Big Data, Simulations and HPC Convergence

Geoffrey Fox, Judy Qiu, Shantenu Jha, Saliya Ekanayake, Supun Kamburugamuve
School of Informatics and Computing ,Indiana University, Bloomington, IN 47408, USA
RADICAL, Rutgers University, Piscataway, NJ 08854, USA

1 Introduction

Two major trends in computing systems are the growth in high performance computing (HPC) with an international exascale initiative, and the big data phenomenon with an accompanying cloud infrastructure of well publicized dramatic and increasing size and sophistication. In studying and linking these trends one needs to consider multiple aspects: hardware, software, applications/algorithms and even broader issues like business model and education. Here we focus on software and applications/algorithms and make comments on the other aspects. We discuss applications/algorithms in section 2, software in section 3 and link them and other aspects in section 4.

2 Applications and Algorithms

We extend the analysis given by us [1,2], which used ideas in earlier parallel computing studies [3-5] to build a set of Big Data application characteristics with 50 features-- called facets -- divided into 4 views. As it incorporated the approach of the Berkeley dwarfs and included features from the NRC Massive Data Analysis Report's Computational Giants, we termed these characteristics as Ogres. Here we generalize approach to integrate Big Data and Simulation applications into a single classification that we call convergence diamonds with a total of 64 facets split between the same 4 views.

The central idea is that any problem -- whether Big Data or Simulation, and whether HPC or cloud-based, can be broken up into Data plus Model. In a Big Data problem, the Data is large and needs to be collected, stored, managed and accessed. Then one uses Data Analytics to compare some Model with this data. The Model could be small such as coordinates of a few clusters or large as in a deep learning network; almost by definition the Data is large! On the other hand for simulations the model is nearly always big -- as in values of fields on a large space-time mesh. The Data could be small and is essentially zero for Quantum Chromodynamics simulations and corresponds to the typically small boundary conditions for many simulations; however climate and weather simulations can absorb large amounts of assimilated data. Remember Big Data has a model, so there are model diamonds for big data – they describe analytics.

Comparing Big Data and simulations is not so clear; however comparing the model in simulations and the model in Big Data is straightforward while the data in both cases can be treated similarly. This simple idea lies at heart of our approach to Big Data - Simulation convergence. In the convergence diamonds given in Table 1, one divides the facets into three types

1) Facet n (without D or M) refers to a facet of system including both data and model -- 16 in total.
2) Facet nD is a Data only facet -- 16 in Total
3) Facet nM is a Model only facet -- 32 in total

The increase in total facets and large number of model facets corresponds mainly to adding Simulation facets to the Processing View of the Diamonds. Note we have included characteristics (facets) present in the Berkeley Dwarfs and NAS Parallel Benchmarks as well the NRC Massive Data Analysis Computational Giants. For some facets there are separate data and model facets. A good example in “Diamond Micropatterns or Execution Features” is that 4D is Data Volume and 4M Model size.

The four views are Problem Architecture (Macro pattern); Execution Features (Micro patterns); Data Source and Style; and finally the Processing (runtime) View. These are respectively mainly System Facets, a mix of system, model and data facets; mainly data facets with the final view entirely model facets. The facets tell us how to compare diamonds (instances of big data and simulation applications) and see which system architectures are needed to support each diamond and which architectures cross multiple diamonds including those from both simulation and big data areas.

Convergence Diamonds		
Facet and View		**Comments**
PA: Problem Architecture View of Diamonds (Meta or MacroPatterns); Nearly all are the system of Data and Model		
1	Pleasingly Parallel	As in BLAST, Protein docking. Includes Local Analytics or Machine Learning – ML or filtering pleasingly parallel, as in bio-imagery, radar images (pleasingly parallel but sophisticated local analytics)
2	Classic MapReduce	Search, Index and Query and Classification algorithms like collaborative filtering.
3	Map-Collective	Iterative maps + communication dominated by “collective”

