# Big Data, Simulations and HPC Convergence

Geoffrey Fox[1], Judy Qiu, Shantenu Jha[2], Saliya Ekanayake[1], and Supun Kamburugamuve[1]

[1] School of Informatics and Computing, Indiana University, Bloomington, IN 47408, USA
[2] RADICAL, Rutgers University, Piscataway, NJ 08854, USA

**Abstract.** Two major trends in computing systems are the growth in high performance computing (HPC) with in particular an international exascale initiative, and big data with an accompanying cloud infrastructure of dramatic and increasing size and sophistication. In this paper, we study an approach to convergence for software and applications/algorithms and show what hardware architectures it suggests. We start by dividing applications into data plus model components and classifying each component (whether from Big Data or Big Compute) in the same way. This leads to 64 properties divided into 4 views, which are Problem Architecture (Macro pattern); Execution Features (Micro patterns); Data Source and Style; and finally the Processing (runtime) View. We discuss convergence software built around HPC-ABDS (High Performance Computing enhanced Apache Big Data Stack) and show how one can merge Big Data and HPC (Big Simulation) concepts into a single stack and discuss appropriate hardware.

**Keywords:** Big Data, HPC, Simulations

## 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. There has been substantial discussion of the convergence of big data analytics, simulations and HPC [1, 11–13, 29, 30] highlighted by the Presidential National Strategic Computing Initiative [5]. In studying and linking these trends and their convergence, 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 [18, 21], which used ideas in earlier parallel computing studies [8, 9, 31] 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 [8] and included features from the NRC Massive Data Analysis Reports Computational Giants [27], 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 four views are Problem Architecture (Macro pattern abbreviated PA); Execution Features (Micro patterns abbreviated EF); Data Source and Style (abbreviated DV); and finally the Processing (runtime abbreviated Pr) View.

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. The DDDAS approach is an example where this idea is explicit [3]. 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. The diamonds and their facets are given in a table put in the appendix. They are summarized above in Figure 1.

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 presented in Appendix, 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 Convergence Diamond Micropatterns or Execution Features is that EF-4D is Data Volume and EF-4M Model size.

The views Problem Architecture; Execution Features; Data Source and Style; and Processing (runtime) 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

