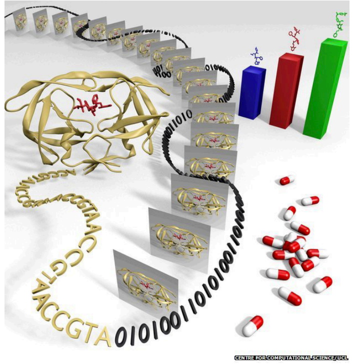
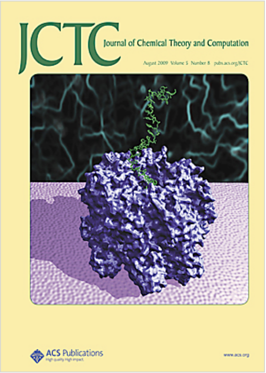
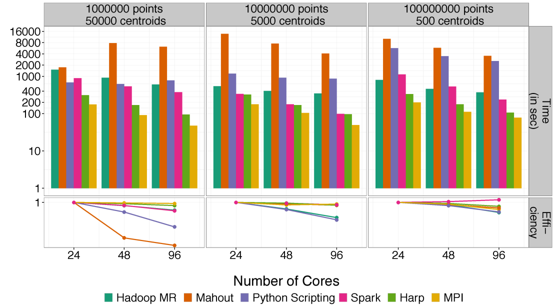






RADICAL-Cybertools: Abstractions-based Tools for Large-Scale Data-Intensive Applications

Shantenu Jha,
http://radical.rutgers.edu

RUTGERS

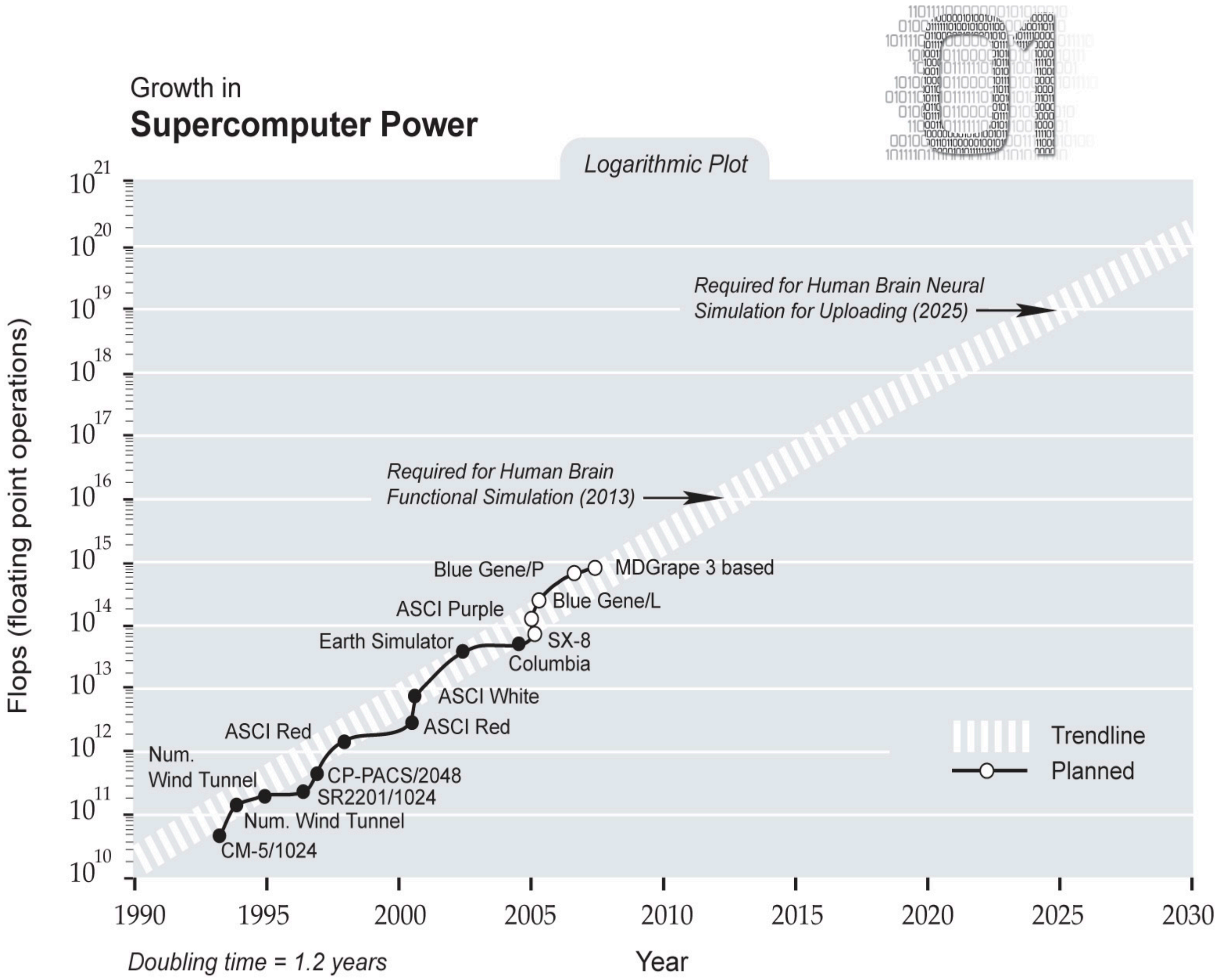
Outline: Part I (Lecture)

- A broad perspective on data-intensive science
  - Rich and diverse landscape of data-intensive architectures, applications and software systems
  - Need for Balanced, Interoperable and Federated CI
    - Architectures for data-intensive applications
    - HPC vs Grids vs Clouds?
- A Tale of Two Data Intensive Paradigms
  - BigData Ogres (mini-app, macro/micro patterns, skeleton)
    - Variations of “task level” parallelism and variants, Kmeans
  - HPC vs Hadoop/Apache Big Data stack (ABDS)
    - Convergence? Consilience between HPC and Apache/Hadoop?
- Introduction to RADICAL-Cybertools
  - Abstractions-based tools for interoperable extensible “task-level parallelism”

Outline: Part II (Hands On)

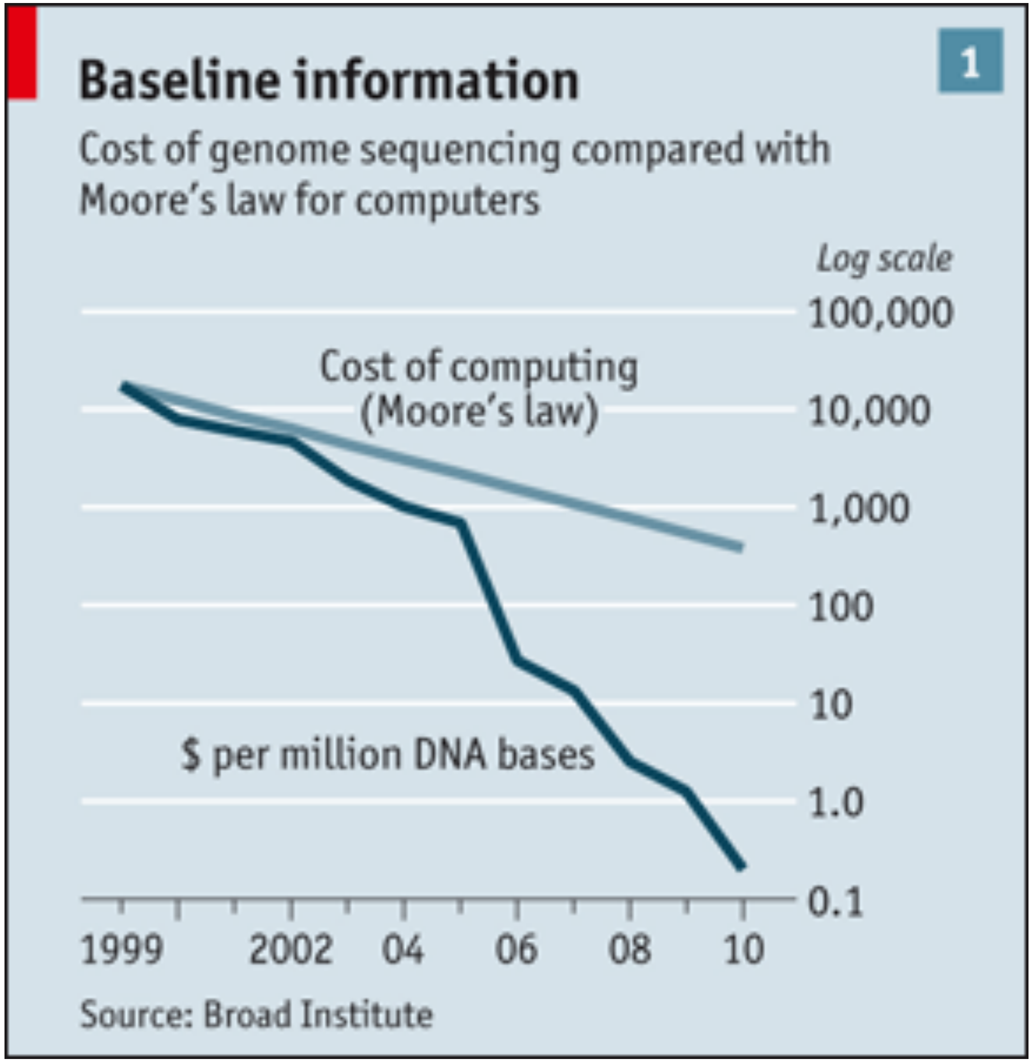
- RADICAL Cybertools:
  - RADICAL-SAGA Interoperability Layer
    - Basics
    - Tutorial
    - Side detour: SAGA-Hadoop
  - RADICAL-Pilot
    - Basics
    - Tutorial
  - Kmeans redux
    - Multiple Kmeans concurrently
    - Trade-off: number vs size
    - Kmeans Map Reduce

Exponential Growth of High-end Computing



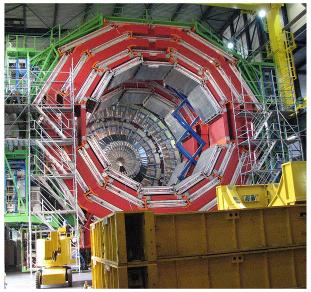
HW: Think about the implications of this graph.

Compute & Data: Two sides of the same coin
An Interesting Observation



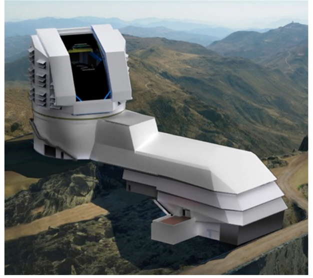
HW: Think about the implications of this graph.

Data-intensive Sciences



High Energy Physics:

- LHC at CERN produces petabytes of data per day.
- Data is processed and distributed across Tier 1 and 2 sites.



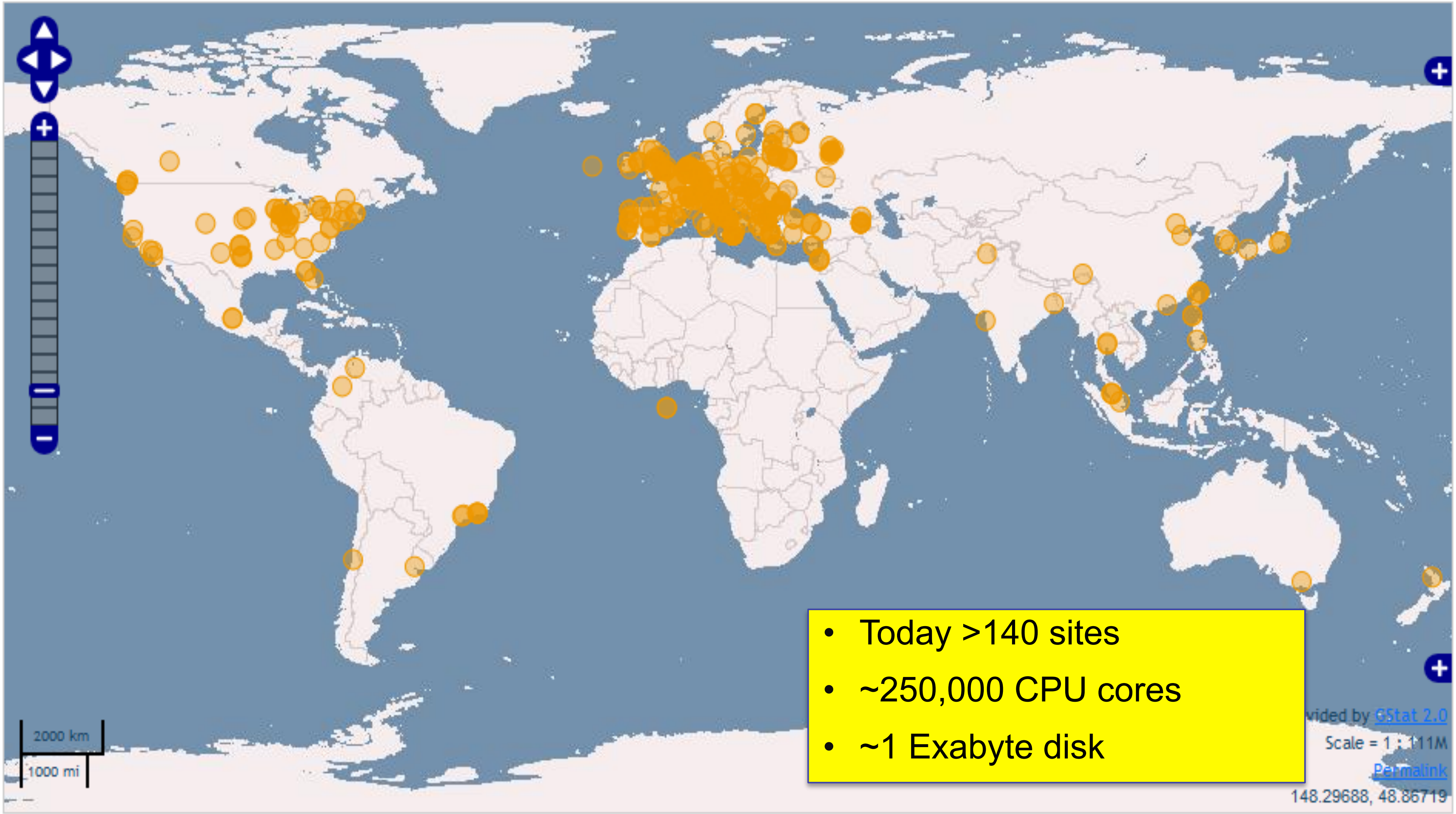
Astronomy:

- Sloan Digital Sky Survey (80 TB over 7 years).
- LSST will produce 40 TB per day (for 10 years).