		operations as in reduction, broadcast, gather, scatter. Common datamining pattern but also seen in simulations
4	Map Point-to-Point	Iterative maps + communication dominated by many small point to point messages as in graph algorithms and simulations
5	Map Streaming	Describes streaming, steering and assimilation problems
6	Shared memory (as opposed to distributed parallel algorithm)	Corresponds to problem where shared memory implementations important. Tend to be dynamic and asynchronous
7	SPMD	Single Program Multiple Data, common parallel programming feature
8	Bulk Synchronous Processing BSP	well-defined compute-communication phases
9	Fusion	Full applications often involves fusion of multiple methods. Only present for composite Diamonds
10	Dataflow	Important application features often occurring in composite Diamonds
11M	Agents	Used in areas like epidemiology (swarm approaches)
12	Orchestration (workflow)	All applications often involve orchestration (workflow) of multiple components
	EF: Diamond Micropatterns or Execution Features	
1	Performance Metrics	Result of Benchmark
2	Flops per Byte (Memory or I/O). Flops per watt (power).	I/O Not needed for “pure in memory” benchmark.
3	Execution Environment	Core libraries needed: matrix-matrix/vector algebra, conjugate gradient, reduction, broadcast; Cloud, HPC, threads, message passing etc. Could include details of machine used for benchmarking here
4D	Data Volume	Property of a Diamond Instance. Benchmark measure
4M	Model Size	

5D	Data Velocity	Associated with streaming facet but value depends on particular problem. Not applicable to model
6D	Data Variety	Most useful for composite Diamonds. Applies separately for model and data
6M	Model Variety	
7	Veracity	Most problems would not discuss but potentially important
8M	Communication Structure	Interconnect requirements; Is communication BSP, Asynchronous, Pub-Sub, Collective, Point to Point? Distribution and Synch
9D	D=Dynamic or S=Static Data	Clear qualitative properties. Importance familiar from parallel computing and important separately for data and model
9M	D=Dynamic or S=Static Model	
10D	R=Regular or I=Irregular Data	
10M	R=Regular or I=Irregular Model	
11M	Iterative or not?	Clear qualitative property of Model. Highlighted by Iterative MapReduce and always present in classic parallel computing
12D	Data Abstraction	e.g. key-value, pixel, graph, vector, bags of words or items. Clear quantitative property although important data abstractions not agreed upon. All should be supported by Programming model and run time
12M	Model Abstraction	e.g. mesh points, finite element, Convolutional Network.
13D	Data in Metric Space or not?	Important property of data.
13M	Model in Metric Space or not?	Often driven by data but model and data can be different here
14M	$O(N^2)$ or $O(N)$ Complexity?	Property of Model algorithm
DSS: Data Source and Style View of Diamonds (No model involvement except in 9)		
1D	SQL/NoSQL/ NewSQL?	Can add NoSQL sub-categories such as key-value, graph, document, column, triple store

2D	Enterprise data model	e.g. warehouses. Property of data model highlighted in database community / industry benchmarks
3D	Files or Objects?	Clear qualitative property of data model where files important in Science; objects in industry
4D	File or Object System	HDFS/Lustre/GPFS. Note HDFS important in Apache stack but not much used in science
5D	Archived or Batched or Streaming	Streaming is incremental update of datasets with new algorithms to achieve real-time response; 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)
	Streaming Category S1)	S1) Set of independent events where precise time sequencing unimportant.
	Streaming Category S2)	S2) Time series of connected small events where time ordering important.
	Streaming Category S3)	S3) Set of independent large events where each event needs parallel processing with time sequencing not critical
	Streaming Category S4)	S4) Set of connected large events where each event needs parallel processing with time sequencing critical.
	Streaming Category S5)	S5) Stream of connected small or large events to be integrated in a complex way.
6D	Shared and/or Dedicated and/or Transient and/or Permanent	Clear qualitative property of data whose importance is not well studied. Other characteristics maybe needed for auxiliary datasets and these could be interdisciplinary, implying nontrivial data movement/replication
7D	Metadata and Provenance	Clear qualitative property but not for kernels as important aspect of data collection process
8D	Internet of Things	Dominant source of commodity data in future. 24 to 50 Billion devices on Internet by 2020
9	HPC Simulations generate Data	Important in science research especially at exascale
10D	Geographic Information Systems	Geographical Information Systems provide attractive access to geospatial data

Pr: Processing (runtime) View of Diamonds		
Big Data and Simulation Processing Kernels		
1M	Micro-benchmarks	Important subset of small kernels
2M	Local Analytics or Informatics or Simulation	Executes on a single core or perhaps node and overlaps Pleasingly Parallel
3M	Global Analytics or Informatics or simulation	Requiring iterative programming models across multiple nodes of a parallel system
12M	Linear Algebra Kernels	Important property of some analytics
	Many important subclasses	Conjugate Gradient, Krylov, Arnoldi iterative subspace methods
		Full Matrix
		Structured and unstructured sparse matrix methods
13M	Graph Algorithms	Clear important class of algorithms – often hard
14M	Visualization	Clearly important aspect of analysis in simulations and big data analyses
15M	Core Libraries	Functions of general value such as Sorting, Math functions, Hashing
Big Data Processing Diamonds		
4M	Base Data Statistics	Describes simple statistical averages needing simple MapReduce in problem architecture
5M	Recommender Engine	Clear type of big data machine learning of especial importance commercially
6M	Data Search/Query/Index	Clear important class of algorithms – especially in commercial applications.
7M	Data Classification	Clear important class of big data algorithms
8M	Learning	Includes deep learning as category
9M	Optimization Methodology	Includes Machine Learning, Nonlinear Optimization, Least Squares, expectation maximization, Dynamic Programming, Linear/Quadratic

