Big Data - State of the Art

Sean Chalmers, Cécile Bothorel, Romain Picot Clemente

Big Data - State of the Art

Sean Chalmers
Science & Engineering Faculty
Queensland University of Technology,
Gardens Point, Brisbane, Queensland, Australia.
Email: sean.chalmers@student.qut.edu.au

Lab-STICC UMR CNRS 6285, LUSSI Department
Télécom Bretagne, Institut Mines-Télécom
Brest, France
Email: sean.chalmers@telecom-bretagne.eu

Cécile Bothorel (co-author)
Lab-STICC UMR CNRS 6285, LUSSI Department
Télécom Bretagne, Institut Mines-Télécom
Brest, France
Email: cecil.bothorel@telecom-bretagne.eu

Romain Picot-Clemente (co-author)
Lab-STICC UMR CNRS 6285, LUSSI Department
Télécom Bretagne, Institut Mines-Télécom
Brest, France
Email: romain.picotclemente@telecom-bretagne.eu

***Abstract*—This report[1] is an investigation into the current state of the art with respect to 'Big Data' frameworks and libraries. The primary purpose of this report is to investigate some of the available processing and analytical frameworks and/or libraries, identify some of their strengths and weaknesses through the application of a set of criteria. This criteria can then be used to compare other frameworks, systems, or libraries that are not present here to enable rapid and objective comparison.**

CONTENTS

[1]This work has been supported by the French FUI Project Pay2You Places, "labelled" by three Ples de Compétitivité (French clusters).

I. Initial List & Description of Possible Candidates

What is contained in this section is a description of various big data libraries and frameworks, to give a feel for how they are designed, their focus and abilities and how they may fit into a large scale processing environment. In the subsequent section we will begin to analyse how these various solutions compare to our list of criteria. To date we have examined the following, they are categorised based on their primary purpose.

Processing/Compute:

- Hadoop[1],
- Nvidia CUDA[2],
- Twitter Storm[3],
- Bulk Synchronous Parallel Processing[4],
- GraphLab[5],
- Disk-Based Graph Processing (GraphChi/TurboGraph)

Storage:

- neo4j[6],
- Titan[7],
- HDFS[1]

Analytics:

- MLPACK[8],
- Mahout[16],

A. Hadoop (HDFS)

Hadoop is an open source Apache project that aims to provide "reliable and scalable distributed computing" [9]. The Hadoop package aims to provide all the tools one could need for large scale computing: Fault Tolerant Distributed Storage in the form of the HDFS. A job scheduling and monitoring framework[10], as well as the popular MapReduce[11] Java libraries that allow for scalable distributed parallel processing.

Hadoop has also proven to be extremely popular, as shown by the community and proprietary support available as well as the number organisations that have declared their "Powered By" status on the Apache project page[12]. There are multiple organisations that have made a business out of providing Hadoop implementations and training (Cloudera[13], HortonWorks[14]). Additionally because of its widespread use there is an extremely large amount of free documentation and assistance available[15].

Hadoop has a strong ecosystem of complementary applications and frameworks that have grown up around it, that either simplify, improve, or extend its capabilities. These range from monitoring tools such as Ambari[106](deployment and management of Hadoop clusters), or Zookeeper [107] (synchronisation and configuration management). To storage frameworks like HBase[16], a distributed column-oriented data store. Allowing you to store extremely large tables of data and have support for real-time read/write access to the entire table. As well as inheriting the fault tolerant attributes of HDFS.

The other interesting component is called Mahout[17]. Mahout is a scalable machine learning and statistics library that runs atop the Hadoop framework. According to their website they currently support the following data mining algorithms and techniques:

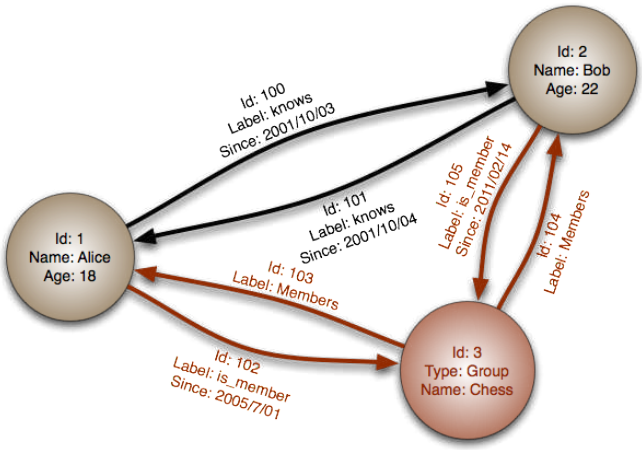
Collaborative Filtering, User and Item based recommenders, K-Means, Fuzzy K-Means clustering, Mean Shift clustering, Dirichlet process clustering, Latent Dirichlet Allocation, Singular value, decomposition, Parallel Frequent Pattern mining, Complementary Naive Bayes classifier, Random forest decision tree based classifier

There is also a very strong community surrounding Mahout that can provide support, insight, and assistance with its use[18]. Mahout is also a very reliable library and can work in both stand alone or distributed application environments.

B. *Neo4j, Titan*

Graph databases are persistent storage mechanisms that forego the table, field, and column structure of traditional relational databases and store all information as nodes or edges. Their entire structure and operation is built on graph theory, and as such they are able to do away with expensive join operations or index lookups.[19]

Data is stored as a property graph, with the nodes representing objects of data, analogous to objects in object-oriented programming, and edges represent the various relationships between them. A straightforward example is represented in the image below taken from the Wikipedia page:



Graph databases have built on the success of NoSQL databases in that nodes and edges do not require a predefined schema for their respective properties, and whilst there are rules to follow for each given implementation when establishing nodes and edges. It is possible to grow a graph database organically based on the changing requirements.