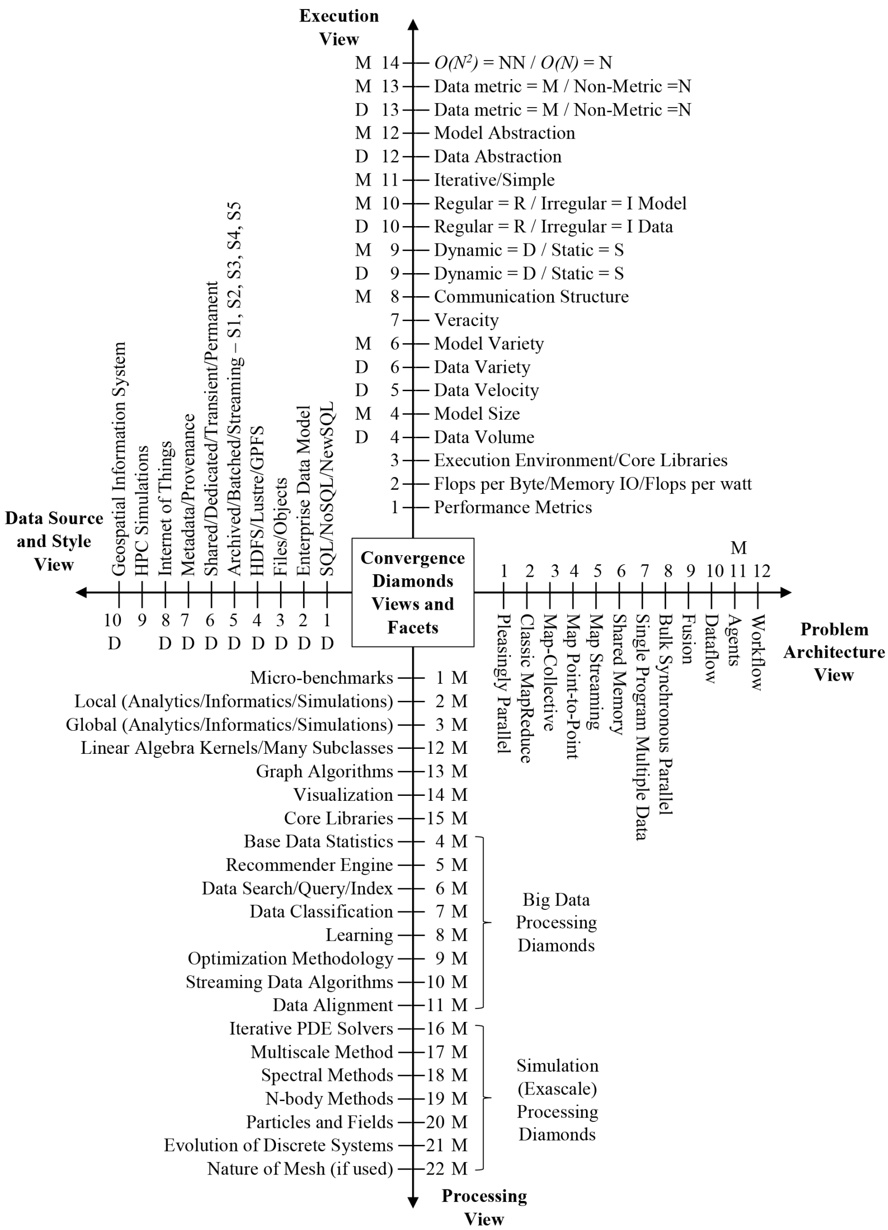


**Fig. 1.** Summary of the 64 facets in the Convergence Diamonds

simulation applications) and see which system architectures are needed to support each diamond and which architectures across multiple diamonds including those from both simulation and big data areas.

In several papers [17, 33, 34] 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 sometimes 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 [32], Flink [7], Twister [15, 16] approach. Note that pleasingly parallel (PA-1) local (Pr-2M) structure is often seen in both simulations and big data.

In [33, 34] 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 [17], we look at performance in great detail showing excellent data analytics speed up on an Infiniband connected HPC cluster using MPI. Deep Learning [14, 24] 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 [26].

We believe there is an immediate need to investigate the overlap of application characteristics and classification from high-end computing and big data ends of the spectrum. Here we have shown how initial work [21] to classify big data applications can be extended to include traditional high-performance applications. Can traditional classifications for high-performance applications [8] be extended in the opposite direction to incorporate big data applications? And if so, is the end result similar, overlapping or very distinct to the preliminary classification proposed here? Such understanding is critical in order to eventually have a common set of benchmark applications and suites [10] that will guide the development of future systems that must have a design point that provides balanced performance.

Note applications are instances of Convergence Diamonds. Each instance will exhibit some but not all of the facets of Fig. 1. We can give an example of the NAS Parallel Benchmark [4] LU (Lower-Upper symmetric Gauss Seidel) using MPI. This would be a diamond with facets PA-4, 7, 8; Pr-3M,16M with its size specified in EF-4M. PA-4 would be replaced by PA-2 if one used (unwisely) MapReduce for this problem. Further if you read initial data from MongoDB, the data facet DV-1D would be added. Many other examples are given in section 3 of [18]. For example non-vector clustering in Table 1 of this section is a nice data analytics example. It exhibits Problem Architecture view PA-3, PA-7, and PA-8; Execution Features EF-9D (Static), EF-10D (Regular), EF-11M (iterative), EF-12M (bag of items), EF-13D (Non-metric), EF-13M(Non metric), and EF-14M(O(N2) algorithm); Processing view Pr-3M, Pr-9M (Machine learning and Expectation maximization), and Pr-12M (Full matrix, Conjugate Gradient).

## 3 HPC-ABDS Convergence Software

In previous papers [20, 25, 28], we introduced the software stack HPC-ABDS (High Performance Computing enhanced Apache Big Data Stack) shown online [4] and in Figures 2 and 3. These were combined with the big data application analysis [6, 19, 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 [18, 22]. In [23] we described the software model of Figure 2 while further details of the stack can be found in an online course [2] that includes a section with about 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

