Towards an Understanding of Facets and Exemplars of Big Data Applications

Geoffrey C. Fox
School of Informatics and Computing
Indiana University, Bloomington
IN 47408, USA

Shantenu Jha
RADICAL, Rutgers University
Piscataway
NJ 08854, USA

Judy Qiu
School of Informatics and Computing
Indiana University, Bloomington
IN 47408, USA

Andre Luckow
RADICAL, Rutgers University
Piscataway
NJ 08854, USA

ABSTRACT

We study many Big Data applications from a variety of research and commercial areas and suggest a set of characteristic features and possible kernel benchmarks that stress those features for data analytics. We draw conclusions for the hardware and software architectures that are suggested by this analysis.

1. INTRODUCTION

With the proliferation of data intensive applications, there is a critical and timely need to understand these properties and the relationship between different applications. The aim of our work is to capture the essential and fundamental Big Data properties, and then to understand applications with those properties.

There are many different types of Big Data applications, and we cover them broadly including both research and commercial cases. However our focus is on Science and Engineering research of data-intensive applications. We compare and contrast some general properties of Big Data applications with classical HPC simulation applications. Pulling together these observations, we identify six key system architectures and note different emphases of commercial and research use cases. Furthermore we point out that combining ideas from HPC and commercial Big Data systems leads to an attractive and powerful Big Data software model.

Section 2 describes the sources of information for our study and their properties. It also details lessons from related studies of parallel computing. Section 3 showcases the features of Big Data use cases and the facets into which we group them, and introduce Ogres to designate broad groupings of applications that exhibit facets. We describe some generic kernels (mini-applications), or instances of Ogres, in the data analytics area. In section 4, we present implications for needed hardware and software while conclusions are in section 5.

2. SOURCES OF INFORMATION

2.1 Data Intensive Use Cases

In discussing the structure of Big Data applications, let us first examine the inevitably incomplete input that we used to do our analysis. We have gained quite a bit of experience from our research over the years, but 3 explicit sources that we used were a recent use case survey by NIST from Fall 2013 [1]; a survey of data intensive research applications by Jha et al. [2, 3]; and a study of members of data analytics libraries including R [4], Mahout [5] and MLLib [6]. Following is a summary of the first two sources.

The NIST Big Data Public Working Group (NBD-PWG) was launched in June 2013 with a set of working groups covering Big Data Definitions, Taxonomies, Requirements, Security and Privacy Requirements, Reference Architectures White Paper Survey, Reference Architectures, Security and Privacy Reference Architectures and Big Data Technology Roadmap. The Requirements working group gathered 51 use cases from a public call and then analyzed them in terms of requirements of a reference architecture [7]. Here we will strive to identify common patterns and characteristics, which can be used to guide and evaluate Big Data hardware and software. The 51 use cases are organized into nine broad areas with the number of associated use cases in parentheses: Government Operation (4), Commercial (8), Defense (3), Healthcare and Life Sciences (10), Deep Learning and Social Media (6), The Ecosystem for Research (4), Astronomy and Physics (5); Earth, Environmental and Polar Science (10) and Energy (1).

Note that the majority of use cases come from research applications, although commercial, defense and government operations have some coverage. A template was prepared by the requirements working group, which allowed experts to categorize each use case by 26 features:

Use case Actors/Stakeholders and their roles and responsibilities; use case goals and description. Specification of current analysis covering compute system, storage, networking and software. Characteristics of use case Big Data with Data Source (distributed/centralized), Volume (size), Velocity (e.g. real time), Variety (multiple datasets, mashup), Variability (rate of change). The so-called Big Data Science (collection, curation, analysis) with Veracity (Robustness Issues, semantics), Visualization, Data Quality (syntax), Data Types and Data Analytics. These detailed specifications were complemented by broad comments including Big Data Specific Challenges (Gaps), Mobility issues, Security and Privacy Requirements and identification of issues for generalizing this use case.

The complete set of 51 responses with in addition a summary from the working group of applications, current status and futures as well as extracted requirements can be found in [7]. They are summarized in the Appendix which also gives 20 other use cases coming from the NBD-PWG which do not have the detailed 26 feature template recorded. These 20 cover enterprise data applications and security and privacy.

The impressive NRC report [8] is a rich source of information. It has several examples in chapter 2; most of these are also present in the NIST study, but NRC does have an interesting discussion of Big Data in Networking and Telecommunication that is omitted from NIST compilation. We will return to the important "Giants" in chapter 10, which are related to different facets of our Ogres.

For the case of distributed applications, there are at least two existing attempts to survey and analyze applications. In Jha et al. [3], the authors examine at a high-level approximately 20 distinct scientific applications that have either been distributed by design or were distributed "by nature". They reduce the number of applications carefully examined to six representative applications. These range from the ubiquitous "@home" class of distributed applications, to Montage - an image reconstruction application emblematic of loosely coupled workflows and to highly specialized and performance oriented applications such as NEKTAR.

Building upon [3], Jha et al. [2] seek to understand distributed, dynamic and data-intensive applications (D3 Science) investigating the programming models and abstractions, the run-time and middleware services, and the computational infrastructure. The survey includes the following applications: NGS Analytics, CMB, Fusion, Industrial Incident Notification and Response, MODIS Data Processing, Distributed Network Intrusion Detection, ATLAS/WLCG, LSST, SOA Astronomy, Sensor Network Application, Climate, Interactive Exploration of Environmental Data, and Power Grids.

2.2 Lessons from Parallel Computing

Before we get to discussing features and patterns of Big Data applications, it is instructive to consider the better understood parallel computing situation. Here the application

Table 1: What is Parallelism Over for NIST Use Cases?