One of the benefits of using a graph database is the significant performance gain for traversing and querying data with a high density of relationships when compared to traditional relational databases[20]. In order to traverse the graph, the database only needs to know the relationship criteria and a starting location. So any query which involves searching through multiple nodes via their relationships is extremely fast.

Graph databases also provide a nice benefit when you consider the advantage of having your data be stored and accessed using the same abstraction that is used to explain and structure it, is considered by many to be a significant advantage[21].

Whilst some graph databases such as Titan, and its underlying structure running Faunus provide some level of MapReduce driven graph analysis[22]. I was not able to locate any analytical libraries or frameworks similar to Mahout that are designed for working with Graph Databases. Given that Fanus runs atop Hadoop however, it may be possible to have access to tools from the Hadoop ecosystem as well.

C. *Bulk Synchronous Parallel Processing (BSP)*

Pregel[4], Apache Giraph[23], Apache Hama[24],[25]

BSP is what is known as a 'bridging model' [26] for the creation of parallel algorithms[27]. Bridging models provide a conceptual link between the underlying physical makeup of the machine(s) and the abstraction that is presented to the programmer[108]. Since communication and synchronisation play a significant role in the success of this algorithm the design accounts for these features being provided by either the hardware or software layer of its implementation.

BSP is designed for systems of processors that are connected via some form of communication network, with each individual processor having private, fast, and local memory. The entire process operates in what is known as "supersteps".

1) Concurrent Computation - Each processor actions a set of computations using only the data available in its local memory.
2) Communication - The processors communicate with each other and communicate the processed data around. This is done using methods analogous to 'PUT' and 'GET' from HTTP, that is to say they are not stateful transactions.
3) Barrier Synchronisation - When this step is reached the cycle cannot proceed to the next step or complete until all processes have marked themselves as complete..

Stages one and two do not necessarily have to be completed in order, processors are free to compute and communicate as required. But the barrier will not let the system continue until all of the communication is complete. This communication is key to the operation of the BSP model as it avoids the problem of data locality. It does not matter if a node is not local to the current processor as the model takes into account the need to communicate the results of computations to different machines/processes.

This process means that each processor will act on its own when performing computations, using local memory only for maximum efficiency. But they will be able to communicate and continue processing using any new information they receive. This makes BSP highly effective at processing large amounts

of highly interrelated information, as the computations can run, share their respective results, and continue or restart their computations based on new information.

This algorithm and associated techniques have been applied to the an internal large scale graph processing system in place at Google called Pregel[81], this is an internal implementation only, although Google did release a paper describing the workings of the application. As well as the Apache Hama application, and the Phoebus[54] implementation.

D. General Purpose Graphics Processing Unit Programming (GPGPU) - Nvidia CUDA

In addition to the processing capabilities of MapReduce, we also have another option for large scale number crunching with GPGPU programming. That is taking operations that would normally be performed on a four core CPU and moving them to, as an example, an Nvidia GeForce GTX 780[28] that contains 2304 cores! Utilising the CUDA[29] API as a bridge between the host and the device, you pass data and functions to the graphics card for it to process, after which you then read the results from specially designed buffers and continue as required.

This power comes with a cost however because the API is written for the C & C++ family of languages. So to acquire the true levels of performance that this API is capable of, you must also write your processing application in C & C++. Additionally not all computational problems transfer easily to a GPU, string processing is one area where GPUs are only recently starting to gain ground[30], and although it is possible, it is not where their strength lies.

Due to their true calling being graphics processing, the cores that are built onto a graphics card are purpose built to process floating point precision numbers at extreme speed. Consequently the processing tasks that GPUs are most adept at, are ones where numerical computation is core point of the algorithm.

But that doesn't tell the whole story, as we must create an entire application to perform this task and it must read the data in somehow, and then process it, and then finally output the results somewhere useful. Being a C/C++ application the complexity is increased substantially when compared to leveraging MapReduce and the comparable Java code therein. However, whilst Java is fast[31]. It is still not as fast as C/C++ [32], nor is that likely to change given their inherent architectural differences[33].

Another option similar to CUDA is the OpenCL framework[34], this 'open computing language' is designed to offer the benefits of programming for multiple GPUs and CPUs without having to rely on any vendor specific libraries. Both Nvidia and ATi support the standard, as do Intel and AMD, thus leveraging a very wide range of hardware whilst only needing to write a single application is certainly possible.

E. Twitter Storm

Twitter Storm[3] is a distributed real-time processing environment that is capable of integrating with any queue management system, programming language, or underlying storage structure[35]. Storm is designed to work with streams of data, or direct requests to an "always-on" computation that returns the most recent result.

The Storm architecture is based around the following abstractions:

1) Tuple- Ordered list of elements, this is the data item abstraction within Storm
2) Stream - Unbound list of Tuples.
3) Spout - Represents the source of a Stream within a Topology.
    a) Spouts can provide Streams of Tuples from:
        i) Queues
        ii) Web Logs
        iii) API Calls
        iv) Event Data

1) Bolts - Process the Streams of Tuples from Spouts to create new Streams
    a) Bolts action the desired parallel processing functionality:
        i) Apply functions/transforms
        ii) Filter
        iii) Aggregation
        iv) Streaming Joins[36]
        v) Access DBs/APIs…etc

1) Topologies - a directed graph of Spouts and Bolts

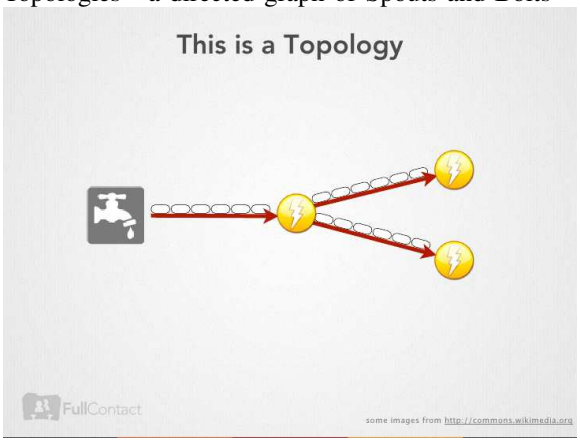


Image Source: Storm: The Real-Time Layer - GlueCon 2012 (Slide 48)

1) Tasks - A process that executes work over Streams and Bolts.