		Programming, Combinatorial Optimization
10M	Streaming Data Algorithms	Clear important class of algorithms associated with Internet of Things. Can be called DDDAS Dynamic Data-Driven Application Systems
11M	Data Alignment	Clear important class of algorithms as in BLAST
	Simulation (Exascale) Processing Diamonds	
16M	Iterative PDE Solvers	Jacobi, Gauss Seidel etc.
17M	Multiscale Method?	Multigrid and other variable resolution approaches
18M	Spectral Methods	Fast Fourier Transform
19M	N-body Methods	Fast multipole, Barnes-Hut
20M	Particles and Fields	Particle in Cell
21M	Evolution of Discrete Systems	Electrical Grids, Chips, Biological Systems, Epidemiology. Needs ODE solvers
22M	Nature of Mesh if used	Structured, Unstructured, Adaptive

In several papers we have looked at the model in big data problems and studied the model performance on both cloud and HPC systems. We have shown similarities and differences between models in simulation and big data area. In particular the latter often need HPC hardware and software enhancements to get good performances. There are special features of each class; for example simulations often have local connections between model points corresponding either to the discretization of a differential operator or a short range force. Big data sometimes involve fully connected sets of points and these formulations have similarities to long range force problems in simulation. In both regimes we often see linear algebra kernels but the sparseness structure is rather different. Graph data structures are present in both cases but that in simulations tends to have more structure. The linkage between people in Facebook social network is less structured than the linkage between molecules in a complex biochemistry simulation. However both are graphs with some long range but many short range interactions. Simulations nearly always involve a mix of point to point messaging and collective operations like broadcast, gather, scatter and reduction. Big data problems often are dominated by collectives as opposed to point to point messaging and this motivates the map collective problem architecture facet PA-3 above. In simulations and big data, one sees a similar BSP (loosely synchronous PA-8), SPMD (PA-7) Iterative (EF-11M) and this motivates the Spark [6], Flink [7], Twister [8,9] approach. Note that pleasingly parallel (PA-1) local (Pr-2M) structure is often seen in both simulations and big data.

In [10,11] we introduce Harp as a plug-in to Hadoop with scientific data abstractions, support of iterations and high quality communication primitives. This runs with good performance on several important data analytics including Latent Dirichlet Allocation LDA, clustering and dimension reduction. Note LDA has a non trivial structure sparse structure coming from an underlying bag of words model for documents. In [12], we look at performance in great detail showing excellent data analytics speed up on an Infiniband connected HPC cluster using MPI. Deep Learning [13,14] has clearly shown importance of HPC and uses many ideas originally developed for simulations.

Above we discuss models in the big data and simulation regimes; what about the data? Here we see the issue as perhaps less clear but convergence does not seem difficult technically. Given models can be executed on HPC systems when needed, it appears reasonable to use a different architecture for the data with the big data approach of hosting data on clouds quite attractive. HPC has tended not to use big data management tools but rather to host data on shared file systems like Lustre. We expect this to change with object stores and HDFS approaches gaining popularity in the HPC community. It is not clear if HDFS will run on HPC systems or instead on co-located clouds supporting the rich object, SQL, NoSQL and NewSQL paradigms. This co-location strategy can also work for streaming data with in the traditional Apache Storm-Kafka map streaming model (PA-5) buffering data with Kafka on a cloud and feeding that data to Apache Storm that may need HPC hardware for complex analytics (running on bolts in Storm). In this regard we have introduced HPC enhancements to Storm [15].

