**Towards a Comprehensive set of Big Data Benchmarks**

*Geoffrey C. Fox1, Shantenu Jha2, Judy Qiu*1*, Saliya Ekanayake1, Andre Luckow2*

*(1) School of Informatics and Computing, Indiana University, Bloomington, IN 47408, USA*

*(2) RADICAL, Rutgers University, Piscataway, NJ 08854, USA*

# Overview of Ogres

## What is an Ogre?

The Berkeley Dwarfs [1] were an important step towards defining an exemplar set of parallel (high performance computing) applications. The recent NRC report [2] gave Seven Computational Giants Of Massive Data Analysis, which start to define critical types of data analytics problems. We propose [3] Ogres ― an extension of these ideas based on an analysis by NIST of 51 big data applications [4]. Big Data Ogres provide a systematic approach to understanding applications, and as such they have facets which represent key characteristics defined both from our experience and from a bottom-up study of features from several individual applications. The facets capture common characteristics which are inevitably multi-dimensional and often overlapping. We note that in HPC, the Berkeley Dwarfs were very successful as patterns but did not get adopted as a standard benchmark set. Rather the NAS Parallel Benchmarks [5], Linpack [6], and (mini-)applications played this role. This suggests that benchmarks do not follow directly from patterns, but the latter can help by allowing one to understand breadth of applications covered by a benchmark set.