F. MLPACK - Scalable C++ Machine Learning Library

For simplicity we will defer to the MLPACK paper for its introduction:

> "MLPACK is a state-of-the-art, scalable, multi-platform C++ machine learning library released in

> late 2011 offering both a simple, consistent API accessible to novice users and high performance and flexibility to expert users by leveraging modern features of C++. MLPACK provides cutting-edge algorithms whose benchmarks exhibit far better performance than other leading machine learning libraries. MLPACK version 1.0.3, licensed under the LGPL, is available at http://www.mlpack.org."

- Curtin, Ryan R et al. "MLPACK: A scalable C++ machine learning library." Journal of Machine Learning Research 14 (2013): 801-805.

Built to be the machine learning analog to the LAPACK C++ library for linear algebra[37]. Internally it uses the Armadillo[38] Linear Algebra C++ library for it's matrix operations due to its efficiency. MLPACK aims to achieve the following goals for users that are students, experts, or anywhere in between[39]:

- to implement scalable, fast machine learning algorithms,
- to design an intuitive, consistent, and simple API for non-expert users,
- to implement a variety of machine learning methods, and
- to provide cutting-edge machine learning algorithms unavailable elsewhere.

The MLPACK API includes support for the following algorithms:

- nearest/furthest neighbor search with cover trees or kd-trees (k-nearest-neighbors)
- range search with cover trees or kd-trees
- Gaussian mixture models (GMMs)
- hidden Markov models (HMMs)
- LARS / Lasso regression
- k-means clustering
- **fast hierarchical clustering** (Euclidean MST calculation) (March et al., 2010)
- kernel PCA (and regular PCA)
- **local coordinate coding** (Yu et al., 2009)
- sparse coding using dictionary learning
- RADICAL (Robust, Accurate, Direct ICA algorithm) (Learned-Miller and Fisher, 2003)
- **maximum variance unfolding** (MVU) via LRSDP (Burer and Monteiro, 2003)
- the naive Bayes classifier
- **density estimation trees** (Ram and Gray, 2011)

At the time that the MLPACK report (Journal of Machine Learning Research - Volume 14, 2013, pages 801-805) was published it was claimed that no other library included support for the algorithms marked in bold[32].

G. GraphLab - Graph Oriented Machine Learning

GraphLab is a combination of a central C++ API and various data mining and machine learning toolkits. All designed to operate on either a single machine or on top of an existing distributed cluster. GraphLab is designed primarily for machine learning and computations based on graph structures, or processes that have computational or data dependencies that are difficult or impossible to account for on a traditional MapReduce framework.

From the GraphLab Abstraction overview page:

> "Unlike Map-Reduce where computation is applied to independent records, computation in GraphLab is applied to dependent records which are stored as vertices in a large distributed data-graph. Computation in GraphLab is expressed as vertex-programs which are executed in parallel on each vertex and can interact with neighboring vertices. In contrast to the more general message passing and actor models, GraphLab constrains the interaction of vertex-programs to the graph structure enabling a wide range of system optimizations. GraphLab programs interact by directly reading the state of neighboring vertices and by modifying the state of adjacent edges. In addition, vertex-programs can signal neighboring vertex-programs causing them to be rerun at some point in the future."

- The Abstraction (GraphLab Website)[40]

Primarily as the name implies, GraphLab is a C++ API for the investigation and processing of graphs. GraphLab contains the following toolkits:

- Collaborative Filtering
- Clustering
- Computer Vision
- Graphical Models
- Graph Analytics
- Topic Modelling
- Linear Solves

GraphLab is capable of running in a distributed environment as well as locally for experimentation and testing. Depending on how your data is structured, or if you have a particular problem that involves computations on highly interrelated data, then GraphLab may be a suitable alternative.

H. Single Machine High Performance Graph Processing - TurboGraph / GraphChi.

TurboGraph[41] and its counterpart GraphChi[42] are graph processing applications and APIs, respectively, that aim to provide billion/web scale graph processing capabilities on a single machine. These two implementations and others like them use disk-based processing techniques. Temporarily storing carefully structured reference information on a storage device (traditional hard disk or SSD) to aid in computations.

The computation engines implement a 'vertex-centric'[43] model, similar to what is proposed in the Pregel paper from earlier. This technique creates distinctly separated sets of data that can be computed asynchronously.

TurboGraph takes this concept one step further and leverages the asynchronous IO capabilities of FlashSSD hard drives

to speed up processing even further[41]. TurboGraph won't actually run without an SSD being present as it relies on that for its processing capabilities to function.

Both TurboGraph and GraphChi claim impressive performance capabilities:

- TurboGraph: http://wshan.net/turbograph/results.html
- GraphChi: https://github.com/GraphChi/graphchi-cpp#performance

These implementations are useful because they both remove the need for a dedicated cluster of machines to perform computations and other related tasks over massive scale graphs. Although TurboGraph requires the not insignificant investment in a professional grade SSD, this pales in comparison to the cost operating a cluster of machines, especially with respect to the amount of time required.

