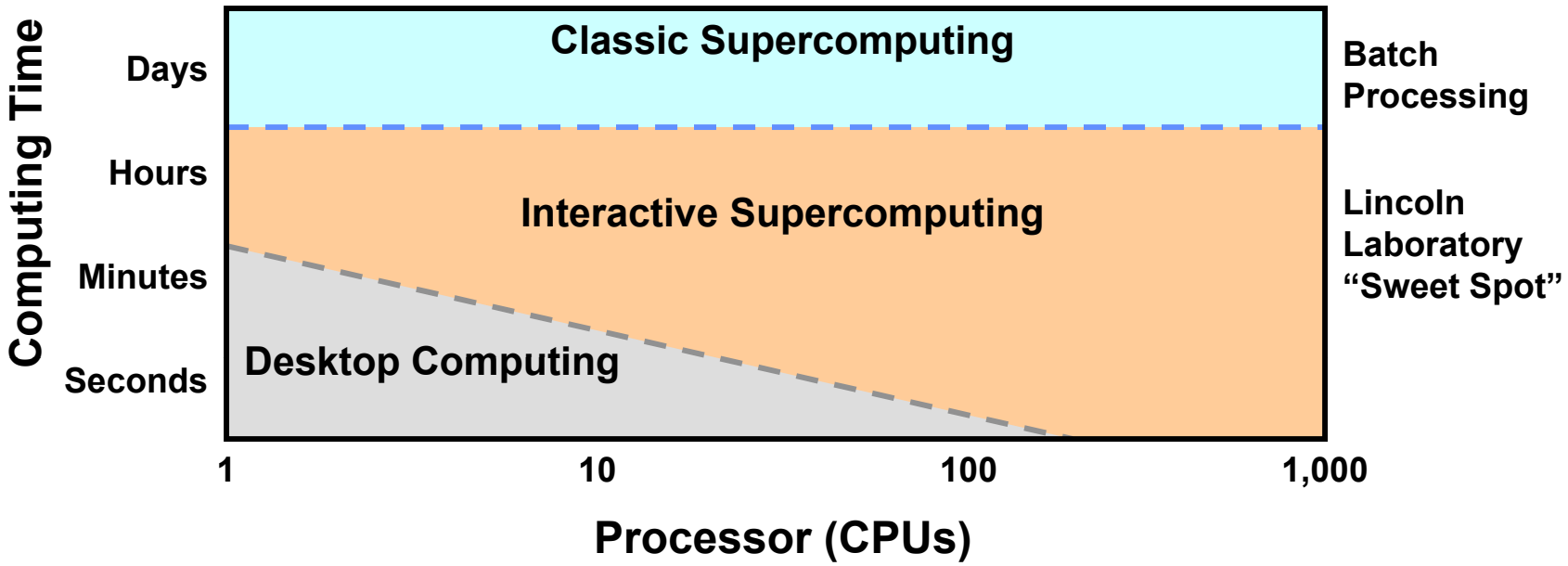
Interactive …
"what if scenarios"

Good for experts, great for novices



- LLGrid is a ~400 user ~2000 processor system
- World's only desktop interactive supercomputer
  - Dramatically easier to use than any other supercomputer
  - Highest fraction of staff using (20%) supercomputing of any organization on the planet

LLGrid Interactive Supercomputing



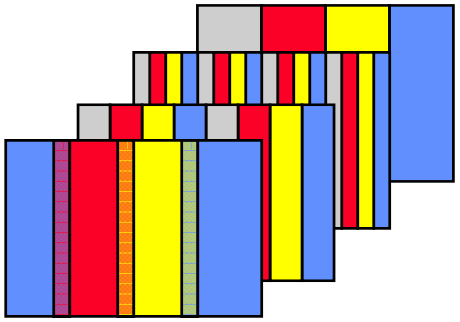
- Classic supercomputing: Jobs take hours/days to run but jobs tolerate waiting in a queue
- Interactive supercomputing: Jobs are large requiring answers in minutes/hours but can not tolerate waiting in a queue
- Desktop computing: Jobs take minutes on a desktop (e.g., algorithm proof-of-principles)

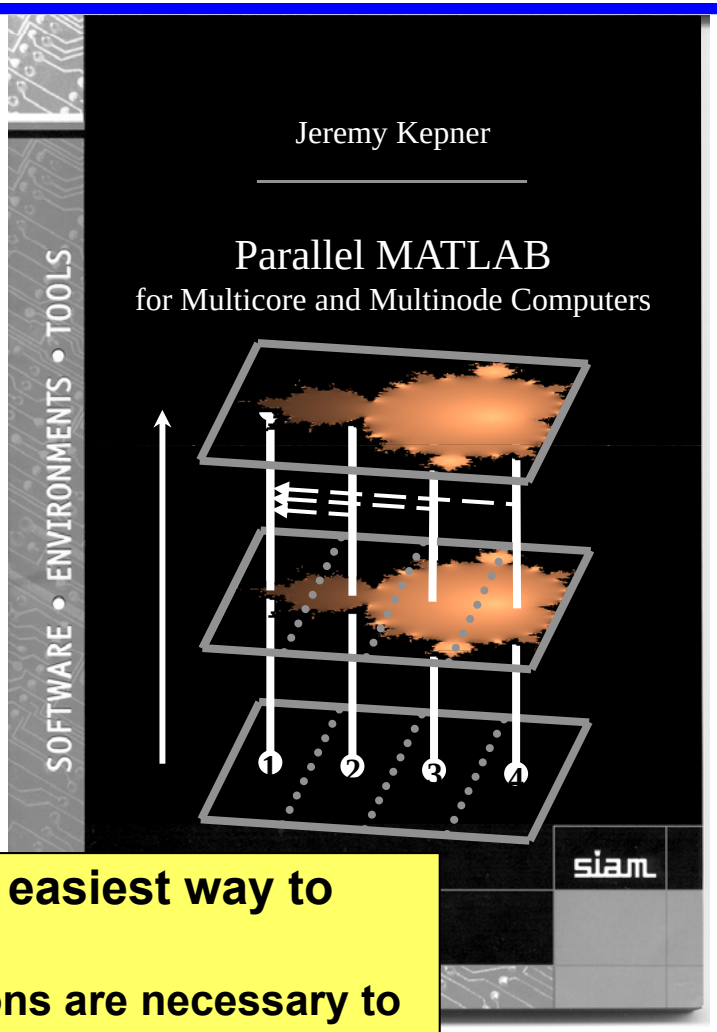
Why is LLGrid easier to use?