| 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, KeystoneML |
| **1) Message and Data Protocols:** Avro, Thrift, Protobuf | **16) Application and Analytics:** Mahout , MLlib , MLbase, DataFu, R, pbdR, Bioconductor, ImageJ, OpenCV, Scalapack, PetSc, PLASMA MAGMA, 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, Parasol, Dream:Lab, Google Fusion Tables, CINET, NWB, Elasticsearch, Kibana, Logstash, Graylog, Splunk, Tableau, D3.js, three.js, Potree, DC.js, TensorFlow, CNTK |
| **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<br>**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, Neptune, Google MillWheel, Amazon Kinesis, LinkedIn, Twitter Heron, Databus, Facebook Puma/Ptail/Scribe/ODS, Azure Stream Analytics, Floe, Spark Streaming, Flink Streaming, DataTurbine<br>**14A) Basic Programming model and runtime, SPMD, MapReduce:** Hadoop, Spark, Twister, MR-MPI, Stratosphere (Apache Flink), Reef, Disco, Hama, Giraph, Pregel, Pegasus, Ligra, GraphChi, Galois, Medusa-GPU, MapGraph, Totem |
| | **13) Inter process communication Collectives, point-to-point, publish-subscribe:** MPI, HPX-5, Argo BEAST HPX-5 BEAST PULSAR, 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, VoltDB, H-Store |
| | **12) Object-relational mapping:** Hibernate, OpenJPA, EclipseLink, DataNucleus, ODBC/JDBC |
| | **12) Extraction Tools:** UIMA, Tika |
| **21 layers Over 350 Software Packages**<br><br>**January 29 2016** | **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, Spark SQL |
| | **11B) NoSQL:** Lucene, Solr, Solandra, Voldemort, Riak, ZHT, 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, graphdb, Yarcdata, AllegroGraph, Blazegraph, Facebook Tao, Titan:db, Jena, Sesame<br>**Public Cloud:** Azure Table, Amazon Dynamo, Google DataStore |
| | **11A) File management:** iRODS, NetCDF, CDF, HDF, OPeNDAP, FITS, RCFile, ORC, Parquet |
| | **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<br>**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, QEMU, 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<br>**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

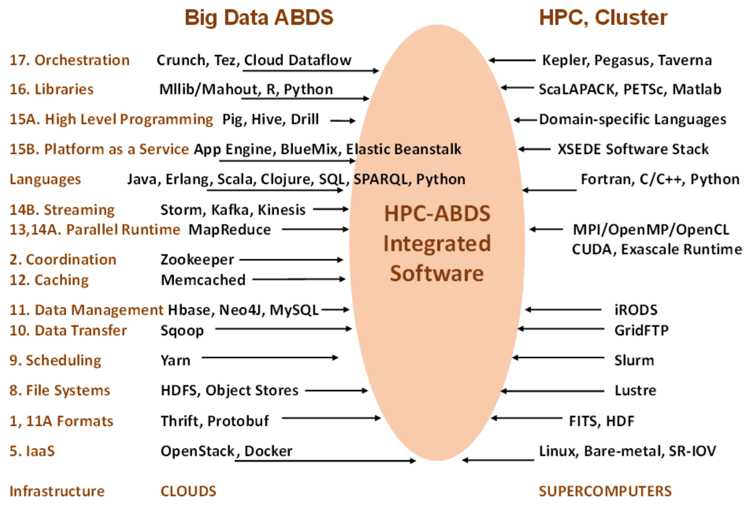


**Fig. 3.** Comparison of Big Data and HPC Simulation Software Stacks

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 [23]. 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 2 in more detail.

The essential idea of our Big Data HPC convergence for software is to 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 [33,34], Storm [26] and the basic Java environment [17]. We suggest using the resultant HPC-ABDS for both big data and big simulation applications. In the language of Figure 2, 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 its 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 [34] 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 [17]. 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

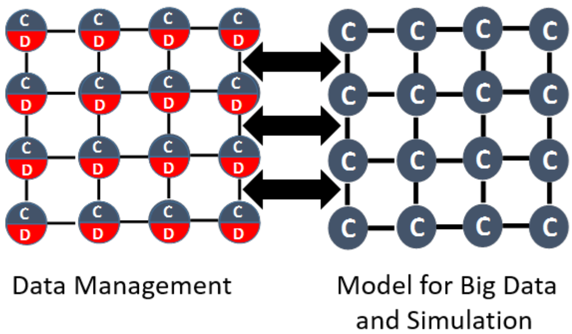


**Fig. 4.** Dual Convergence Architecture

Figure 3 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 3, 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 2, 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 4 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 we pick and chose from HPC-ABDS on both machines but we neednt 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).

**Acknowledgments.** This work was partially supported by NSF CIF21 DIBBS 1443054, NSF OCI 1149432 CAREER. and AFOSR FA9550-13-1-0225 awards. We thank Dennis Gannon for comments on an early draft.