According to the benchmarks that TurboGraph provide, it is orders of magnitude faster than GraphChi, however the price of this performance is not only the requirement for a SSD, but that TurboGraph is not open source, only runs on a computer that has Windows as its operating system, and only provides a very limited number of processing algorithms (http://wshan.net/turbograph/binaryfiles.html). It is for these reasons that although TurboGraph is powerful and should certainly be investigated if your requirements are met by the algorithms it offers. It will not be included for detailed comparison.

GraphChi, being a direct relative of GraphLab, is an open source API that has both C++ and Java support and its analysis will be included with the GraphLab comparison as it shares many of the same fundamental considerations.

II. Discussion of Candidates

To ensure the various options are analysed in an appropriate manner, we will first define a set of criteria to be used in their evaluation. The starting point for our criteria will be the concept of the "Four 'V's of Big Data" [[44] [45] [46]]; Volume, Velocity, Variety, and Variability.

There are some considerations that will not be included in our criteria:

- Vulnerability / Security - System security represents an entire discipline in and of itself. The sheer extent and complexity of this aspect is beyond the scope of this paper as even the most robust framework or library may have its efforts thwarted by inappropriate memory usage within an application or poor firewall configurations.
- Failure handling - This particular topic is far too subjective to be included in our criteria.
- Availability - This aspect cannot be included as your requirements in regard to it are simply that, "your requirements". It is beyond the scope of this document to consider the many different interpretations of availability and how they may apply to each solution.
- Discussion of Programming Language Suitability / Solution Debugging - Although there is a consideration of the programming languages related to each solution. Efforts have been made to keep this as objective and brief as possible. This topic is simply too deep and too specific for any sensible generic discussion to be included here. Abilities and opinions vary wildly and will not be the same for two given situations.

It is important to state that these are not trivial concerns, however any discussion around these topics is requires explanation of specifics and that is not what this report is about. This elements are often so greatly affected by individual circumstances that their inclusion in this report is considered inappropriate.

For the criteria that we will create, first we will cover what we mean by these terms and then list the questions that will be posed from each topic to create our set of criteria.

A. Volume

This is the consideration of the scale of the dataset that you intend on dealing with and how a particular solution would deal with that scale.

- What scale of data is the solution best suited for?
  - A constant volume?
  - Smaller datasets?
- How much data do you have to process?
- Will your dataset grow?
  - If so, by how much and at what rate?

B. Velocity

When discussing the velocity of data, we're referring to an abstraction of how the resulting data is intended to be consumed, and how fast results can be produced ([39] - Slide 9). Velocity is the consideration of both how fast fresh data is expected to be fed into the system and how quickly the results are expected to be generated.

Below are some examples of what constitutes a high velocity system versus a low velocity. Both aspects of input *and* output are included when determining velocity.

- A solution that may be applicable for financial trading might have a high velocity requirement.
  - High volume input stream -> Expectancy of real-time results
- A solution for crop growth analysis could have a low velocity requirement.
  - Low volume input stream/Batches of data -> Real-time results are not a priority.

C. Variety

This term refers to the range of data types or sources that the solutions supports. The Hadoop platform provides two Java Interfaces "InputFormat"[47] and "OutputFormat"[48] that can be extended to support custom data types, this allows for unlimited variation on how data is handled. Some solutions may require that you conform to a particular file format or data structure before they will function, these solutions could be more difficult to implement because of the additional pre/post-processing steps that are needed to ensure your data conforms to the needs of the solution.

D. Variability

This refers to the breadth of analytics, or configurations of data that are capable within a given solution, and the complexity of the process for implementing it[39]. In addition to be being a representation of the adaptability of the solution to changes in the data or the analytics.

Graph databases can be seen to have a low variability as they are purpose built to store data that has a high density of relationships[49] and as such are extremely efficient and effective when used this way. However their benefits are lost when attempting to store large volumes of simple tabular information, such as user logs or system logs.

E. Context

Making correct decisions with respect to Information Systems (IS) is a complex problem[50], and maintaining an awareness of the problem that needs to be solved is key. The criteria that make up each particular aspect of a big data solution (Volume,Variety,Velocity,Variability) must be weighed against your specific requirements[51].

Selecting a computation solution that is built for real-time analytics and high velocity throughput is inappropriate for a problem that operates on batches of data that are updated once per month. Likewise with choosing an analytical algorithm as each has it's own purpose and properties that must be taken into account[52].

F. Criteria Construction

We will now establish our criteria for comparison of the big data / machine learning solutions. This criteria will be constructed as a series of questions and topics that are, as a best effort, categorised into the above labels.

G. Criteria for Comparison

Volume

- Is there evidence to indicate that this solution can handle the scale of data you require?
    - This is relevant for both scaling up, and scaling down. Some solutions are efficient for extremely large datasets, but due to the stack they are built on perform quite poorly on smaller datasets.

Variety

- Does the solution support or provide a mechanism to support, the format your data is currently in and the format you require for output?
    - Either support for file formats or various forms of data store connectors
- If support is not built in, how complex is it to implement this support?
- If you data structure changes, how complex is it to adjust the solution to take this into account?
    - Do you need to reprocess all of your data if you make a change?

Variability

- Given the current scope of your problem requirements, does this solution provide sufficient coverage?
    - A consideration is if the solution covers 80% of your requirements, how much effort is required to achieve the final 20% and is that effort justifiable[53].
- If the solution does not provide sufficient coverage -
    - Is it possible to alter/update/extend the solution so that it does?
- When considering the possible future scope of your problem (future research, or expanding business analytics), does the solution provide sufficient coverage?

