**High Performance Data Analytics on HPC-Cloud**

**Judy Qiu, Indiana University**

The Indiana University Intel Parallel Processing Center (IPCC@IU) is established to address grand challenges in High Performance simulation and data analytics with innovative solutions and software development using Intel architecture. Many real-world problems depend on the ability to analyze and compute on large amounts of data. This analysis often does not scale well; its effectiveness is hampered by the increasing volume, variety and rate of change (velocity) of big data. This project is designing, developing and implementing building blocks that will enable a fundamental improvement in the ability to support data intensive analysis.



In a data-driven world, our work is to support Big Data Machine Learning by

training the “Elephant” (Hadoop) to “run” as fast as a supercomputer.

1. Introduction

Big Data Application Analysis [1] [2] identifies features of data intensive applications that need to be supported in software and represented in benchmarks. This analysis has been extended to support application-driven community software for science. In a data-driven world, Machine Learning (including Deep Learning) can “learn” from training big data through classification, predication and generation. This remarkable capability of converting data to knowledge and insights has great potential. According to recent Mckinsey Global Institute Analysis report [3], Machine Learning has potential impact across industries and use case types, including automotive, manufacturing, consumer, finance, agriculture, energy, health care, pharma-ceuticals, public/social, media, telecom, transport and logistics. Google is remaking itself as a “Machine Learning first” company and so do Facebook and Microsoft rebranding themselves around Artificial Intelligence (AI).

A corresponding trend is that early cloud data centers are evolving with new technologies to better support massive data analytics and machine learning. Programming models and tools are one point of divergence between the scientific computing and big data ecosystems. Analysis of Big Data use cases identifies the need for High Performance Computing (HPC) technologies in the Apache Big Data Stack (HPC-ABDS) [4] [5]. Deep Learning, using GPU clusters, is a clear example. But many Machine Learning algorithms also need iteration [6] [7] [8], high performance communication and other HPC optimizations. A major challenge is to bridge the performance gap between Big Data tools (e.g. Hadoop) and HPC systems (e.g. MPI) for complex data analytics problems (e.g. Global Machine Learning).

Many of the primary software tools used to do the large-scale data analysis were born in the Cloud. Batch-based big data processing frameworks like Apache Hadoop are designed to run on large-scale commodity CPU clusters connected by Ethernet. Hadoop can scale horizontally over a thousand of computer nodes. The design is suitable for pleaslingly parallel computation as depicted in “Map only” and “MapReduce” patterns of Figure 1. Hadoop can also run in a multi-tennants environment and support elasticity that ramps up/down with computation workload. With proper extensions of the mapreduce programming model, it can effectively support iterative computation with “Iterative MapReduce” and model-centric synchronization through Harp collective communication [4]. We’ve expanded the applicability of Hadoop (with Harp plugin) for more classes of big data applications, especially complex data analytics such as machine learning, neural network and graph which we define as “Global Machine Learning”. Modern scale-up servers like Xeon, Xeon Phi and GPU provide substantial processing, memory, and I/O capabilities. We investigate and re-design optimized software stacks to effectively utilize scale-up servers in the Cloud for machine learning and data analytics applications.