## Ogres have Facets

We suggest that Ogres would have properties that we classify in four distinct dimensions or views. Each view consists of facets; when multiple facets are linked together, they describe classes of big data problems represented as an Ogre. One view of an Ogre is the overall **problem architecture** which is naturally related to the machine architecture needed to support data intensive application while still being different. Then there is the **execution (computational) features** view, describing issues such as I/O versus compute rates, iterative nature of computation and the classic V’s of Big Data: defining problem size, rate of change, etc. The **data source & style** view includes facets specifying how the data is collected, stored and accessed. The final **processing** view has facets which describe classes of processing steps including algorithms and kernels. Ogres are specified by the particular value of a set of facets linked from the different views. The views contain the following facets given in Table 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Facet and View | | Comments | Dibbs | DB | NIST |
| Facets in Problem Architecture View: | | | | | |
| 1 | Pleasingly Parallel | Clear qualitative property overlapping Local Analytics | M | H | H |
| 2 | Classic MapReduce | Clear qualitative property | M | H | H |
| 3 | Map-Collective | Clear qualitative property | S | N | H |
| 4 | Map Point-to-Point (graphs) | Clear qualitative property | H | S | M |
| 5 | Map Streaming | Property of growing importance. Not well benchmarked | N | N | H |
| 6 | Shared memory (as opposed to distributed parallel algorithm) | Corresponds to problem where shared memory implementations important. Tend to be dynamic asynchronous | S | N | S |
| 7 | Single Program Multiple Data SPMD | Clear qualitative property | H | H | H |
| 8 | Bulk Synchronous Processing BSP | Needs to be defined but reasonable qualitative property | H | S | H |
| 9 | Fusion | Only present for composite Ogres | N | S | H |
| 10 | Dataflow | Only present for composite Ogres | N | M | H |
| 11 | Agents | Clear but uncommon qualitative property | N | N | S |
| 12 | Orchestration (workflow) | Only present for composite Ogres | N | H | H |
|  | | | | | |
| Facets in Execution View: | | | | | |
| 1 | Performance Metrics | Result of Benchmark | - | - | - |
| 2 | Flops per Byte (Memory or I/O) | I/O Not needed for “pure in memory” benchmark. Value not clear in broad overview. Could depend on implementation | - | - | - |
| 3 | Execution Environment (LN = Libraries needed, C= Cloud, HPC = HPC) | Depends on how benchmark set up | - | - | - |
| 4 | Volume | Depends on data size | - | - | - |
| 5 | Velocity | Associated with streaming facet but value depends on particular problem | N | S | H |
| 6 | Variety | Most useful for composite Ogres | N | S | H |
| 7 | Veracity | Most problems would not discuss but potentially important | N | N | M |
| 8 | Communication Structure (D=Distributed, I=Interconnect, S=Synchronization) | Qualitative property – related to BSP and Shared memory | U | U | U |
| 9 | D=Dynamic or S=Static | Clear qualitative property | H | H | H |
| 10 | R=Regular or I=Irregular | Clear qualitative property | H | H | H |
| 11 | Iterative? | Clear qualitative property | H | S | H |
| 12 | Data Abstraction(K= key-value, BW= bag of words, BI = bag of items, P= pixel/spatial, V= vectors/matrices, S= sequence, G= graph) | Clear quantitative property although important data abstractions not agreed | H | H | H |
| 13 | M= Metric Space or N= not? | Clear qualitative property | H | N | H |
| 14 | NN= O(N2) or N= O(N)? | Clear qualitative property | H | N | H |
|  | | | | | |
| Facets in Data Source&Style View: | | | | | |
| 1 | SQL/NoSQL/NewSQL? | Clear qualitative property. Need to decide on categories such as key-value, graph, document … | N | H | H |
| 2 | Enterprise data model (warehouses) | Clear qualitative property of data model | N | H | M |
| 3 | Files/Objects? | Clear qualitative property of data model | N | N | H |
| 4 | HDFS/Lustre/GPFS? | Clear qualitative property of data model | N | H | H |
| 5 | Archive/Batched/Streaming | Clear qualitative property but not for kernels as describes how data collected | N | N | H |
| 6 | Shared/Dedicated/Transient/Permanent | Clear qualitative property of data | N | N | H |
| 7 | Metadata/Provenance | Clear qualitative property but not for kernels as important aspect of data collection process | N | N | H |
| 8 | Internet of Things | Clear qualitative property. | N | S | H |
| 9 | HPC Simulations | Clear qualitative property | N | N | H |
| 10 | Geographic Information Systems; | Clear property but not for kernels | S | N | H |
|  | | | | | |
| Facets in Processing View: | | | | | |
| 1 | Micro-benchmarks | Important subset of kernels | N | H | N |
| 2 | Local Analytics | Well defined but overlaps Pleasingly Parallel | H | H | H |
| 3 | Global Analytics | Clear qualitative property | H | S | H |
| 4 | Base Statistics | Describes simple statistical averages needing simple MapReduce. MRStat in [4] | N | S | M |
| 5 | Recommender Engine | Clear type of machine learning of especial importance commercially | N | M | H |
| 6 | Search/Query/Index | Clear important class of algorithms | S | H | H |
| 7 | Classification | Clear important class of algorithms | S | M | H |
| 8 | Learning | Includes deep learning as category. | S | M | H |
| 7 | Optimization Methodology ( ML= Machine Learning, NO = Nonlinear Optimization, LS = Least Squares, EM = expectation maximization, LQP = Linear/Quadratic Programming, CO = Combinatorial Optimization) | LQP and CO overshadowed by machine learning but important where used. ML includes many analytics which are often NO and EM and sometimes LS (or similar Maximum Liklihood) | H | M | H |
| 10 | Streaming | Clear important class of algorithms | N | S | H |
| 11 | Alignment | Clear important class of algorithms | N | N | M |
| 12 | Linear Algebra Kernels | Important property of some analytics | H | N | H |
| 13 | Graph Algorithms | Clear important class of algorithms | H | S | M |
| 14 | Visualization | Clearly important aspect of data analysis but different in character to most other facets | S | N | H |

**Table 1: The four views and their constituent facets: High Use; M Medium use; S Small use; N essentially no use; - inapplicable; U Unknown. Dibbs is [7]. DB is Database analysis [8]. NIST is [4]**