Velocity

- What levels of velocity of data are supported by the solution?
- Does the solution provide support for your requirements? This aspect is where performance related information is the most valuable. Trying to use a batch based system to give you near real-time data is not likely to provide the performance you require.

- Input -
    - Static Dataset
    - Dataset updated via batches at regular or infrequent intervals
    - Near constant/real-time updates

- Output / Results
    - By request. Results are provided only after a request is made, often using information provided in the request. Database queries, for example.
    - By batch. Regular or semi-regular 'job' runs to provide results, usually automated.
    - Near constant/real-time updates. Results constantly updated to match any new data.

H. General Considerations

The following criteria is not exhaustive and is meant as a general guide to complement the above considerations. It is all too easy to become caught up in technical minutiae when examining different big data solutions and forget some key aspects. This list is quite anecdotal and the details are left as an exercise for the reader.

People

- What skills are currently available within the circle of people that will be involved in your project? (Previous experience, programming ability, architectural knowledge, etc)
- It would not be prudent to decide on a solution centred on Phoebus[54], which is written entirely in Erlang, if your team only has strong Java skills.
- If training or recruitment is required, how will this affect your timelines/budget?
- Solutions that use esoteric languages or architectures sometimes make it difficult to locate suitable people, if the need arises.

Infrastructure

- What is the state of the operating environment where you are intending the solution to exist?
- Is it compatible?
- Is that the optimum environment for the solution?
- If it is not the optimum environment, what is the impact of this?
- How is the deployment and management of the infrastructure to be managed?
- There are many organisations dedicated to hosting and maintaining Hadoop installations for example, but these can be expensive. Or they operate on a timeshare basis so your processing must wait until an agreed upon time before it can start.
- Do you have room for experimentation/testing/growth over time?

III. Evaluation of Candidates

A. Hadoop

Category / Criteria *1) Volume:*
Is there evidence to indicate that this solution can handle the scale of data you require? This is taking into account the ability to handle scaling up to massive scale, as well as handling smaller scale issues.
Evaluation
In 2010, Facebook declared they had the largest Hadoop infrastructure in the world; Storing 21PB across 2000 machines in a single HDFS cluster[55]. Hadoop however fails to deal well with small files[56]. The preferred file size is 128MB, due to the architecture of HDFS and how the MapReduce operations are performed. Having multiple smaller files on HDFS can greatly impede the ability of Hadoop to operate efficiently. So if you are dealing with files or data that can broken into 128MB blocks then Hadoop would be able to handle it efficiently and scale as designed.

Category / Criteria *2) Variety:*
Does the solution support or provide a mechanism to support, the format your data is currently in and the format you require for output? If support is not built in, how complex is it to implement this support? If you data structure changes, how complex is it to adjust the solution to take this into account?
Evaluation
Hadoop is written using the Java language, and the constructs it uses to read information are Java interfaces that are designed to be extended to suit individual needs. There is a built in class and interface hierarchy within the MapReduce libraries that assist with the support for new or unique input/output formats[57]. Provided you have access to the right Java skills then the process is allegedly quite straightforward[58],[59]. Based on the previous information it would be a case of adjusting, or creating the required Java classes to ensure the new requirements were met.

Category / Criteria *3) Variability:*
Given the current scope of your problem requirements, does this solution provide sufficient coverage? If the solution does not provide sufficient coverage - Is it possible to alter/update/extend the solution so that it does? When considering the possible future scope of your problem (future research, or expanding business analytics), does the solution provide sufficient coverage?
Evaluation
Hadoop scales up effectively, although it can be demanding when it comes to hardware as it grows due to the space required for HDFS and the memory usage of MapReduce. Works best on text based tabular data inputs in 64/128MB 'chunks'. Reacts poorly to lots of smaller files, that is any file significantly smaller than the lower block size of 64Mb[109]. although with effort it can be adapted to combine or stream this type of data. Data under the 64/128MB threshold will cause significant performance issues. There are solutions to this problem, the simplest being to merge multiple smaller files into successively larger files. The different solutions are suited to different situations and there are various performance and complexity penalties (or benefits!) that are involved[109]. Unsuited to highly interrelated data processing algorithms (graph traversal etc.) without due care, due to the distributed nature of the processing model there is no guarantee of locality of data. There are plenty of tools that remove the requirement for writing Java code every time you need a query run. Saving time and allowing for more experimental investigation of your dataset. An incredible amount of support exists in the open source community. Java is an extremely popular language[60]. Hadoop is also open sourced under the Apache licence so you have access to the source code of MapReduce. There are also plenty of vendors that are dedicated to providing services centred on Hadoop. Extending the platform would be a significant undertaking and whether or not this would be worth the effort is down your specific needs. However the platform already has a high level of support for extensibility and growth. If planning for growth, or you have evidence of impending growth then Hadoop would be a wise choice given its ability to scale, stability, and popularity. However if in that future scope is a change in requirements from tabular data to highly structured or more real-time requirements, Hadoop may not be the best choice.

Category / Criteria *4) Velocity:*
What levels of velocity of data is supported by the solution?
Evaluation
Hadoop is a batch oriented system and whilst it is capable of very high performance, its model is centred on 'Jobs' and scheduled runs[61]. Whether or not this is an advantage depends entirely on your requirements and circumstances. Hadoop is considered a low velocity system because of this. There are systems that are built on top of the Hadoop stack that are considered high velocity systems but they bear little resemblance to Hadoop itself[62].