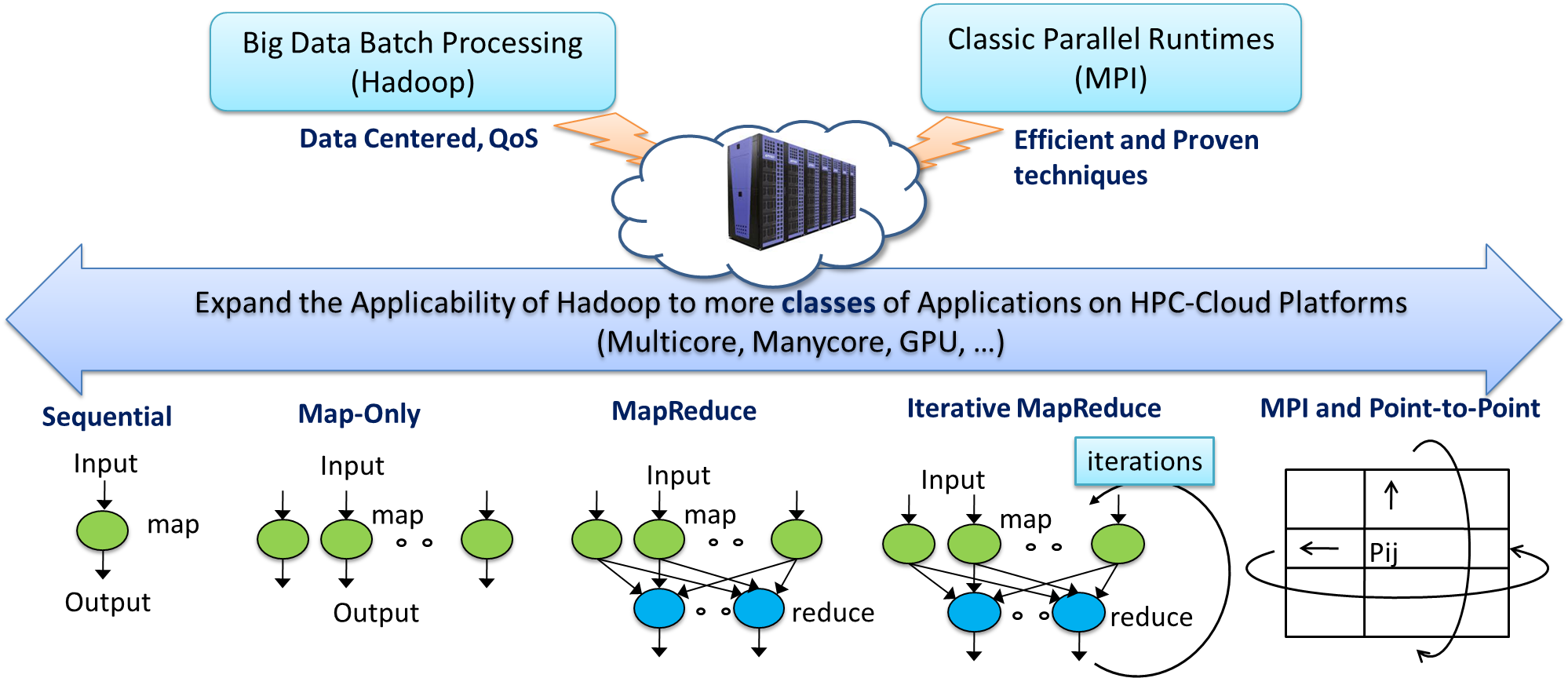


Figure 1 Cloud-HPC interoperable software for High Performance Big Data Analytics at Scale

2. Summary of the Progress for High Performance Data Anlaytics

Our work has concentrated on data processing runtime and data management to support HPC-ABDS [4]. HPC-ABDS is Cloud-HPC interoperable software with performance of HPC (High Performance Computing) and the rich functionality of the commodity Apache Big Data Stack. This is illustrated by our open source software Harp, a plug-in for native Apache Hadoop, which has a convenient science interface, high performance communication and can invoke Intel’s Data Analytics Acceleration Library (DAAL) (ref. Figure 2). [Harp-DAAL](https://dsc-spidal.github.io/harp/docs/harpdaal/harpdaal/) is a high performance data analytics framework that enables big data processing tools like Hadoop to run iterative computation for machine learning and achieve HPC performance. Harp [16] and Harp-DAAL [17] allow our Machine Learning applications to be scalable and interoperable across a range of computing systems including clouds (Azure, Amazon) [6] [7], clusters (Haswell, Knights Landing) [9] [10] [11] [12] [13] and supercomputers (IU Big Red II) [8].

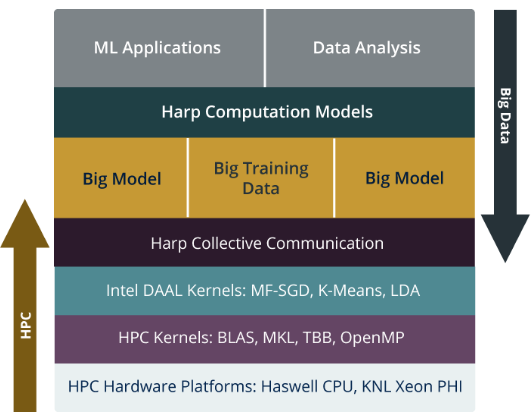


Figure 2. Architecture of Harp-DAAL

We have shown in three PhD thesis research that previous standalone enhanced versions of MapReduce (Twister [3] and Twister4Azure [4]) can be replaced by Harp (a Hadoop plug-in) [5] that offers both data abstractions useful for high performance iterative computation and MPI-quality communication and can drive libraries like DAAL. The three examples listed below show impressive performance, where Hadoop (with Harp and DAAL enhancements) can run as fast as (or faster than) state-of-the-art MPI implementations for Global Machine Learning applications.