General Class	Examples
People	Users (but see below) or Subjects of application and often both
Decision makers	Researchers or doctors (users of application)
Items	Experimental observations Contents of online store Images or Electronic Information nuggets EMR: Electronic Medical Records (often similar to people parallelism) Protein or Gene Sequences Material properties, Manufactured Object specifications, etc., in custom dataset
Modeled entities	Vehicles and people
Sensors	Internet of Things
Events	Detected anomalies in telescope, credit card or atmospheric data
Graph Nodes	RDF databases
Regular Nodes	Simple nodes as in a learning network
Information Units	Tweets, Blogs, Documents, Web Pages, etc. and characters/words in them
Files or data	To be backed up, moved or assigned metadata
Particles/cells/ mesh points	Used in parallel simulations

requirements have been captured in many ways:

A. **Benchmark Sets:** These vary from full applications [9] to kernels or mini-applications such as the NAS Parallel Benchmarks [9, 10] or Parkbench [11], with the Top500 [12] pacing application Linpack (HPL) being particularly well-known [13]. The new sparse HPCG conjugate gradient benchmark is worthy of mention [13]. Note benchmarks can be specified via explicit code and/or by a "pencil and paper specification" that can be optimized in any way for a particular platform.

B. **Patterns or Templates:** These can be similar to benchmarks but have different goals, such as providing a generic framework that can be modified by users with details of their application as in Template book [14, 15]. Alternatively they can be aimed at illustrating different applications as in the original Berkeley Dwarfs [16].

In this paper, our approach adheres closest to the Dwarfs framework; this is one motivation for choosing to name it the Big Data 'Ogres'. In looking at this previous work, we note that benchmarks often cover a variety of different application aspects and are accompanied by principles or folklore that can guide the writing of parallel code or designing suitable hardware and software. For example, data locality and cost of data movement, sparseness, Amdahl's law, communication latency, bisection bandwidth and scaled speedup are associated with substantial folklore.

The famous NAS Parallel Benchmarks (NPB) [17] consist of: *MG: Multigrid, CG: Conjugate Gradient, FT: Fast Fourier Transform, IS: Integer sort, EP: Embarrassingly Parallel,* *BT: Block Tridiagonal, SP: Scalar Pentadiagonal, and LU: Lower-Upper symmetric Gauss Seidel.* All these are fairly uniform. With the exception of EP, which is an application class, the other members are typical constituents of a low level library for parallel simulations. On the other hand, the Berkeley Dwarfs are 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 and Finite State Machines. The Dwarfs are not exact kernels, but instead describe problems from different points of view, including programming model (MapReduce), numerical method (Grids, Spectral method), kernel structure (dense or sparse linear algebra), algorithm (dynamic programming) and application class (N-body), etc. We believe it is generally accepted that both parallel computing and Big Data cannot be characterized with a single criterion, and so we introduce multiple Ogres exhibiting a set of facets in four different directions. We anticipate that there will be a correlation between the values of specific facet and application type and the needed computing architecture to support them.

2.3 Properties of the 51 NIST Use Cases

Tables 1 to 3 summarize characteristics of the 51 use cases, which we will combine with other input for the Ogres. Note that Big Data and parallel programming are intrinsically linked, as any Big Data analysis is inevitably processed in parallel. Parallel computing is almost always implemented by dividing the data between processors (data decomposition); the richness here is illustrated in Table 1, which lists the members of space that are decomposed for different use cases. Of course these sources of parallelism are broadly applicable outside the 51 use cases from which they were extracted. In Table 2, we identify 15 use case features that will be used later as facets of the Ogres. The second column of Table 2 lists our estimate of the number of use cases that illustrate this feature; note these are not exclusive, so any one use case will illustrate many features.

It is important to note that while machine learning is commonly used, there is an interesting distinction between what are termed Local Machine Learning (LML) and Global Machine Learning (GML) in Table 2. In LML, there is parallelism over items of Table 1 and machine learning is applied separately to each item; needed machine learning parallelism is limited, typified by the use of accelerators (GPU). In GML, the machine learning is applied over the full dataset with MapReduce, MPI or an equivalent. Typically GML comes from maximum likelihood or with a sum over the data items — documents, sequences, items to be sold, images, etc., and often links (point-pairs). Usually GML is a sum of positive numbers, as in least squares, and is illustrated by algorithms like PageRank, clustering/community detection, mixture models, topic determination, Multidimensional scaling, and (Deep) Learning Networks. Somewhat quixotically, GML can be termed Exascale Global Optimization or EGO.

The difference between LML and GML is illustrated in Table 3, which contrasts 9 of the 51 NIST use cases that involve image-based data. For example, use case 18 with light source data is largely independent machine learning on each image from the source, i.e. LML. In contrast deep learning in use case 26 is constructing a learning network integrating all the images.

Table 2: Some Features of NIST Use Cases

Abbrev.	#	Description
PP	26	Pleasingly Parallel or Map Only
MR	18	Classic MapReduce MR (add MRStat below for full count)
MRStat	7	Simple version of MR where key computations are simple reduction as found in statistical averages such as histograms and averages
MRIter	23	Iterative MapReduce or MPI
Graph	9	Complex graph data structure needed in analysis
Fusion	11	Integrate diverse data to aid discovery/decision making; could involve sophisticated algorithms or could just be a portal
Streaming	41	Some data comes in incrementally and is processed this way
Classify	30	**Classification:**divide data into categories
S/Q	12	**Index**, Search and Query
CF	4	**Collaborative Filtering** for recommender engines
LML	36	**Local Machine Learning** (Independent for each parallel entity)
GML	23	**Global Machine Learning:** Deep Learning, Clustering, LDA, PLSI, MDS, Large Scale Optimizations as in Variational Bayes, MCMC, Lifted Belief Propagation, Stochastic Gradient Descent, L-BFGS, Levenberg-Marquardt . Can call EGO or Exascale Global Optimization with scalable parallel algorithm
	51	**Workflow**: Universal so no label
GIS	16	**Geotagged data** and often displayed in ESRI, Microsoft Virtual Earth, Google Earth, GeoServer etc.
HPC	5	Classic **large-scale simulation** of cosmos, materials, etc. generating (visualization) data
Agent	2	Simulations of models of data-defined macroscopic entities represented as **agents**

2.4 Properties of Distributed Use Cases

In the process of reduction and classification, the authors of [2, 3] analyze the structure of applications and find commonalities; they introduce the term "vectors" to capture four essentially orthogonal but critical properties that determine both the development and the execution of the application. These vectors are: execution unit, communication, coordination, and an execution environment. The first three are internal properties of a distributed application, whereas the latter is essentially an external property. Based upon recurring values of vectors, the authors propose a set of common patterns that help elucidate the structure of the distributed applications. It is worth noting, that vectors and patterns for distributed applications do not provide insight into performance aspects of the applications. Table 4 shows seven example applications and their properties.