3 HPC-ABDS Convergence Software

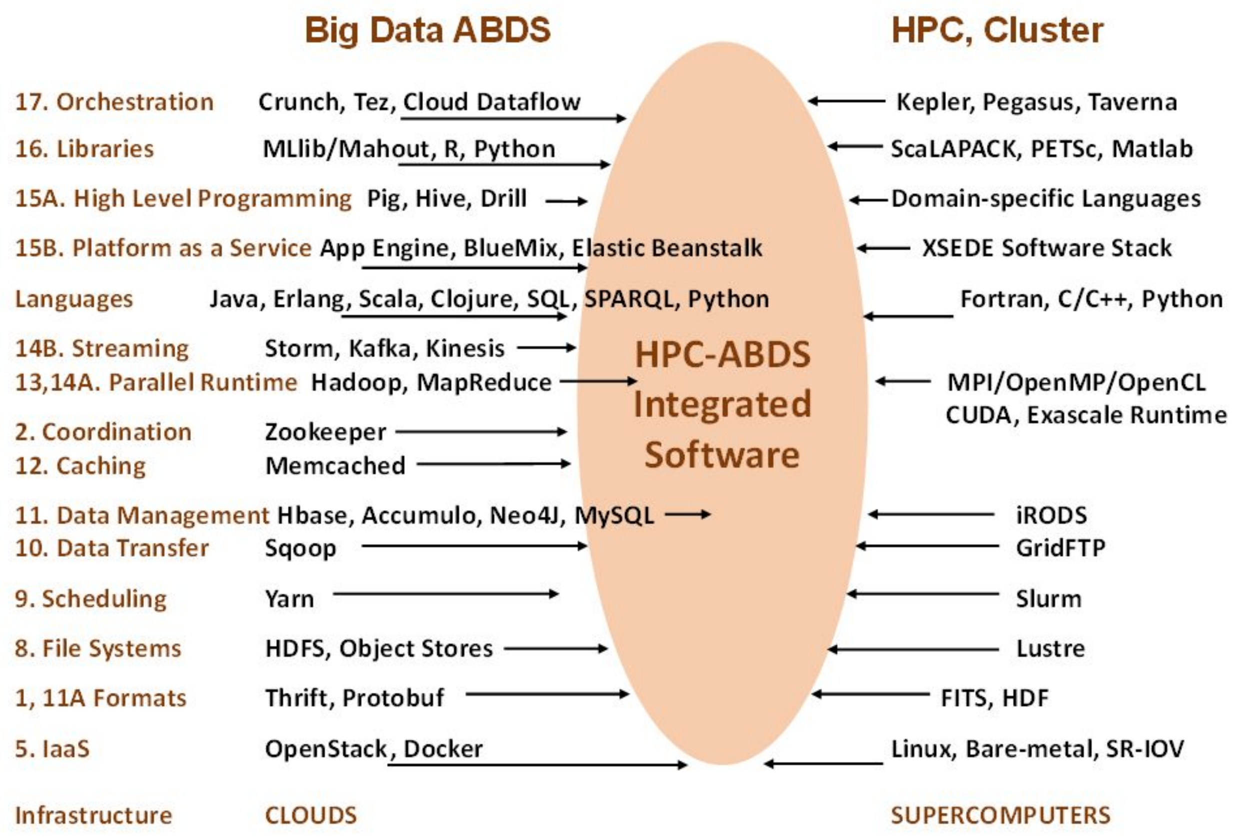
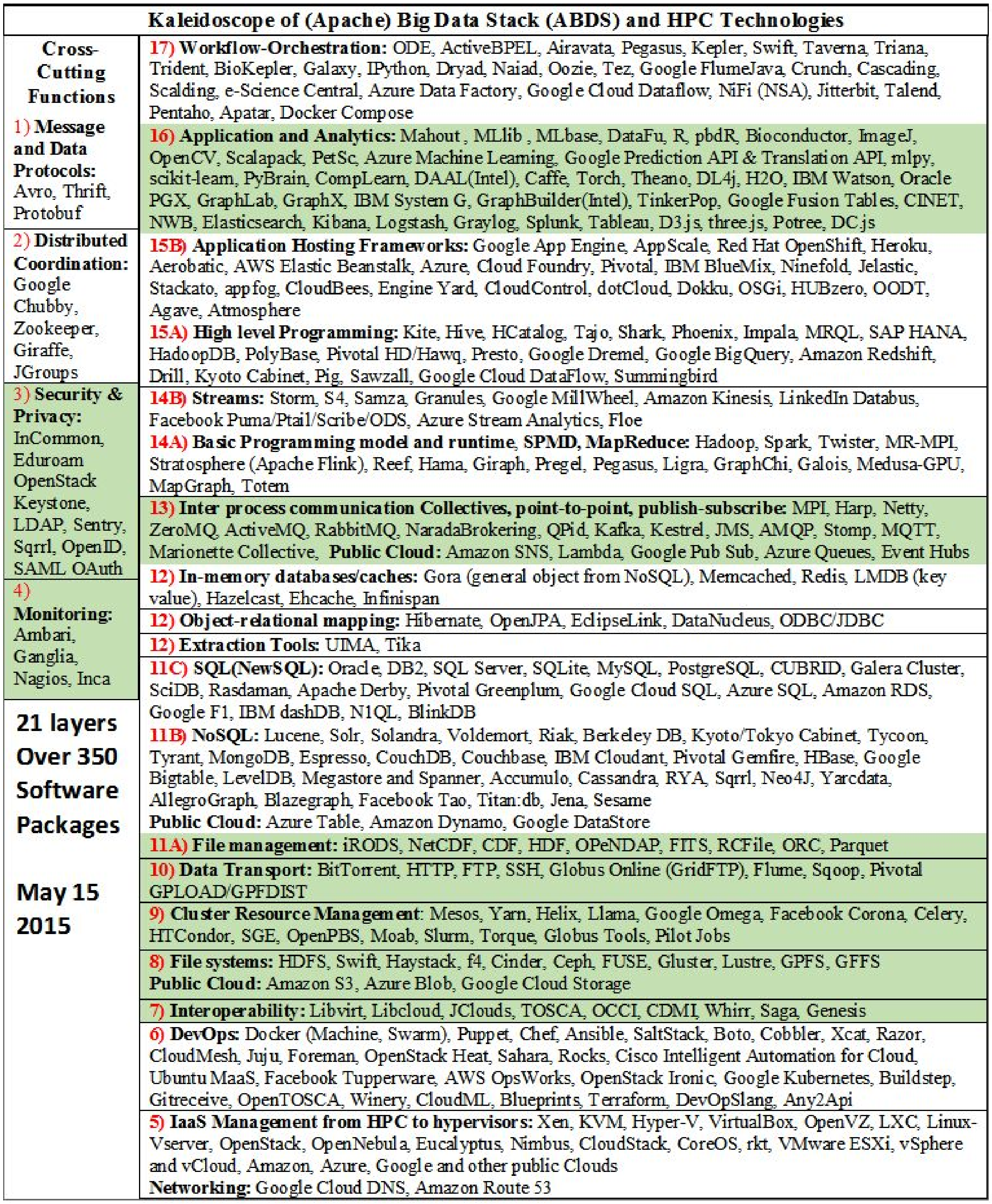