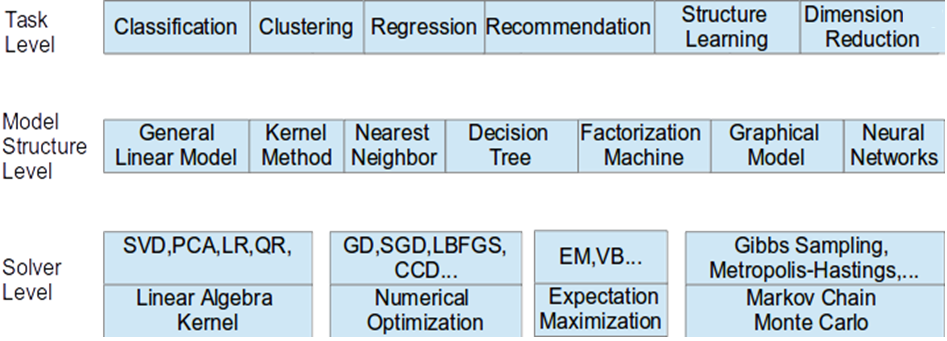
* Harp-DAAL combines advanced communication operations from Harp and high performance computation kernels from DAAL. It achieves 15x to 40x speedups over Spark-Kmeans and 25x to 40x speedups to Spark-ALS on KNL clusters [12].
* HarpLDA+, a Hadoop and Java based Latent Dirichlet Allocation (LDA) training system, outperforms three other MPI/C++ based state-of-the-art systems, which are LightLDA, F+NomadLDA, and WarpLDA. Note that the HarpLDA+ algorithm has a time complexity of O(Kd + Kw) while that of LightLDA is O(1) [14].
* HarpSubgraph, a Hadoop and Java based subgraph mining, achives better performance than the MPI implementation of related work on large Twitter data set (44 million vertices and 1 billion edges) on Haswell clusters.

We are building a scalable parallel Machine Learning library that enables routines in Apache Mahout, MLlib and others built in an NSF funded collaboration [15]. This already has 20 (12 using DAAL) library members and is being tested while we add more functionality in Table 2. Our preliminary results show that Model-Centric parallelism (see Figure 3) extends our understanding of distributed and parallel computation to further advancements in handling Big Model parameters (that cannot fit in memory) and speed of convergence [9] [13]. This finding demonstrates the effectiveness of using HPC machines for Big Data problems.

**3. Taxonomy for Machine Learning Algorithms**

In order to build a general machine learning framework, we have studied different parallel patterns (kernels) of machine learning applications, looking in particular at Gibbs Sampling [5], Stochastic Gradient Descent (SGD) [6], Cyclic Coordinate Descent (CCD) [7] and K-Means clustering. These algorithms are fundamental for large-scale data analysis and cover several important categories: Markov Chain Monte Carlo (MCMC), Gradient Descent and Expectation and Maximization (EM).

Table 1 Taxonomy for Machine Learning Algorithms



Although machine learning algorithms become increasingly common and can easily fit into memory, they require fine-grained parallelism for high performance. Table 1 presents optimization and parallel implementation approaches. We conclude that

* Task level alone can't capture the traits of Computation Model, which is the key for iterative algorithms.
* Data structure such as vectors, matrix, tree, matrices and their sizes are critical for performance optimization.
* Solver has specific computation and communication pattern, which is fundamental for building high performance ML kernel libraries.

**4. Performance Benchmarking**

We focus on modernizing applications to increase parallelism and scalability through optimizations. A subset of machine learning algorithms has been implemented with optimal performance using Hadoop/Harp-DAAL [3]. The code were tested on Intel’s Haswell and Knights Landing clusters at Indiana University. For future work, we will select a subset of machine learning and data analysis algorithms from state-of-the-art libraries and tools and compare them on different platforms (Xeon, Xeon Phi and GPU). This will help to clarify the performance and effectiveness of Harp-DAAL and other existing tools. The benchmark results will be well archived with tutorials developed for community access. In addition, we will create a machine learning library that covers some routines from Apache Maout, Spark, Flink, Tensorflow and other libraries as listed in Table 2. The last 6 algorithms in Table 2 are selected from the NSF DIBBs community applications.