Universal Parallel Matlab programming

```
Amap = map([Np 1],{},0:Np-1);
Bmap = map([1 Np],{},0:Np-1);
A = rand(M,N,Amap);
B = zeros(M,N,Bmap);
B(:,:) = fft(A);
```

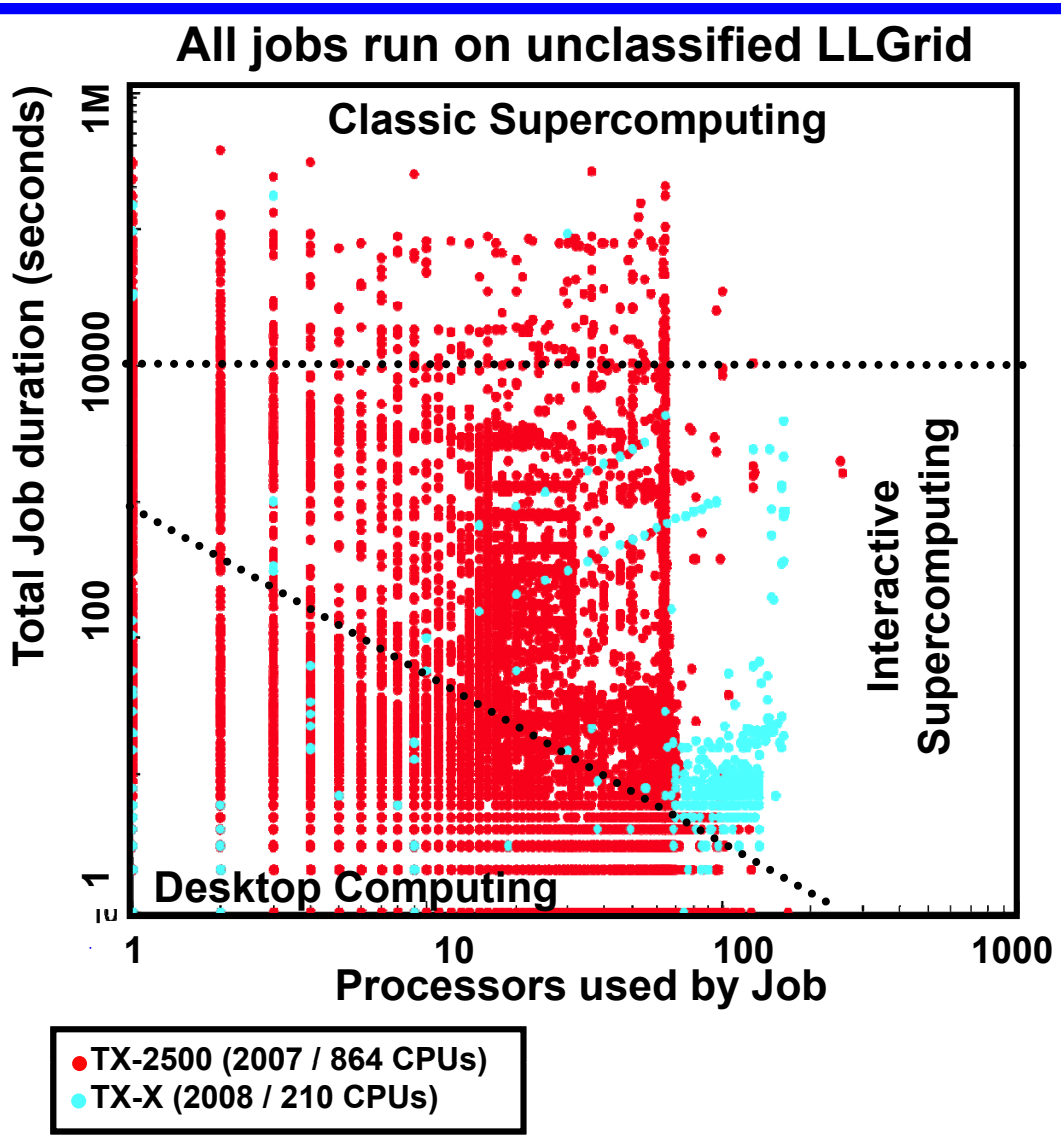
- pMatlab runs in all parallel Matlab environments
- Only a few functions are needed
  - Np
  - Pid
  - map
  - local
  - put_local
  - global_index
  - agg
  - SendMsg/RecvMsg





- **Distributed arrays have been recognized as the easiest way to program a parallel computers since the 1970s**
  - **Only a small number of distributed array functions are necessary to write nearly all parallel programs**
- **LLGrid is the first system to deploy interactive distributed arrays**

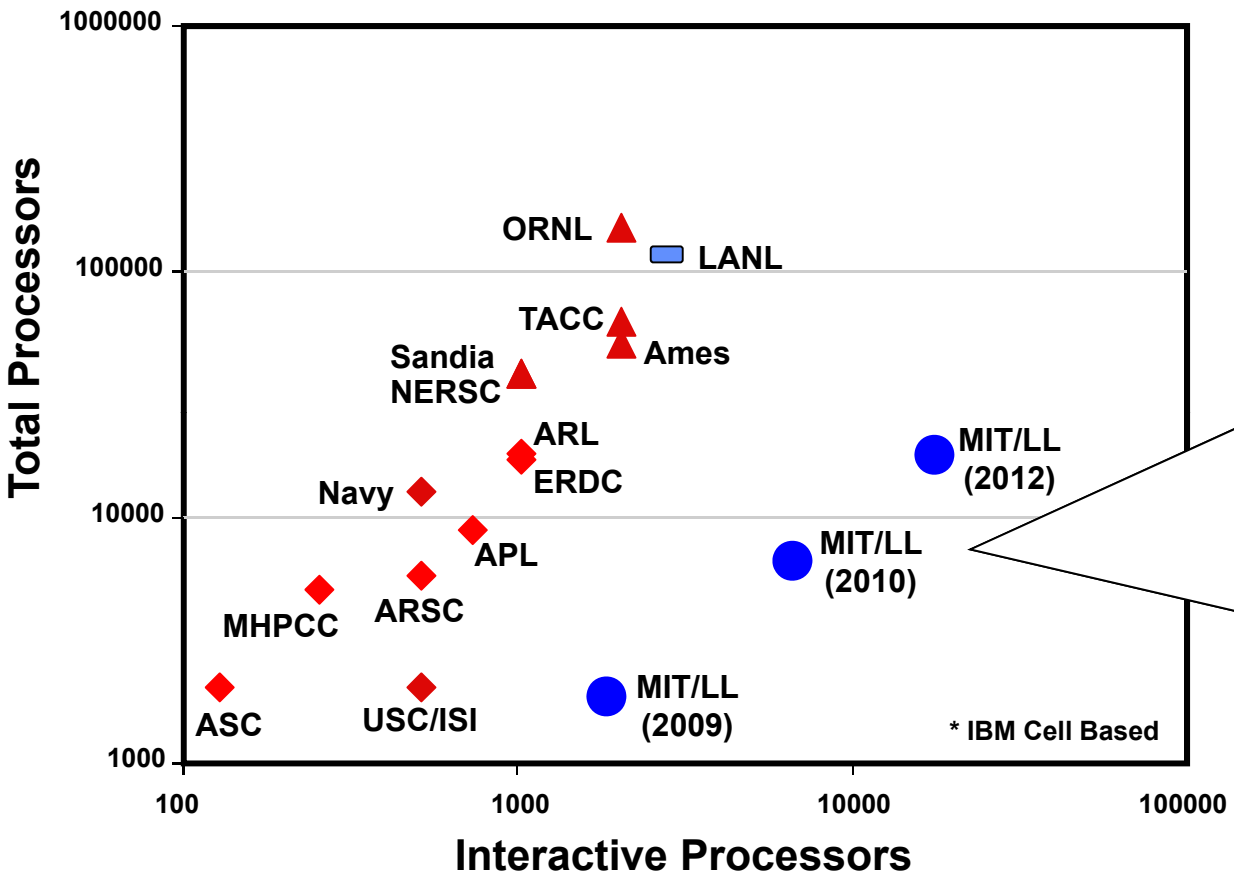
LLGrid Usage (Past Year)