In [2], the authors propose a framework for describing applications, distributed and dynamic data and infrastructure. Figure 1 shows the data lifecycle model used for the analysis capturing both applications using sensors and computation-

Table 3: 9 Image-based NIST Use Cases

Use Case	Title	Application	Features
17	Pathology Imaging/ Digital Pathology	Moving to terabyte size 3D images, Global Classification	PP, LML, MR for search
18	Light sources	Biology and Materials	PP, LML
26	Large-scale Deep Learning	Stanford ran 10 million images and 11 billion parameters on a 64 GPU HPC; vision (drive car), speech, and Natural Language Processing	GML
27	Organizing large-scale, unstructured collections of photos	Fit position and camera direction to assemble 3D photo ensemble	GML
36	Catalina Real-Time Transient Synoptic Sky Survey (CRTS)	Processing of individual images for events based on classification of image structure (GML)	PP, LML, GML
43	Radar Data Analysis for CReSIS Remote Sensing of Ice Sheets	Identify glacier beds and snow layers See GML when one addresses full ice sheet	PP, LML moving to GML
44	UAVSAR Data Processing	Find and display slippage from radar images. Includes Data Product Delivery, and Data Services	PP
45, 46	Analysis of Simulation visualizations	Find paths, classify orbits, classify patterns that signal earthquakes, instabilities, climate, turbulence	PP, LML, GML

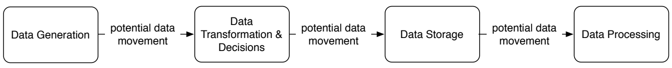

Figure 1: Application Stages

Table 4: Distributed Applications Characteristics

Application	Execution Unit	Communication	Coordination	Execution Environment
Montage	Multiple sequential and parallel executable	Files	Dataflow (DAG)	Dynamic process creation, execution
NEKTAR	Multiple concurrent parallel executables	Stream based	Dataflow	Co-scheduling, data streaming, async. I/O
Replica-Exchange	Multiple seq. and parallel executables	Pub/sub	Dataflow and events	Decoupled coordination and messaging
Climate Prediction (analysis)	Multiple seq. and parallel executables	Files and messages	Dataflow	Dynamics process creation, workflow execution
SCOOP	Multiple Executable	Files and messages	Dataflow	Preemptive scheduling, reservations
Coupled Fusion	Multiple executable	Stream-based	Dataflow	Co-scheduling, data streaming, async I/O

Table 5: Computational Giants of Massive Data Analysis [8]

G1	Basic Statistics
G2	Generalized N-Body Problems
G3	Graph-Theoretic Computations
G4	Linear Algebraic Computations
G5	Optimizations
G6	Integration
G7	Alignment Problems

ally generated data.

The authors call out the Big Data aspects, the dynamic aspects and the distributed aspects of a large set of applications, and introduce quantitative estimates for various performance related properties.

Table 4 (from [3]) shows the specific values of the "DPA vectors" for the set of six distinct applications investigated. It is interesting to note that the categorization did not lead to a well-defined and non-overlapping classification of applications, as the complexity of considering the end-to-end aspects and the diverse ways in which applications are utilized resulted in classes that had overlapping characteristics.

3. THE BIG DATA OGRES AND THEIR FOUR FACETS

Synthesizing lessons learned from HPC, distributed applications and the NIST use case given above, we argue that there is a need to construct classes of mini-applications that facilitate the understanding and characterization of the Big Data properties of these applications. We further introduce facets or features in 4 classification dimensions or views to categorize Big Data applications. These are the Problem architecture, Execution features, Data Source or Style, and Processing views. There are of course other ways of looking at the Ogres and our work should be treated as an initial suggestion for further discussion. These views and their facets build on earlier discussions, especially Table 2. Note that a given application can be made up of components with different facets in Ogre classification. We will reference the 7 computational giants G1-G7 from the NRC report recorded in Table 5. These are important Big Data patterns, although the Ogres go into more detail. The final subsection discusses a selection of kernels focusing on analytics which are instances of Ogres. We intend to follow up with other Ogre "mini-app" or "kernel" instances covering a broader set of facets, including those from database benchmarking [18].

3.1 Problem Architecture View of Ogres

The Problem Architecture view has facets that describe the overall structure of the application, which determines the overall software and is an important driver of the software and hardware architecture discussed later.

Table 6: Problem Architecture Facet of Ogres (Meta or Macro Pattern)

Pleasingly Parallel	Seen in BLAST, Protein docking, some (bio-)imagery including Local Analytics or Local Machine Learning with pleasingly parallel filtering (light source data, radar images)
Classic MapReduce	Search, Index and Query and Classification algorithms like collaborative filtering (G1 for MR-Stat in Table 2, G7)
Map Collective	Seen in machine learning - especially with linear algebra kernels
Map P2P	Point to Point Communication seen in parallel simulation and graph algorithms
Map Streaming	Point to Point Communication seen in parallel simulation and graph algorithms
Shared Memory	As opposed to distributed data (memory). Corresponds to problem where shared memory implementations are important. Tend to be dynamic asynchronous
Graph	Problem set up as a graph as opposed to vector, grid (G3)
SPMD	Single Program Multiple Data well-known parallel computing style
BSP	Bulk Synchronous Processing: well-defined compute-communication phases
Fusion	Knowledge discovery often involves fusion of multiple methods. All applications often involve orchestration (workflow) of multiple components
Dataflow	Composite structure with multiple components linked by exchanged data
Agents	As used in epidemiology, discrete event simulations etc. Swarm approaches
Workflow	Many applications often involve orchestration (workflow) of multiple components

3.2 Execution Features View of Ogres

This facet contains application characteristics that are familiar from the simulation domain as well as the famous V's of Big Data. The data abstraction layer is a key facet that we highlight in the software architecture rather than burying it as is done now in particular packages like Hadoop (key-value) and Giraph (graph). Simulations are often set up in well-defined physical spaces, however data is generally more abstract and the algorithms are typically quite different for metric and non-metric spaces. In contrast to the problem architecture facet, the computational features facet has a direct handle/relevance to performance. Note non-metric space algorithms are often $O(N^2)$. As discussed in the NRC report, there is a great deal of opportunity to incorporate sophisticated new algorithms to reduce $O(N^2)$ to $O(N \log n)$. This is commonly used in search and sort algorithms but not yet applied in computation despite promising initial work [8, 19, 20].