Table 2 Core Machine Learning Algorithms implemented in different frameworks

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | [Harp-DAAL](https://dsc-spidal.github.io/harp/docs/harpdaal/harpdaal/) | [Intel DAAL](https://github.com/01org/daal) | [Mahout](https://mahout.apache.org/users/basics/algorithms.html) | [Spark](https://spark.apache.org/docs/latest/mllib-guide.html) | [Flink](https://ci.apache.org/projects/flink/flink-docs-release-1.2/dev/libs/ml/index.html) | [Turi](https://turi.com/products/create/docs/graphlab.toolkits.html#machine-learning-applications) | [Petuum](http://pmls.readthedocs.io/en/latest/index.html) | [DMTK](https://github.com/Microsoft/DMTK) | [H2O](http://docs.h2o.ai/h2o/latest-stable/index.html) | [ANGEL](https://github.com/Tencent/angel) |
| K-means | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |  | ✓ | ✓ |
| MF-SGD | ✓ |  | ✓ |  |  |  | ✓ |  |  | ✓ |
| Implicit-ALS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |  |  |  |  |
| Neural Network | ✓ | ✓ |  |  |  | ✓ | ✓ |  | ✓ |  |
| PCA | ✓ | ✓ | ✓ | ✓ |  |  |  |  | ✓ |  |
| SVD | ✓ | ✓ | ✓ | ✓ |  |  |  |  |  |  |
| QR | ✓ | ✓ | ✓ |  |  |  |  |  |  |  |
| Covariance | ✓ | ✓ |  |  |  |  |  |  |  |  |
| Linear regression | ✓ | ✓ |  | ✓ | ✓ | ✓ |  |  |  |  |
| Lasso Regression |  |  |  |  |  |  | ✓ |  |  |  |
| logistic Regression |  |  | ✓ | ✓ |  | ✓ | ✓ |  |  | ✓ |
| Ridge Regression | ✓ | ✓ |  |  |  |  |  |  |  |  |
| Moments | ✓ | ✓ |  |  |  |  |  |  |  |  |
| Naive Bayes | ✓ | ✓ | ✓ | ✓ |  |  |  |  | ✓ |  |
| Association Rules | Incoming | ✓ |  | ✓ |  |  |  |  |  |  |
| Decision Forest |  | ✓ |  | ✓ |  | ✓ |  |  |  |  |
| K nearest neighbor | Incoming | ✓ |  |  | ✓ | ✓ |  |  |  |  |
| Outlier Detection |  | ✓ |  |  |  | ✓ |  |  |  |  |
| SVM | Incoming | ✓ |  | ✓ | ✓ | ✓ | ✓ |  |  | ✓ |
| Random Forest |  |  | ✓ | ✓ |  | ✓ | ✓ |  | ✓ |  |
| Distance Metric Learning |  |  |  |  |  |  | ✓ |  |  |  |
| LDA | Incoming |  | ✓ | ✓ |  | ✓ | ✓ | ✓ |  | ✓ |
| GBM (Generalized Linear Modeling) |  |  |  |  |  |  |  | ✓ | ✓ |  |
| Multiverso |  |  |  |  |  |  |  | ✓ |  |  |
| GLM (Generalized Linear Modeling) |  |  |  |  |  |  |  |  | ✓ |  |
| GLRM (Generalized Low Rank Models) |  |  |  |  |  |  |  |  | ✓ |  |
| Stacked Ensembles |  |  |  |  |  |  |  |  | ✓ |  |
| GBDT (Gradient Boosting Decision Tree) |  |  |  |  |  |  |  |  |  | ✓ |
| Multidimentional Scaling (MDS) |  |  |  |  |  |  |  |  |  |  |
| DA-MDS (Deterministic Annealing-MDS) |  |  |  |  |  |  |  |  |  |  |
| Force-Directed Graph Drawing |  |  |  |  |  |  |  |  |  |  |
| Subgraph Mining |  |  |  |  |  |  |  |  |  |  |
| Irregular DAVS Clustering |  |  |  |  |  |  |  |  |  |  |
| DA Semimetric Clustering |  |  |  |  |  |  |  |  |  |  |

We are inspired by the beneficial impact that scientific libraries such as PETSc, MPI and ScaLAPACK have had for supercomputer simulations and hope that our building blocks MIDAS and SPIDAL [15] for high performance data anlaytics will have the same impact on data analytics. Harp and Harp-DAAL will allow our libraries to be scalable and interoperable across a range of computing systems including clouds, clusters and supercomputers.