Geonomics:

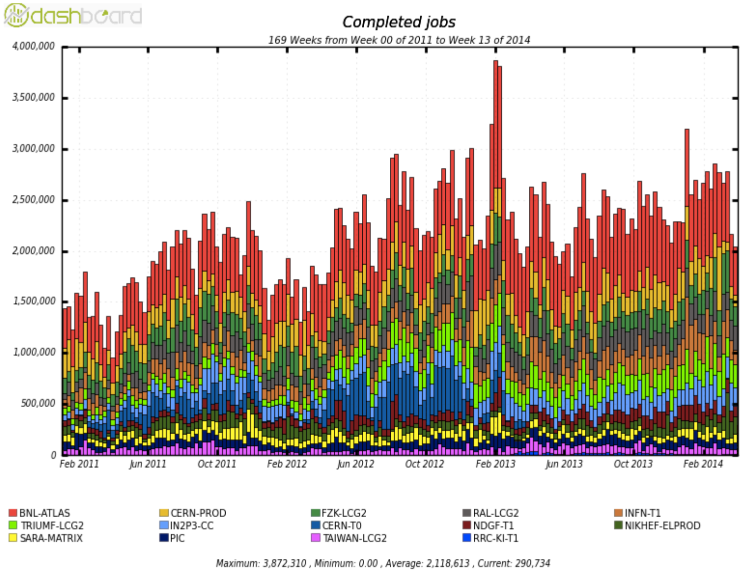
- Data volume increasing with every new generation of sequence machine. A machine can produce TB/day.
- Costs for Sequencing are decreasing.

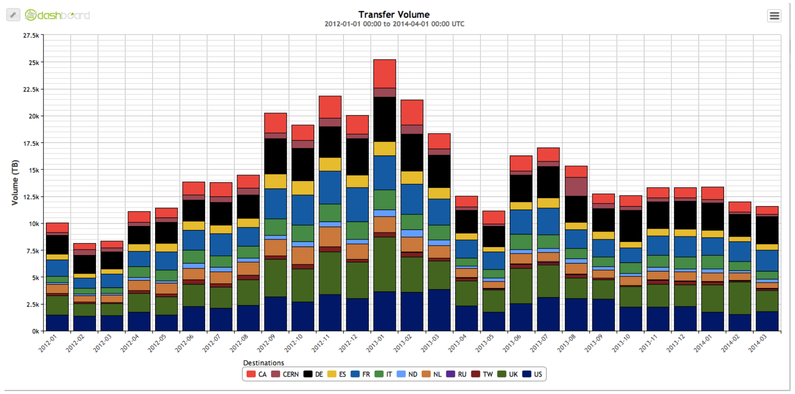
WLCG: Worldwide LHC Computing Grid
The First “Small” BigData Problem



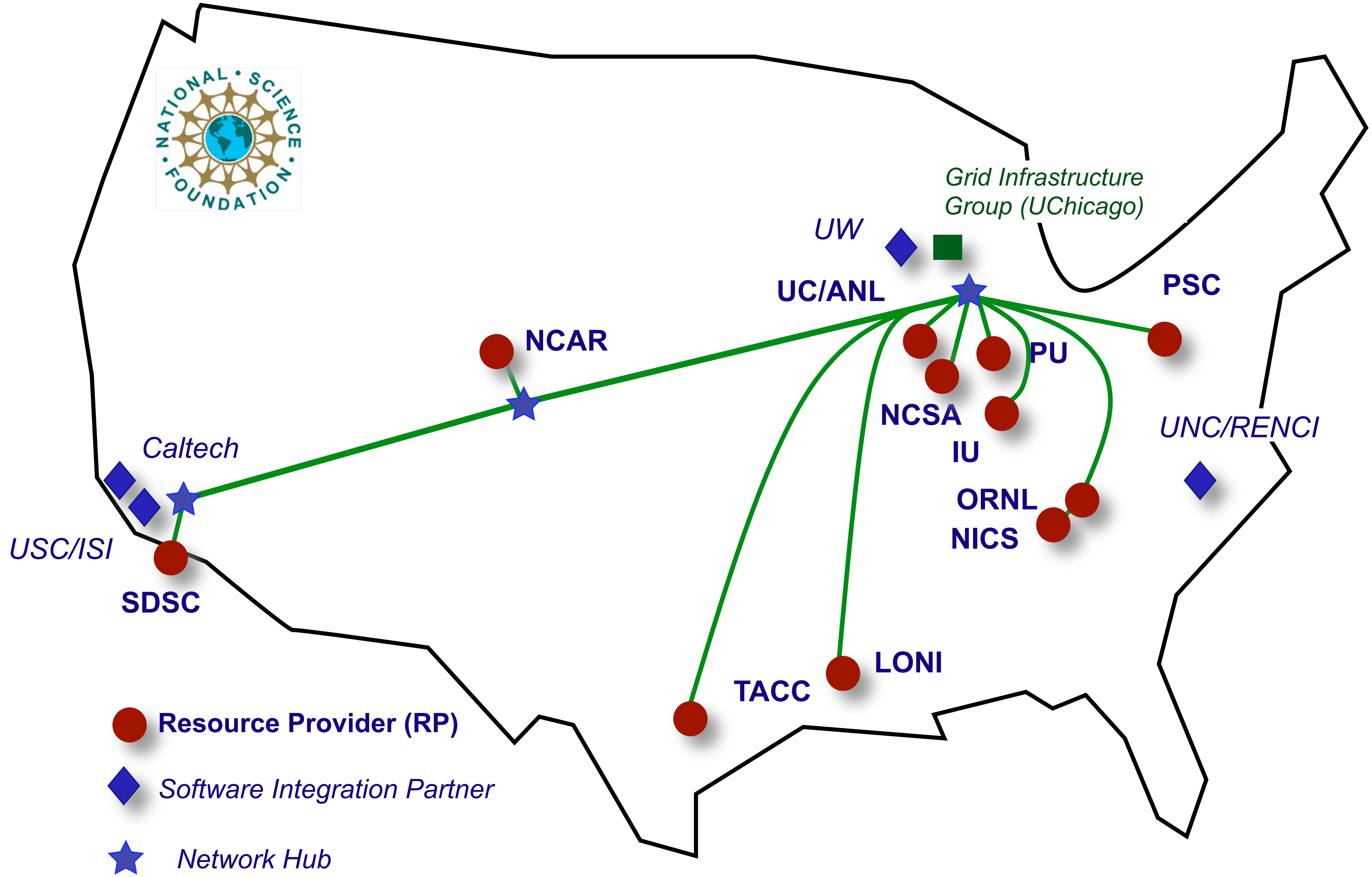
- Today >140 sites
- ~250,000 CPU cores
- ~1 Exabyte disk

ATLAS Project: Setting the Scale

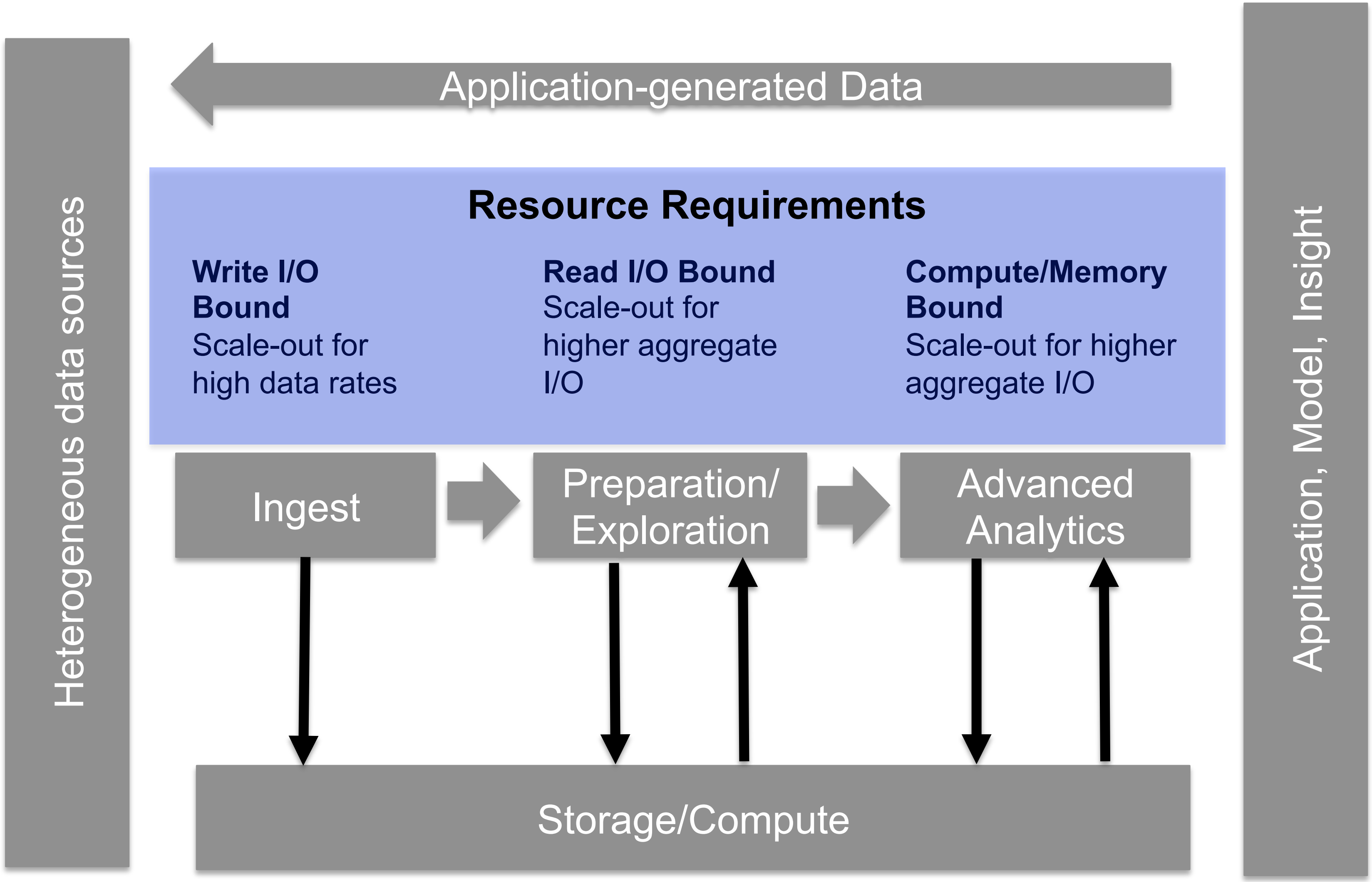




- ATLAS – one of the LHC experiments responsible for discovery of Higgs
  - Relies heavily on DCI for all computing needs
- PANDA: WMS for ATLAS. 1.2 Exabytes processed in 2013!!
- Current scale:
  - 25M jobs completed every month at > O(100) sites
  - ~O(10^9) CPU hours a year!
- Scale and complexity of computing will increase by factor of O(10)
- Example of Big Data before it was "Big"!
- Using RADICAL Cybertools (SAGA) to manage access to supercomputers


Grid Infrastructure Group (UChicago)
UW
UC/ANL
PSC
NCAR
PU
NCSA
IU
UNC/RENCI
Caltech
ORNL
NICS
USC/ISI
SDSC
LONI
TACC
● Resource Provider (RP)
◆ Software Integration Partner
★ Network Hub

Data Lifecycle and Challenges



Application-generated Data
Heterogeneous data sources
Application, Model, Insight
Ingest
Preparation/ Exploration
Advanced Analytics
Storage/Compute

Resource Requirements

Write I/O Bound
Scale-out for high data rates

Read I/O Bound
Scale-out for higher aggregate I/O

Compute/Memory Bound
Scale-out for higher aggregate I/O

Diversity of Data-Intensive Applications [courtesy GCF]

- http://bigdatawg.nist.gov/usecases.php
- **51 Detailed Use Cases: Contributed July-September 2013 Covers goals, data features such as 3 V's, software, hardware**
- **Government Operation(4):** National Archives and Records Administration, Census Bureau
- **Commercial(8):** Finance in Cloud, Cloud Backup, Mendeley (Citations), Netflix, Web Search, Digital Materials, Cargo shipping (as in UPS)
- **Defense(3):** Sensors, Image surveillance, Situation Assessment
- **Healthcare and Life Sciences(10):** Medical records, Graph and Probabilistic analysis, Pathology, Bioimaging, Genomics, Epidemiology, People Activity models, Biodiversity
- **Deep Learning and Social Media(6):** Driving Car, Geolocate images/cameras, Twitter, Crowd Sourcing, Network Science, NIST benchmark datasets
- **The Ecosystem for Research(4):** Metadata, Collaboration, Language Translation, Light source experiments
- **Astronomy and Physics(5):** Sky Surveys including comparison to simulation, Large Hadron Collider at CERN, Belle Accelerator II in Japan

Towards Balanced, Interoperable, Federated DCI

- What is Federation?
  - Federation is the collective and concurrent utilization of DCI
    - Integration and interoperability are necessary conditions
- Why Federate DCI?
  - Effective application-resource mapping
    - Application characteristics and sophistication increase
  - Application scalability
    - Peak (time) and steady-state demand, heterogeneous workload
  - Resource utilization efficiency
    - Exploit diversity, yet preserve specificity
- How to Federate?
  - Three Architectures:
    - “Grid” vs “Cloud” vs “Hybrid”
  - Types and levels of Federation

A Tale of Two Data-Intensive Pardigms: Architectures, Applications and Abstractions

Collaboratinon with Geoffrey Fox
http://arxiv.org/abs/1403.1528

Data-Intensive Application Pattern (or Structure)

- Capture “essence of these use cases”.. Classify applications into patterns, “small” kernels, mini-apps
  - Focus on cases with detailed analytics
  - Use for benchmarks of computers and software
- In parallel computing, this is well established
  - **Linpack** for measuring performance to rank machines in Top500
  - **NAS Parallel Benchmarks** (originally a pencil and paper specification to allow optimal implementations; then MPI library)
  - Other **specialized Benchmark sets** keep changing and used to guide procurements
    - Last 2 NSF hardware solicitations had NO preset benchmarks – perhaps as no agreement on key applications for clouds and data intensive applications
  - **Berkeley dwarfs** capture different structures that any approach to parallel computing must address
  - **Templates** used to capture parallel computing patterns

HPC Benchmark Classics

- **Linpack** or HPL: Parallel LU factorization for solution of linear equations
- **NPB** version 1: Mainly classic HPC solver kernels
  - MG: Multigrid
  - CG: Conjugate Gradient
  - FT: Fast Fourier Transform
  - IS: Integer sort
  - EP: Embarrassingly Parallel
  - BT: Block Tridiagonal
  - SP: Scalar Pentadiagonal
  - LU: Lower-Upper symmetric Gauss Seidel

7 Original Berkeley Dwarfs (Colella)

1. Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)
2. Unstructured Grids
3. Fast Fourier Transform
4. Dense Linear Algebra
5. Sparse Linear Algebra
6. Particles
7. Monte Carlo

Note “vaguer” than NPB

13 Berkeley Dwarfs

- Dense Linear Algebra
- Sparse Linear Algebra
- Spectral Methods
- N-Body Methods
- Structured Grids
- Unstructured Grids
- MapReduce
- Combinational Logic
- Graph Traversal
- Dynamic Programming
- Backtrack and Branch-and-Bound
- Graphical Models
- Finite State Machines

First 6 of these correspond to Colella's original.