- **Desktop Computing**
  - CPU-time < 20 minutes
- **Classic Supercomputing**
  - Wall-clock time > 3 hours
- **Interactive Supercomputing**
  - Between Desktop and Classic Supercomputing
  - shortens the time to insight
  - 10 development turns/day instead of 1 turn/week

	Unclassified
Users	148
Jobsets	323,151
Jobs	5,245,995
Subjobs	30,087,536
CPU Years	581.8

Next Generation System

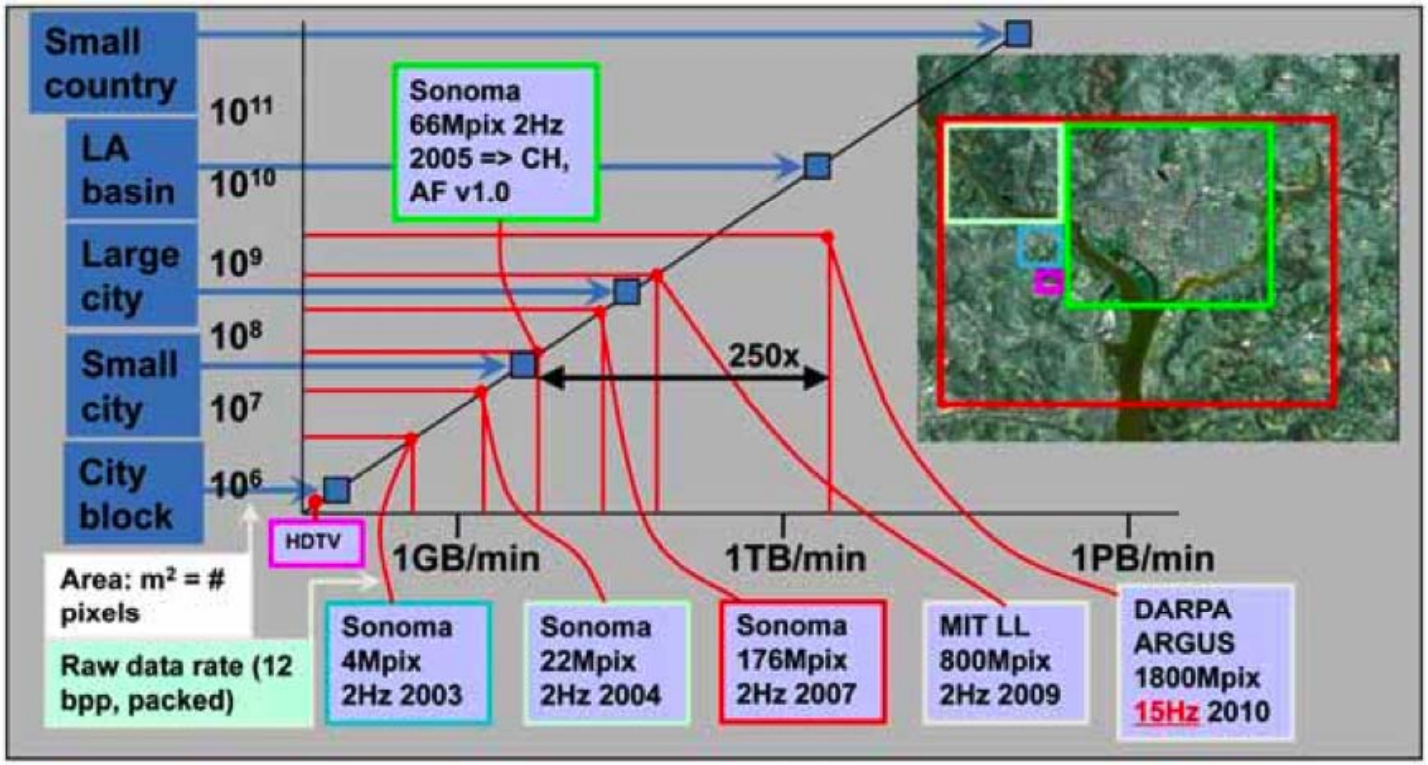


- HPCMP system 4x over current LLGrid (maintains LL leadership position in interactive HPC)
- Container based solution 12x more Flops/Watt (40x more Flops/CO2) over current LLGrid

Outline

- Introduction
- **Front End Processing**
  - ***Data Volumes***
  - ***Cloud Storage***
- Back End Processing
- System Architecture
- Summary

Persistent Surveillance Data Rates



- Persistent Surveillance requires watching large areas to be most effective
- Surveilling large areas produces enormous data streams
- Must use distributed storage and exploitation

Cloud Computing Concepts

Data Intensive Computing

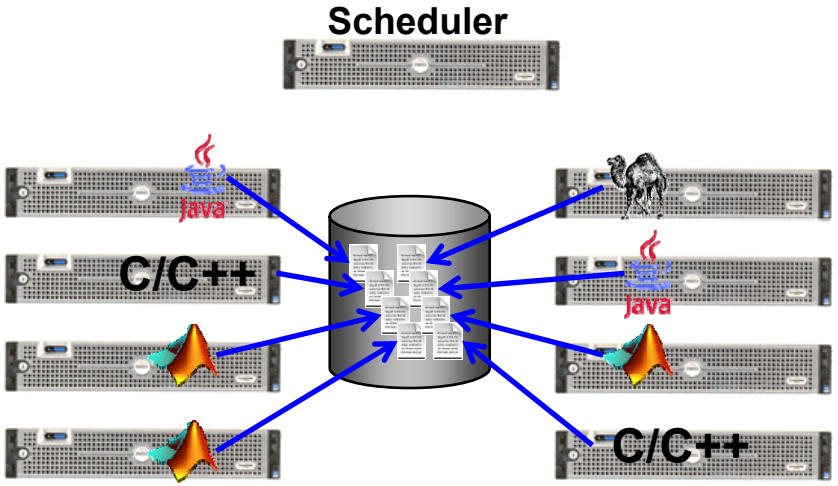
- Compute architecture for large scale data analysis
  - Billions of records/day, trillions of stored records, petabytes of storage
    - Google File System 2003
    - Google MapReduce 2004
    - Google BigTable 2006
- Design Parameters
  - Performance and scale
  - Optimized for ingest, query and analysis
  - Co-mingled data
  - Relaxed data model
  - Simplified programming
- Community
  - Yahoo, Google, Facebook, Windows Live, Cloudera, Apache, …