Pleasingly Parallel

Classic MapReduce

Map-Collective

MP2P Map Point-to-Point

Map Streaming

Shared Memory

Single Program Multiple Data

Bulk Synchronous Parallel

Fusion

Dataflow

Agents

Workflow

Geographic Information System

HPC Simulations

Internet of Things

Metadata/Provenance

Shared / Dedicated / Transient / Permanent

Archived/Batched/Streaming

HDFS/Lustre/GPFS

Files/Objects

Enterprise Data Model

SQL/NoSQL/NewSQL

**Ogre Views and Facets**

Performance Metrics

Flops/Byte

Flops/Byte; Memory I/O

Execution Environment; Core Libraries

Volume

Velocity

Variety

Veracity

Communication Structure

Iterative / Simple

Metric = M / Non-Metric = N

= NN / = N

Regular = R / Irregular = I

Dynamic = D / Static = S

Linear Algebra Kernels

Graph Algorithms

Deep Learning

Classification

Recommender Engine

Search / Query / Index

Basic Statistics

Optimization Methodology

Global Analytics

Local Analytics

Micro-benchmarks

Visualization

Streaming

**Problem**

**Architecture View**

**Data Source and**

**Style View**

1

2

3

4

5

6

7

8

9

10

11

12

10

9

8

7

6

5

4

3

2

1

1

2

3

4

5

6

7

8

9

10

11

13

14

9

8

7

5

4

3

2

1

14

13

12

11

10

Alignment

6

Data Abstraction

12

**Execution View**

**Processing View**

In our language, instances of Ogres can form benchmarks. One can consider composite or atomic (simple, basic) benchmarks. For example, a clustering benchmark is an instance of an Ogre with a Map-Collective facet in the Problem Architecture view and the machine learning facet in the Processing view. The Execution view describes properties that could differ for different clustering algorithms and would often be measured in a benchmarking process. Note a simple benchmark like this could ignore the data source & style view and just be studied for in-memory data. Alternatively we can consider a composite benchmark linking clustering to different data storage mechanisms. A given benchmark can be associated with multiple facets in a single view, i.e. clustering has other problem architecture facets including SPMD, BSP, and Global Analytics.

# Particular Benchmarks as instances of Ogres

Our approach suggests choosing benchmarks from Ogre instances that cover a diverse range of facets. Rather than trying to be comprehensive at this stage, we give some examples. Note that kernel benchmarks are instances of Ogre Processing facets; this is where the NAS parallel benchmarks or TeraSort [9] would fit. On the other hand, micro-benchmarks such as MPI ping-pong and SPEC [10] are measures of Ogre execution facets.

Baru and Rabl’s tutorial [8] has a thorough discussion of benchmarks including the TPC series [11], HiBench [12], Yahoo Cloud Serving Benchmark [13], BigDataBench [14], BigBench [15] and Berkeley Big Data Benchmark [16] that quantify the Ogre data source & style facets.

The processing view has the well-known Graph500 [17] benchmarks (and associated machine ranking), but of course libraries like R [18], Mahout [19] and MLlib [20] also include many candidates for analytics benchmarks. We are part of a recent NSF project from the DIBBs (Data Infrastructure Building Blocks) program where one can use Ogres to classify Building Blocks that are the focus of this program. Below we list a few examples of problems we are studying, with a full set available at [7, 21]. Note each problem can provide benchmarks for many different execution view facets.