Category / Criteria *5) People:*
Required Skills/Experience. Recruiting talent/finding knowledge.
Evaluation
The MapReduce component of Hadoop is written in Java, ranked by Tiobe as the number one most popular programming language for 2012. Java knowledge is plentiful and it is a familiar language to many developers. However Hadoop itself is a complex framework and debugging MapReduce applications is often a complicated undertaking, moreso when taking into account that you may have to debug a problem that only occurs when running in a distributed environment.

Category / Criteria *6) Infrastructure:*
How is the deployment and management of the infrastructure to be managed with regard to the chosen solution? Do you have room for experimentation/testing/growth over time?
Evaluation
Hadoop at scale is regarded as a very complex system, it is very low level due to its design and not many people have sufficient experience to properly create and manage a system such as Hadoop[63]. There are organisations that provide access to shared or private Hadoop clusters - Cloudera, Hortonworks, Amazon. However this has to be considered as an additional ongoing cost. Additionally there are costs in time as data must be pushed into the 'cloud'. Depending on your data there may be security issues with utilising an external provider. Hadoop has dozens of ancillary tools as part of its ecosystem that aid anything from machine learning (Mahout), or querying the data (Pig,HBase). A Hadoop cluster when properly managed is able to scale by simply increasing the amount of hardware it has access to. Thereby increasing the number of Hadoop Nodes and the potential processing power. Hadoop will run more than adequately on virtualised hardware - Both Cloudera and Hortonworks provide single node virtual machine images for training and experimentation. So having 'local' nodes for testing and development is feasible and considered good practice.

B. Mahout

Category / Criteria *1) Volume:*
Is there evidence to indicate that this solution can handle the scale of data you require? This is taking into account the ability to handle scaling up to massive scale, as well as handling smaller scale issues.
Evaluation
Mahout scales commensurate with the environment it is running on and the algorithms it has been asked to run[64]. Whilst Mahout does partner very well with Hadoop, the algorithms that are included are highly optimised and are capable of being leveraged from a normal Java project.[65] Any project that would benefit from leveraging an existing, tested, and support library of machine learning algorithms could benefit from the use of Mahout, provided it has an interface to Java.

Category / Criteria *2) Variety:*
Does the solution support or provide a mechanism to support, the format your data is currently in and the format you require for output? If support is not built in, how complex is it to implement this support? If you data structure changes, how complex is it to adjust the solution to take this into account?
Evaluation
To use Mahout with Hadoop there are predetermined file formats and structures that must be adhered to. This may mean that your data has to be modified to suit the input of Mahout prior to any processing being done. The support appears to be required all from the user side as Mahout expects particular formats for its algorithms to function correctly within the Hadoop environment. However as Mahout is built using Java just like Hadoop, the same benefits apply in terms of flexibility and extensibility. Mahout is less affected by changes in data structure as its primary interest is only in certain key data points. Consequently it would only be a change to your data that affects these elements that would require a change to any Mahout implementation.

Category / Criteria *3) Variability:*
Given the current scope of your problem requirements, does this solution provide sufficient coverage? If the solution does not provide sufficient coverage - Is it possible to alter/update/extend the solution so that it does? When considering the possible future scope of your problem (future research, or expanding business analytics), does the solution provide sufficient coverage?
Evaluation
Mahout supports a wide range of algorithms[66] that are applicable to a variety of fields and problems. The source code for Mahout is available online as it is released as open source under the Apache Licence. There is also a ticket managing system for tracking any bugs or issues, and a guide for anyone interested in committing fixes and updates to the codebase[67]. Mahout is an active open source project with a very large user base including corporate, academic, and individual users. A wide array of documentation is available online and there is always scope for expansion of the code base due to its open source nature.

Category / Criteria *4) Velocity:*
What levels of velocity of data is supported by the solution?
Evaluation
The recommender engine within Mahout demonstrates the ability to provided extremely high velocity results at massive scale[68], [69].

Category / Criteria *5) People:*
Required Skills/Experience and Recruiting talent/finding knowledge.
Evaluation
Mahout requires an understanding of the algorithm(s) you intend to leverage. Not possessing this information may result in the wrong information being provided to Mahout, and thus incorrect results. In addition to requiring an intermediate to advanced level of Java knowledge to ensure a smooth and efficient implementation[70].

Category / Criteria *6) Infrastructure:*
How is the deployment and management of the infrastructure to be managed with regard to the chosen solution? Do you have room for experimentation/testing/growth over time?
Evaluation
Mahout is a collection of libraries for the Java language, it will work with or without Hadoop. But apart from the inclusion of the Mahout Java libraries it has not special infrastructure requirements. New algorithms are being added to Mahout by the community. As an open source project there is plenty of scope to provide enhancements and fixes back to the community. Since Mahout does not necessarily require a Hadoop install, it is feasible to set up Mahout locally for experimentation and testing.

C. Graph Databases - (Disk Based - Neo4j, OrientDB), (Distributed - Titan)

Category / Criteria *1) Volume:*
Is there evidence to indicate that this solution can handle the scale of data you require? This is taking into account the ability to handle scaling up to massive scale, as well as handling smaller scale issues.
Evaluation
Both Neo4j[71] and OrientDB[72] have the capacity to operate at massive scale. Titan is a distributed only graph database that runs on either Cassandra[73] or HBase[74] and is built to scale up. Neo4j and OrientDB can also operate at smaller scale, running on single nodes or smaller clusters. Titan can operate in what is called 'single-node' mode, but this is more for testing and experimentation purposes rather than production use as it is distributed by design.