## References

1. Big Data and Extreme-scale Computing (BDEC), http://www.exascale.org/bdec/, Accessed: 2016 Jan 29
2. Data Science Curriculum: Indiana University Online Class: Big Data Open Source Software and Projects. 2014, http://bigdataopensourceprojects.soic.indiana.edu/, accessed Dec 11 2014
3. DDDAS Dynamic Data-Driven Applications System Showcase, http://www.1dddas.org/, Accessed July 22 2015
4. HPC-ABDS Kaleidoscope of over 350 Apache Big Data Stack and HPC Technologies, http://hpc-abds.org/kaleidoscope/
5. NSCI: Executive Order – Creating a National Strategic Computing Initiative, https://www.whitehouse.gov/the-press-office/2015/07/29/executive-order-creating-national-strategic-computing-initiative, July 29 2015
6. NIST Big Data Use Case & Requirements. V1.0 Final Version 2015 (Jan 2016), http://bigdatawg.nist.gov/V1_output_docs.php
7. Apache Software Foundation: Apache Flink open source platform for distributed stream and batch data processing, https://flink.apache.org/, Accessed: Jan 16 2016
8. Asanovic, K., Bodik, R., Catanzaro, B.C., Gebis, J.J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W., et al.: The landscape of parallel computing research: A view from berkeley. Tech. rep., UCB/EECS-2006-183, EECS Department, University of California, Berkeley (2006), Available from http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

9. Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., et al.: The NAS parallel benchmarks. International Journal of High Performance Computing Applications 5(3), 63–73 (1991)
10. Baru, C., Rabl, T.: Tutorial 4 "Big Data Benchmarking" at 2014 IEEE International Conference on Big Data. 2014 [accessed 2015 January 2]
11. Baru, C.: BigData Top100 List, http://www.bigdatatop100.org/, Accessed: 2016 Jan
12. Bryant, R.E.: Data-Intensive Supercomputing: The case for DISC. http://www.cs.cmu.edu/~bryant/pubdir/cmu-cs-07-128.pdf, CMU-CS-07-128 May 10 2007
13. Bryant, R.E.: Supercomputing & Big Data: A Convergence. https://www.nitrd.gov/nitrdgroups/images/5/5e/SC15panel_RandalBryant.pdf, Supercomputing (SC) 15 Panel- Supercomputing and Big Data: From Collision to Convergence Nov 18 2015 - Austin, Texas https://www.nitrd.gov/apps/hecportal/index.php?title=Events#Supercomputing_.28SC.29_15_Panel
14. Coates, A., Huval, B., Wang, T., Wu, D., Catanzaro, B., Andrew, N.: Deep learning with COTS HPC systems. In: Proceedings of the 30th international conference on machine learning. pp. 1337–1345 (2013)
15. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.H., Qiu, J., Fox, G.: Twister: a runtime for iterative mapreduce. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. pp. 810–818. ACM (2010)
16. Ekanayake, J., Pallickara, S., Fox, G.: Mapreduce for data intensive scientific analyses. In: eScience, 2008. eScience'08. IEEE Fourth International Conference on. pp. 277–284. IEEE (2008)
17. Ekanayake, S., Kamburugamuve, S., Fox, G.: SPIDAL: High Performance Data Analytics with Java and MPI on Large Multicore HPC Clusters. http://dsc.soic.indiana.edu/publications/hpc2016-spidal-high-performance-submit-18-public.pdf (Jan 2016), Technical Report
18. Fox, G., Jha, S., Qiu, J., Ekanazake, S., Luckow, A.: Towards a Comprehensive Set of Big Data Benchmarks. Big Data and High Performance Computing 26, 47 (Feb 2015), Available from http://grids.ucs.indiana.edu/ptliupages/publications/OgreFacetsv9.pdf
19. Fox, G., Chang, W.: 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. pp. 18–21 (2014)
20. Fox, G., Qiu, J., Jha, S.: High Performance High Functionality Big Data Software Stack. Big Data and Extreme-scale Computing (BDEC) (2014), Available from http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/fox.pdf
21. Fox, G.C., Jha, S., Qiu, J., Luckow, A.: 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
22. Fox, G.C., Jha, S., Qiu, J., Luckow, A.: Ogres: A Systematic Approach to Big Data Benchmarks. Big Data and Extreme-scale Computing (BDEC) pp. 29–30 (2015)
23. Fox, G.C., Qiu, J., Kamburugamuve, S., Jha, S., Luckow, A.: HPC-ABDS High Performance Computing Enhanced Apache Big Data Stack. In: Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on. pp. 1057–1066. IEEE (2015)