3.3 Data Source and Data Style Facet of Ogres

The facets of Table 8 cover the acquisition, storage, management and access to the data. The mantra of bringing computing to the data is an important principle, especially for the Internet of Things when it is often not practical since backend (clouds) are needed to provide adequate computing.

Table 7: Execution Features View Facets of Ogres

Performance metrics	As measured in benchmarks
Flops per byte	Important for performance
Execution Environment	Cloud or HPC; are Core libraries needed such as matrix-matrix/vector algebra, conjugate gradient, reduction, broadcast (G4)
Volume	Data size
Velocity	Measures Streaming
Variety	Multiple data sources are often mixed. See Fusion facet
Veracity	Accuracy of data affecting pre-processing needed and reliability of answer
Communication Structure	Interconnect structure? Is communication Synchronous or Asynchronous? In latter case shared memory may be attractive
Static or Dynamic?	Does application (graph) change during execution?
Regularity	Most applications consist of a set of interconnected entities; is this regular as a set of pixels or is it a complicated irregular graph?
Iterative or not?	Important algorithm characteristic
Data Abstraction	Key-value, pixel, graph, vector, HDF5, Bag of words, etc.
Data Space?	Are data points in metric or non-metric spaces (G2)?
Complexity	Is algorithm $O(N^2)$ or $O(N)$ (up to logs) for N points per iteration (G2)?

It is interesting that the HPC approach of large shared file systems uses technologies like Lustre, which is rather different from commercial systems that use databases or HDFS. Before the data gets to the 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 or lower (Real time control, streaming). This is measured by Archived/Batched/Streaming facets. Figure 1 stresses that an important source of data is the output of other programs, as data is streamed through a workflow. Other characteristics are needed for permanent auxiliary/comparison datasets which could be interdisciplinary, implying nontrivial data movement/replication. This is covered by the Variety facet in the Execution view.

3.4 Processing or Run-time View of Ogres

We have already stressed the importance and distinction between Local and Global Machine Learning. These are often associated with Expectation Maximization and Steepest Descent methods.

3.5 Analytics Algorithm/Kernels as Ogre Instances

Table 8: Data Source and Style Facet of Ogres

SQL, NewSQL, or NoSQL	NoSQL includes Document, Column, Key-value, Graph, Triple store
Enterprise data systems	10 examples from NIST [1] integrate SQL/NoSQL
Files or Objects	Files as managed in iRODS and extremely common in scientific research. Objects most common in ABDS
HDFS/ Lustre/GPFS	Are data and compute collocated?
Archive/ Batched/ Streaming	Streaming is Incremental update of datasets with new algorithms to achieve real-time response (G7)
Storage system styles	Styles include Shared, Dedicated, Permanent, and Transient
Metadata/ Provenance	Define overall features of data and processing
Internet of Things	24 [21] to 50 (Cisco [22, 23]) billion devices on the Internet by 2020
HPC generated data	Simulations generate visualization output that often needs to be mined
GIS	Geographical Information Systems provide access to geospatial data

Table 10: Ogre Instances for Important Analytics

Pleasingly Parallel (Map Only) or Local Machine Learning: any algorithm
Map-Reduce
Search, Query, Index: Dominant commercial use and important in Science with fewer users
Recommender Systems including Collaborative filtering: Major commercial use, little use in Science
Summarizing statistics (MRStat) as in LHC Data analysis (histograms) (G1)
Linear Classifiers: Bayes, Random Forests
Alignment and Streaming (G7)
Genomic Alignment, Incremental Classifiers
Global Analytics — Nonlinear Solvers (Structure depends on Objective Function) (G5, G6)
Stochastic Gradient Descent SGD
(L-)BFGS approximation to Newton's Method
Levenberg-Marquardt solver
Global Analytics — Map-Collective (See Mahout, MLlib) (G2, G4, G6)
Outlier Detection
Clustering (many methods) related to community identification in networks
Mixture Models, LDA (Latent Dirichlet Allocation), PLSI (Probabilistic Latent Semantic Indexing)
SVM and Logistic Regression
PageRank (find leading eigenvector of sparse matrix)
SVD (Singular Value Decomposition)
MDS (Multidimensional Scaling)
Learning Neural Networks (Deep Learning)
Hidden Markov Models
Global Analytics — Map-Communication (targets for Giraph) (G3)
Graph Structure (Communities, subgraphs/motifs, diameter, maximal cliques, connected components)
Network Dynamics - Graph simulation Algorithms (epidemiology)
Global Analytics — Asynchronous Shared Memory (may be distributed algorithms)
Graph Structure (Betweenness centrality, shortest path) (G3)
Linear/Quadratic Programming, Combinatorial Optimization, Branch and Bound (G5)

Table 9: Processing/Run-time View Facets of Ogres

Micro Benchmarks	A simple kernel or mini-app used to measure core system performance
LML	Local Analytics or Local machine Learning
GML	Global Analytics or Machine Learning requiring iterative runtime (G5, G6)
Base Statistics	Simple statistics seen in Table 2 as MRStat
Recommendations	Collaborative Filtering and other recommender analytics
Search/Query/ Index	Rich set of technologies used in Search, Query and Indexing data
Classification	Technologies to label data (SVM, Bayes, deep learning, clustering)
Learning	Training algorithms
Optimization Methodology	Machine Learning, Nonlinear Optimization, Least Squares, Linear/Quadratic Programming, Combinatorial Optimization, expectation maximization, Monte Carlo, Variational Bayes, Global Inference
Streaming	Growing class of fast online $O(N)$ algorithms
Alignment	Variant of Search seen in sequence comparison as in BLAST
Linear Algebra	Many machine learning algorithms build on linear algebra kernels
Graph	Problem set up as a graph as opposed to vector, grid, etc. (G3)
Visualization	Important component of many analysis pipelines