Fig. 1: Comparison of Big Data and HPC Simulation Software Stacks



Kaleidoscope of (Apache) Big Data Stack (ABDS) and HPC Technologies	
Cross-Cutting Functions	**17) 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, Docker Compose
1) **Message and Data Protocols:** Avro, Thrift, Protobuf	**16) 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, H2O, IBM Watson, Oracle PGX, GraphLab, GraphX, IBM System G, GraphBuilder(Intel), TinkerPop, Google Fusion Tables, CINET, NWB, Elasticsearch, Kibana, Logstash, Graylog, Splunk, Tableau, D3.js, three.js, Potree, DC.js
2) **Distributed Coordination:** Google Chubby, Zookeeper, Giraffe, JGroups	**15B) 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
	15A) 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
3) **Security & Privacy:** InCommon, Eduroam OpenStack Keystone, LDAP, Sentry, Sqrrl, OpenID, SAML OAuth	**14B) Streams:** Storm, S4, Samza, Granules, Google MillWheel, Amazon Kinesis, LinkedIn Databus, Facebook Puma/Ptail/Scribe/ODS, Azure Stream Analytics, Floe
	14A) Basic Programming model and runtime, SPMD, MapReduce: Hadoop, Spark, Twister, MR-MPI, Stratosphere (Apache Flink), Reef, Hama, Giraph, Pregel, Pegasus, Ligra, GraphChi, Galois, Medusa-GPU, MapGraph, Totem
	13) Inter process communication Collectives, point-to-point, publish-subscribe: MPI, Harp, Netty, ZeroMQ, ActiveMQ, RabbitMQ, NaradaBrokering, QPid, Kafka, Kestrel, JMS, AMQP, Stomp, MQTT, Marionette Collective, **Public Cloud:** Amazon SNS, Lambda, Google Pub Sub, Azure Queues, Event Hubs
4) **Monitoring:** Ambari, Ganglia, Nagios, Inca	**12) In-memory databases/caches:** Gora (general object from NoSQL), Memcached, Redis, LMDB (key value), Hazelcast, Ehcache, Infinispan
	12) Object-relational mapping: Hibernate, OpenJPA, EclipseLink, DataNucleus, ODBC/JDBC
	12) Extraction Tools: UIMA, Tika
	11C) 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 350 Software Packages	**11B) 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, Blazegraph, Facebook Tao, Titan:db, Jena, Sesame **Public Cloud:** Azure Table, Amazon Dynamo, Google DataStore
	11A) File management: iRODS, NetCDF, CDF, HDF, OPeNDAP, FITS, RCFile, ORC, Parquet
May 15 2015	**10) Data Transport:** BitTorrent, HTTP, FTP, SSH, Globus Online (GridFTP), Flume, Sqoop, Pivotal GPLOAD/GPFDIST
	9) Cluster Resource Management: Mesos, Yarn, Helix, Llama, Google Omega, Facebook Corona, Celery, HTCondor, SGE, OpenPBS, Moab, Slurm, Torque, Globus Tools, Pilot Jobs
	8) File systems: HDFS, Swift, Haystack, f4, Cinder, Ceph, FUSE, Gluster, Lustre, GPFS, GFFS **Public Cloud:** Amazon S3, Azure Blob, Google Cloud Storage
	7) Interoperability: Libvirt, Libcloud, JClouds, TOSCA, OCCI, CDMI, Whirr, Saga, Genesis
	6) DevOps: Docker (Machine, Swarm), Puppet, Chef, Ansible, SaltStack, Boto, Cobbler, Xcat, Razor, CloudMesh, Juju, Foreman, OpenStack Heat, Sahara, Rocks, Cisco Intelligent Automation for Cloud, Ubuntu MaaS, Facebook Tupperware, AWS OpsWorks, OpenStack Ironic, Google Kubernetes, Buildstep, Gitreceive, OpenTOSCA, Winery, CloudML, Blueprints, Terraform, DevOpSlang, Any2Api
	5) IaaS Management from HPC to hypervisors: Xen, KVM, Hyper-V, VirtualBox, OpenVZ, LXC, Linux-Vserver, OpenStack, OpenNebula, Eucalyptus, Nimbus, CloudStack, CoreOS, rkt, VMware ESXi, vSphere and vCloud, Amazon, Azure, Google and other public Clouds **Networking:** Google Cloud DNS, Amazon Route 53