Applications and their characteristics are illustrated in Table 2 with concrete use cases. In particular, we classify them into **1) Graph Problems:** *Community detection, Subgraph/motif finding, Finding diameter, Clustering coefficient, Page rank, Maximal cliques, Connected component, Betweenness centrality, Shortest path*; all are instances of the Graph Algorithm facet of the Processing view and either the Map Point-to-Point and/or Shared memory facets in the Problem architecture view. 2) **Spatial Analytics:** *Spatial relationship-based queries* from the Search/Query/Index and MapReduce facets; *Spatial Clustering* from Global Machine Learning, Map-Collective and Global Analytics facets; *Distance-based queries* from Pleasingly Parallel and Search/Query/Index facets. These 3 benchmarks all have the spatial data abstraction facet. 3) **Machine Learning in general and for image processing:** Several *Clustering* algorithms illustrating O(N), O(N2), and Metric (non-metric) space execution view facets; *Levenberg-Marquardt Optimization* and *SMACOF Multi-Dimensional Scaling* with Linear Algebra Kernels and Expectation maximization facets from Processing view; *TFIDF Search and Random Forest* with Pleasingly Parallel facets. All heavily emphasize the machine learning facet of the processing view.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Applications** | **Problem Architecture View** | **Execution View** | **Processing View** |
| **Graph Analytics** | | | | |
| Community detection | Social networks, webgraph | 4, 7 | 9S, 10I, 11, 12G | 3, 9ML, 13 |
| Subgraph/motif finding | Webgraph, biological/social networks | 4, 7 | 9D, 10I, 12G | 3, 9ML, 13 |
| Finding diameter | Social networks, webgraph | 4, 7 | 9D, 10I, 12G | 3, 9ML, 13 |
| Clustering coefficient | Social networks | 4, 7 | 9S, 10I, 11, 12G | 3, 9ML, 13 |
| Page rank | Webgraph | 3, 4, 7 | 9S, 10I, 11, 12V | 3, 9ML, 12, 13 |
| Maximal cliques | Social networks, webgraph | 4, 7 | 9D, 10I, 12G | 3, 9ML, 13 |
| Connected component | Social networks, webgraph | 4, 7 | 9D, 10I, 12G | 3, 9ML, 13 |
| Betweenness centrality | Social networks | 6 | 9D, 10I, 12G, 13N | 9ML, 13 |
| Shortest path | Social networks, webgraph | 6 | 9D, 10I, 12G, 13N | 9ML, 13 |
| **Spatial Queries and Analytics** | | | | |
| Spatial relationship based queries | GIS/social networks/pathology informatics (add GIS execution view) | 2 | 6 | 6 |
| Distance based queries | 1 | 12P | 2 |
| Spatial clustering | 3, 7, 8 | 12P | 3, 9ML,EM |
| Spatial modeling | 1 | 12P | 2 |
| **Core Image Processing** | | | | |
| Image preprocessing | Computer vision/pathology informatics | 1 | 13M | 2 |
| Object detection & segmentation | 1 | 13M | 2, 9ML |
| Image/object feature computation | 1 | 13M | 2, 9ML |
| 3D image registration | 1 | 13M | 2, 9ML |
| Object matching | 1 | 13N | 2, 9ML |
| 3D feature extraction | 1 | 13N | 2, 9ML |
| **General Machine Learning** | | | | |
| DA Vector Clustering | Accurate Clusters | 3, 7, 8 | 9D, 10I, 11, 12V, 13M, 14N | 9ML, 9EM, 12 |
| DA Non metric Clustering | Accurate Clusters, Biology, Web | 3, 7, 8 | 9S, 10R, 11, 12BI, 13N, 14NN | 9ML, 9EM, 12 |
| Kmeans; Basic, Fuzzy and Elkan | Fast Clustering | 3, 7, 8 | 9D, 10I(Elkan), 11, 13M, 14N | 9ML, 9EM |
| Levenberg-Marquardt Optimization | Non-linear Gauss-Newton, use in MDS | 3, 7, 8 | 9D, 10R, 11, 12V, 14NN | 9ML, 9NO, 9LS, 9EM, 12 |
| DA, Weighted SMACOF | MDS with general weights | 3, 7, 8 | 9S, 10R, 11, 12BI, 13N, 14NN | 9ML, 9NO, 9LS, 9EM, 12, 17 |
| TFIDF Search | Find nearest neighbors in document corpus | 1 | 9S, 10R, 12BI, 9NMN, 14N | 2, 9ML |
| All-pairs similarity search | Find pairs of documents with TFIDF distance below a threshold | 3, 7, 8 | 9S, 10R, 12BI, 9NMN, 14NN | 9ML |
| Support Vector Machine SVM | Learn and Classify | 3, 7, 8 | 9S, 10R, 11, 12V, 9NMM, 14N | 7, 8, 9ML |
| Random Forest | Learn and Classify | 1 | 9S, 10R, 12BI, 9NMM, 14N | 2, 7, 8, 9ML |
| Gibbs sampling (MCMC) | Solve global inference problems | 3, 7, 8 | 9S, 10R, 11, 12BW, 9NMN, 14N | 9ML, 9NO, 9NM |
| Latent Dirichlet Allocation LDA with Gibbs sampling or Var. Bayes | Topic models (Latent factors) | 3, 7, 8 | 9S, 10R, 11, 12BW, 9NMN, 14N | 9ML, 9NM |
| Singular Value Decomposition SVD | Dimension Reduction and PCA | 3, 7, 8 | 9S, 10R, 11, 12V, 9NMM, 14NN | 9ML, 12 |
| Hidden Markov Models (HMM) | Global inference on sequence models | 3, 7, 8 | 9S, 10R, 11, 12BI | 2, 9ML, 12 |