Monte Carlo dropped
N-body methods are a subset of Particle

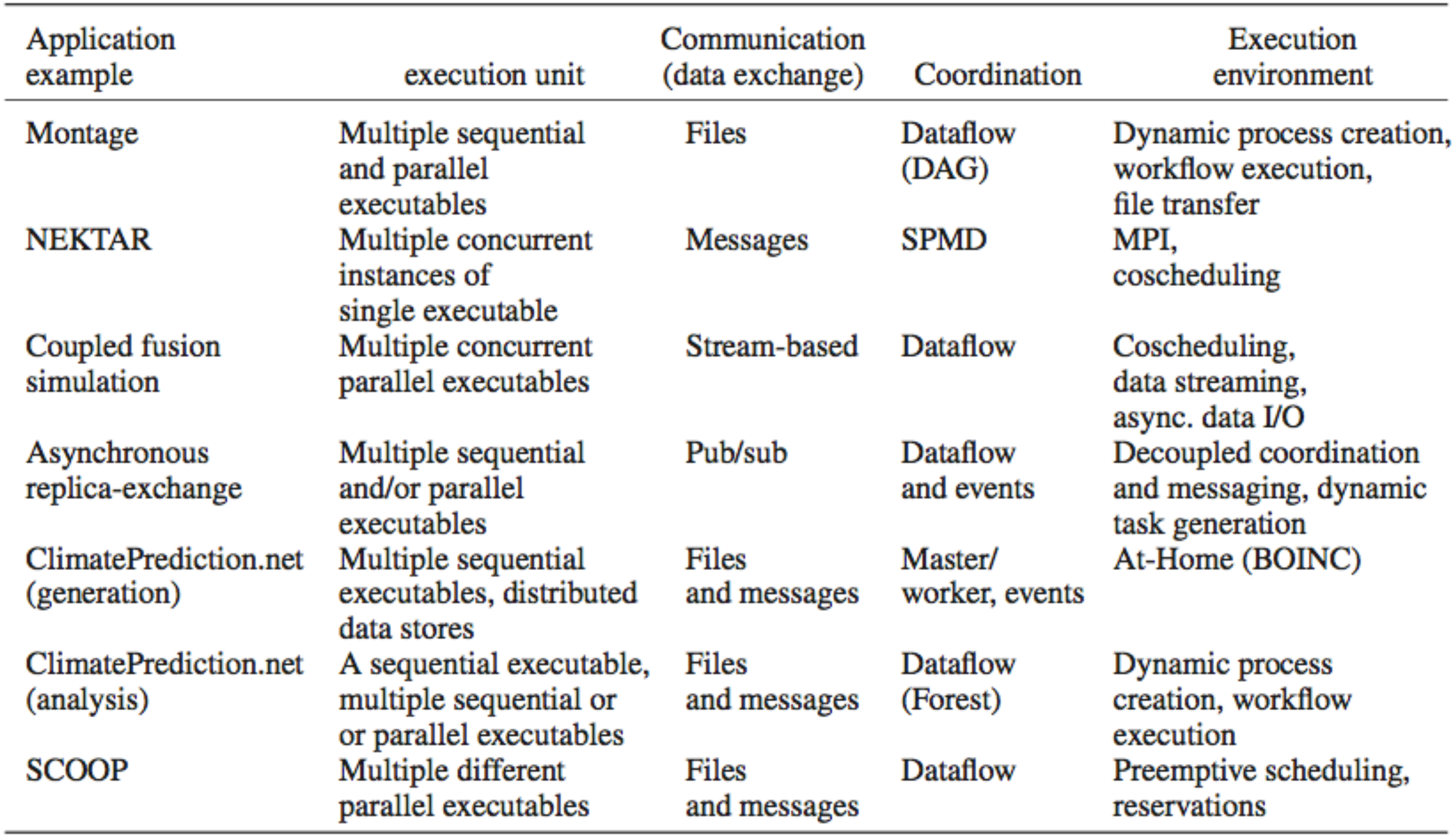
Note a little inconsistent in that MapReduce is a programming model and spectral method is a numerical method

Need multiple facets!

Distributed Computing MetaPatterns I

Jha et al, 1532-0634, 2012. DOI: 10.1002/cpe.2897CCPE

Table I. Characteristics of the set of applications arranged according to the identified vectors.



Application example	execution unit	Communication (data exchange)	Coordination	Execution environment
Montage	Multiple sequential and parallel executables	Files	Dataflow (DAG)	Dynamic process creation, workflow execution, file transfer
NEKTAR	Multiple concurrent instances of single executable	Messages	SPMD	MPI, coscheduling
Coupled fusion simulation	Multiple concurrent parallel executables	Stream-based	Dataflow	Coscheduling, data streaming, async. data I/O
Asynchronous replica-exchange	Multiple sequential and/or parallel executables	Pub/sub	Dataflow and events	Decoupled coordination and messaging, dynamic task generation
ClimatePrediction.net (generation)	Multiple sequential executables, distributed data stores	Files and messages	Master/ worker, events	At-Home (BOINC)
ClimatePrediction.net (analysis)	A sequential executable, multiple sequential or or parallel executables	Files and messages	Dataflow (Forest)	Dynamic process creation, workflow execution
SCOOP	Multiple different parallel executables	Files and messages	Dataflow	Preemptive scheduling, reservations

Distributed Computing MetaPatterns II

Jha et al

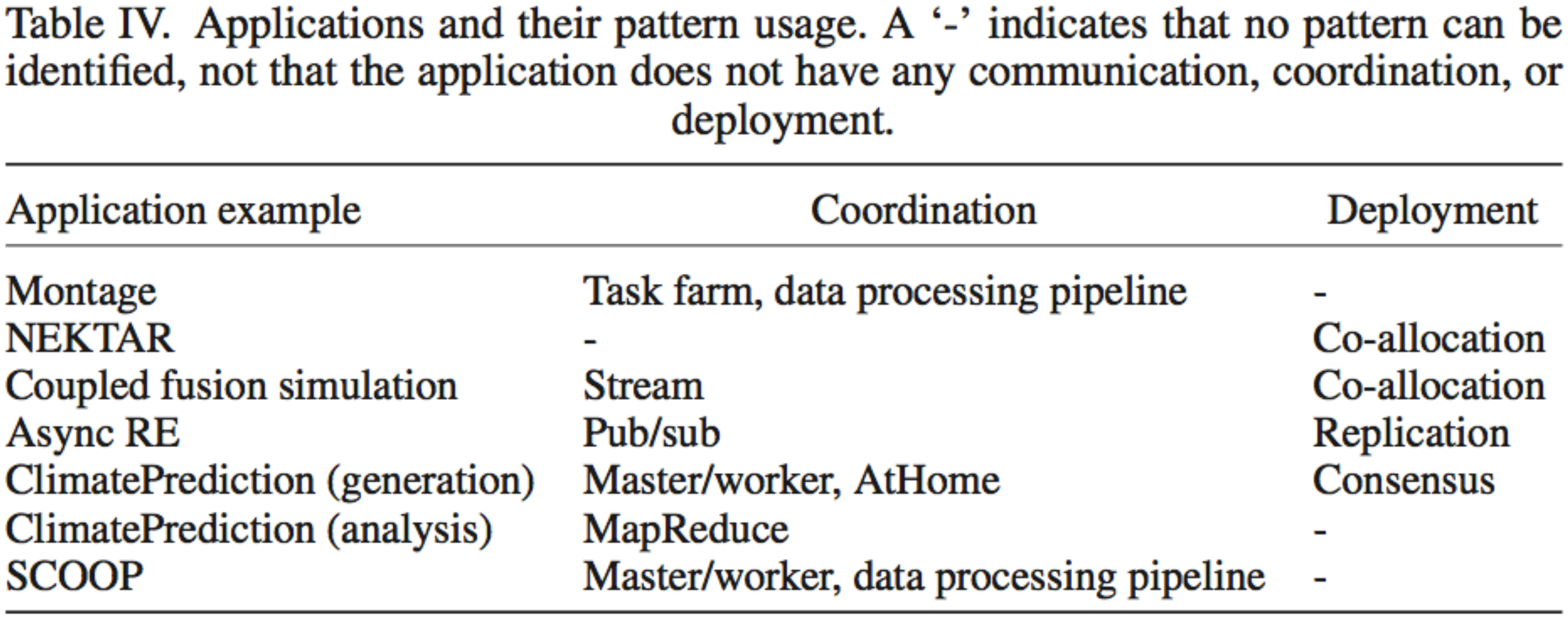


Table IV. Applications and their pattern usage. A '-' indicates that no pattern can be identified, not that the application does not have any communication, coordination, or deployment.

Application example	Coordination	Deployment
Montage	Task farm, data processing pipeline	-
NEKTAR	-	Co-allocation
Coupled fusion simulation	Stream	Co-allocation
Async RE	Pub/sub	Replication
ClimatePrediction (generation)	Master/worker, AtHome	Consensus
ClimatePrediction (analysis)	MapReduce	-
SCOOP	Master/worker, data processing pipeline	-

Distributed Computing MetaPatterns III

Jha et al

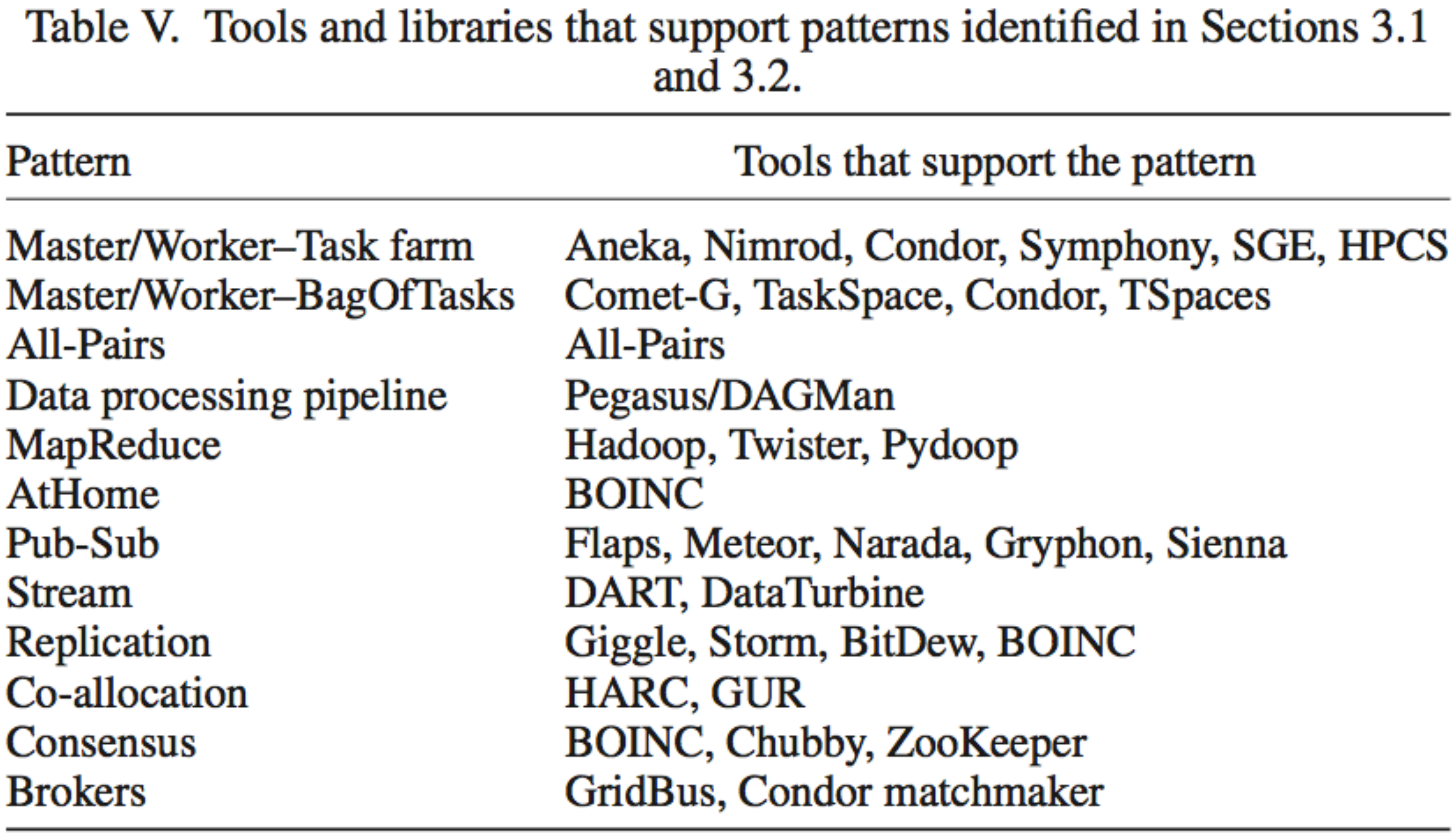


Table V. Tools and libraries that support patterns identified in Sections 3.1 and 3.2.

Pattern	Tools that support the pattern
Master/Worker–Task farm	Aneka, Nimrod, Condor, Symphony, SGE, HPCS
Master/Worker–BagOfTasks	Comet-G, TaskSpace, Condor, TSpaces
All-Pairs	All-Pairs
Data processing pipeline	Pegasus/DAGMan
MapReduce	Hadoop, Twister, Pydoop
AtHome	BOINC
Pub-Sub	Flaps, Meteor, Narada, Gryphon, Sienna
Stream	DART, DataTurbine
Replication	Giggle, Storm, BitDew, BOINC
Co-allocation	HARC, GUR
Consensus	BOINC, Chubby, ZooKeeper
Brokers	GridBus, Condor matchmaker

Comparison of Data Analytics with Simulation -- I

- **Pleasingly parallel** often important in both
- Both are often **SPMD** and **BSP**
- **Non-iterative MapReduce** is major big data paradigm
  - not a common simulation paradigm except where "Reduce" summarizes pleasingly parallel execution
- Big Data often has **large collective communication**
  - Classic simulation has a lot of smallish point-to-point messages
- Simulation dominantly **sparse** (nearest neighbor) data structures
  - "Bag of words (users, rankings, images..)" algorithms are sparse, as is PageRank
  - Important data analytics involves full matrix algorithms

Comparison of Data Analytics with Simulation - II

- There are similarities between some **graph problems and particle simulations** with a **strange cutoff force.**
  - Both **Map-Communication**
- Note many big data problems are “**long range force**” as all points are linked.
  - Easiest to parallelize. Often full matrix algorithms
  - e.g. in DNA sequence studies, distance $\delta(i, j)$ defined by BLAST, Smith-Waterman, etc., between all sequences *i*, *j.*
  - Opportunity for “fast multipole” ideas in big data.
- In image-based **deep learning**, neural network weights are block sparse (corresponding to links to pixel blocks) but can be formulated as full matrix operations on GPUs and MPI in blocks.
- In HPC benchmarking, Linpack being challenged by a new sparse conjugate gradient benchmark HPCG, while we use **non- sparse conjugate gradient solvers** in clustering and Multi-dimensional scaling.