Fig. 2: Big Data and HPC Software subsystems arranged in 21 layers. Green layers have a significant HPC integration.

In previous papers [16-18], we introduced the software stack HPC-ABDS (High Performance Computing enhanced Apache Big Data Stack) shown online [19] and in Figures 1 and 2. These were combined with the big data application analysis [1, 20, 21] in terms of Ogres that motivated the extended convergence diamonds in section 2. We also use Ogres and HPC-ABDS to suggest a systematic approach to benchmarking [2, 22]. In [24] we described the software model of Fig. 2 in detail while further details of the stack can be found in an online course [23] that includes a section with approximately one slide (and associated lecture video) for each entry in Figure 2.

Figure 2 collects together much existing relevant systems software coming from either HPC or commodity sources. The software is broken up into layers so software systems are grouped by functionality. The layers where there is especial opportunity to integrate HPC and ABDS are colored green in figure 2. This is termed HPC-ABDS (High Performance Computing enhanced Apache Big Data Stack) as many critical core components of the commodity stack (such as Spark and Hbase) come from open source projects while HPC is needed to bring performance and other parallel computing capabilities [24]. Note that Apache is the largest but not only source of open source software; we believe that the Apache Foundation is a critical leader in the Big Data open source software movement and use it to designate the full big data software ecosystem. The figure also includes proprietary systems as they illustrate key capabilities and often motivate open source equivalents. We built this picture for big data problems but it also applies to big simulation with caveat that we need to add more high level software at the library level and more high level tools like Global Arrays. This will become clearer in the next section when we discuss Figure 1 in more detail.

The essential idea of our Big Data HPC convergence for software is to use make use of ABDS software where possible as it offers richness in functionality, a compelling open-source community sustainability model and typically attractive user interfaces. ABDS has a good reputation for scale but often does not give good performance. We suggest augmenting ABDS with HPC ideas especially in the green layers of Figure 2. We have illustrated this with Hadoop [10, 11], Storm [15] and the basic Java environment [12]. We suggest using the resultant HPC-ABDS for **both** big data and big simulation applications. In the language of Figure 1, we use the stack on left enhanced by the high performance ideas and libraries of the classic HPC stack on the right. As one example we recommend using enhanced MapReduce (Hadoop, Spark, Flink) for parallel programming for simulations and big data where it's the model (data analytics) that has similar requirements to simulations. We have shown how to integrate HPC technologies into MapReduce to get performance expected in HPC [10] and that on the other hand if the user interface is not critical, one can use a simulation technology (MPI) to drive excellent data analytics performance [12]. A byproduct of these studies is that classic HPC clusters make excellent data analytics engine. One can use the convergence diamonds to quantify this result. These define properties of applications between both data and simulations and allow one to specify hardware and software requirements uniformly over these two classes of applications.