24. Iandola, F.N., Ashraf, K., Moskewicz, M.W., Keutzer, K.: FireCaffe: near-linear acceleration of deep neural network training on compute clusters. arXiv preprint arXiv:1511.00175 (2015)
25. Jha, S., Qiu, J., Luckow, A., Mantha, P., Fox, G.C.: A tale of two data-intensive paradigms: Applications, abstractions, and architectures. In: 2014 IEEE International Congress on Big Data (BigData Congress). pp. 645–652. IEEE (2014)
26. Kamburugamuve, S., Ekanayake, S., Pathirage, M., Fox, G.: Towards High Performance Processing of Streaming Data in Large Data Centers. http://dsc.soic.indiana.edu/publications/high_performance_processing_stream.pdf (2016), Technical Report
27. National Research Council: Frontiers in Massive Data Analysis. The National Academies Press, Washington, DC (2013)
28. Qiu, J., Jha, S., Luckow, A., Fox, G.C.: Towards HPC-ABDS: An Initial High-Performance Big Data Stack. Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data pp. 18–21 (2014), Available from http://grids.ucs.indiana.edu/ptliupages/publications/nist-hpc-abds.pdf
29. Reed, D.A., Dongarra, J.: Exascale computing and big data. Communications of the ACM 58(7), 56–68 (2015)
30. Trader, T.: Toward a Converged Exascale-Big Data Software Stack, http://www.hpcwire.com/2016/01/28/toward-a-converged-software/-stack-for-extreme-scale-computing-and-big-data/, January 28 2016
31. Van der Wijngaart, R.F., Sridharan, S., Lee, V.W.: Extending the BT NAS parallel benchmark to exascale computing. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. p. 94. IEEE Computer Society Press (2012)
32. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. vol. 10, p. 10 (2010)
33. Zhang, B., Peng, B., Qiu, J.: Parallel LDA Through Synchronized Communication Optimizations. http://dsc.soic.indiana.edu/publications/LDA_optimization_paper.pdf (2015), Technical Report
34. Zhang, B., Ruan, Y., Qiu, J.: Harp: Collective communication on hadoop. In: IEEE International Conference on Cloud Engineering (IC2E) conference (2014)

## Appendix: Convergence Diamonds with 64 Facets

These are discussed in Section 2 and summarized in Figure 1

Table 1: Convergence Diamonds and their Facets.

| Facet and View | Comments |
|---|---|
| **PA: Problem Architecture View of Diamonds (Meta or MacroPatterns) Nearly all are the system of Data and Model** | |

Continued on next page

| Facet and View | | Comments |
|---|---|---|
| PA-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) |
| PA-2 | Classic MapReduce | Search, Index and Query and Classification algorithms like collaborative filtering. |
| PA-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 |
| PA-5 | Map Streaming | Describes streaming, steering and assimilation problems |
| PA-6 | Shared memory (as opposed to distributed parallel algorithm) | Corresponds to problem where shared memory implementations important. Tend to be dynamic and asynchronous |
| PA-7 | SPMD | Single Program Multiple Data, common parallel programming feature |
| PA-8 | Bulk Synchronous Processing (BSP) | Well-defined compute-communication phases |
| PA-9 | Fusion | Full applications often involves fusion of multiple methods. Only present for composite Diamonds |
| PA-10 | Dataflow | Important application features often occurring in composite Diamonds |
| PA-11M | Agents | Modelling technique used in areas like epidemiology (swarm approaches) |
| PA-12 | Orchestration (workflow) | All applications often involve orchestration (workflow) of multiple components |
| **EF: Diamond Micropatterns or Execution Features** | | |
| EF-1 | Performance Metrics | Result of Benchmark |
| EF-2 | Flops/byte (Memory or I/O). Flops/watt (power). | I/O Not needed for pure in memory benchmark. |

Continued on next page

| Facet and View | | Comments |
|---|---|---|
| EF-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 |
| EF-4D | Data Volume | Property of a Diamond Instance. Benchmark measure |
| EF-4M | Model Size | |
| EF-5D | Data Velocity | Associated with streaming facet but value depends on particular problem. Not applicable to model |
| EF-6D | Data Variety | Most useful for composite Diamonds. Applies separately for model and data |
| EF-6M | Model Variety | |
| EF-7 | Veracity | Most problems would not discuss but potentially important |
| EF-8M | Communication Structure | Interconnect requirements; Is communication BSP, Asynchronous, Pub-Sub, Collective, Point to Point? Distribution and Synch |
| EF-9D | D=Dynamic or S=Static Data | Clear qualitative properties. Importance familiar from parallel computing and important separately for data and model |
| EF-9M | D=Dynamic or S=Static Model | Clear qualitative properties. Importance familiar from parallel computing and important separately for data and model |
| EF-10D | R=Regular or I=Irregular Data | |
| EF-10M | R=Regular or I=Irregular Model | |
| EF-11M | Iterative or not? | Clear qualitative property of Model. Highlighted by Iterative MapReduce and always present in classic parallel computing |
| EF-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 |
| EF-12M | Model Abstraction | e.g. mesh points, finite element, Convolutional Network. |