Table 10 records particular data analysis algorithms that play the same role as the members of the NAS parallel benchmarks. They form instances of Ogres covering a range of facets already introduced. These are deliberately kernels and further work is needed to specify more precise mechanisms. For example, there are many very different outlier and clustering algorithms corresponding to different scenarios (such as metric or non-metric spaces) and goals (such as tradeoff between performance and quality). Working with colleagues, we are developing benchmarks in the areas identified in Table 9. One should also introduce Ogre instances corresponding to full applications and workflows. These are important but not discussed here. We intend to investigate further work to introduce mini-apps as Ogre instances with broad coverage of the different facets in the 4 views.

4. HARDWARE AND SOFTWARE ARCHITECTURE ISSUES

4.1 Six Important Architectures

In Table 12, we present 6 problem architectures that map into 6 distinct system architectures which seem to cover the Ogres and their facets discussed in previous sections. Category 11.6 is the shared memory architecture needed for some graph algorithms that perform better here as well as for some large memory applications. The central batch architectures are 11.1 to 11.4 which correspond exactly to the four forms of MapReduce we have presented previously [24] summarized in Figures 2a) and Figure 2b), which introduces the Map-Streaming architecture. Note these six architectures only describe some of the facets in Tables 6-9. There are many other issues that need to be addressed including support of workflow and the data systems captured in the facets of Table 8.

Note that we separate Map-Collective [25, 26] and Map-

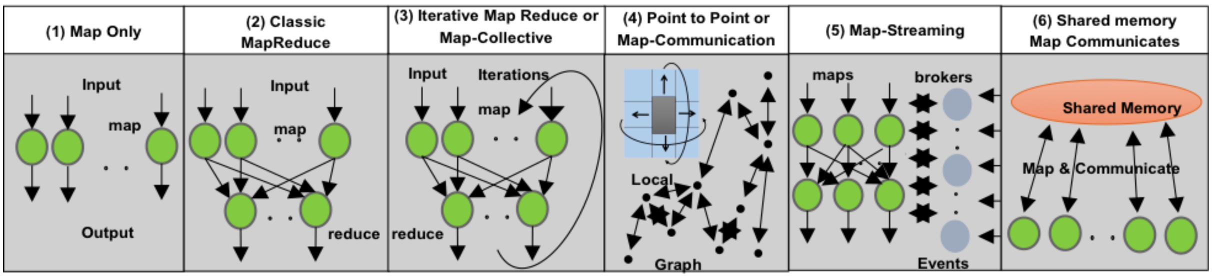

Figure 2: Six Distinctive Software/Hardware Architectures for Data Analytics

(Point to Point) Communication following the Apache projects Hadoop, Spark and Giraph which focus on these cases. These programming models or runtimes differ in communication style (bandwidth versus latency), application abstraction (key-value versus graph), possible scheduling or load-balancing. HPC with MPI suggests that one could integrate categories 11.3 and 11.4 into a single environment. This approach is illustrated by the Harp plug-in for Hadoop which supports both models [27]. We recently added the map-streaming architecture of Table 11.5 and Figure 2b); recall that Table 2 listed 41 streaming applications in the 51 use cases.

Table 11: Distinctive Software/Hardware Architectures for Data Analytics

1	Pleasingly Parallel (Map Only)	Includes local machine learning (LML) as in parallel decomposition over items and
2	Classic MapReduce	Includes MRStat, search applications and those using collaborative filtering and motif finding implemented using classic MapReduce (Hadoop)
3	Iterative Map - Collective	Iterative MapReduce using Collective Communication as needed in clustering — Hadoop with Harp, Spark, etc.
4	Iterative Map - (Point to Point) Communication	Iterative MapReduce such as Giraph with point-to-point communication; includes most graph algorithms such as maximum clique, connected component, finding diameter, community detection. Vary in difficulty of finding partitioning (classic parallel load balancing)
5	Map-Streaming	Architecture such as that of Apache Storm that supports streaming data. [28] A set of brokers holding data as it streams, supplying a set of long running tasks that filter and accumulate data.
6	Shared (Large) Memory	Thread-based (event driven) graph algorithms such as shortest path and Betweenness centrality. Large memory applications

4.2 Comparison between Data Intensive and Simulation Problems

We can use the Ogre facet analysis and the data analytics architectures to compare data intensive and simulation applications. Looking back at Table 6, there are some clear similarities between them, with "Pleasingly parallel" (11.1), BSP and SPMD being common in both arenas. However the Classic MapReduce architecture (11.2) is a major Big Data paradigm though much less common in simulations. One example is the execution of multiple simulations (as in Quantum Monte Carlo) followed by a reduce operation to collect the results of different simulations. The Iterative Map-Collective architecture (category 11.3) is common in Big Data analytics such as clustering where there is no local graph structure and the parallel algorithms involve large-scale collectives but no point-to-point communication. The same structure is seen in N-body (long range force) or other "all-pairs" simulations without the locality typical from discretizing differential operators.

Many simulation problems have the Map-Communication (category 11.4) architecture with many smallish point-to-point messages coming from local interactions between points defining systems to be simulated. The importance of sparse data structures and algorithms is well understood in simulations and is seen in some Big Data problems such as PageRank, which calculates the leading eigenvector of the sparse matrix formed by internet site links. Other Big Data sparse data structures are seen in user-item ratings and bags of words problems (although these have feature that suggest sparseness corresponds to missing information and not to zero values). Most items are rated by only a few users and many documents contain a small fraction of the word vocabulary. However important data analytics involve full matrix algorithms. For example, recent papers [27, 29, 30] on a new Multidimensional Scaling method use conjugate gradient solvers with full matrices as opposed to the new sparse conjugate gradient benchmark HPCG being developed for supercomputer (Top500) evaluations [31].