4 Convergence Systems

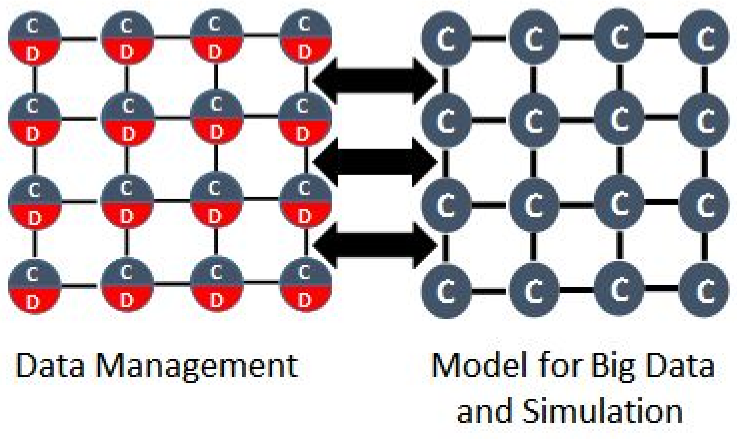


Figure 3: Dual Convergence System

Figure 1 contrasts modern ABDS and HPC stacks illustrating most of the 21 layers and labelling on left with layer number used in Figure 2. The omitted layers in Figure 2 are Interoperability, DevOps, Monitoring and Security (layers 7, 6, 4, 3) which are all important and clearly applicable to both HPC and ABDS. We also add in Figure 1, an extra layer corresponding to programming language, which feature is not discussed in Figure 2. Our suggested approach is to build around the stacks of Figure 1, taking the best approach at each layer which may require merging ideas from ABDS and HPC. This converged stack is still emerging but we have described some features in the previous section. Then this stack would do both big data and big simulation as well the data aspects (store, manage, access) of the data in the “data plus model” framework.Although the stack architecture is uniform it will have different emphases in hardware and software that will be optimized using the convergence diamond facets. In particular the data management will usually have a different optimization from the model computation.

Thus we propose a canonical “dual” system architecture sketched in Figure 3 with data management on the left side and model computation on the right. As drawn the systems are the same size but this of course need not be true. Further we depict data rich nodes on left to support HDFS but that also might not be correct -- maybe both systems are disk rich or maybe we have a classic Lustre style system on the model side to mimic current HPC practice. Finally the systems may in fact be coincident with data management and model computation on the same nodes. The latter is perhaps the canonical big data approach but we see many big data cases where the model will require hardware optimized for performance and with for example high speed internal networking or GPU enhanced nodes. In this case the data may be more effectively handled by a separate cloud like cluster. This depends on properties recorded in the facets of the Convergence Diamonds for application suites. These ideas are built on substantial experimentation but still need significant testing as they have not be looked at systematically.

We suggested using the same software stack for both systems in the dual Convergence system. Now that means that we pick and chose from HPC-ABDS on both machines but we needn't make same choice on both systems; obviously the data management system would stress software in layers 10 and 11 of Figure 2 while the model computation would need libraries (layer 16) and programming plus communication (layers 13-15).

References

1. Geoffrey C.Fox, Shantenu Jha, Judy Qiu, and Andre Luckow, Towards an Understanding of Facets and Exemplars of Big Data Applications, in 20 Years of Beowulf: Workshop to Honor Thomas Sterling's 65th Birthday October 14, 2014. Annapolis http://dx.doi.org/10.1145/2737909.2737912 http://dsc.soic.indiana.edu/publications/OgrePaperv11.pdf
2. Geoffrey C. FOX , Shantenu JHA, Judy QIU, Saliya EKANAYAKE, and Andre LUCKOW, Towards a Comprehensive Set of Big Data Benchmarks. February 15, 2015. http://grids.ucs.indiana.edu/ptliupages/publications/OgreFacetsv9.pdf.
3. 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.
4. NASA Advanced Supercomputing Division. *NAS Parallel Benchmarks*. 1991 [accessed 2014 March 28]; Available from: https://www.nas.nasa.gov/publications/npb.html.
5. Rob F. Van der Wijngaart, Srinivas Sridharan, and Victor W. Lee, *Extending the BT NAS parallel benchmark to exascale computing*, in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 2012, IEEE Computer Society Press. Salt Lake City, Utah. pages. 1-9.
6. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). *Spark: cluster computing with working sets*. Paper presented at the Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, Boston, MA. http://www.cs.berkeley.edu/~matei/papers/2010/hotcloud_spark.pdf
7. Apache Flink open source platform for distributed stream and batch data processing https://flink.apache.org/
8. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., & Fox, G. (2010). *Twister: A Runtime for iterative MapReduce*. Paper presented at the Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference June 20-25, 2010, Chicago, Illinois. http://grids.ucs.indiana.edu/ptliupages/publications/hpdc-camera-ready-submission.pdf
9. Ekanayake, J., S. Pallickara, and G. Fox, *MapReduce for Data Intensive Scientific Analyses,* in Fourth IEEE International Conference on eScience. 2008, IEEE Press.