**5. Approaches of Building A Scalable Big Data Machine Learning Library**

We establish principles for designing parallel machine learning algorithms supporting a variety of model synchronization paradigms. There is a vast amount of literature on distributed machine learning and data analytics, much of it continuing a long tradition of developing special ways to speed up or parallelize individual algorithms or applications. However, specialized implementation rarely leads to wide-spread deployment since it yields no generalization of parallelization techniques. Therefore the focus of our work is to develop a general and exact parallelization technique for a large class of machine learning algorithms. We aim to provide the software building blocks (kernels) that are portable to manycore architectures, as we migrate from the multicore to manycore era.

|  |  |
| --- | --- |
|  |  |
| Figure 3 The Process of Parallel Machine Learning | Figure 4 Computation Models |

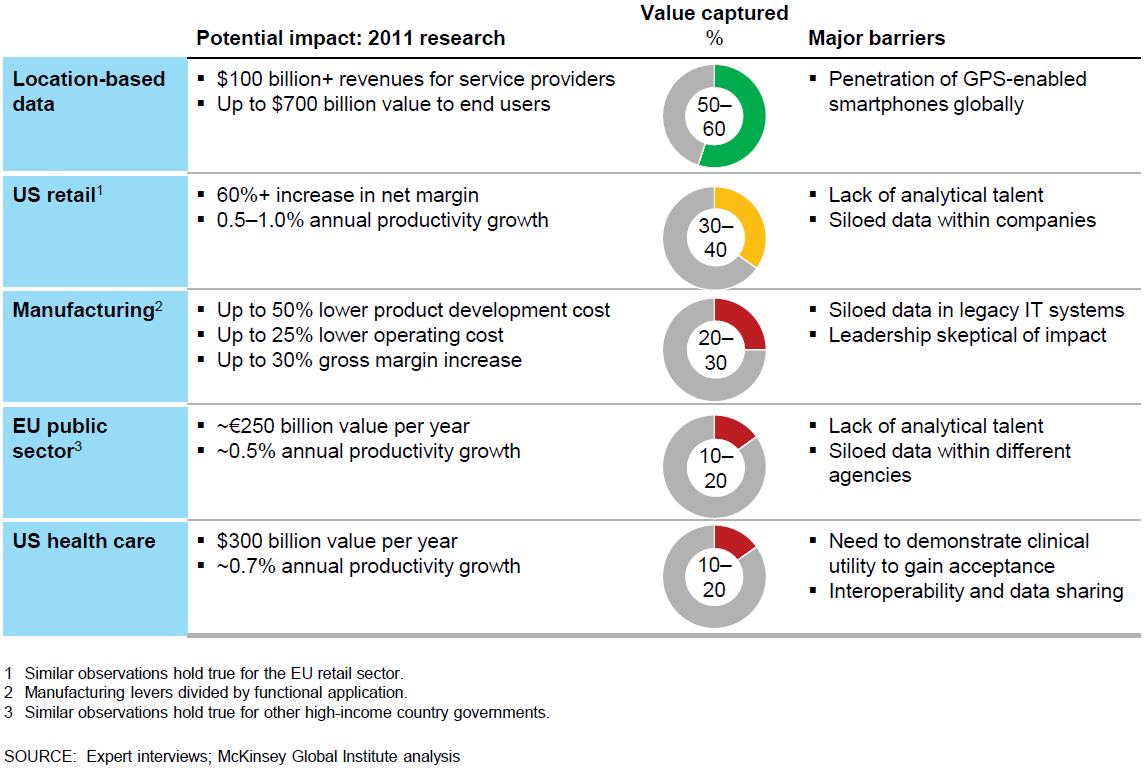
The process for parallelization of machine learning algorithms is shown in Figure 3: the first step is to choose an algorithm for a given big data analysis problem. It may occur that there are multiple solutions to the same problem. An implementation is often optimized for a selected algorithm. Such a tightly coupled cycle (ref. top triangle of Figure 3) works well for a specific application but becomes difficulty to sustain due to diverse choices as well as changes of technology at algorithm, system and hardware levels. This motivates us to investigate the fundamental issue of computation and parallelization abstractions that are effective for a set of domain problems.

We propose a systematic approach with categorizations based on “Computation Model”, which effectively expresses kernel computation characteristics and synchronization or communication mechanisms. The separation of Computation Model, Programming Interface and Implementation details allows us to adapt the variants and make the optimization easier for parallel and distributed machine learning algorithms. Programming interface in particular provides MapReduce style and Harp collective communication APIs to application users. We further categorize parallel machine learning applications into four types of computation models (a) Locking, (b) Rotation, (c) Allreduce, (d) Asynchronous, based on the synchronization patterns and the effectiveness of the model parameter update (see Figure 4).