**Table 2: Classification of Applications, their characteristics and Ogres benchmarks**

# Ogre-Driven Benchmarking

First, we note some qualitatively different types of benchmark. There are at simplest “micro-benchmarks” which capture some core machine performance. Then we have “atomic” kernels – simple non trivial algorithms or problems that cannot usefully be broken up. Then we see a class we call “mini-apps” that are the most complex and can be constructed in two different ways: top-down and bottom-up. In the top-down case, we start with a real complex application and simplify it to capture some “key” capabilities but we do not necessarily make it “atomic”. In the bottom-up approach, we take multiple “atomic kernels” and link them together as for example in benchmarking Mahout’s clustering algorithm reading data from Hbase.

The suggested process is to examine current benchmarking and list facets that they cover, then augment with new benchmarks to cover those facets not addressed in the initial choice. We can already see from Table 1 that the kernels do not cover many of the facets. This can be understood as facets come from an analysis of full applications and table 1 has by construction only analytics kernels. However we can suggest the following systematic approach

1. Consider analytic kernels in Table 1
2. Add the data view benchmarks from Baru and Rabl’s tutorial [8]
3. Add well-known “micro-benchmarks”
4. Examine applications that produced Table 1 which will for example link data and applications and have interesting sources such as HPC. Add these as “mini-application” benchmarks
5. Examine list of facets (such as streaming) that are not (well) represented in collection of benchmarks. Add benchmarks with these missing facets – if necessary returning to applications that motivated benchmarks.
6. Prune collection by noting and perhaps removing those whose facets are already represented

One must also address the many well studied general points of benchmarking, such as agreeing on datasets with various sizes (Volume facet in execution view), requiring correct answers for each implementation, and the choice between pencil and paper and source code specification of a benchmark.

**References**

1. Asanovic, K., R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, and K.A. Yelick. *The Landscape of Parallel Computing Research: A View from Berkeley*. 2006 December 18 [accessed 2009 December]; Available from: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html>.

2. Committee on the Analysis of Massive Data; Committee on Applied and Theoretical Statistics; Board on Mathematical Sciences and Their Applications; Division on Engineering and Physical Sciences; National Research Council, *Frontiers in Massive Data Analysis*. 2013: National Academies Press. <http://www.nap.edu/catalog.php?record_id=18374>

3. Shantenu Jha, Judy Qiu, Andre Luckow, Pradeep Mantha, and Geoffrey C. Fox, *A Tale of Two Data-Intensive Approaches: Applications, Architectures and Infrastructure*, in *3rd International IEEE Congress on Big Data Application and Experience Track*. June 27- July 2, 2014. Anchorage, Alaska. <http://arxiv.org/abs/1403.1528>.