Continued on next page

| Facet and View | | Comments |
|---|---|---|
| EF-13D | Data in Metric Space or not? | Important property of data. |
| EF-13M | Model in Metric Space or not? | Often driven by data but model and data can be different here |
| EF-14M | $O(N^2)$ or $O(N)$ Complexity? | Property of Model algorithm |
| **DV: Data Source and Style View of Diamonds (No model involvement except in DV-9)** | | |
| DV-1D | SQL/NoSQL/ NewSQL? | Can add NoSQL sub-categories such as key-value, graph, document, column, triple store |
| DV-2D | Enterprise data model | e.g. warehouses. Property of data model highlighted in database community / industry benchmarks |
| DV-3D | Files or Objects? | Clear qualitative property of data model where files important in Science; objects in industry |
| DV-4D | File or Object System | HDFS/Lustre/GPFS. Note HDFS important in Apache stack but not much used in science |
| DV-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. |

Continued on next page

| Facet and View | | Comments |
|---|---|---|
| | Streaming Category S5) | S5) Stream of connected small or large events to be integrated in a complex way. |
| DV-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 |
| DV-7D | Metadata and Provenance | Clear qualitative property but not for kernels as important aspect of data collection process |
| DV-8D | Internet of Things | Dominant source of commodity data in future. 24 to 50 Billion devices on Internet by 2020 |
| DV-9 | HPC Simulations generate Data | Important in science research especially at exascale |
| DV-10D | Geographic Information Systems | Geographical Information Systems provide attractive access to geospatial data |
| **Pr: Processing (runtime) View of Diamonds Useful for Big data and Big simulation** | | |
| Pr-1M | Micro-benchmarks | Important subset of small kernels |
| Pr-2M | Local Analytics or Informatics or Simulation | Executes on a single core or perhaps node and overlaps Pleasingly Parallel |
| Pr-3M | Global Analytics or Informatics or simulation | Requiring iterative programming models across multiple nodes of a parallel system |
| Pr-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 |
| Pr-13M | Graph Algorithms | Clear important class of algorithms often hard especially in parallel |
| Pr-14M | Visualization | Clearly important aspect of analysis in simulations and big data analyses |
| Pr-15M | Core Libraries | Functions of general value such as Sorting, Math functions, Hashing |
| **Big Data Processing Diamonds** | | |

Continued on next page

| Facet and View | | Comments |
|---|---|---|
| Pr-4M | Base Data Statistics | Describes simple statistical averages needing simple MapReduce in problem architecture |
| Pr-5M | Recommender Engine | Clear type of big data machine learning of especial importance commercially |
| Pr-6M | Data Search/ Query/Index | Clear important class of algorithms especially in commercial applications. |
| Pr-7M | Data Classification | Clear important class of big data algorithms |
| Pr-8M | Learning | Includes deep learning as category |
| Pr-9M | Optimization Methodology | Includes Machine Learning, Nonlinear Optimization, Least Squares, expectation maximization, Dynamic Programming, Linear/Quadratic Programming, Combinatorial Optimization |
| Pr-10M | Streaming Data Algorithms | Clear important class of algorithms associated with Internet of Things. Can be called DDDAS Dynamic Data-Driven Application Systems |
| Pr-11M | Data Alignment | Clear important class of algorithms as in BLAST to align genomic sequences |
| **Simulation (Exascale) Processing Diamonds** | | |
| Pr-16M | Iterative PDE Solvers | Jacobi, Gauss Seidel etc. |
| Pr-17M | Multiscale Method? | Multigrid and other variable resolution approaches |
| Pr-18M | Spectral Methods | Fast Fourier Transform |
| Pr-19M | N-body Methods | Fast multipole, Barnes-Hut |
| Pr-20M | Particles and Fields | Particle in Cell |
| Pr-21M | Evolution of Discrete Systems | Electrical Grids, Chips, Biological Systems, Epidemiology. Needs ODE solvers |
| Pr-22M | Nature of Mesh if used | Structured, Unstructured, Adaptive |