**6. Impact of Machine Learning and Big Data Analytics Applications**

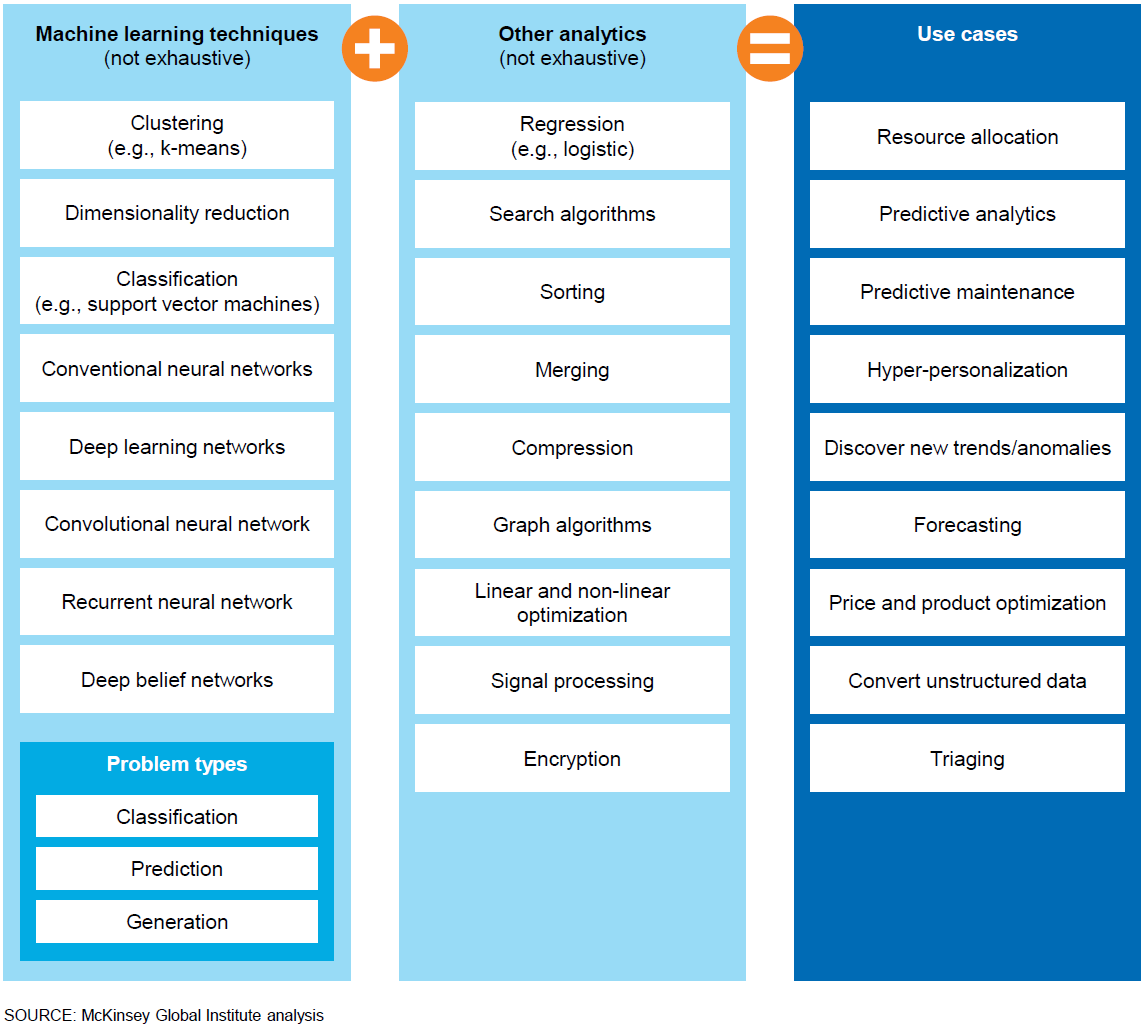
An interesting statement from the Mckinsey Global Institute Analysis suggests that there is tremendeous potential of growth for Data Anlaytics. Table 3 illustrates the value being captured based on their well known predition in 2011, which is still a small fraction of the revenues (of 100~300 billions dollars) in the sectors of location based data, US rail, Manufaturing, EU public sector and US Health care.

Table 3 Potential of Growth for Data and Analytics



Machine Learning can fuel innovation and discovery in all domains. Table 4 shows that when combined with other types of analytics, it can solve a large class of business problems.