Big Data Ogres and Their “Facets”

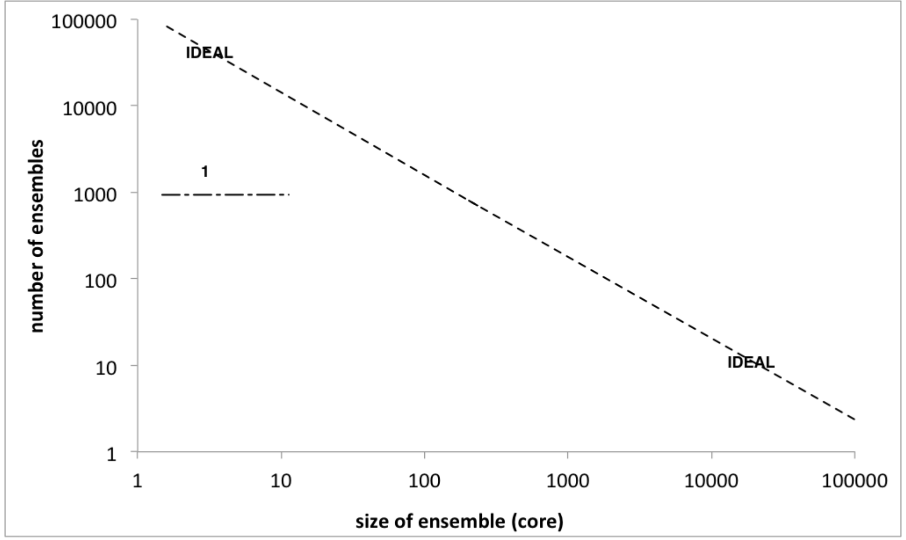
- **The first Ogre Facet captures different problem “architecture”**. Such as (i) **Pleasingly Parallel** – as in Blast, Protein docking, imagery (ii) **Local Machine Learning** – ML or filtering pleasingly parallel as in bio-imagery, radar (iii) **Global Machine Learning** seen in LDA, Clustering etc. with parallel ML over nodes of system (iv) **Fusion:** Knowledge discovery often involves fusion of multiple methods. (v) **Workflow**
- **The second Ogre Facet captures source of data** (i) **SQL**, (ii) **NOSQL** based, (iii) Other Enterprise data systems (10 examples at NIST) (iv)**Set of Files** (as managed in iRODS), (v) **Internet of Things**, (vi) **Streaming** and (vii) **HPC simulations**.
- Before data gets to compute system, there is often an initial data gathering phase which is characterized by a block size and timing. Block size varies from month (Remote Sensing, Seismic) to day (genomic) to seconds (Real time control, streaming)
- There are storage/compute system styles: Dedicated, Permanent, Transient
- Other characteristics are need for permanent auxiliary/comparison datasets and these could be interdisciplinary implying nontrivial data movement/replication

Detailed Structure of Ogres

- **The third Ogre Facet is distinctive system features** such as (i) **Agents**, as in epidemiology (swarm approaches) and (ii) **GIS** (Geographical Information Systems).
- **The fourth Ogre Facet captures Style of Big Data applications.** (i) Are data points in **metric or non-metric spaces** (ii) **Maximum Likelihood**, (iii) χ^2 minimizations, and (iv) **Expectation Maximization** (often Steepest descent)
- **The fifth Facet is Ogres themselves classifying core analytics kernels** (i) Recommender Systems (**Collaborative Filtering**) (ii) **SVM** and Linear Classifiers (Bayes, Random Forests), (iii) **Outlier Detection** (iORCA) (iv) **Clustering** (many methods), (v) **PageRank**, (vi) **LDA** (Latent Dirichlet Allocation), (vii) **PLSI** (Probabilistic Latent Semantic Indexing), (viii) **SVD** (Singular Value Decomposition), (ix) **MDS** (Multidimensional Scaling), (x) **Graph Algorithms** (seen in neural nets, search of RDF Triple stores), (xi) Learning Neural Networks (**Deep Learning**), and (xii) **Global Optimization** (Variational Bayes).
- Flops per byte and Communication Interconnect requirements characterize fifth facet

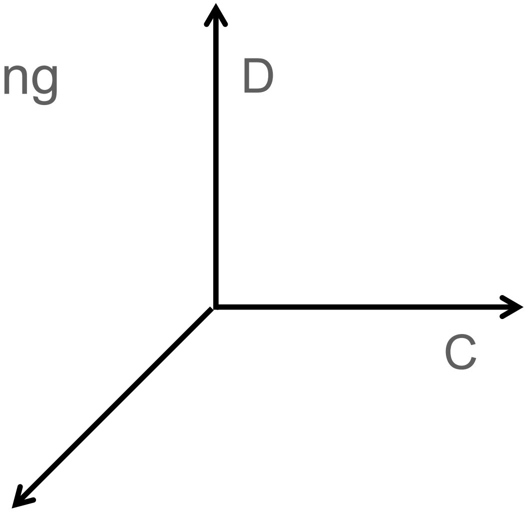
“Many Tasks” Pathway to Extreme Scale

- Problems in computational science *naturally* amenable to “task level” parallelism model of computing:
  - – “Embarrassingly Parallel” data-intensive applications
  - – Many free energy calculations, enhanced sampling problems.
  - – Many multi-physics simulations are also multi components.
- Single “application” might be broken into many smaller simulations
- This is not *just* HTC or HPC, but complex application objectives
  - Isn’t about just peak perf, nor maximal throughput
  - Given access to X cores/ nodes – slice/dice or distribute as needed

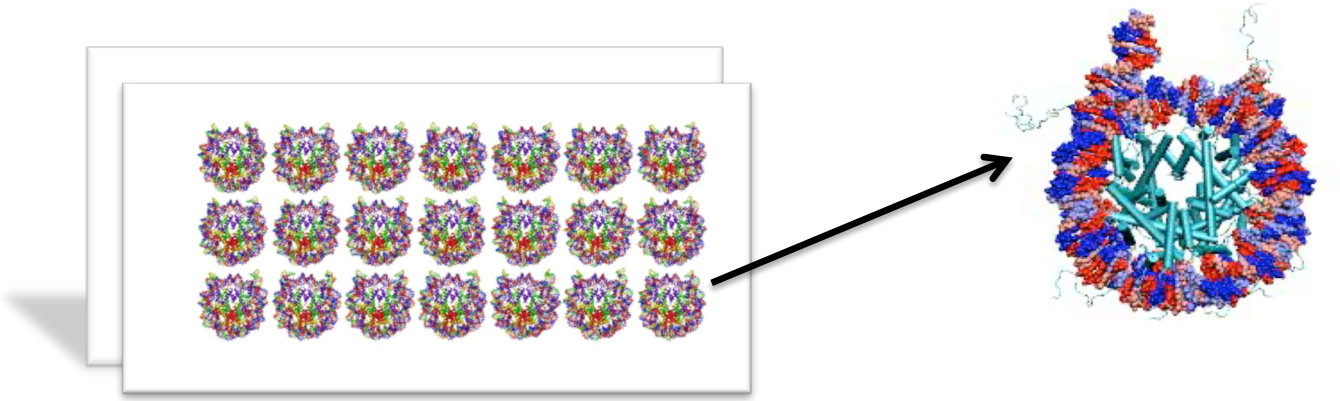


From Many Tasks to Complex Applications

- Starting from uncoupled heterogeneous simulations, varying levels of coordination and dependency can be gradually added and “tuned”
  - Homogeneous/Heterogeneous
    - Complexity of simulation-resources mapping
  - Coupling between simulations
    - Different coordination mechanism
  - Dependencies
    - Constraints, scheduling, data transfer

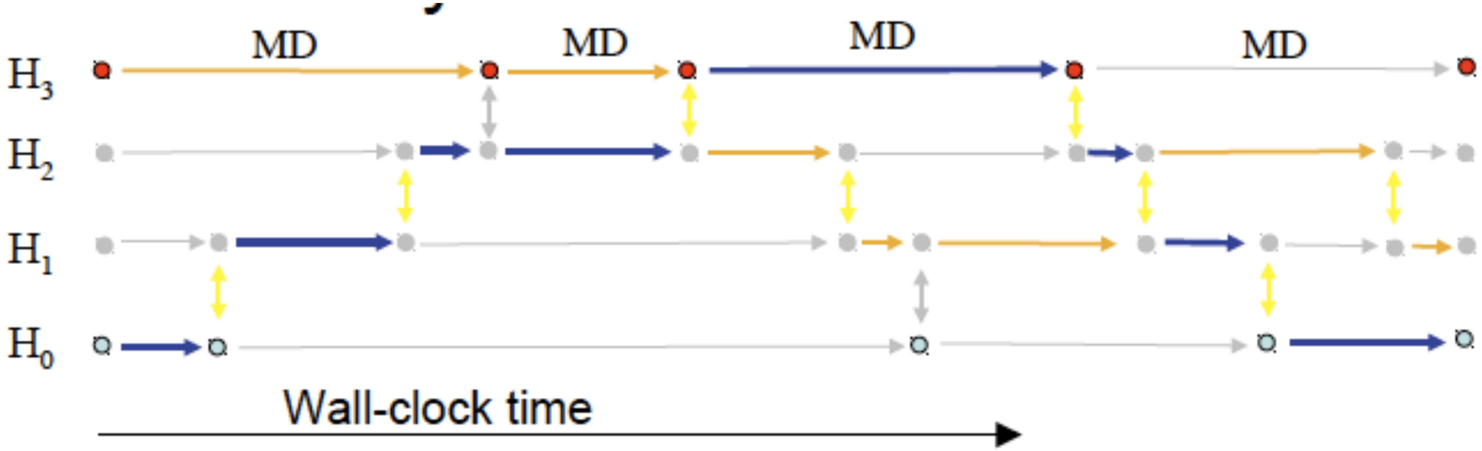


- Depending upon the above properties, the importance and feasibility of distribution varies



Scalable Online Comparative Genomics of Mononucleosomes.

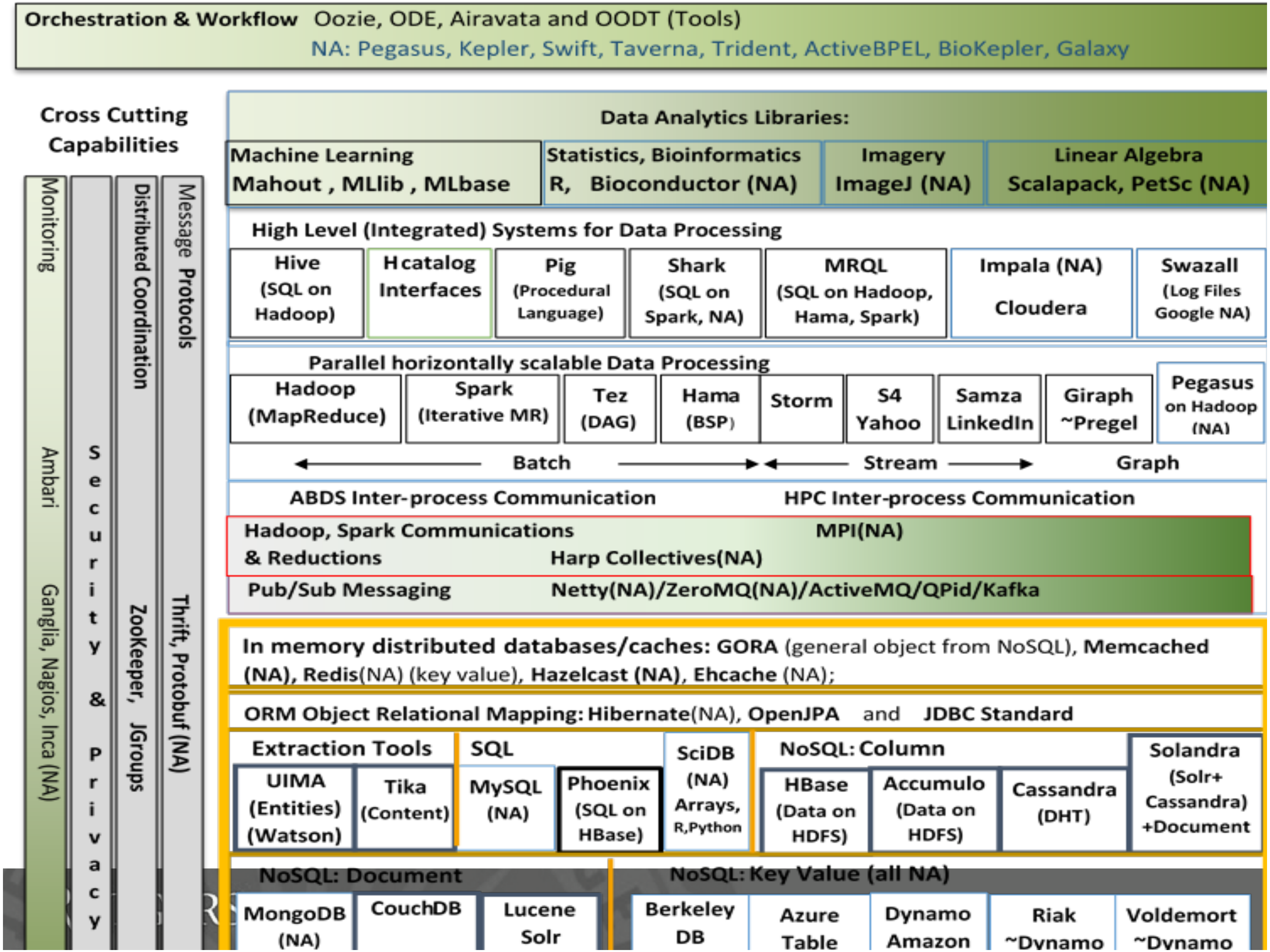
“Scalable Online Comparative Genomics of Mononucleosomes: A BigJob” , *Proceedings of Conference on XSEDE*, 2013.



Asynchronous Replica-Exchange: Advanced Algorithms for enhanced sampling.

“A Framework for Flexible and Scalable Replica-Exchange on Production Distributed CI”, *Proceedings of Conference on XSEDE*, 2013.