Utility Computing

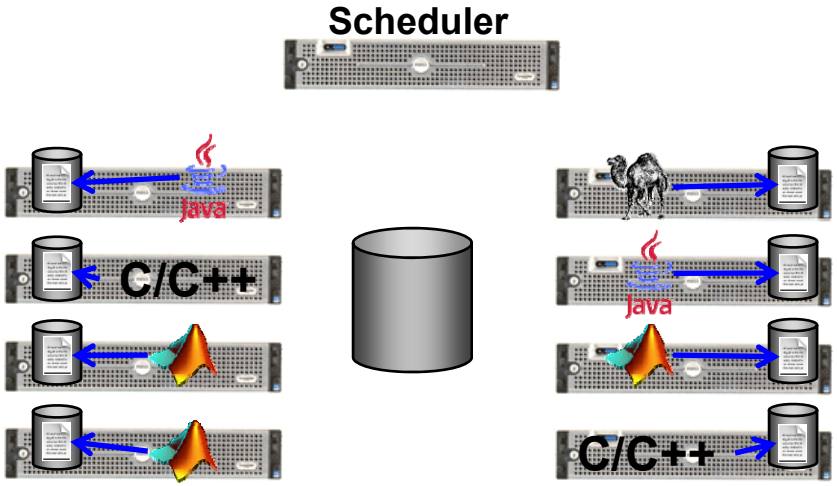
- Compute services for outsourcing IT
  - Concurrent, independent users operating across millions of records and terabytes of data
    - IT as a Service
    - Infrastructure as a Service (IaaS)
    - Platform as a Service (PaaS)
    - Software as a Service (SaaS)
- Design Parameters
  - Isolation of user data and computation
  - Portability of data with applications
  - Hosting traditional applications
  - Lower cost of ownership
  - Capacity on demand
- Community
  - Google, Salesforce.com, Amazon, Windows Azure, Cloudnine, iTricity, …

Advantages of Data Intensive Cloud: Disk Bandwidth

Traditional:
Data from central store to compute nodes



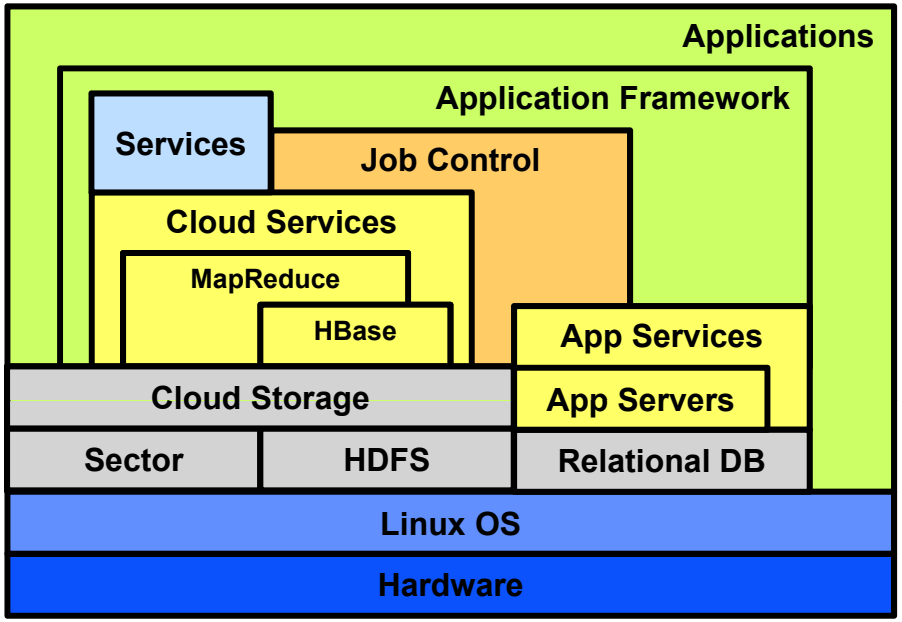
Cloud:
Data replicated on nodes, computation sent to nodes



- Cloud computing moves computation to data
  - Good for applications where time is dominated by reading from disk
- Replaces expensive shared memory hardware and proprietary database software with cheap clusters and open source
  - Scalable to hundreds of nodes

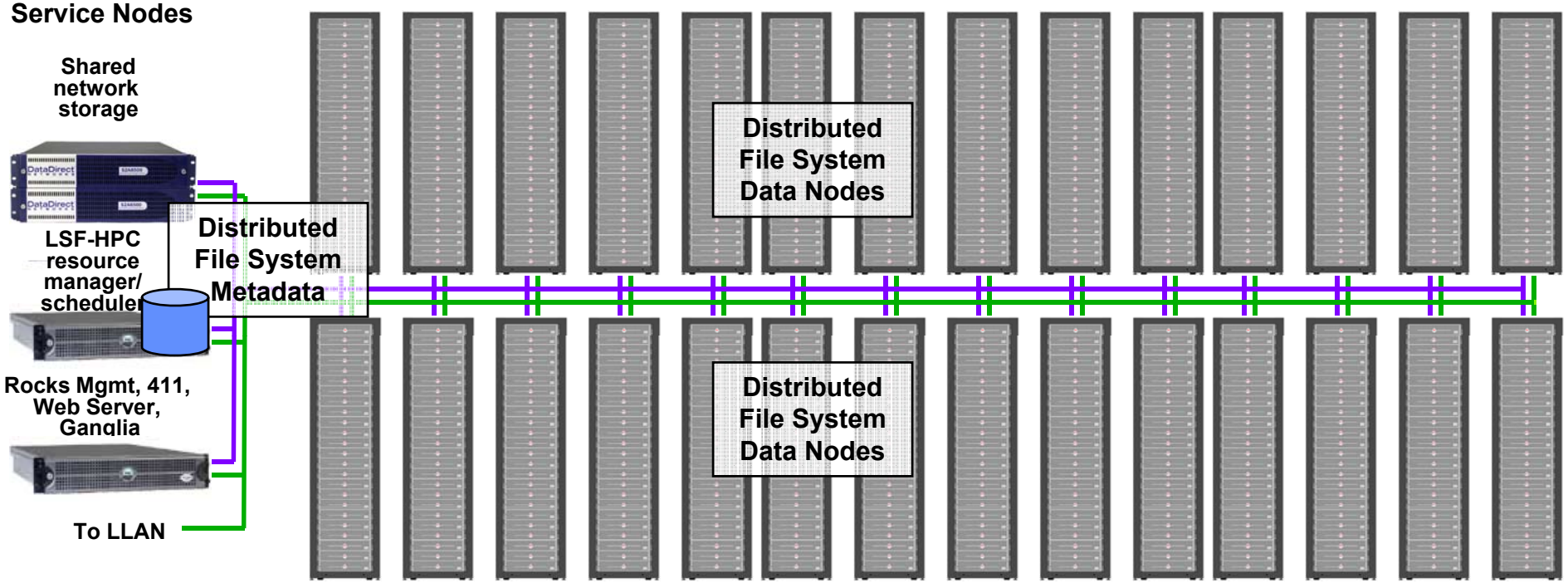
Cloud Software: Hybrid Software Stacks

- Cloud implementations can be developed from a large variety of software components
  - Many packages provide overlapping functionality
- Effective migration of DoD to a cloud architecture will require mapping core functions to the cloud software stack
  - Most likely a hybrid stack with many component packages
- MIT-LL has developed a dynamic cloud deployment architecture on its computing infrastructure
  - Examining performance trades across software components