Table 4 Machine Learning techniques and their use cases



**7. Training and Engagements**

The Harp framework has been used by over 350 students at Indiana University for their course projects over the past two years. Now it has been released as open source project that is available at public github domain [16]. Harp provides a collection of iterative machine learning and data analysis algorithms that have been tested and benchmarked on Cloud and HPC platforms including Haswell and Knights Landing architectures. Harp is implemented as a Hadoop plugin to make it to run machine learning and complex data applications on heterogeneous HPC-Cloud environments. To lower the barrier for domain scientists and data scientists to use the library, Harp-DAAL may integrate with a high level user interface such as python.

The Apache Software Foundation (ASF) is a community-driven open source organization for hosting projects. To become a successful ASF member, it is important to build a vibrant community around our work. Most projects entering into Apache have previously been available as open source versions in places such as Github. Upon achieving a certain level of community support and publicity they can be introduced under the Apache umbrella for better visibility and a wider pool of contributors. This way when a project joins, it will have a substantial community and a process that it can use to build upon in the future.

Harp project is attracting interest from the development community of HPC and Big Data Ecosystems. The next step is to move it to Apache. Becoming a successful open source project requires strategic partnerships between interested parties. Also a more community-driven development methodology has to be introduced. This means discussing the architectural and design issues in public, lowering the entry requirements for new developers, and making it easier for users to adopt the Harp software. After entering ASF, Harp can follow the Apache Incubation process to become a top-level project or a sub-project under a larger effort like Hadoop, depending on the community requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| **Intel PCC @IU** | | | |
| **Statement of Work – *Year 1*** | | | |
| * **Year 1:** project start date is 9/1/2015 through end date 8/31/2016. * **Success Metrics** for Year 1 is:   + Preparing HPC incubator project with Apache to bring HPC-ABDS to community.   + Continuing support of a postdoc, Langshi Chen, who designs the Harp-DAAL framework and library.   + Hiring a new professional software engineer, who port machine learning algorithms to Xeon, Xeon Phi and new architectures   + Select a subset of machine learning algorithms in Table 2 and benchmark on Xeon, Xoen Phi and GPU hardware   + Select a subset of machine learning algorithms in Table 2 and benchmark and compare with other ML and DL frameworks   + Well documented code, user’s guide, examples, tutorials and benchmark results | | | |
| **Deliverables** | **Timeframe** | **Progress** | **Comments** |
| *Specific actions of work performed* | *Completion Date* | *Complete / In-Progress* | *Significant results (performance improvements), help needed, etc.* |
| Harp-DAAL benchmark on Haswell, KNL and GPU | Year 1, Q1 |  |  |
| Harp-DAAL benchmark and comparison with other ML and DL frameworks | Year 1, Q2 |  |
| Harp-DAAL Machine Learning library with optimial performance, invoking routines from Apache Mahout, MLlib, Tensorflow and other libraries in Q1-Q4 | Year 1, Q3 |  |  |
| Harp-DAAL tuning on single node with advanced memory access techiques in Q1-Q4  Harp-DAAL scaling test on large clusters in Q3-Q4 | Year 1, Q4 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Statement of Work – *Year 2***  ***We understand plans change over the course of the program. Please update our records to show your “year 2” statement of work.*** | | | |
| * **Year 2:** project start date is 9/1/2016 through end date 8/31/2017. * **Success Metrics** for Year 2 is:   + Continue development and benchmarking on Harp-DALL algorithms in Table 2   + Discussing HPC incubator project with Apache to bring HPC-ABDS to community.   + Engage community developers   + Port applications of a community infrastructure through *NSF DIBBs project on middleware and high performance analytics libraries* in collaboration with Rutgers University, Virginia Tech, University of Utah, Arizona State and Stony Brook University   + Explore new application-driven requirements for Harp-DAAL | | | |
| **Deliverables** | **Timeframe** | **Progress** | **Comments** |
| *Specific actions of work performed* | *Completion Date* | *Complete / In-Progress* | *Significant results (performance improvements), help needed, etc.* |
| Produce a high-quality ML library based on Table 2 subset with well documented performance results in Q1 and as a suitable starting system for Apache. | Year 2, Q1 |  |  |
| Complete Table 2 entries with benchmarking and documentation in Q1-Q4 | Year 2, Q2 |  |  |
| Port applications of a community infrastructure for NSF DIBBs project and any applications identified by Intel in Q2-Q4 | Year 2, Q3 |  |  |
| Continue improvement of Harp-DAAL as a framework including high-level user environment as in TensorFlow using Python in Q3-Q4 | Year 2, Q4 |  |  |

Acknowledgements

We gratefully acknowledge generous support from the Intel© Parallel Computing Center (IPCC) grant, NSF OCI-114932 (Career: Programming Environments and Runtime for Data Enabled Science), CIF-DIBBS 143054: Middleware and High Performance Analytics Libraries for Scalable Data Science, and NSF XPS 1439007: Collaborative: Rapid Prototyping HPC Environment for Deep Learning grants.