The Case for an Integrating Apache/Hadoop Big Data Stack with HPC



Orchestration & Workflow Oozie, ODE, Airavata and OODT (Tools)

NA: Pegasus, Kepler, Swift, Taverna, Trident, ActiveBPEL, BioKepler, Galaxy

Cross Cutting Capabilities

Monitoring: Ambari, Ganglia, Nagios, Inca (NA)

Security & Privacy

Distributed Coordination: ZooKeeper, JGroups

Message Protocols: Thrift, Protobuf (NA)

Data Analytics Libraries:

Machine Learning	Statistics, Bioinformatics	Imagery	Linear Algebra
Mahout , MLlib , MLbase	R, Bioconductor (NA)	ImageJ (NA)	Scalapack, PetSc (NA)

High Level (Integrated) Systems for Data Processing

Hive (SQL on Hadoop)	Hcatalog Interfaces	Pig (Procedural Language)	Shark (SQL on Spark, NA)	MRQL (SQL on Hadoop, Hama, Spark)	Impala (NA) Cloudera	Swazall (Log Files Google NA)

Parallel horizontally scalable Data Processing

Hadoop (MapReduce)	Spark (Iterative MR)	Tez (DAG)	Hama (BSP)	Storm	S4 Yahoo	Samza LinkedIn	Giraph ~Pregel	Pegasus on Hadoop (NA)

← Batch → ← Stream → Graph

ABDS Inter-process Communication **HPC Inter-process Communication**

Hadoop, Spark Communications & Reductions MPI(NA) Harp Collectives(NA)

Pub/Sub Messaging Netty(NA)/ZeroMQ(NA)/ActiveMQ/QPid/Kafka

In memory distributed databases/caches: GORA (general object from NoSQL), **Memcached (NA)**, **Redis**(NA) (key value), **Hazelcast (NA)**, **Ehcache** (NA);

ORM Object Relational Mapping: Hibernate(NA), OpenJPA and JDBC Standard

Extraction Tools: UIMA (Entities) (Watson); Tika (Content)

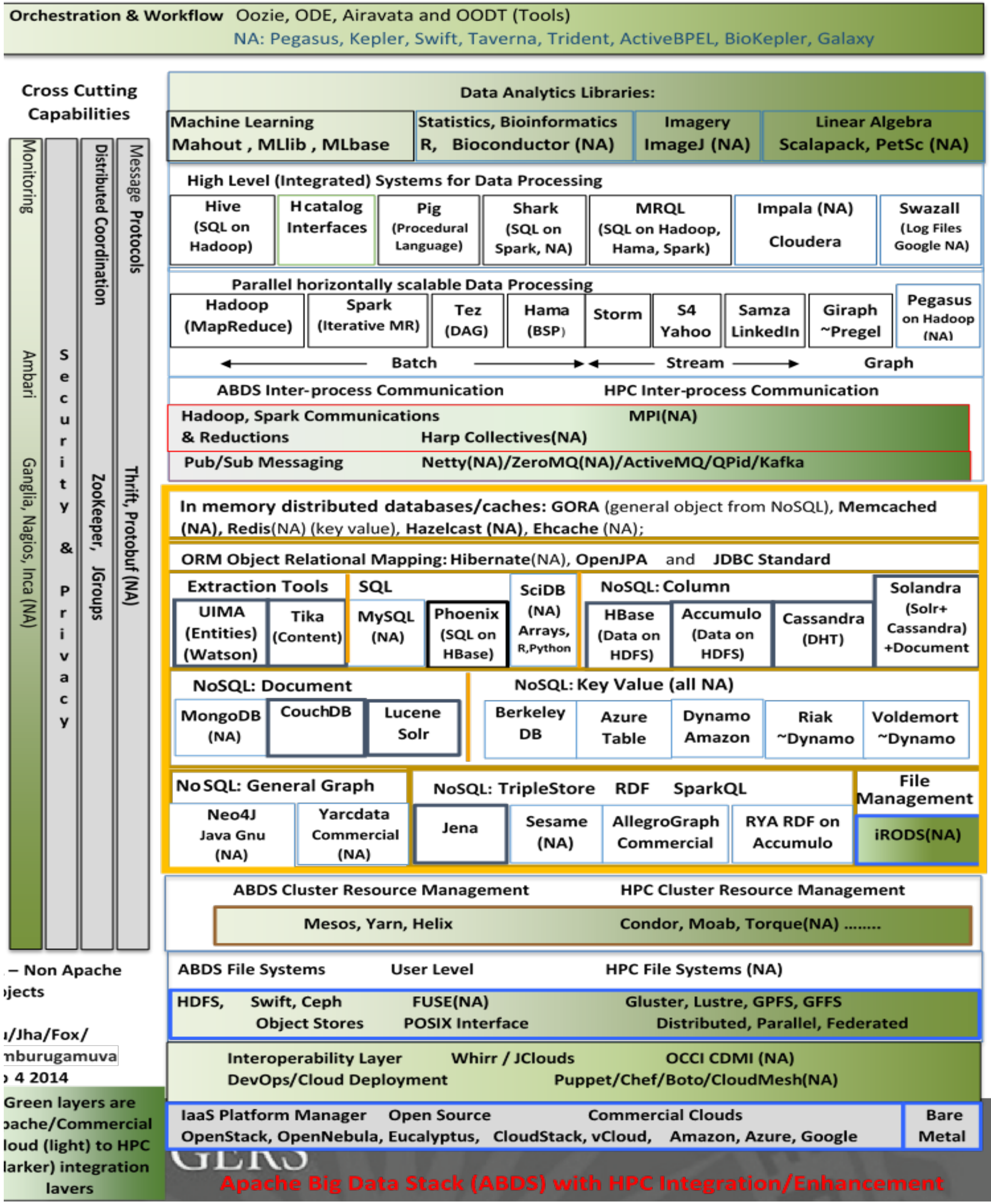
SQL: MySQL (NA); Phoenix (SQL on HBase)

SciDB (NA) Arrays, R,Python

NoSQL: Column: HBase (Data on HDFS); Accumulo (Data on HDFS); Cassandra (DHT); Solandra (Solr+ Cassandra) +Document

NoSQL: Document: MongoDB (NA); CouchDB; Lucene Solr

NoSQL: Key Value (all NA): Berkeley DB; Azure Table; Dynamo Amazon; Riak ~Dynamo; Voldemort ~Dynamo



Orchestration & Workflow Oozie, ODE, Airavata and OODT (Tools)
NA: Pegasus, Kepler, Swift, Taverna, Trident, ActiveBPEL, BioKepler, Galaxy

Cross Cutting Capabilities

Monitoring Ambari Ganglia, Nagios, Inca (NA)

Security & Privacy

Distributed Coordination ZooKeeper, JGroups

Message Protocols Thrift, Protobuf (NA)

Data Analytics Libraries:

Machine Learning
Mahout , MLlib , MLbase

Statistics, Bioinformatics
R, Bioconductor (NA)

Imagery
ImageJ (NA)

Linear Algebra
Scalapack, PetSc (NA)

High Level (Integrated) Systems for Data Processing

Hive (SQL on Hadoop) | Hcatalog Interfaces | Pig (Procedural Language) | Shark (SQL on Spark, NA) | MRQL (SQL on Hadoop, Hama, Spark) | Impala (NA) Cloudera | Swazall (Log Files Google NA)

Parallel horizontally scalable Data Processing

Hadoop (MapReduce) | Spark (Iterative MR) | Tez (DAG) | Hama (BSP) | Storm | S4 Yahoo | Samza LinkedIn | Giraph ~Pregel | Pegasus on Hadoop (NA)

Batch Stream Graph

ABDS Inter-process Communication HPC Inter-process Communication

Hadoop, Spark Communications & Reductions MPI(NA) Harp Collectives(NA)

Pub/Sub Messaging Netty(NA)/ZeroMQ(NA)/ActiveMQ/QPid/Kafka

In memory distributed databases/caches: GORA (general object from NoSQL), Memcached (NA), Redis(NA) (key value), Hazelcast (NA), Ehcache (NA);

ORM Object Relational Mapping: Hibernate(NA), OpenJPA and JDBC Standard

Extraction Tools: UIMA (Entities) (Watson) | Tika (Content)

SQL: MySQL (NA) | Phoenix (SQL on HBase)

SciDB (NA) Arrays, R,Python

NoSQL: Column: HBase (Data on HDFS) | Accumulo (Data on HDFS) | Cassandra (DHT)

Solandra (Solr+ Cassandra) +Document

NoSQL: Document: MongoDB (NA) | CouchDB | Lucene Solr

NoSQL: Key Value (all NA): Berkeley DB | Azure Table | Dynamo Amazon | Riak ~Dynamo | Voldemort ~Dynamo

NoSQL: General Graph: Neo4J Java Gnu (NA) | Yarcdata Commercial (NA)

NoSQL: TripleStore RDF SparkQL: Jena | Sesame (NA) | AllegroGraph Commercial | RYA RDF on Accumulo

File Management: iRODS(NA)

ABDS Cluster Resource Management HPC Cluster Resource Management

Mesos, Yarn, Helix Condor, Moab, Torque(NA) ....

ABDS File Systems User Level HPC File Systems (NA)

HDFS, Swift, Ceph Object Stores FUSE(NA) POSIX Interface Gluster, Lustre, GPFS, GFFS Distributed, Parallel, Federated

Interoperability Layer Whirr / JClouds OCCI CDMI (NA)
DevOps/Cloud Deployment Puppet/Chef/Boto/CloudMesh(NA)

IaaS Platform Manager Open Source Commercial Clouds Bare Metal
OpenStack, OpenNebula, Eucalyptus, CloudStack, vCloud, Amazon, Azure, Google

– Non Apache
jects

/Jha/Fox/
nburugamuva
4 2014

Green layers are
pache/Commercial
loud (light) to HPC
larker) integration
layers

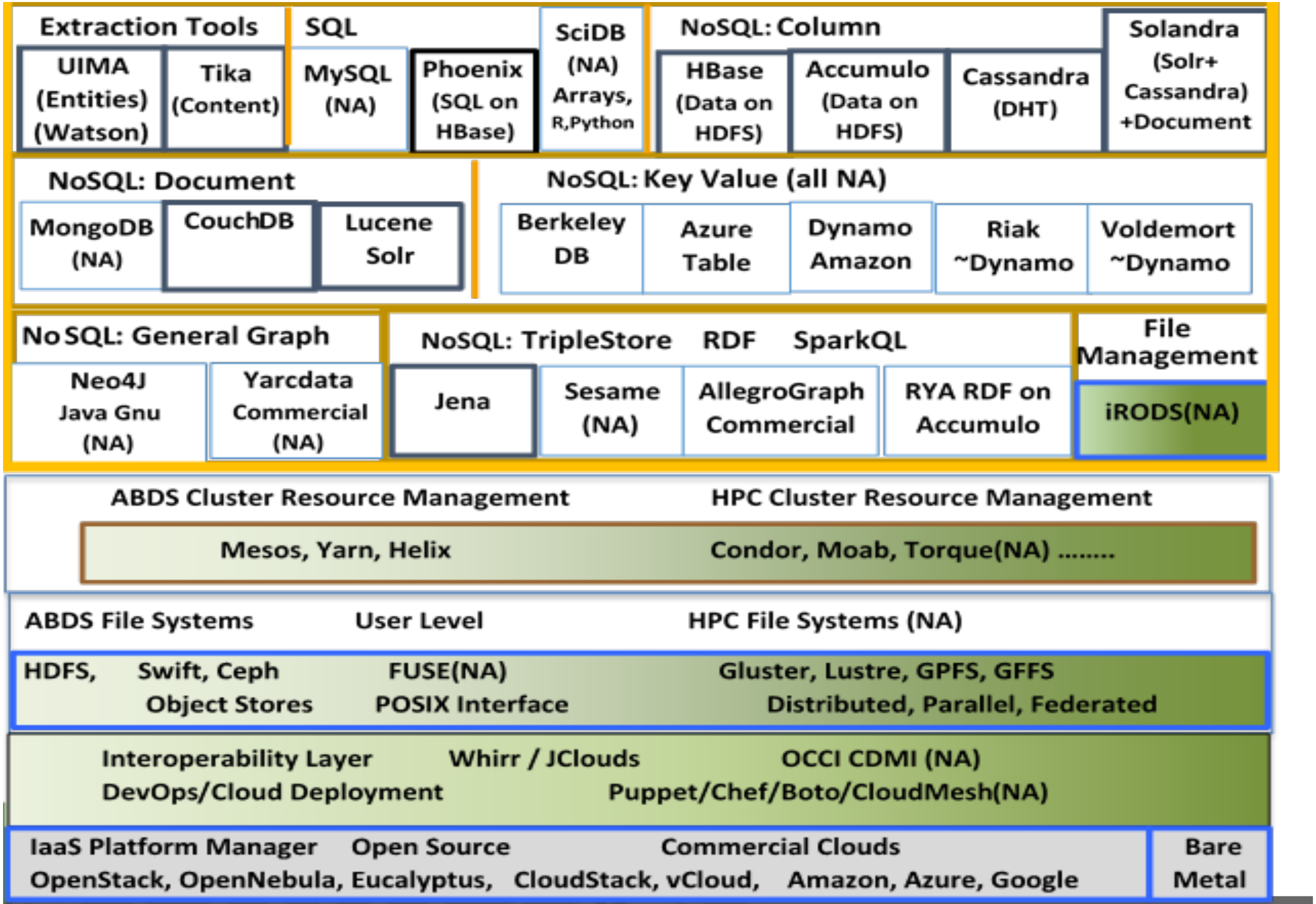
GERS

Apache Big Data Stack (ABDS) with HPC Integration/Enhancement

Enhanced **Apache/Hadoop Big Data Stack** (ABDS)

- ~120 Capabilities
- >40 Apache
- **Green layers have strong HPC Integration opportunities**

- **Goal**
  - **Functionality of ABDS**
  - **Performance of HPC**



Extraction Tools

UIMA (Entities) (Watson)

Tika (Content)

SQL

MySQL (NA)

Phoenix (SQL on HBase)

SciDB (NA) Arrays, R,Python

NoSQL: Column

HBase (Data on HDFS)

Accumulo (Data on HDFS)

Cassandra (DHT)

Solandra (Solr+ Cassandra) +Document

NoSQL: Document

MongoDB (NA)

CouchDB

Lucene Solr

NoSQL: Key Value (all NA)

Berkeley DB

Azure Table

Dynamo Amazon

Riak ~Dynamo

Voldemort ~Dynamo

No SQL: General Graph

Neo4J Java Gnu (NA)

Yarcdata Commercial (NA)

NoSQL: TripleStore RDF SparkQL

Jena

Sesame (NA)

AllegroGraph Commercial

RYA RDF on Accumulo

File Management

iRODS(NA)

ABDS Cluster Resource Management HPC Cluster Resource Management

Mesos, Yarn, Helix Condor, Moab, Torque(NA) ....

ABDS File Systems User Level HPC File Systems (NA)

HDFS, Swift, Ceph Object Stores FUSE(NA) POSIX Interface Gluster, Lustre, GPFS, GFFS Distributed, Parallel, Federated

Interoperability Layer Whirr / JClouds OCCI CDMI (NA)

DevOps/Cloud Deployment Puppet/Chef/Boto/CloudMesh(NA)

IaaS Platform Manager Open Source Commercial Clouds

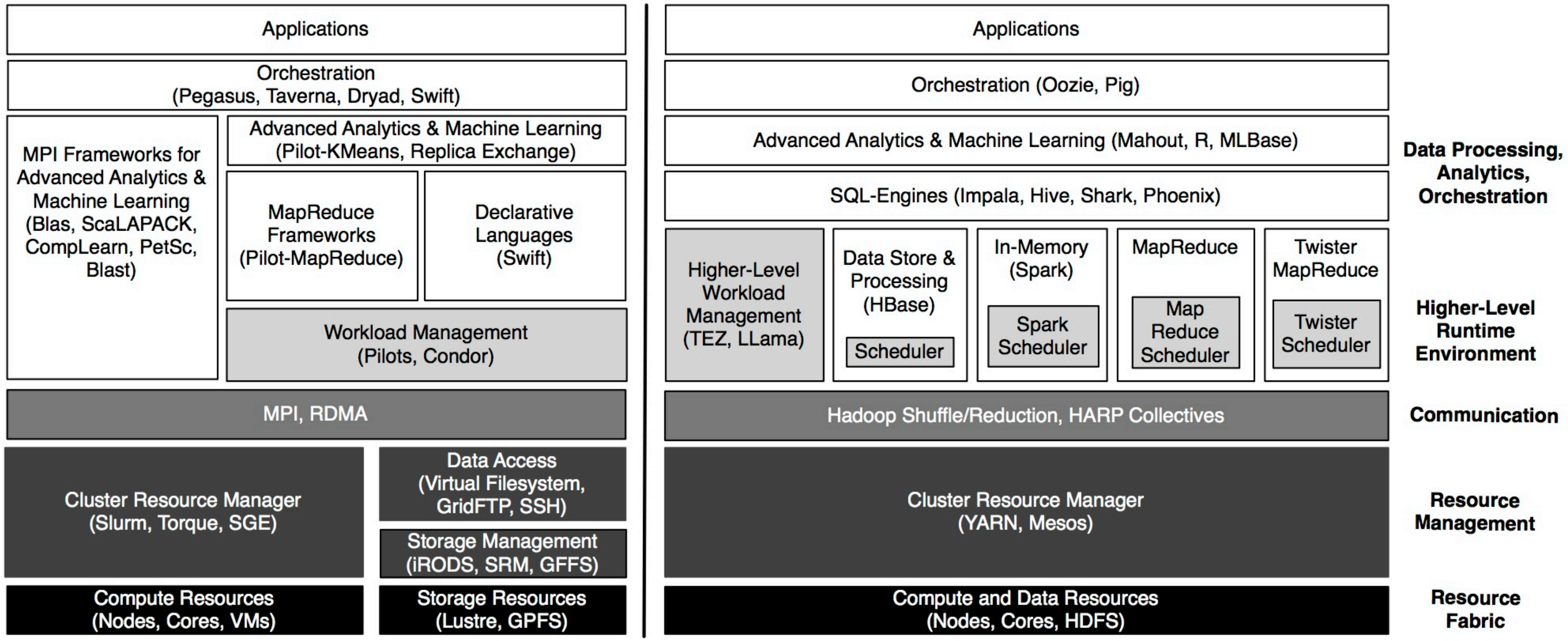
OpenStack, OpenNebula, Eucalyptus, CloudStack, vCloud, Amazon, Azure, Google