- Distributed file systems
  - File-based: Sector
  - Block-based: Hadoop DFS
- Distributed database: HBase
- Compute environment: Hadoop MapReduce

Distributed Cloud File Systems Prototyped On TX-2500 Cluster



432 DELL PowerEdge 2850

Dual 3.2 GHz EM64-T Xeon (P4)
8 GB RAM memory
Two Gig-E Intel interfaces
Infiniband interface
Six 300-GB disk drives

- 432+5 Nodes
- 864+10 CPUs
- 3.4 TB RAM
- 0.78 PB of Disk
- 28 Racks

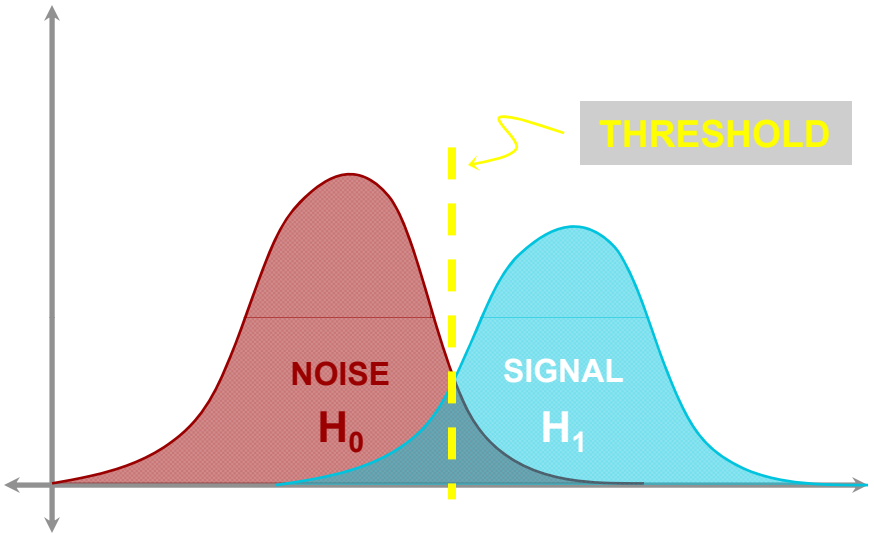
MIT-LL Cloud	Hadoop DFS	Sector
Number of nodes used	350	350
File system size	298.9 TB	452.7 TB
Replication factor	3	2

Outline

- Introduction
- Front End Processing
- **Back End Processing**
  - ***Graph Algorithms***
  - ***D4M***
  - ***Graph Visualization***
- System Architecture
- Summary

Detection Theory

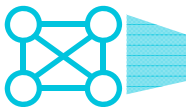
DETECTION OF SIGNAL IN NOISE



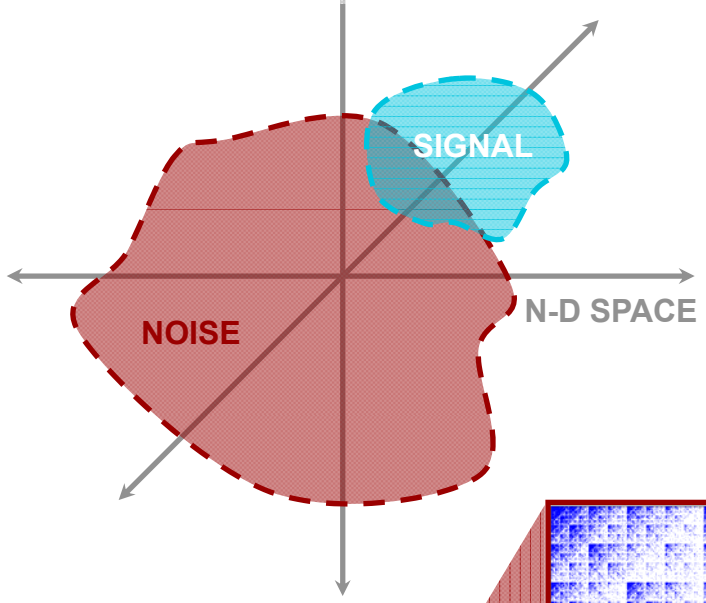
ASSUMPTIONS

- Background (noise) statistics
- Foreground (signal) statistics
- Foreground/background separation
- Model ≈ reality

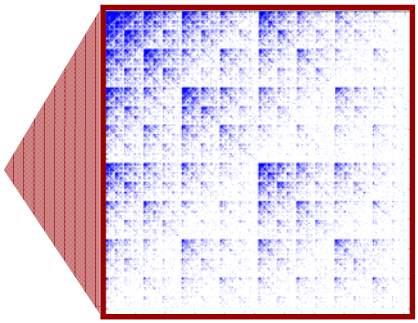
DETECTION OF SUBGRAPHS IN GRAPHS



Example subgraph of interest:
Fully connected (complete)



Example background model:
Powerlaw graph



Goal: Develop basic detection theory for finding subgraphs of interest in large background graphs

Facet Search

- Core analytic of SKS
- Gives keyword distribution of a set of documents that share a common keyword(s)
  - Provides useful guide to what keyword to select next
- Currently implemented with several hundreds of lines of Java/SQL
- Associative array implementation has 1 line

Associative arrays concept

- **Like Perl associative arrays but in 2D and mixed data types**

```
A('alice ','bob ') = 'talked '
```