Category / Criteria *2) Variety:*
Does the solution support or provide a mechanism to support, the format your data is currently in and the format you require for output? If support is not built in, how complex is it to implement this support? If you data structure changes, how complex is it to adjust the solution to take this into account?
Evaluation
Both Neo4j and OrientDB provide the following integrations[75]: • Integrated Java API • Remotely via REST protocols • OrientDB provides additional remote access via Binary protocols for C/PHP/NodeJs, and Ruby (coming soon). Titan provides integration via REST and Binary protocols through its tight integration with the following systems: • TinkerPop[76], BluePrint[77], Gremlin[78], Titan Server[79], Rexter[80] Lack of support for the programming language you're project is using can be circumvented using the REST communication protocol. However this method will be slower than the binary integration provided for some languages or the native Java API. OrientDB is open sourced under the Apache 2 licence, so you have access to the source code if you are willing to commit to extend the application but that could be an incredibly complex task. Neo4j provides integration via a REST API as well as direct integration with several programming languages. It is also open source so the same information from OrientDB applies. Titan provides several methods of integration, as well as different access techniques for different languages. Again it is open source so either extending it yourself or working with the community to achieve your goal is possible, but non-trivial. Each storage mechanism provides methods and queries for updating the schema of a database (if you opted to use one). In addition to providing methods for updating properties and other information on individual vertices/edges. The type of change you are wanting to make will be an indicator of the effect it will have on the existing data.

Category / Criteria *3) Variability:*
Given the current scope of your problem requirements, does this solution provide sufficient coverage? If the solution does not provide sufficient coverage - Is it possible to alter/update/extend the solution so that it does? When considering the possible future scope of your problem (future research, or expanding business analytics), does the solution provide sufficient coverage?
Evaluation
These storage mechanisms are only desirable if you are storing data with a high density of relationships. Or where graph abstractions are clearly relevant and useful. These are dedicated solutions to a specific problem space. Although flexible their usefulness does not extend far outside of this space. Given the specific problems these systems are trying to solve, it should be very clear as to whether or not their use is required.

Category / Criteria *4) Velocity:*
What levels of velocity of data is supported by the solution?
Evaluation
Each of these systems are extremely high velocity systems. Out performing traditional relational databases by orders of magnitude when dealing with high density related data.

Category / Criteria *5) People:*
Required Skills/Experience and Recruiting talent/finding knowledge.
Evaluation
Primarily a sound knowledge of graph theory is required to understand how to be best leverage what these systems offer. Each one offer comprehensive documentation and support, either through internal support requests or through their respective communities.

Category / Criteria *6) Infrastructure:*
How is the deployment and management of the infrastructure to be managed with regard to the chosen solution? Do you have room for experimentation/testing/growth over time?
Evaluation
All three solutions provide detailed information on how to complete their various installations. Deploying any of them as a distributed solution is more complex and will involve a far more detailed installation and setup procedure. Each of the solutions offer the ability for running them locally or as 'single-nodes'. Allowing for experimentation with data structures and various forms of query and integration. Each one is an active open source project with an active community so there should always be support for attempting new things.

D. Bulk Synchronous Parallel Processing (BSP)

Category / Criteria *1) Volume:*
Is there evidence to indicate that this solution can handle the scale of data you require? This is taking into account the ability to handle scaling up to massive scale, as well as handling smaller scale issues.
Evaluation
BSP has been demonstrated to scale effectively up to and beyond graphs with one billion vertices[81]. BSP is most likely excessive for any small scale processing given its inherently distributed nature. This is hard to quantify or find evidence for, as the Pregel and BSP designs and implementations are only targeting the large scale solutions.

Category / Criteria *2) Variety:*
Does the solution support or provide a mechanism to support, the format your data is currently in and the format you require for output? If support is not built in, how complex is it to implement this support? If you data structure changes, how complex is it to adjust the solution to take this into account?
Evaluation
Both Hama and Giraph are Apache projects that integrate with the Hadoop technology stack. Consequently they are able to integrate with anything that is supported by the same technology. This gives them very good coverage for any new or existing solution requirements. The Phoebus project, an Erlang implementation, does not yet appear to be production ready[82]. For Hama and Giraph the complexity for additional support is similar to that of Hadoop. Both are open source Java projects, though they do not have the widespread adoption of Hadoop. The severity of the change will determine how much of an impact it causes. These solutions are primarily focused on the large scale processing of data and not as much on the structure of the data itself. Any of the processing code that is affected by a change to the structure of the data will need to be altered.

Category / Criteria *3) Variability:*
Given the current scope of your problem requirements, does this solution provide sufficient coverage? If the solution does not provide sufficient coverage - Is it possible to alter/update/extend the solution so that it does? When considering the possible future scope of your problem (future research, or expanding business analytics), does the solution provide sufficient coverage?
Evaluation
It is hard to define the applicability of the BSP algorithm based processing models as the capability of the BSP model is not restricted to graph processing. Although it is extremely successful in this regard, it's capacity for iterative computations on a massive scale for interrelated data reaches much further. Hama and Giraph are Apache open source projects so contributions by the community are encouraged. However sufficient Java knowledge and an understanding of BSP will be required to make any substantial extensions to these projects. From a processing point of view, these solutions scale to massive proportions in terms of processing and computational ability. Expansion of functionality of these solutions is driven by the community and other interests.