Bare Metal

Apache Big Data Stack (ABDS) with HPC Integration/Enhancement

Bringing High Performance to Data Analytics

- On the systems side, we have two principles
  - The Apache Big Data Stack with ~120 projects has important broad functionality with a vital large support organization
  - HPC including MPI has striking success in delivering high performance with however a fragile sustainability model
- There are key systems abstractions which are levels in HPC-ABDS software stack where careful integration needed
  - Resource management
  - Resource Fabric: Storage and Compute
  - Programming model -- horizontal scaling parallelism
  - Collective and Point to Point communication
  - Support of iteration
  - Data interface (not just key-value)



Applications

Orchestration (Pegasus, Taverna, Dryad, Swift)

MPI Frameworks for Advanced Analytics & Machine Learning (Blas, ScaLAPACK, CompLearn, PetSc, Blast)

Advanced Analytics & Machine Learning (Pilot-KMeans, Replica Exchange)

MapReduce Frameworks (Pilot-MapReduce)

Declarative Languages (Swift)

Workload Management (Pilots, Condor)

MPI, RDMA

Cluster Resource Manager (Slurm, Torque, SGE)

Data Access (Virtual Filesystem, GridFTP, SSH)

Storage Management (iRODS, SRM, GFFS)

Compute Resources (Nodes, Cores, VMs)

Storage Resources (Lustre, GPFS)

Applications

Orchestration (Oozie, Pig)

Advanced Analytics & Machine Learning (Mahout, R, MLBase)

SQL-Engines (Impala, Hive, Shark, Phoenix)

Higher-Level Workload Management (TEZ, LLama)

Data Store & Processing (HBase)

Scheduler

In-Memory (Spark)

Spark Scheduler

MapReduce

Map Reduce Scheduler

Twister MapReduce

Twister Scheduler

Hadoop Shuffle/Reduction, HARP Collectives

Cluster Resource Manager (YARN, Mesos)

Compute and Data Resources (Nodes, Cores, HDFS)

Data Processing, Analytics, Orchestration

Higher-Level Runtime Environment

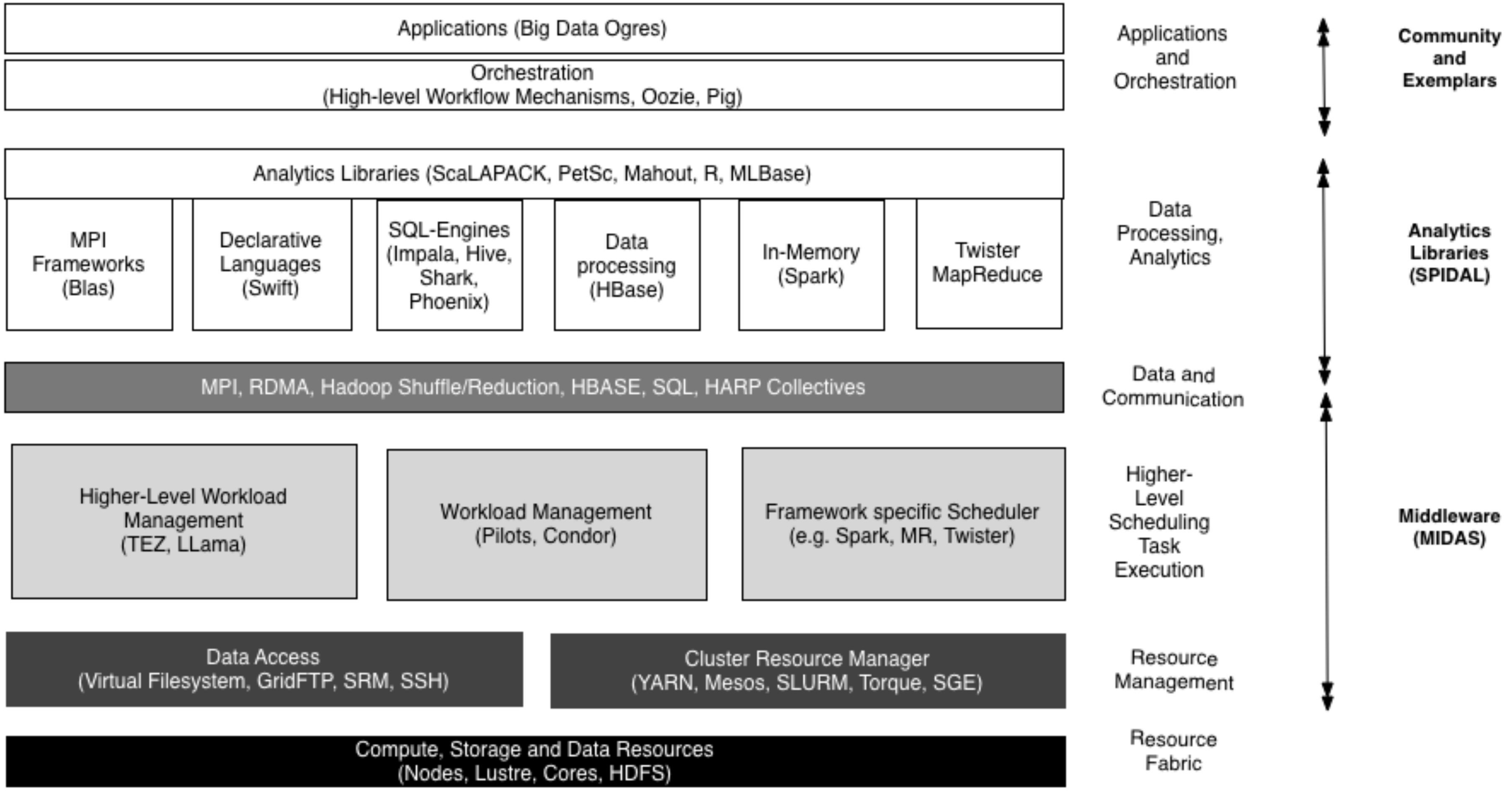
Communication

Resource Management

Resource Fabric

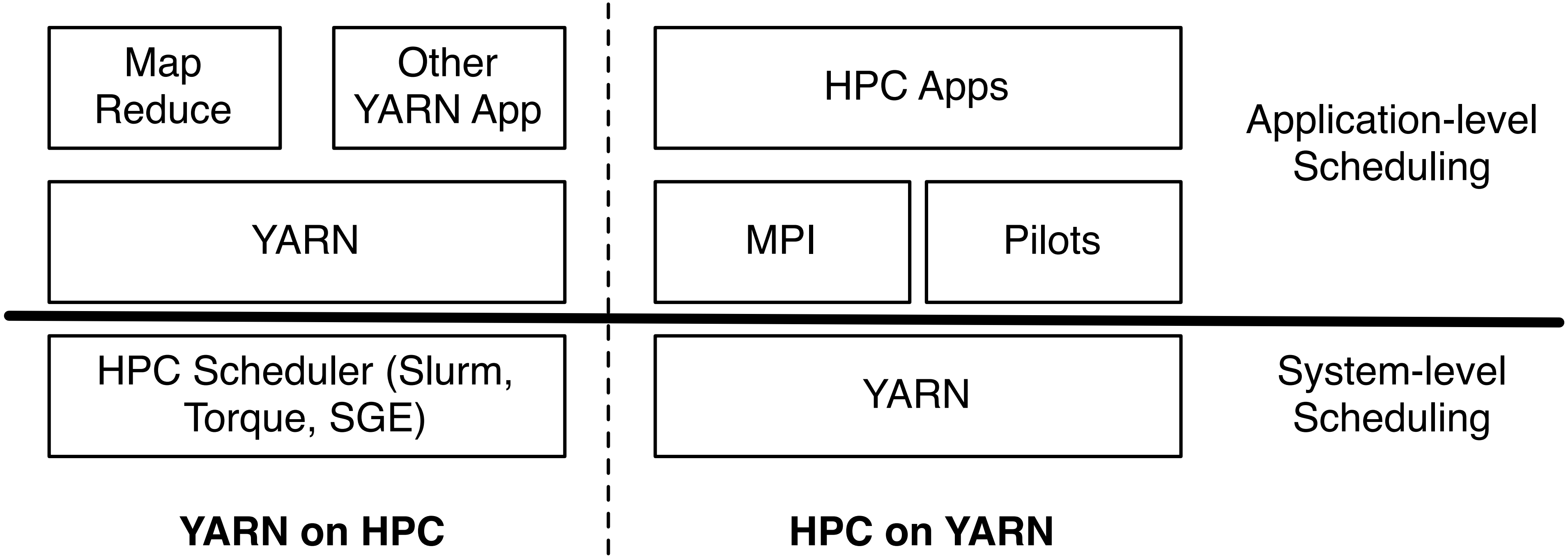
High-Performance Computing

Apache Hadoop Big Data

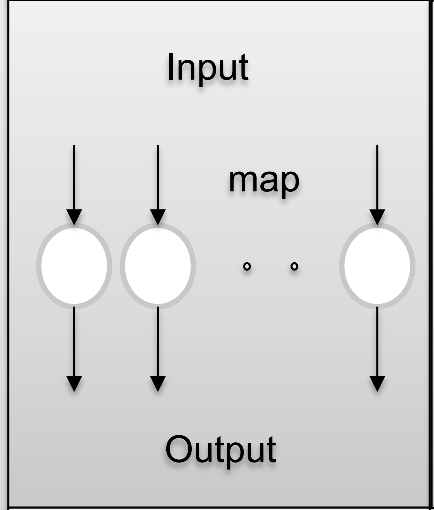
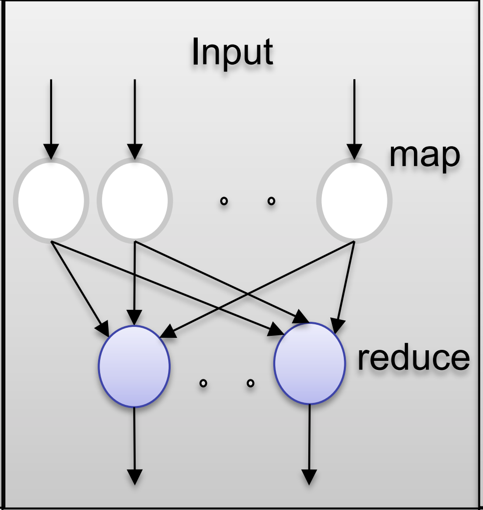
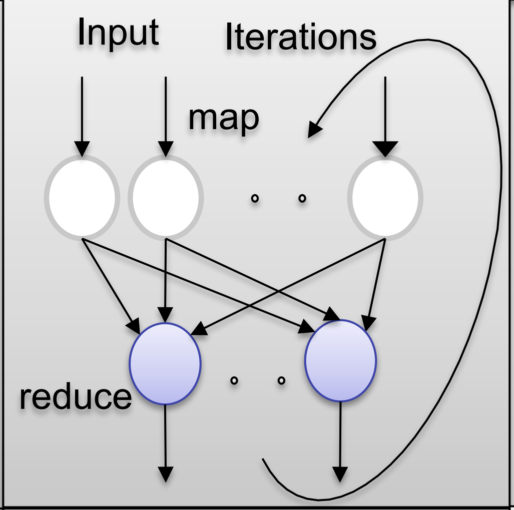
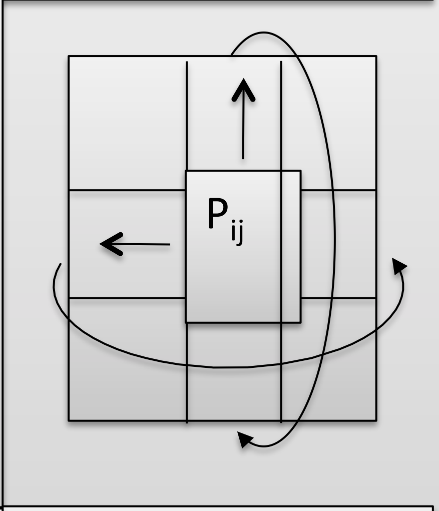

Applications (Big Data Ogres)
Orchestration (High-level Workflow Mechanisms, Oozie, Pig)
Analytics Libraries (ScaLAPACK, PetSc, Mahout, R, MLBase)
MPI Frameworks (Blas)
Declarative Languages (Swift)
SQL-Engines (Impala, Hive, Shark, Phoenix)
Data processing (HBase)
In-Memory (Spark)
Twister MapReduce
MPI, RDMA, Hadoop Shuffle/Reduction, HBASE, SQL, HARP Collectives
Higher-Level Workload Management (TEZ, LLama)
Workload Management (Pilots, Condor)
Framework specific Scheduler (e.g. Spark, MR, Twister)
Data Access (Virtual Filesystem, GridFTP, SRM, SSH)
Cluster Resource Manager (YARN, Mesos, SLURM, Torque, SGE)
Compute, Storage and Data Resources (Nodes, Lustre, Cores, HDFS)
Applications and Orchestration
Data Processing, Analytics
Data and Communication
Higher-Level Scheduling Task Execution
Resource Management
Resource Fabric
Community and Exemplars
Analytics Libraries (SPIDAL)
Middleware (MIDAS)