Note that there are similarities between some Big Data graph problems and particle simulations with an unusual potential defined by the graph node connectivity. Both use the Map-Communication architecture, and the links in a Big Data graph are equivalent to strength of force between the graph nodes considered as particles. In this analogy, many Big Data problems are "long range force" corresponding to a graph where all nodes are linked to each other. As in simulation cases, these $O(N^2)$ problems are typically very compute intense but straightforward to parallelize efficiently. It is interesting to consider the analogue of the "fast multipole" methods for the fully connected Big Data problems, which can dramatically improve the performance to $O(N)$ or $O(NlogN)$ as discussed in Sec. 3.3. Finally note the network connections used in deep learning are indeed sparse, but in recent image interpretation studies [32], the network weights are block sparse (corresponding to links to pixel blocks) and can be formulated as full matrix operations with GPUs and MPI running efficiently with these blocks.

The map-streaming architecture (11.5) is seen in problems such as Twitter analysis and data assimilation where large-scale simulations are updated by streaming data. The final architecture of category11.6 (Shared Memory) is important in some applications but not heavily used in either simulations or Big Data, although large memory systems are used extensively in gene assembly applications.

The above discussion focuses on a qualitative comparison of Big Data applications with traditional simulation (HPC) applications, namely comparing the structure. As is evident, there are similarities as well as points of distinction. It is likely, however, that there will be significant differences in values of facets of the "execution features" view for the two application classes; for instance the distribution of the values of different ratios (e.g., ratio of computing to I/O, ratio of memory to I/O, etc.) characterizing the computational feature will be different. We will investigate both quantitative and qualitative differences in future work.

4.3 A Big Data Software Environment

We have previously described [33, 34, 35] how we propose to implement Big Data applications exploiting the HPBDS architecture sketched in Table 12 [36]. This combines the best practice commercial Big Data software with an emphasis on Apache projects with HPC subsystems. Table 12 illustrates by green shading those layers where HPC adds significant value to the Apache stack ABDS. Note that high performance communication is known to be critical for simulations but is also essential for many scientific Big Data applications. Commercial applications have large "search" (10.2) components corresponding to the huge number of users accessing commercial Big Data systems. In science, this step is necessary – especially for good data management – but is a much lower fraction of system use as the number of scientists accessing data is far lower than the number of users of commercial Big Data.

5. DISCUSSION AND CONCLUSION

This is only an initial discussion about our objectives, scope and methodology, and is by no means a complete or comprehensive body of work. It is motivated by the fact that there are several existing efforts at describing and highlighting Big Data applications, yet many are domain or usage specific. We move beyond any specific set of applications, and focus on Big Data applications and analytics kernels that are generally considered to be of relevance/importance to science and engineering using a context that includes a limited set of commercial problems. Using this broad range of Big Data applications as our working set, this paper is an attempt at distilling the Big Data properties (facets divided into 4 views) and organizing the plethora of disparate Big Data applications using these properties. Although we validate using analytics kernels, this classification/organization will in turn shed light on and help provide better understanding of both the structure of science and engineering Big Data applications, as well as determinants of their performance. In Section 4, we show how a deeper appreciation of the Ogre facets will help design and implement better hardware and software systems.

Acknowledgements

Work at Indiana was partially supported by NSF CIF21 DIBBS ACI-1443054, CAREER OCI-1149432 and AFOSR FA9550-13-1-0225 awards. Work at Rutgers was supported by NSF CAREER ACI-1253644 and NSF ACI-1443054. We thank Nikhil Shenoy for help with Latex and conversion.

Appendix Table 13 summarizes the 51 NIST use cases.

6. REFERENCES

[1] NIST, "Big data initiative reports from v1," 2013, http://bigdatawg.nist.gov/_uploadfiles/M0357_v1_5179958773.docx.

[2] S. Jha, N. C. Hong, S. Dobson, D. S. Katz, A. Luckow, O. Rana, and Y. Simmhan, "Introducing distributed dynamic data-intensive (d3) science: Understanding applications and infrastructure," 2014.

[3] S. Jha, M. Cole, D. Katz, O. Rana, M. Parashar, and J. Weissman, "Distributed computing practice for large-scale science & engineering applications," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 11, pp. 1559–1585, 2013.

[4] "R open source statistical library," http://www.r-project.org/.

[5] "Apache mahout scalable machine learning and data mining," http://mahout.apache.org/.

[6] "Machine learning library (mllib)," https://spark.apache.org/docs/1.3.0/mllib-guide.html.

[7] NIST, "Nist big data public working group (nbd-pwg) use cases and requirements," 2013, http://bigdatawg.nist.gov/usecases.php.

[8] Committee on the Analysis of Massive Data and Committee on Applied and Theoretical Statistics and Board on Mathematical Sciences and Their Applications and Division on Engineering and Physical Sciences, *Frontiers in Massive Data Analysis.* The National Academies Press, 2013.

[9] M. Berry *et al.*, "The perfect club benchmarks: Effective performance evaluation of supercomputers," *International Journal of High Performance Computing Applications*, vol. 3, no. 3, pp. 5–40, 1989.

[10] R. F. V. der Wijngaart, S. Sridharan, and V. W. Lee, "Extending the bt nas parallel benchmark to exascale computing," pp. 1–9, 2012.

[11] "Parkbench (parallel kernels and benchmarks)," http://www.netlib.org/parkbench/, 1996.

[12] J. Dongarra, E. Strohmaier, and M. Resch, "Top 500 supercomputer sites," http://www.top500.org/, 2014.

[13] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary, "Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers," http://www.netlib.org/benchmark/hpl/, September 10 2008.

[14] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.* Philadelphia, PA: SIAM, 1994.