or

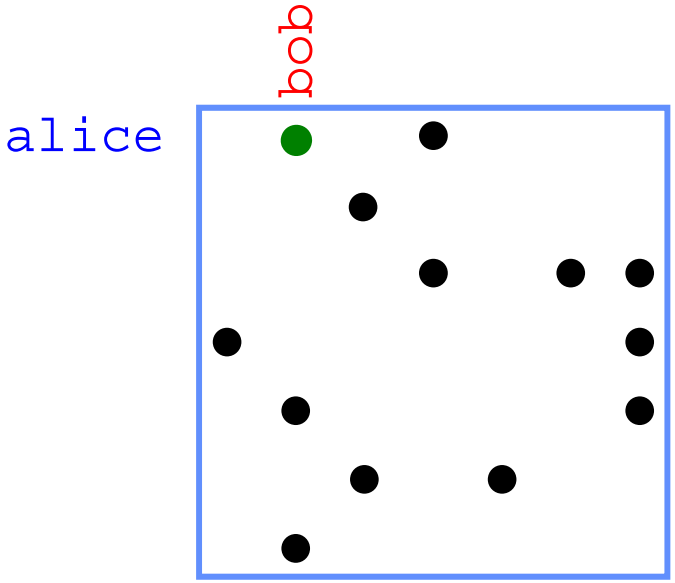
```
A('alice ','bob ') = 47.0
```

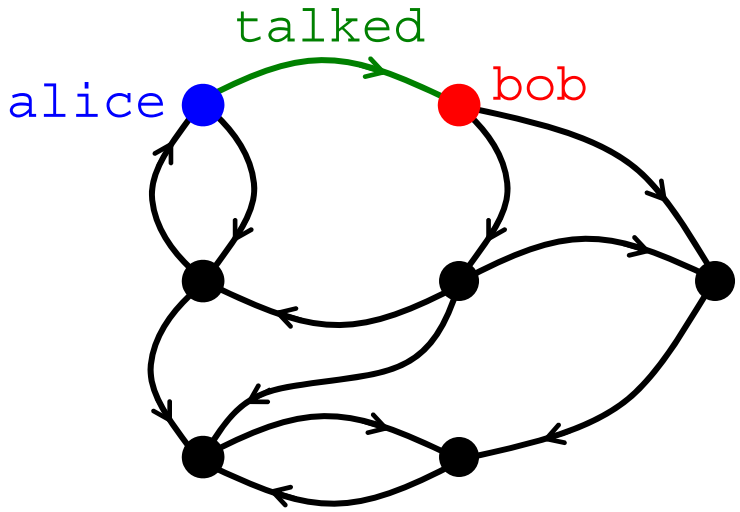
- **1-to-1 correspondence with triple store**

```
('alice ','bob ','talked ')
```

or

```
('alice ','bob ',47.0)
```

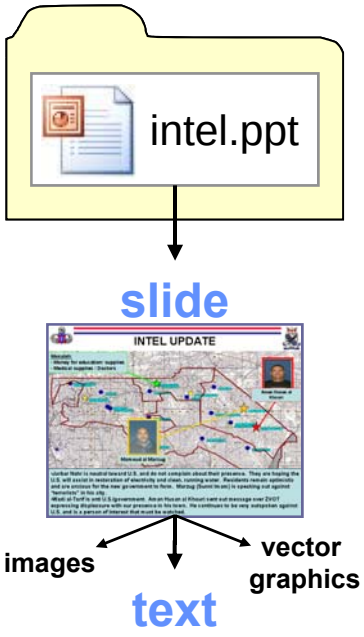




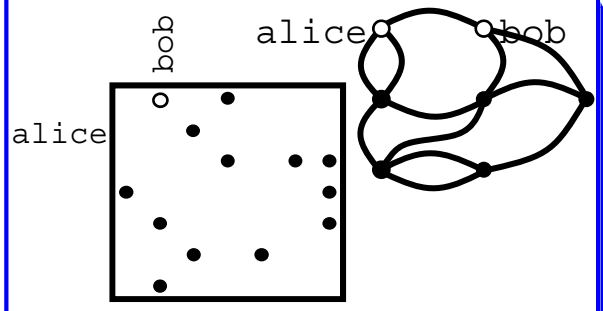
D4M: Dynamic Distributed Dimensional Data Model

SKS

File parsing



D4M



Associative Arrays

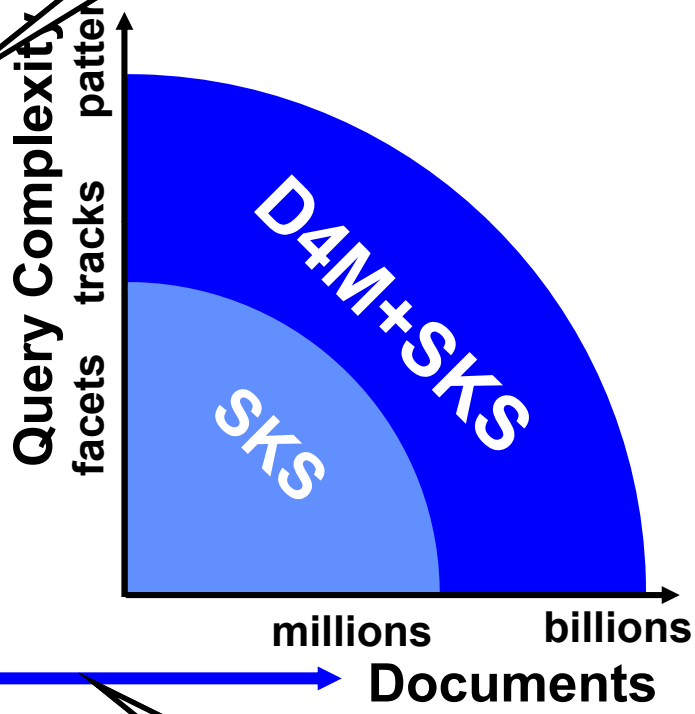
Parallel Library

- Distributed Arrays (pMatlab)
- Scheduler (gridMatlab)
- Math (Matlab)
- Messaging (MatlabMPI)

Cloud Databases

- 3-store
- 3-store

Composable
1:1 w/DB triples
50x less work

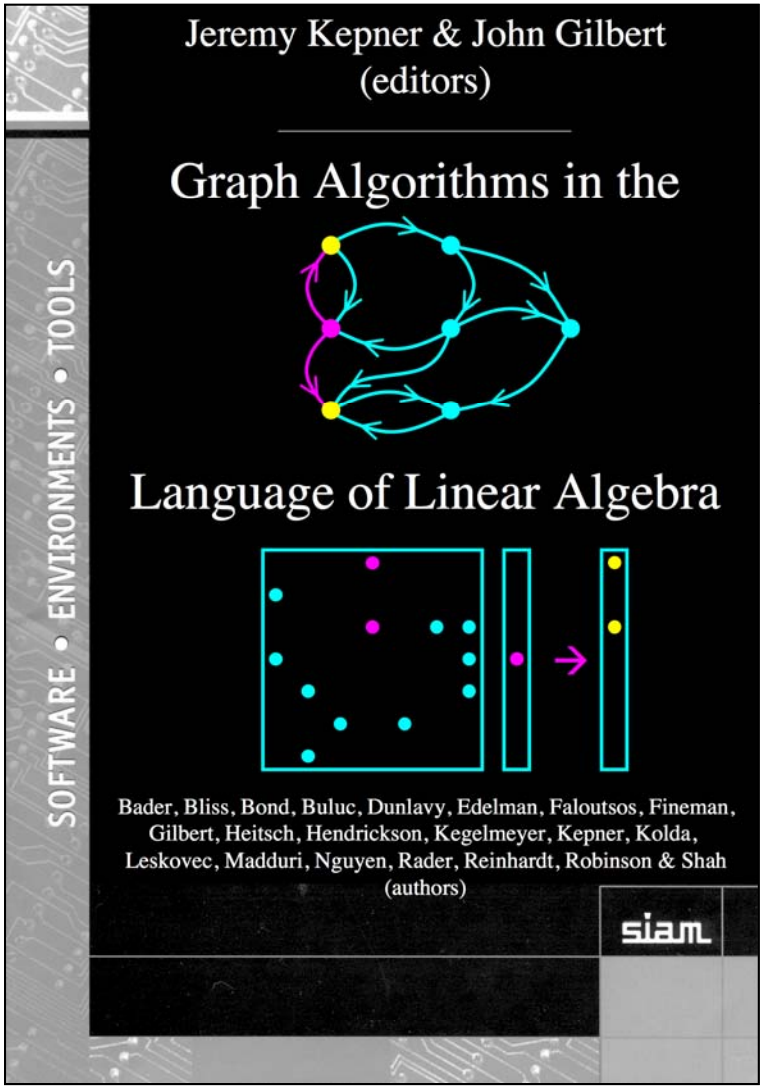


High performance
row/col queries
10x over SQL

- SKS allows analysis of unstructured documents
- DM4+SKS allows the size and complexity of queries to increase dramatically