pages. 277-284. http://grids.ucs.indiana.edu/ptliupages/publications/ekanayake-MapReduce.pdf.

10. Bingjing Zhang, Yang Ruan, Judy Qiu *Harp: Collective Communication on Hadoop* IEEE International Conference on Cloud Engineering (IC2E) conference Tempe AZ March 9-12 2015 http://dsc.soic.indiana.edu/publications/harp9.pdf
11. Bingjing Zhang, Bo Peng, Judy Qiu, *Parallel LDA Through Synchronized Communication Optimizations*, Technical report November 16 2015. http://dsc.soic.indiana.edu/publications/LDA_optimization_paper.pdf
12. Saliya Ekanayake, Supun Kamburugamuve and Geoffrey Fox, "SPIDAL: High Performance Data Analytics with Java and MPI on Large Multicore HPC Clusters", Technical Report January 5 2016 http://dsc.soic.indiana.edu/publications/hpc2016-spidal-high-performance-submit-18-public.pdf
13. Adam Coates, Brody Huval, Tao Wang, David Wu, Bryan Catanzaro, and Andrew Ng. *Deep learning with COTS HPC systems*. in Proceedings of the 30th International Conference on Machine Learning (ICML-13) 2013.
14. Forrest N. Iandola, Khalid Ashraf, Matthew W. Moskewicz, Kurt Keutzer, *FireCaffe: near-linear acceleration of deep neural network training on compute clusters*, January 8 2016 http://arxiv.org/abs/1512.02595
15. Supun Kamburugamuve, Saliya Ekanayake, Milinda Pathirage, Geoffrey Fox, *Towards High Performance Processing of Streaming Data in Large Data Centers*, Technical report January 2016
16. Judy Qiu, Shantenu Jha, Andre Luckow, and Geoffrey C.Fox, Towards HPC-ABDS: An Initial High-Performance Big Data Stack, in Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data. March 18-21, 2014. San Diego Supercomputer Center, San Diego. http://grids.ucs.indiana.edu/ptliupages/publications/nist-hpc-abds.pdf.
17. Geoffrey Fox, Judy Qiu, and Shantenu Jha, High Performance High Functionality Big Data Software Stack, in Big Data and Extreme-scale Computing (BDEC). 2014. Fukuoka, Japan. http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/fox.pdf. 3.
18. 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.
19. HPC-ABDS Kaleidoscope of over 350 Apache Big Data Stack and HPC Tecnologies. [accessed 2016 January 21]; Available from: http://hpc-abds.org/kaleidoscope/.
20. Geoffrey Fox and Wo Chang, Big Data Use Cases and Requirements, in 1st Big Data Interoperability Framework Workshop: Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data March 18 - 21, 2014. San Diego Supercomputer Center, San Diego. http://grids.ucs.indiana.edu/ptliupages/publications/NISTUseCase.pdf.
21. NIST Big Data Use Case & Requirements. 2013 [accessed 2015 March 1]; Available from: http://bigdatawg.nist.gov/V1_output_docs.php.
22. Geoffrey C.Fox, Shantenu Jha, Judy Qiu, and Andre Luckow, Ogres: A Systematic Approach to Big Data Benchmarks, in Big Data and Extreme-scale Computing (BDEC)

January 29-30, 2015. Barcelona. http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/OgreFacets.pdf.

23. Geoffrey Fox. Data Science Curriculum: Indiana University Online Class: Big Data Open Source Software and Projects. 2014 [accessed 2014 December 11]; Available from: http://bigdataopensourceprojects.soic.indiana.edu/.
24. Geoffrey Fox, Judy Qiu, Shantenu Jha, Supun Kamburugamuve and Andre Luckow HPC-ABDS High Performance Computing Enhanced Apache Big Data Stack Invited talk at 2nd International Workshop on Scalable Computing For Real-Time Big Data Applications (SCRAMBL'15) May 4 2015 at.CCGrid2015, the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, held in Shenzhen, Guangdong, China