[15] T. G. Mattson, B. A. Sanders, and B. L. Massingill, *Patterns for Parallel Programming.* Addison-Wesley

Table 12: Big Data Software Environment: ABDS

Cross-Cutting Functionalities	**Workflow-Orchestration:** ODE, ActiveBPEL, Airavata, Pegasus, Kepler, Swift, Taverna, Triana, Trident, BioKepler, Galaxy, IPython, Dryad, Naiad, Oozie, Tez, Google FlumeJava, Crunch, Cascading, Scalding, e-Science Central, Azure Data Factory, Google Cloud Dataflow, NiFi(NSA), Jitterbit, Talend, Pentaho, Apatar
Message and Data Protocols: Avro, Thrift, Protobuf	**Application and Analytics:** Mahout, MLlib , MLbase, DataFu, R, pbdR, Bioconductor, ImageJ, OpenCV, Scalapack, PetSc, Azure Machine Learning, Google Prediction API & Translation API, mlpy, scikit-learn, PyBrain, CompLearn, DAAL(Intel), Caffe, Torch, Theano, DL4j, H2), IBM Watson, Oracle PGX GraphLab, GraphX, IBM System G, GraphBuilder(Intel), TinkerPop, Google Fusion Tables, CINET, NWB, Elasticsearch, Kibana, Logstash, Graylog, Splunk, Tableu, D3.js, three.js, Potree
Distributed Coordination: Google Chubby, Zookeeper, Giraffe, JGroups	**Application Hosting Frameworks:** Google App Engine, AppScale, Red Hat OpenShift, Heroku, Aerobatic, AWS Elastic beanstalk, Azure, Cloud Foundry, Pivotal, IBM BlueMIx, Ninefold, Jelastic, Stackato, appfog, CloudBees, Engine Yard, CloudControl, dotCloud, Dokku, OSGi, HUBzero, OODT, Agave, Atmosphere
	High Level Programming: Kite, Hive, HCatalog, Tajo, Shark, Phoenix, Impala, MRQL, SAP Hana, HadoopDB, PolyBase, Pivotal HD/Hawq, Presto, Google Dremel, Google BigQuery, Amazon Redshift, Drill, Kyoto Cabinet, Pig, Sawzall, Google Cloud DataFlow, Summingbird
Security & Privacy: InCommon, Eduroam, OpenStack, Keystone, LDAP, Sentry, Sqrrl, OpenID, SAML OAuth	**Streams:** Storm, S4, Samza, Granules, Google MillWheel, Amazon Kinesis, LinkedIn Databus, Facebook Puma/Ptail//Scribe/ODS, Azure Stream Analytics
	Basic Programming Model and Runtime, SPMD, MapReduce: Hadoop, Spark, Twister, Stratosphere (Apache Flink), Reef, Hama, Giraph, Pregel, Pegasus, ligra, GraphChi
	Inter-process communication Collectives, , publish-subscribe: MPI, Harp, Netty, ZeroMQ, ActiveMQ, RabbitMQ, NaradaBrokering, QPid, Kafka, Kestrel, JMS, AMQP, Stomp, MQTT **Public Cloud:** Amazon SNS, Lambda, Google Pub Sub, Azure Queues, Event Hubs
	In-memory databases/caches: Gora (general object from NoSQL), Memcached, Redis, LMDB (key-value), Hazelcast, Ehcache, Infinispan
Monitoring: Ambari, Ganglia, Nagios, Inca	**Object-relational mapping:** Hibernate, Open JPA, EclipseLInk, DataNucleus, ODBC/JDBC
	Extraction Tools: UIMA, Tika
	SQL(NewSQL): Oracle, DB2, SQL Server, SQLite, MySQL, PostgreSQL, CUBRID, Galera Cluster, SciDB, Rasdaman, Apache Derby, Pivotal Greenplum, Google Cloud SQL, Azure SQL, Amazon RDS, Google F1, IBM dashDB, N1Ql, BlinkDB
21 layers, Over 300 Software Packages on April 3, 2015	**NoSQL:** Lucene, Solr, Solandra, Voldemort, Riak, Berkeley DB, Kyoto/Tokyo Cabinet, Tycoon, Tyrant, MongoDB, Espresso, CouchDB, Couchbase, IBM Cloudant, Pivotal Gemfire, HBase, Google Bigtable, LevelDB, Megastore and Spanner, Accumulo, Cassandra, RYA, Sqrrl, Neo4J, Yarcdata, AllegroGraph, Facebook Tao, Titan:db, Jena, Sesame **Public Cloud:** Azure Table, Amazon Dynamo, Google DataStore
	File Management: iRODS, NetCDF, CDF, HDF, OPeNDAP, FITS, RCFile, ORC, Parquet
	Data Transport: BitTorrent, HTTP, FTP, SSH, Globus Online (GridFTP), Flume, Sqoop, Pivotal GPLOAD/GPFDIST
	Cluster Resource Management: Mesos, Yarn, Helix, Llama, Google Omega, Facebook Corona, Celery, HTCondor, SGE, OpenPBS, Moab, Slurm, Torque, Globus Tools, Pilot Jobs
	File systems: HDFS, Swift, Haystack, f4, Cinder, Ceph, FUSE, Gluster, Lustre, GPFS, GFFS **Public Cloud:** Amazon S3, Azure Blob, Google Cloud Storage
	Interoperability: Libvirt, Libcloud, JClouds, TOSCA, OCCI, CDMI, Whirr, Saga, Genesis
	DevOps: Docker, Puppet, Chef, Ansible, SaltStack, Boto, Cobbler, Xcat, Razor, CoudMesh, Juju, Foreman, OpenSTack Heat, Rocks, Cisco Intelligent Automation for Cloud, Ubuntu MaaS, Facebook Tupperware, AWS OpsWorks, OpenStack Ironic, Google Kibernetes, Buildstep, Gitreceive
	IaaS Management from HPC to hypervisors: Xen, KVM, Hyper-V, VirtualBox, OpenVZ, LXC, Linux-Vserver, OpenStack, OpenNebula, Eucalyptus, Numbus, CloudStack, CoreOS, VMware ESXi, vSphere, and vCloud, Amazon, Azure, Google and other public Clouds **Networking:** Google Cloud DNS, Amazon Route 53

Professional, 2013.