Reference

- Book: “Graph Algorithms in the Language of Linear Algebra”
- Editors: Kepner (MIT-LL) and Gilbert (UCSB)
- Contributors
  - Bader (Ga Tech)
  - Chakrabart (CMU)
  - Dunlavy (Sandia)
  - Faloutsos (CMU)
  - Fineman (MIT-LL & MIT)
  - Gilbert (UCSB)
  - Kahn (MIT-LL & Brown)
  - Kegelmeyer (Sandia)
  - Kepner (MIT-LL)
  - Kleinberg (Cornell)
  - Kolda (Sandia)
  - Leskovec (CMU)
  - Madduri (Ga Tech)
  - Robinson (MIT-LL & NEU), Shah (UCSB)



Outline

- Introduction
- Front End Processing
- Back End Processing
- **System Architecture**
  - ***Requirements***
  - ***System Trades***
  - ***Energy Efficiency***
- Summary

Next Generation System Goals

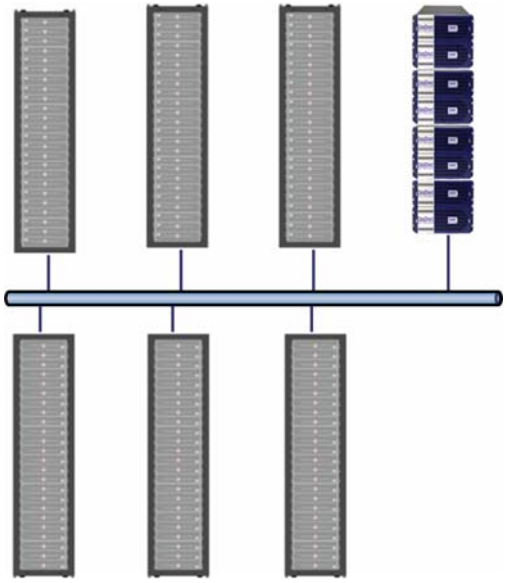
Minimum Requirements	
Processing Cores	7200
RAM	19.2 TB
Storage	2.4 PB
Storage Configuration	Hybrid Solution Preferred
Data Center Infrastructure	Shipping Container - Self Contained Data Center Module
Network	10GbE 2:1 Oversubscription

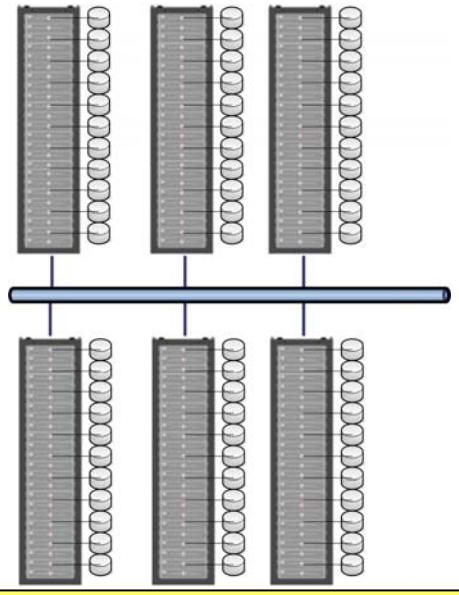
Proposed Systems
Storage System Solutions

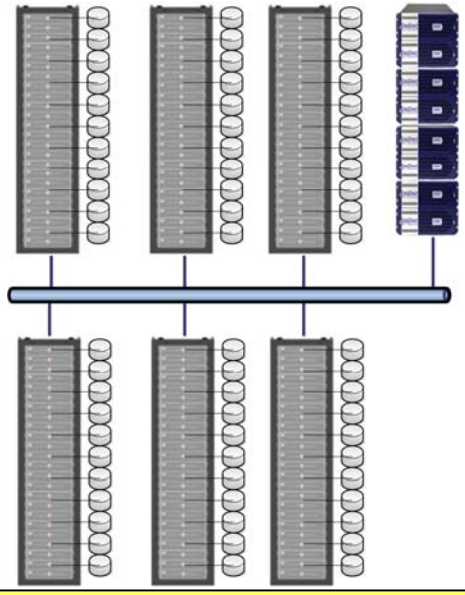
- **Centralized Storage**
  - Bandwidth Limited by Central Storage Devices
  - Greater Redundancy
  - Fewer Components

- **Distributed Storage**
  - Very Large Potential Bandwidth
  - Less Redundancy
  - Larger Number of Components
  - Must Tradeoff Compute Capability for Storage

- **Hybrid Storage**
  - Combines the Strength of Both Solutions
  - More Expensive