References

1. Shantenu Jha, Judy Qiu, Andre Luckow, Pradeep Mantha, Geoffrey Fox, A Tale of Two Data-Intensive Paradigms: Applications, Abstractions, and Architectures, Proceedings of the 3rd International Congress on Big Data Conference (IEEE BigData 2014).
2. Geoffrey Fox, Shantenu Jha, Judy Qiu, Andre Luckow, Towards an Understanding of Facets and Exemplars of Big Data Applications, accepted to the Twenty Years of Beowulf Workshop (Beowulf), 2014.
3. The Age of Analytics: Competing in a Data-Driven World, Mckinsey Global Institute Analysis report, December 2016.
4. Judy Qiu, Shantenu Jha, Andre Luckow, Geoffrey Fox, Towards HPC-ABDS: An Initial High-Performance Big Data Stack, accepted to the proceedings of ACM 1st Big Data Interoperability Framework Workshop: Building Robust Big Data ecosystem, NIST special publication, 2014.
5. HPC-ABDS Kaleidoscope of over 350 Apache Big Data Stack and HPC Technologies. Available from: <http://hpc-abds.org/kaleidoscope/>. [accessed 2016 January 20].
6. J. Ekanayake et. al, “Twister: A Runtime for Iterative MapReduce”, in Proceedings of the 1st International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference.
7. T. Gunarathne et. al, “Portable Parallel Programming on Cloud and HPC: Scientific Applications of Twister4Azure”, in Proceedings of 4th IEEE International Conference on Utility and Cloud Computing (UCC 2011).
8. B. Zhang et. al, “Harp: Collective Communication on Hadoop,” in Proceedings of IEEE International Conference on Cloud Engineering (IC2E 2015).
9. [B. Zhang, B. Peng, J. Qiu. Model-Centric Computation Abstractions in Machine Learning Applications](http://ipcc.soic.iu.edu/Computation%20Abstractions.pdf) (BeyondMR of VLDB, 2016)
10. [B. Zhang, B. Peng, J. Qiu. High Performance LDA through Collective Model Communication Optimization](http://ipcc.soic.iu.edu/ICCS-harp-lda.pdf) (ICCS, 2016)
11. [B. Zhang. A Collective Communication Layer for the Software Stack of Big Data Analytics](http://ipcc.soic.iu.edu/A%20Collective%20Communication%20Layer.pdf) (Doctor Symposium in IEEE IC2E, 2016)
12. [L. Chen, B. Peng, B. Zhang, T. Liu, Y. Zou, L. Jiang, R. Henschel, C. Stewart, Z. Zhang, E. Mccallum, T. Zahniser, O. Jon, J. Qiu. Benchmarking Harp-DAAL: High Performance Hadoop on KNL Clusters](http://dsc.soic.indiana.edu/publications/2017CLOUDResearchTrack_12261.pdf) (IEEE Cloud Computing, 2017)
13. [B. Zhang, B. Peng, J. Qiu. Parallelizing Big Data Machine Learning Applications with Model Rotation](http://dsc.soic.indiana.edu/publications/Parallelizing%20Big%20Data%20Machine%20Learning%20Applications%20with%20Model%20Rotation.pdf) (Book Chapter in series Advances in Parallel Computing, IOS Press, 2017)
14. B. Peng, B. Zhang, L. Chen, M. Avram, R. Henschel, C. Stewart, S. Zhu, E. Mccallum, L. Smith, T. Zahniser, O. Jon, J. Qiu. [HarpLDA+: Optimizing Latent Dirichlet Allocation for Parallel Efficiency](http://dsc.soic.indiana.edu/publications/HarpLDA+%20Optimizing%20Latent%20Dirichlet%20Allocation%20for%20Parallel%20Efficiency.pdf), submitted to IEEE International Conference on Big Data (IEEE Big Data 2017)
15. SPIDAL. NSF CIF21 DIBBs project on Middleware and High Performance Data Analytics Libraries for Scalable Data Science

[16] Harp. <https://dsc-spidal.github.io/harp/>

[17] Harp-DAAL. <https://dsc-spidal.github.io/harp/docs/harpdaal/harpdaal/>