4. NIST, *NIST Big Data Public Working Group (NBD-PWG) Use Cases and Requirements.* 2013. <http://bigdatawg.nist.gov/usecases.php>

5. NASA Advanced Supercomputing Division. *NAS Parallel Benchmarks*. 1991 [accessed 2014 March 28]; Available from: <https://www.nas.nasa.gov/publications/npb.html>.

6. A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. *HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*. 2008 September 10 [accessed 2014 July 19,]; Available from: <http://www.netlib.org/benchmark/hpl/>.

7. Indiana, Rutgers, Virginia Tech, Kansas, Stony Brook, Arizona State, and Utah. *CIF21 DIBBs: Middleware and High Performance Analytics Libraries for Scalable Data Science Summary*. 2014 December 18 [accessed 2015 January 12]; Available from: <http://grids.ucs.indiana.edu/ptliupages/presentations/Dibbs%20-%20Overall%20-%20Dec18-2014.pptx>

8. Chaitan Baru and Tilmann Rabl. *Tutorial 4 " Big Data Benchmarking" at 2014 IEEE International Conference on Big Data*. 2014 [accessed 2015 January 2 ]; Available from: <http://cci.drexel.edu/bigdata/bigdata2014/tutorial.htm>.

9. Apache Hadoop. *TeraSort map/reduce sort*. [accessed 2015 January 12]; Available from: <http://hadoop.apache.org/docs/current/api/org/apache/hadoop/examples/terasort/package-summary.html>.

10. *SPEC's Benchmarks*. [accessed 2015 January 12]; Available from: <https://www.spec.org/benchmarks.html>.

11. Transaction Processing Performance Council. *TPC Benchmarks*. [accessed 2015 January 12]; Available from: <http://www.tpc.org/information/benchmarks.asp>.

12. Lan Yi. *Experience with HiBench: From Micro-Benchmarks toward End-to-End Pipelines at WBDB 2013 Workshop* 2013 [accessed 2015 January 12];

13. Brian Cooper. *Yahoo Cloud Serving Benchmark with Key-value store (Accumulo, Cassandra, Hbase, MongoDB etc.) benchmarks*. [accessed 2015 January 12]; Available from: <https://github.com/brianfrankcooper/YCSB/>

14. Jianfeng Zhan. *BigDataBench: A Big Data Benchmark Suite from ICT, Chinese Academy of Sciences*. [accessed 2015 January 12]; Available from: <http://prof.ict.ac.cn/BigDataBench/>.

15. Tilmann Rabl. *Big Data Analytics Benchmark (BigBench)*. [accessed 2015 January 12]; Available from: <http://www.msrg.org/projects/project_view.php?id=12>.

16. AMPLab. *Berkeley Big Data Benchmark: Redshift, Hive, Shark, Impala, Stinger/Tez benchmarks*. [accessed 2015 January 12]; Available from: <https://amplab.cs.berkeley.edu/benchmark/>.

17. *Graph 500 Benchmarks*. [accessed 2015 January 12]; Available from: <http://www.graph500.org/specifications>.

18. *R open source statistical library*. [accessed 2012 December 8]; Available from: <http://www.r-project.org/>.

19. *Apache Mahout Scalable machine learning and data mining* [accessed 2012 August 22]; Available from: <http://mahout.apache.org/>.

20. *Machine Learning Library (MLlib)*. [accessed 2014 April 1]; Available from: <http://spark.apache.org/docs/0.9.0/mllib-guide.html>.

21. *HPC-ABDS Kaleidoscope of over 270 Apache Big Data Stack and HPC Tecnologies*. [accessed 2014 April 8]; Available from: <http://hpc-abds.org/kaleidoscope/>.