HPBDS

Integrating Hadoop/Yarn with HPC



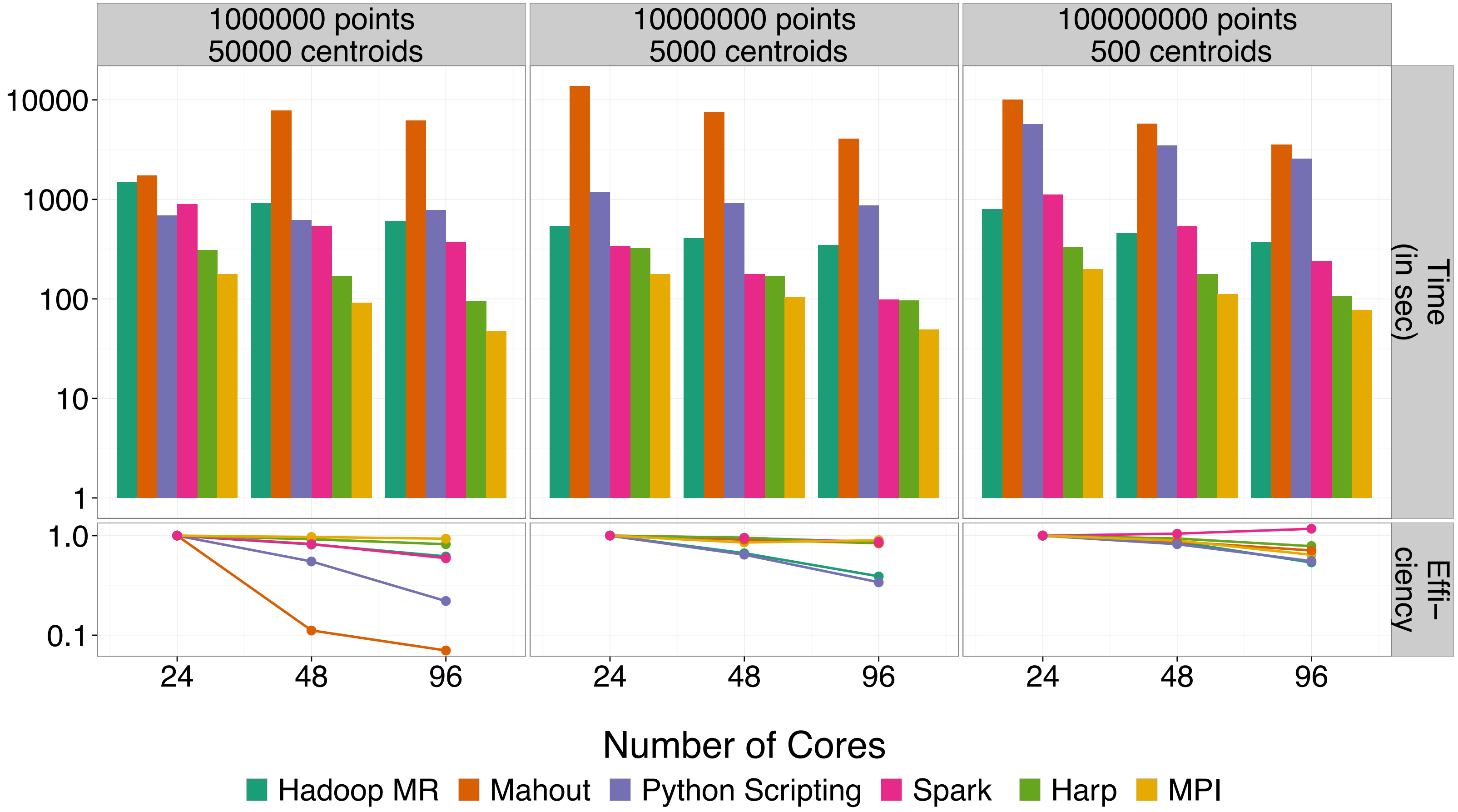
4 Forms of MapReduce

(a) Map Only	(b) Classic MapReduce	(c) Iterative MapReduce	(d) Loosely Synchronous
			
BLAST Analysis Parametric sweep Pleasingly Parallel	High Energy Physics (HEP) Histograms Distributed search	Expectation maximization Clustering e.g. Kmeans Linear Algebra, Page Rank	Classic MPI PDE Solvers and particle dynamics
← Domain of MapReduce and Iterative Extensions → Science Clouds			MPI Giraph

MPI is Map followed by Point to Point or Collective Communication – as in style c) plus d) [slide courtesy Geoffrey Fox]

Increasing Communication

Identical Computation



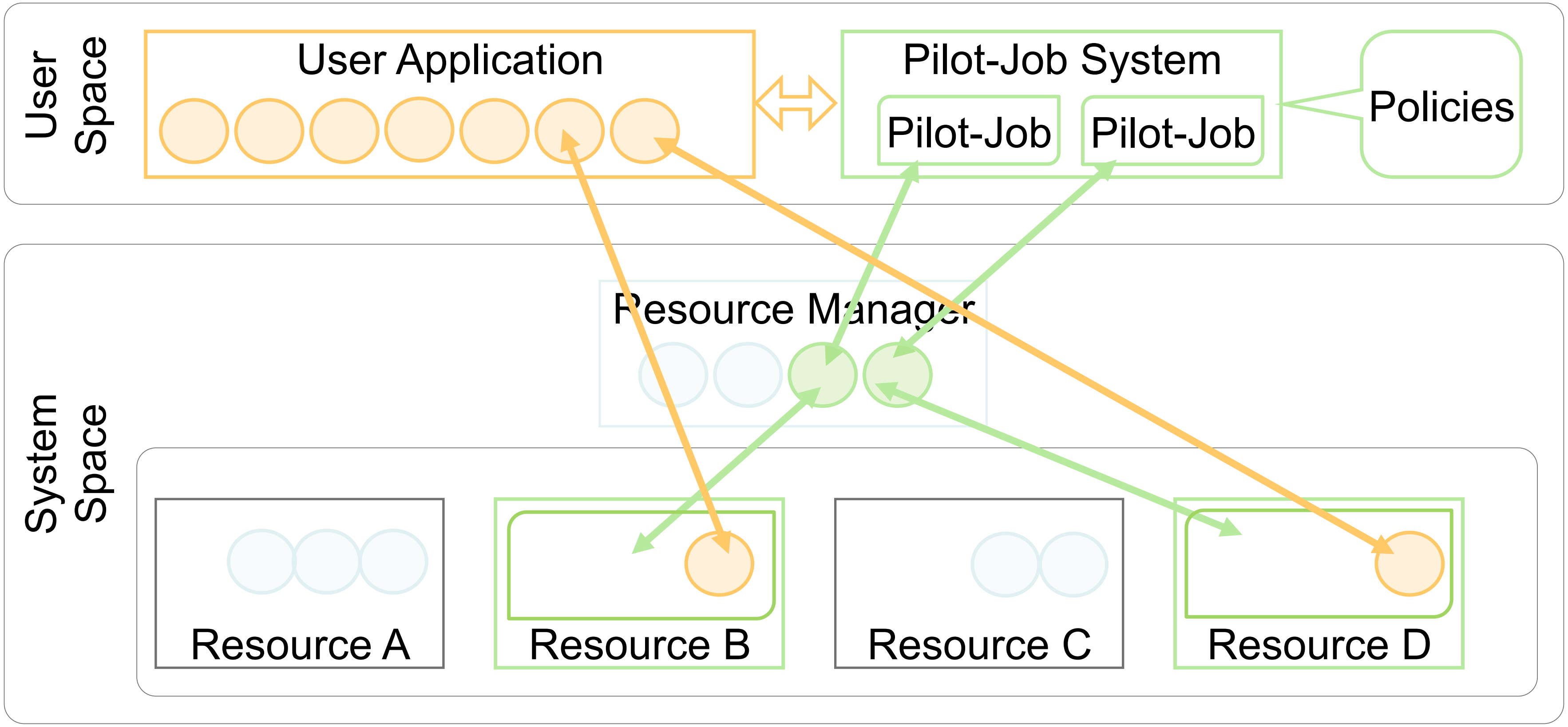
Mahout and Hadoop MR – Slow due to MapReduce
Python slow as Scripting; **MPI** fastest
Spark Iterative MapReduce, non optimal communication
Harp Hadoop plug in with ~MPI collectives

RADICAL-Cybertools: Usage, Usage Modes and Applications

http://radical-cybertools.github.io/

Pilot Abstractions

Working definition: A system that generalizes a placeholder job to provide multi-level scheduling to allow application-level control over the system scheduler via a scheduling overlay.



Introduction to Pilot-Abstraction (2)

- Working definitions:
  - A system that generalizes a placeholder job to provide multi-level scheduling to allow application-level control over the system scheduler via a scheduling overlay
  - *".. defined as an abstraction that generalizes the reoccurring concept of utilizing a placeholder job as a container for a set of compute tasks; an instance of that placeholder job is referred to as Pilot-Job or pilot."*

- Advantages of Pilot-Abstractions:
  - The Perfect Pilot: Decouples workload from resource management
  - Flexible Resource Management
    - Enables the fine-grained (ie "slicing and dicing") of resources
    - Tighter temporal control and other advantages of application-level Scheduling (avoid limitations of system-level only scheduling)
  - Move control, extensibility and flexibility "upwards"
    - Build higher-level capabilities without explicit resource management

Landscape of Pilot-Job Systems

- There are many PJS offerings, often semantically distinct
  - PanDA, DIANE, DIRAC, Condor Glide-In, SWIFT, ToPoS Falkon, BigJob…
    - *Why do you think there has been a proliferation of PJs?*

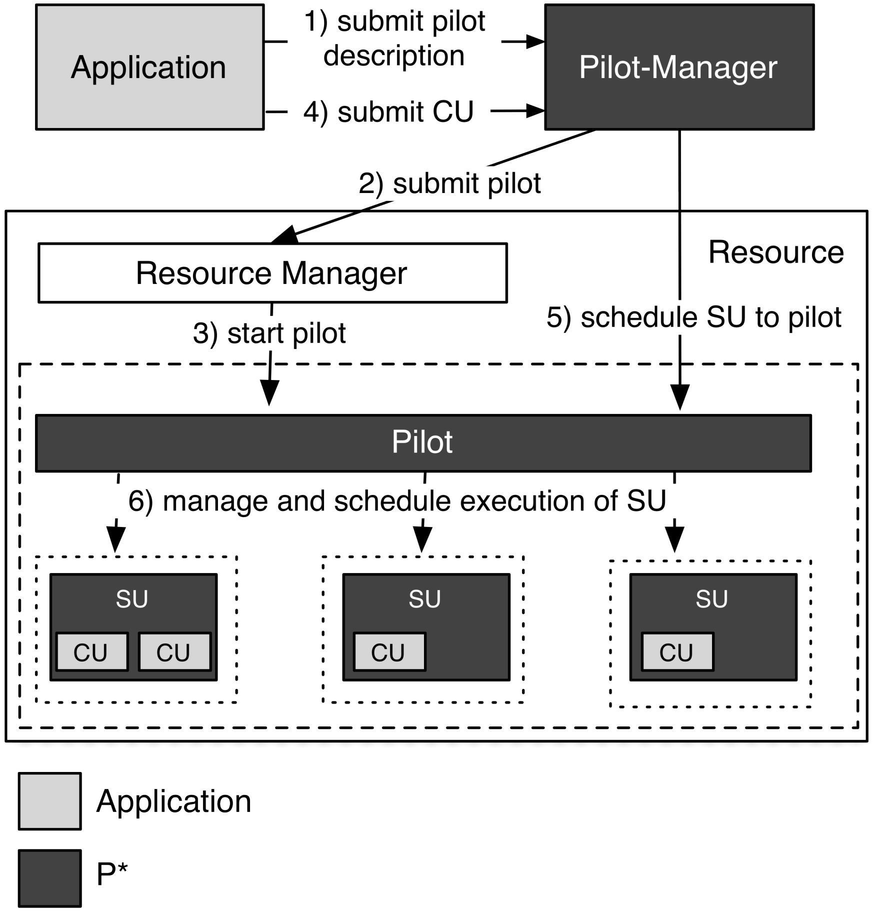
- Conceptual & practical barriers to extensibility (& interoperability)
  - The landscape of PJS reflects, in addition to PJS specifics, the broader eco-system of distributed middleware & infrastructure
  - Software Engineering issues, interfaces, standardization

- Difference in the execution models of the PJ
  - We know “what” pilot-jobs do, but the “how” remains less clear
    - How to map tasks to pilot-jobs? How to choose/map optimal resource?
    - How to “slice and dice” resources?

- Data remains a dependent variable, not a primary variable
  - Introduce the concept of Pilot-data

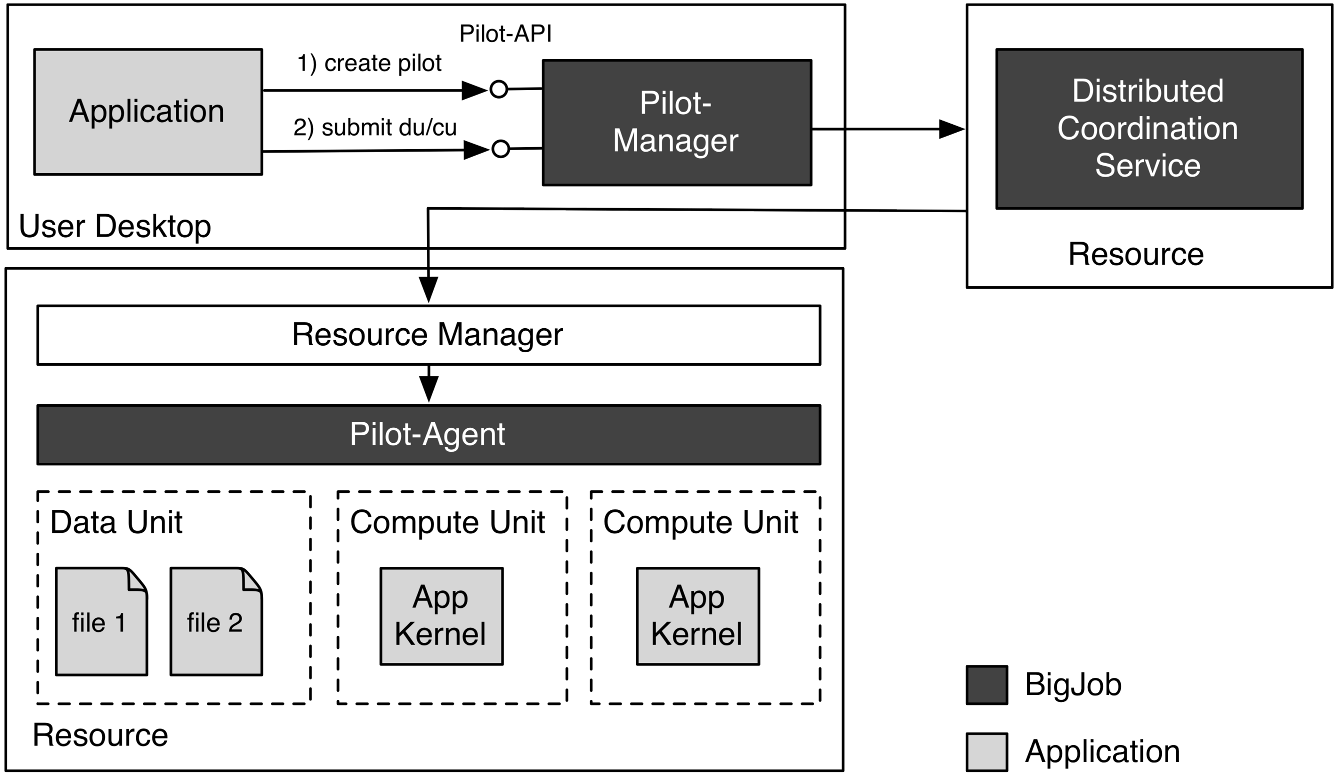
P* Model: Elements, Characteristics and API

- Elements:
  - Pilot-Compute (PC).
  - Pilot-Data (PD).
  - Compute Unit (CU).
  - Data Unit (DU).
  - Scheduling Unit (SU).
  - Pilot-Manager (PM).
- Characteristics:
  - Coordination.
  - Communication.
  - Scheduling.
- Pilot-API.