- **Ongoing Analysis of Tradeoffs as Vendor Proposals Mature**

Big & Green

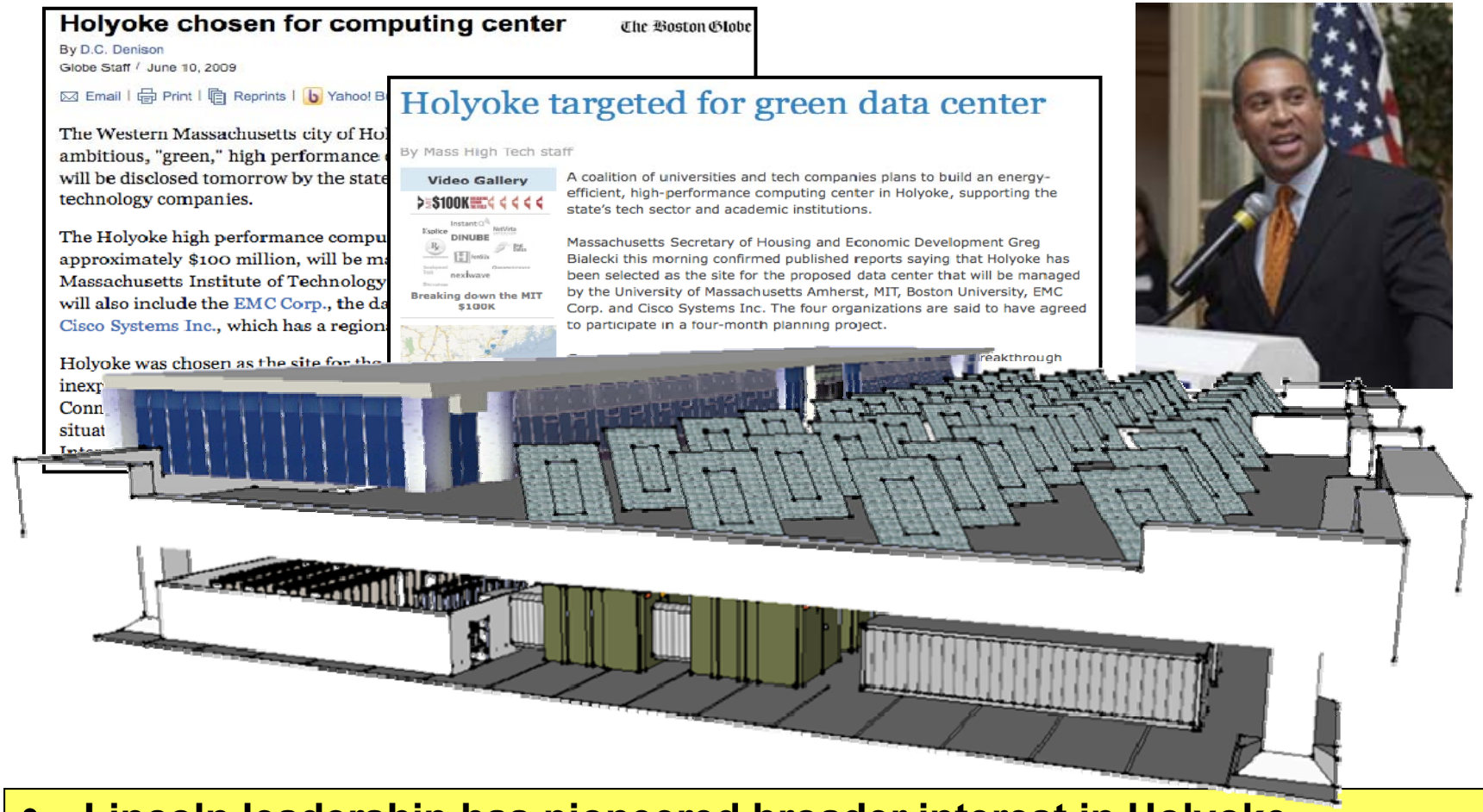
- Hundreds of nodes and thousands of cores
- Teraflops of Compute, Terabytes of RAM, Petabytes of disk
- Shipping container (no building required), located near green energy
- 3x better Flops/Watt, 10x better Flops/CO2

Delivered Flops

	Batch	Interactive
User Load	N_P	N_P
Utilization	90%	50%
Time dilation	1.8	1
Overkill*	1.8	1
Required system	$1.8N_P/0.9 = 2N_P$	$N_P/0.5 = 2N_P$
Machine Flops	$1.8\ N_P$	N_P
User Flops	N_P	N_P
User Flops/Watt	N_P	$2N_P$
PUE	1.5	1.1
Relative CO2/Watt	1	0.3
Center Flops/Watt	$N_P/1.5 = 0.6\ N_P$	$2N_P/1.1 = 1.8\ N_P$
Center Flops/CO2	$0.6\ N_P$	$6\ N_P$

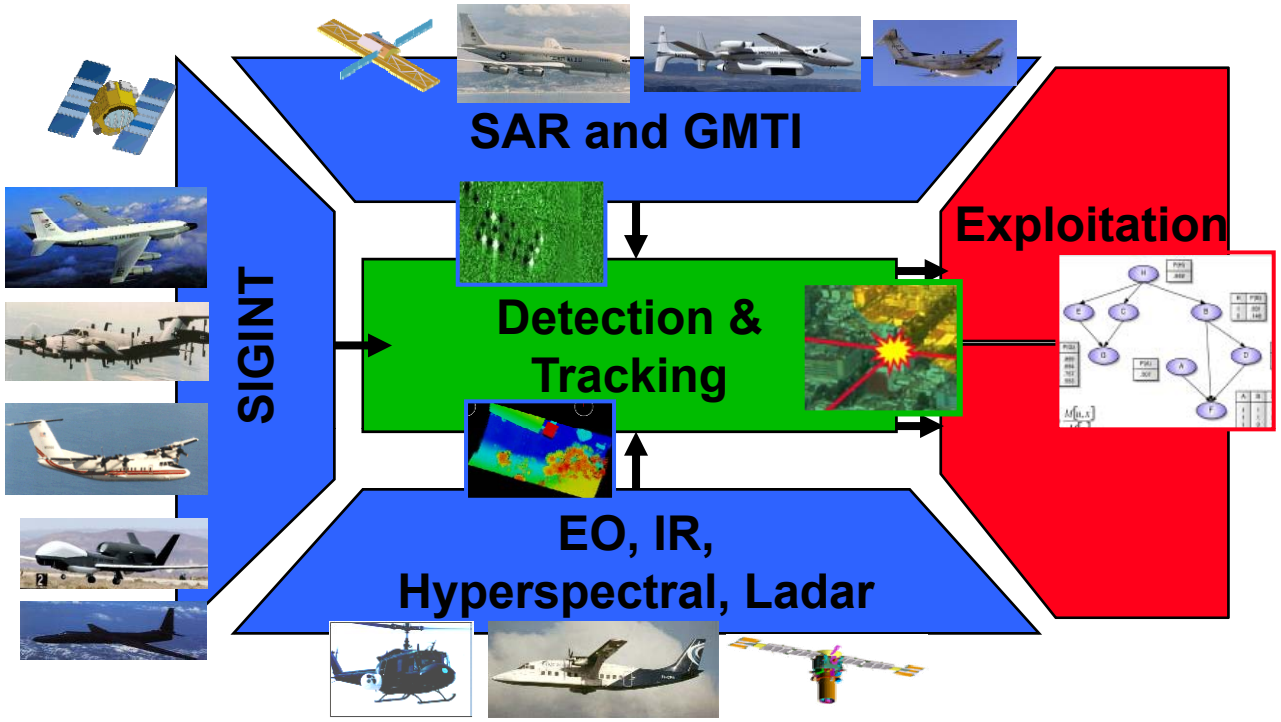
*Inefficiencies from scaling bugs, time dilation, queue hacking, unmonitored jobs, allocation burning, and job overkill

Holyoke High Performance Computing Center



- Lincoln leadership has pioneered broader interest in Holyoke
- Statewide collaboration: State, MIT, BU, UMass, NEU, EMC & Cisco
- Budget: $50M-$80M, timeframe 2012

Summary: Persistent Surveillance Supercomputing Enables



Stage	Calibration & registration	Detection & tracking	Exploitation
Algorithms	Front end signal processing	Back end signal processing	Graph analysis
Data	Sensor inputs	Dense Arrays	Graphs
Kernels	FFT, FIR, SVD, …	Kalman, MHT, …	BFS, DFS, SSSP, …
Architecture	Embedded	Cloud	Graph processor
Efficiency	25% - 100%	10% - 25%	< 0.1%

Algorithm prototyping
- Front end
- Back end
- Exploitation

Processor prototyping
- Embedded
- Cloud
- Graph