**Need**

Hour glass figure

Tables of NIST use cases and their properties: Done

Tables of components in OGRE facets: Done

HPC-ABDS figure

**Sources of Information**

In discussing the structure of Big Data Applications, let us first discuss the inevitably incomplete input that we used to do our analysis. We have gained of course quite a bit of experience from our research over many years but 3 explicit sources that we used were a recent use case survey by NIST from Fall 2013; a survey of data intensive research applications (Distributed Computing MetaPatterns) by Jha et al.; and study of members of data analytics libraries including R, Mahout and MLLib. We follow with a summary of 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 in terms of requirements of a reference architecture. Here we will look at them differently to identify common patterns and characteristics which can be used to guide and evaluate Big Data hardware and software.

The use cases are organized into nine broad areas as follows, 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)
* Energy (1)

Note that the majority of use cases come from research applications but 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 that included

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 & 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 []

Need a discussion of Jha et al.

**Lessons from Parallel Computing**

Before discussing features and patterns of Big Data applications, it is instructive to consider the better understood parallel computing situation. Here the application requirements have been captured in many ways

1. **Benchmark Sets.** These vary from full applications [Kuck] to kernels or mini-applications such as the NAS Parallel Benchmarks[] or Parkbench[http://www.netlib.org/parkbench/] with the Top500 pacing application Linpack (HPL) particularly well known. The new sparse HPCG conjugate gradient benchmark is notable. Note benchmarks can be specified via explicit code and/or specified by a “pencil and paper specification” that can be optimized in any way for a particular platform.
2. **Patterns or Templates.** These can be similar to benchmarks but with different goals such as providing a generic framework that can be modified by users with details of their application as in Template book [Siam, ?recent UIUC]. Alternatively they can be aimed at illustrating different applications as in original Berkeley Dwarves [].

In this paper, our approach is nearest that of the Dwarves and one motivation for us calling our work 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 folk lore 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 and bisection bandwidth and scaled speedup are associated with substantial folklore.

The famous NAS Parallel Benchmarks NPB consists 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, are pretty 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 dwarves 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 dwarves are not exact kernels but describe problem 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 think that it is inevitable that both parallel computing and Big Data cannot be characterized with a single criteria and so we introduce multiple facets for our Ore characterization.

Add Jha et al. lessons

**Properties of the 51 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 is decomposed for different use cases. In Table 2, we identify 15 use case features that will be used later as components of the Ogre facets. 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.

|  |  |
| --- | --- |
| 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 |
| Modelled 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 |

|  |  |  |
| --- | --- | --- |
| Table 2: Some Features of NIST Use Cases | | |
| Abbreviation | **#** | **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 |
| Class | 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, Lifted Belief Propagation, Stochastic Gradient Descent, L-BFGS, Levenberg-Marquardt . Sometimes 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** |

It’s important to note that machine learning is commonly used but there is an interesting distinction between what are termed Local (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 and is typified by use of accelerators (GPU). In GML, the machine learning is applied over the full dataset with MapReduce, MPI or equivalent. Typically GML comes from maximum likelihood or χ2 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 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 |
| 43 | Radar Data Analysis for CReSIS Remote Sensing of Ice Sheets | Identify glacier beds and snow layers | PP, LML moving to GML for full ice-sheet |
| 44 | UAVSAR Data Processing, Data Product Delivery, and Data Services | Find and display slippage from radar images | PP |
| 45, 46 | Analysis of Simulation visualizations | Find paths, classify orbits, classify patterns that signal earthquakes, instabilities, climate, turbulence | PP LML ?GML |

**Properties of other use cases**

Other NIST and Jha et al.

**The 5 Facets of the Big Data Ogres**

We introduce 5 facets or classification dimensions to categorize Big data applications. These are Application Style, Problem architecture, Computational features, Data Source or Style and Analytics Algorithm/Kernel. There are of course other ways of looking at the Ogres and our work should be treated as an initial suggestion for further discussion.

**Application Style Facet of Ogres**

|  |  |
| --- | --- |
| # | Table 4: Application Style Facet of Big Data Ogres |
| 30 | Classification divide data into categories including collaborative filtering |
| 12 | Search Index and query |
|  | Maximum Likelihood or χ2 minimizations |
|  | Expectation Maximization (often Steepest descent) |
| 36 | Local (pleasingly parallel) Machine Learning |
| 23 | GML or (Exascale) Global Optimization such as Learning Networks, Variational Bayes and Gibbs Sampling |
| 2 | Use Agents as in epidemiology (swarm approaches)? |

**Problem architecture Facet of Ogres**

|  |
| --- |
| Table 5: Problem Architecture Facet of Ogres (Meta or Macro Pattern) |
| Pleasingly Parallel as in BLAST, Protein docking, some (bio-)imagery including Local Analytics or Local Machine Learning with pleasingly parallel filtering, as in light source data, radar images |
| Classic MapReduce for Search and Query |
| Global Analytics or Machine Learning requiring iterative programming models |
| Problem set up as a graph as opposed to vector, grid |
| SPMD (Single Program Multiple Data) |
| Bulk Synchronous Processing: well-defined compute-communication phases |
| Fusion: Knowledge discovery often involves fusion of multiple methods. |
| Workflow (often used in fusion) |

**Computational features Facet of Ogres**

|  |
| --- |
| Table 6: Computational Features Facet of Ogres |
| Flops per byte: important for performance |
| Communication Interconnect requirements; |
| Is application (graph) constant or dynamic? |
| Most applications consist of a set of interconnected entities; is this regular as a set of pixels or is it a complicated irregular graph? |
| Is communication BSP or Asynchronous? In latter case shared memory may be attractive; |
| Are algorithms Iterative or not? |
| Data Abstraction: key-value, pixel, graph, vector |
| Are data points in metric or non-metric spaces? |
| Core libraries needed: matrix-matrix/vector algebra, conjugate gradient, reduction, broadcast …. |

Hdf5 etc.