"P*: A Model of Pilot-Abstractions", *8th IEEE International Conference on e-Science 2012*, 2012

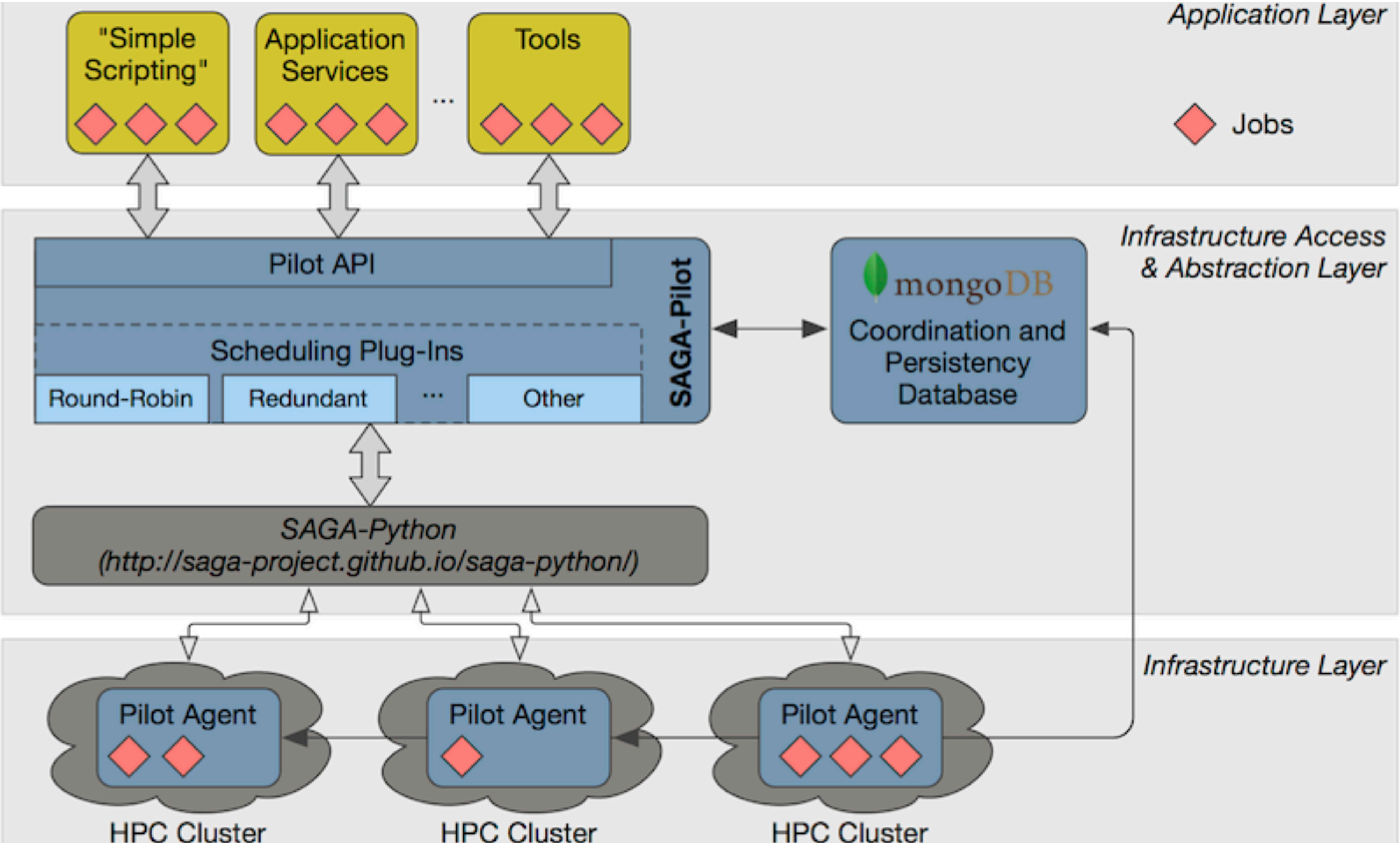
RADICAL-Pilot (BigJob): Architecture



“Coarse-Grained” RADICAL-Pilot Performance

- Number of zero-payload tasks that BJ can dispatch per second:
  - Distributed: O(1)
  - Locally: > O(10)
- Number of Pilots (Pilot-Agents) that can be marshaled
  - Locally/Distributed: O(100)
- Typical number of tasks per Pilot-Agent:
  - Locally/distributed: O(1000)
- Number of tasks concurrently managed = Number of Pilot-Agents x tasks per each agent :
  - O(100) x O(1000)
- (Obviously) The above depends upon data per task:
  - BigJob has been used over O(1)--O(10^9) bytes/task, for tasks of duration O(1) second to O(10^5) seconds

RADICAL-Pilot http://radical-cybertools.github.io/radical-pilot



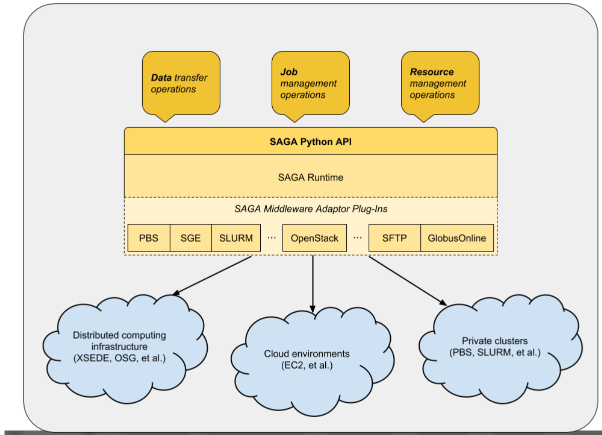
RADICAL-Pilot

http://radical-cybertools.github.io/radical-pilot

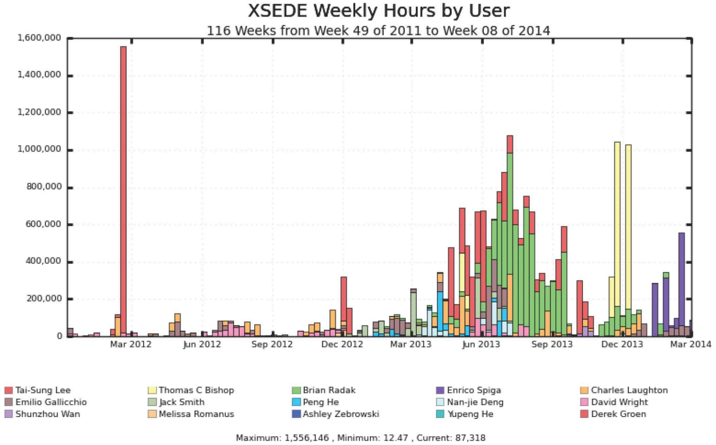
- Lightweight, portable, fast, scalable pilot framework
- Scalability (up and out)
  - Lightweight data model
  - Bulk operations
  - Notifications / support for async programming
- Portability
  - Pure Python
  - Modular pilot agent
  - SAGA-Python as plumbing layer
- Supports Research
  - Pluggable schedulers
  - High degree of introspection, provenance
  - Consistent and verifiable performance

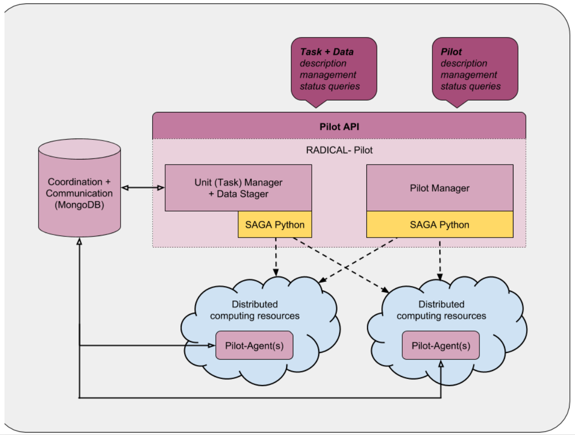
SAGA: A Standardized Interoperability Layer

- SAGA – Simple API for Distributed (“Grid”) Applications:
  - Allows access to different middleware / services through Python implementation of Open Grid Forum GFD.90.
  - Also unified semantics across middleware, backend plug-ins (“adaptors”).
- HOW SAGA is Used?
  - Uniform Access-layer to DCI, e.g, XSEDE,
  - Build tools, middleware services and capabilities
    - Pilot-Jobs, workflow systems, science gateways and web portals
  - Domain-specific (distributed) applications, libraries and frameworks
- Functional component as well as a specific component XSEDE Architecture



Data transfer operations
Job management operations
Resource management operations
SAGA Python API
SAGA Runtime
SAGA Middleware Adaptor Plug-Ins
PBS
SGE
SLURM
...
OpenStack
...
SFTP
GlobusOnline
Distributed computing infrastructure (XSEDE, OSG, et al.)
Cloud environments (EC2, et al.)
Private clusters (PBS, SLURM, et al.)



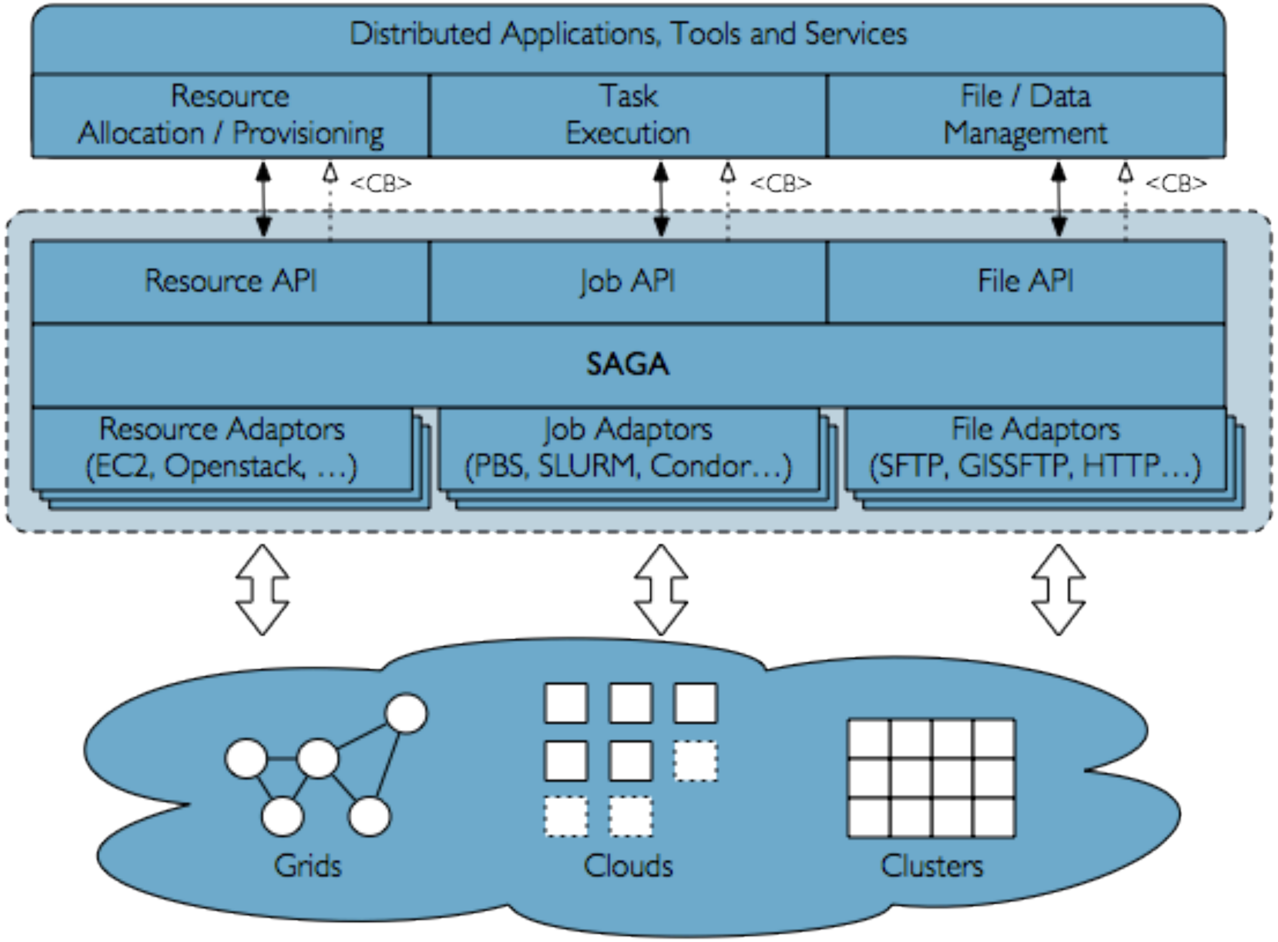


Task + Data description management status queries
Pilot description management status queries
Pilot API
RADICAL- Pilot
Coordination + Communication (MongoDB)
Unit (Task) Manager + Data Stager
Pilot Manager
SAGA Python
SAGA Python
Distributed computing resources
Pilot-Agent(s)
Distributed computing resources
Pilot-Agent(s)

SAGA: Interoperability Layer for RP

- SAGA – Simple API for Distributed (“Grid”) Applications:
  - Application level standardized (Open Grid Forum GFD.90) API.
  - Application is a broad term: “one person’s application is another person’s tool (building block)”.
- SAGA-Python:
  - Native Python implementation of Open Grid Forum GFD.90.
  - Allows access to different middleware / services through a unified interface
  - Provides access via different backend plug-ins (“adaptors”).
  - SAGA-Python provides both a common API, but also unified semantics across heterogeneous middleware:
    - Transparent Remote operations (SSH / GSISSH tunneling).
    - Asynchronous operations.
    - Callbacks.
    - Error Handling.

SAGA Schematic


Distributed Applications, Tools and Services
Resource Allocation / Provisioning
Task Execution
File / Data Management
<CB>
<CB>
<CB>
Resource API
Job API
File API
SAGA
Resource Adaptors (EC2, Openstack, …)
Job Adaptors (PBS, SLURM, Condor…)
File Adaptors (SFTP, GISSFTP, HTTP…)
Grids
Clouds
Clusters

Conclusion

- A fundamental need for abstractions to support diverse set of data-intensive applications
  - *Need for a balanced, interoperable and federated data CI*
- Towards High-performance Data-Intensive Computing
  - Best of both: Hadoop/Apache Big Data stack meets HPC
- Set against these objectives: “Pilot Abstraction Works!”
  - Better integration into HPC
    - Platform independent libraries: different application types, execution modes, coupling schemes
  - Similar “abstractions” emerging in the Hadoop BDS

References

- RADICAL-Cybetools:
  - http://radical-cybertools.github.com/
- RADICAL-SAGA:
  - http://saga-project.github.io/radical-saga
- RADICAL-Pilot: An implementation of P*
  - http://saga-project.github.io/radical-pilot/
- RADICAL:
  - http://radical.rutgers.edu/
- Publications:
  - http://radical.rutgers/edu/publications

Acknowledgements

Graduate Students:

- Mark Santcroos
- Antons Trekalis
- Vivek Balasubramanian

Research Scientists:

- Andre Luckow
- Andre Merzky
- Matteo Turilli
- Ole Weidner

Collaborators

- Geoffrey Fox, Judy Qiu