[16] K. Asanovic *et al.*, “The landscape of parallel computing research: A view from berkeley,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-183, Dec 2006. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

[17] N. A. S. Division, “Nas parallel benchmarks,” 1991, https://www.nas.nasa.gov/publications/npb.html.

[18] C. Baru and T. Rabl, “Tutorial 4 ” big data benchmarking” at 2014 ieee international conference on big data,” 2014. [Online]. Available: http://cci.drexel.edu/bigdata/bigdata2014/tutorial.htm

[19] P. Ram, D. Lee, W. March, and A. Gray, “Linear-time algorithms for pairwise statistical problems,” in *Advances in Neural Information Processing Systems. NIPS*, Conference Proceedings, pp. 1527–1535.

[20] J. Qiu, J. Ekanayake, T. Gunarathne, J. Y. Choi, S.-H. Bae, Y. Ruan, S. Ekanayake, S. Wu, S. Beason, G. Fox, M. Rho, and H. Tang, *Data Intensive Computing for Bioinformatics*. IGI Publishers, 2011.

[21] O. Malik, “Internet of things will have 24 billion devices by 2020,” http://gigaom.com/2011/10/13/internet-of-things-will-have-24-billion-devices-by-2020/.

[22] Cisco, “Visual networking index: Forecast and methodology, 2012–2017,” http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html, May 29 2013.

[23] D. Evans, “The internet of things: How the next evolution of the internet is changing everything,” http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, April 2011.

[24] J. Ekanayake, T. Gunarathne, J. Qiu, G. Fox, S. Beason, J. Y. Choi, Y. Ruan, S.-H. Bae, and H. Li,

Table 13: 51 General NIST uses and 20 Specialized

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
Earth, Environmental and Polar Science(10): Radar Scattering in Atmosphere, Earthquake, Ocean, Earth Observation, Ice sheet Radar scattering, Earth radar mapping, Climate simulation datasets, Atmospheric turbulence identification, Subsurface Biogeochemistry (microbes to watersheds), AmeriFlux and FLUXNET gas sensors
Energy(1): Smart grid
Enterprise Data Systems (10): Multiple users performing interactive queries and updates on a database with basic availability and eventual consistency (BASE); Perform real time analytics on data source streams and notify users when specified events occur; Move data from external data sources into a highly horizontally scalable data store, transform it using highly horizontally scalable processing (e.g. Map-Reduce), and return it to the horizontally scalable data store (ELT); Perform batch analytics on the data in a highly horizontally scalable data store using highly horizontally scalable processing (e.g. MapReduce) with a user-friendly interface (e.g. SQL); Perform interactive analytics on data in analytics-optimized database; Visualize data extracted from horizontally scalable Big Data store; Move data from a highly horizontally scalable data store into a traditional Enterprise Data Warehouse; Extract, process, and move data from data stores to archives; Combine data from Cloud databases and on premise data stores for analytics, data mining, and/or machine learning; Orchestrate multiple sequential and parallel data transformations and/or analytic processing using a workflow manager
Security & Privacy (10): Consumer Digital Media Usage; Nielsen Homescan; Web Traffic Analytics; Health Information Exchange; Personal Genetic Privacy; Pharma Clinic Trial Data Sharing; Cyber-security; Aviation Industry; Military - Unmanned Vehicle sensor data; Education - "Common Core" Student Performance Reporting

"Applicability of dryadlinq to scientific applications," http://grids.ucs.indiana.edu/ptliupages/publications/DryadReport.pdf, Community Grids Laboratory, Indiana University., Report, January 30 2010.

[25] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and . G. Fox, "Twister: A runtime for iterative mapreduce," New York, NY, USA, pp. 810–818, 2010. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851593

[26] B. Zhang, Y. Ruan, T.-L. Wu, J. Qiu, A. Hughes, and G. Fox, "Applying twister to scientific applications," http://grids.ucs.indiana.edu/ptliupages/publications/PID1510523.pdf, November 30-December 3 2010.

[27] B. Zhang, Y. Ruan, and J. Qiu, "Harp: Collective communication on hadoop," October 10 2014. [Online]. Available: http://grids.ucs.indiana.edu/ptliupages/publications/HarpQiuZhang.pdf

[28] S. Kamburugamuve, L. Christiansen, and G. Fox, "A framework for real-time processing of sensor data in the cloud," Report, November 7 2014. [Online]. Available: http://grids.ucs.indiana.edu/ptliupages/publications/iotcloud_hindavi_two_column_final_2.pdf

[29] Y. Ruan and G. Fox, "A robust and scalable solution for interpolative multidimensional scaling with weighting," October 22-25 2013.

[30] Y. Ruan, G. House, S. Ekanayake, U. Schutte, J. Bever, H. Tang, and G. Fox, "Integration of clustering and multidimensional scaling to determine phylogenetic trees as spherical phylograms visualized in 3 dimensions," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, May 2014, pp. 720–729.

[31] J. Dongarra and M. A. Heroux, "Toward a new metric for ranking high performance computing systems," http://www.sandia.gov/~maherou/docs/HPCG-Benchmark.pdf, pp. SANDIA REPORT SAND2013–4744 (Defines HPCG), June 2013.

[32] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and A. Ng, "Deep learning with cots hpc systems," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, Conference Proceedings, pp. 1337–1345.

[33] W. Chang, "Iso/iec jtc 1 study group on big data," in *1st Big Data Interoperability Framework Workshop: Building Robust Big Data Ecosystem*, vol. 2014. NIST, Conference Proceedings.

[34] G. Fox, J. Qiu, and S. Jha, "High performance high functionality big data software stack," http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/fox.pdf, 2014.

[35] S. Jha, J. Qiu, A. Luckow, P. Mantha, and G. C. Fox, "A tale of two data-intensive approaches: Applications, architectures and infrastructure," http://arxiv.org/abs/1403.1528, June 27- July 2 2014.

[36] "Hpc-abds kaleidoscope of 300 apache big data stack and hpc technologies," http://hpc-abds.org/kaleidoscope/.