Composition multiple styles

**Data Source or Style Facet of Ogres**

|  |
| --- |
| Data Source and Style Facet of Ogres |
| SQL |
| NOSQL based |
| Other Enterprise data systems (10 examples from NIST []) |
| Set of Files (as managed in iRODS) |
| Internet of Things |
| Streaming |
| HPC simulations |
| Involve GIS (Geographical Information Systems) |
| 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 or lower (Real time control, streaming) |
| There are storage/compute system styles: Shared, Dedicated, Permanent, Transient |
| Other characteristics are needed for permanent auxiliary/comparison datasets and these could be interdisciplinary, implying nontrivial data movement/replication |

**Analytics Algorithm/Kernel Facet of Ogres**

|  |
| --- |
| Table 7: Core Analytics Facet of Ogres (microPattern) |
| Pleasingly Parallel (Map Only) or Local Machine Learning: ~any algorithm |
| Map-Reduce |
| Search, Query, Index: Dominant commercial use and important in Science with less users |
| Recommender Systems including Collaborative filtering: Dominant commercial use, Little Science |
| Summarizing statistics (MRStat) as in LHC Data analysis (histograms) |
| Linear Classifiers: Bayes, Random Forests |
| Global Analytics – Nonlinear Solvers (Structure depends on Objective Function) |
| Stochastic Gradient Descent SGD |
| (L-)BFGS approximation to Newton’s Method |
| Levenberg-Marquardt solver |
| Global Analytics – Map-Collective (See Mahout, MLlib) |
| Outlier Detection |
| Clustering (many methods) |
| 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) |
| Graph Structure (Communities, subgraphs/motifs, diameter, maximal cliques, connected components) |
| Network Dynamics - Graph simulation Algorithms (epidemiology) |
| Global Analytics – Asynchronous Shared Memory |
| Graph Structure (Betweenness centrality, shortest path) |

**Data/Simulation Comparison**

**Hardware and Software Architecture Issues**

**---- To Delete**

**Commercial Big data is big**

**HPC-ABDS Overview**

Layers

Target 5 Architectures

Principles including Hourglass

**NIST Review**

Template

Table of 51 use cases

Examples of V's properties

20 additional use case

**Building Ogres**

Summary of mini-apps

NPB

Berkeley dwarfs

Distributed Computing MetaPatterns (see CloudDB talk)

**Properties of properties of 51 use cases**

What is parallelism Over

Low-Level (Run-time) Computational Types

Higher-Level Computational Types or Features

Global Machine Learning aka EGO – Exascale Global Optimization

**A set of Image data related use cases (Better summarized in Cetraro)**

Facebook etc.

17:Pathology Imaging/ Digital Pathology

18: Computational Bioimaging

26: Large-scale Deep Learning

27: Organizing large-scale, unstructured collections of consumer photos

36: Catalina Real-Time Transient Survey (CRTS)

43: Radar Data Analysis for CReSIS Remote Sensing of Ice Sheets

44: UAVSAR Data Processing, Data Product Delivery, and Data Services

**Facets of the Ogres**

Application Class Facet of Ogres

Problem Architecture Facet of Ogres (Meta or MacroPattern)

4 Forms of MapReduce

Computational Features

Data Source and Style Facet

**Analytics Facet (kernels) of the Ogres**

Several examples

**Parallel Global Machine Learning Examples**

MDS and Clustering pictures

Harp WDA-SMACOF efficiency results

Summary of Harp

Spherical Phylogram from MDS

More results from Azure and original hpc-abds paper (In Spare Slides and CloudDB talk)

**Comparing Data Intensive and Simulation Problems**

Communication structure

Graphs v Particle Dynamics

**Comparing Commercial and Science uses**

Cetraro: Note key difference between commercial and science. Former has huge number of users which map into classic mapreduce. Latter needs much of this capability for management/access functions but different scale. However backend use issues comparable in 2 cases with of course different algorithms

**Sensors and the Internet of Things - Streaming**

Sensors/Streaming as a Service

Relevance of cloud

Projection of IoT size

IOTCloud and performance

39: Particle Physics: Analysis of LHC Large Hadron Collider Data: Discovery of Higgs particle

13: Large Scale Geospatial Analysis and Visualization

50: DOE-BER AmeriFlux and FLUXNET Networks

51: Consumption forecasting in Smart Grids

**Java Grande**

Kernel mpiJava performance

DAVS Java clustering performance C# v Java

Results of DAVS

**Conclusions**

Integrate (don’t compete) HPC with “Commodity Big data” (Google to Amazon to Enterprise Data Analytics)

i.e. improve Mahout; don’t compete with it

Use Hadoop plug-ins rather than replacing Hadoop

Enhanced Apache Big Data Stack HPC-ABDS has ~120 members

Opportunities at Resource management, Data/File, Streaming, Programming, monitoring, workflow layers for HPC and ABDS integration

Data intensive algorithms do not have the well developed high performance libraries familiar from HPC

Strong case for high performance Java (Grande) run time supporting all forms of parallelism