Category / Criteria *4) Velocity:*
What levels of velocity of data is supported by the solution?
Evaluation
BSP and the Pregel design that is based off of it are designed for both low and high velocity systems. For a more detailed description of the initial performance capabilities of Pregel consult the paper, but here are some example benchmarks: "Yoo et al [46] report on a BlueGene/L implementation of breadth-first search (s-t shortest path) on 32,768 PowerPC processors with a high performance torus network, achieving 1.5 seconds for a Poisson distributed random graph with 3.2 billion vertices and 32 billion edges. Bader and Madduri [2] report on a Cray MTA-2 implementation of a similar problem on a 10 node, highly multithreaded system, achieving .43 seconds for a scale-free R-MAT random graph with 134 million vertices and 805 million edges." - [83]

Category / Criteria *5) People:*
Required Skills/Experience and Recruiting talent/finding knowledge.
Evaluation
Knowledge of graph algorithms, and BSP is essential to taking the most advantage of these solutions. Java knowledge is required to implement any changes.

Category / Criteria *6) Infrastructure:*
How is the deployment and management of the infrastructure to be managed with regard to the chosen solution? Do you have room for experimentation/testing/growth over time?
Evaluation
Both Hama and Giraph run as part of a Hadoop install, and thus require Hadoop to be present in order to function. So any infrastructure will be Hadoop first, then these solutions applied after. Each solution has the ability to run in a 'single node' environment for testing purposes. Contributions back into the community would always be appreciated.

E. General Purpose GPU Programming (GPGPU) - Nvidia CUDA, OpenCL

Category / Criteria *1) Volume:*
Is there evidence to indicate that this solution can handle the scale of data you require? This is taking into account the ability to handle scaling up to massive scale, as well as handling smaller scale issues.
Evaluation
A GPU is a hardware component that provides a highly specialised array of cores with dedicated high speed video memory. By utilising the design of the hardware and CUDA, one of the leading APIs, GPGPU processing can scale nearly indefinitely. Additionally there is growing evidence that suggest many of the modern machine learning and processing algorithms perform substantially faster on GPUs[84],[85],[86]. But these are often algorithms that center on numerical computations or possess the property of being "embarrassingly parallel".

Category / Criteria *2) Variety:*
Does the solution support or provide a mechanism to support, the format your data is currently in and the format you require for output? If support is not built in, how complex is it to implement this support? If you data structure changes, how complex is it to adjust the solution to take this into account?
Evaluation
GPGPUs are specialised and suitable for extremely high performance computations, but the method of interaction is only via a programming API. So integration with other solutions or your data must often be done via custom development of a computational pipeline. GPGPUs operate at a very low level and utilising their full potential is not an easy task[87]. The APIs for GPU programming are very low level and require manual memory management on behalf of the developer. Detailed knowledge of the computation requirements of the task at hand, and the capabilities of the GPU are required. Both CUDA and OpenCL are native to C/C++, with some bindings available for other languages; Java through "jcuda". Python through "pycuda". Often GPU programming is done to suit a specific set of requirements. New data structures or requirements may result in new applications being required.

Category / Criteria *3) Variability:*
Given the current scope of your problem requirements, does this solution provide sufficient coverage? If the solution does not provide sufficient coverage - Is it possible to alter/update/extend the solution so that it does? When considering the possible future scope of your problem (future research, or expanding business analytics), does the solution provide sufficient coverage?
Evaluation
GPGPU processing has a wide range of uses and the number of algorithms that benefit from being adopted to running on the GPU is increasing 81. Although care must be taken when considering the use of GPGPU processing due to its complexity and potentially difficult implementation process. GPGPUs provides a means of computation through programming APIs, the only real limits are the time and ability available to implement solutions to any problems that may exist. GPGPUs have a demonstrated ability to scale and through their API can be made to solve nearly any computational problem. Provided the complexity is properly maintained.

Category / Criteria *4) Velocity:*
What levels of velocity of data is supported by the solution?
Evaluation
GPGPU processing is extremely high velocity processing. Given it's low level nature the bottleneck is often other hardware components. An insufficiently powerful CPU or slow PCIe bus speed can be detrimental to performance by limiting the flow of data to and from the device. Poorly allocated or misused memory/cores can also introduce errors or lead to poorly performing solutions. This is due to the nature of how memory and threads are allocated on the GPU. A specific number of threads must be allocated to a block, blocks require a specific allocation of memory. These blocks then operate within a "warp" on the GPU, warps need to be allocated in such a way that there is maximum occupancy of the resources of the GPU.

Category / Criteria *5) People:*
Required Skills/Experience and Recruiting talent/finding knowledge.
Evaluation
"CUDA is hard." - Bryan O'Sullivan[88]. sufficiently detailed knowledge of your problem domain is required, as well as strong C/C++ and GPU programming ability. In order to achieve maximum performance and to ensure that you don't intentionally create inefficient programs there is a very large amount of knowledge that must be absorbed with respect to Nvidia CUDA or OpenCL. These skills can be learned of course, but the complexity of developing solutions using this method should not be underestimated.

Category / Criteria *6) Infrastructure:*
How is the deployment and management of the infrastructure to be managed with regard to the chosen solution? Do you have room for experimentation/testing/growth over time?
Evaluation
There are hundreds of variations of GPUs available, not all have the same computational capabilities. Nvidia's 'Tesla' range of cards that are designed for GPGPU processing support bidirectional overlap. This allows you to stream data onto the card for processing and stream the results from completed computations asynchronously. Most other cards do not support this feature. Given the low level nature of the APIs and the inherent flexibility of the supported languages that provide the most performance (C/C++). GPGPU programming is also a popular area of research[89].

F. Twitter Storm

Category / Criteria *1) Volume:*
Is there evidence to indicate that this solution can handle the scale of data you require? This is taking into account the ability to handle scaling up to massive scale, as well as handling smaller scale issues.
Evaluation
This solution was designed as a real-time processing solution that would complement existing batch based solutions. It has been tuned to achieve extremely high velocities at massive scale. The website for Twitter Storms claims that it is capable of reaching one million, 100 byte messages per second per node[90]. Twitter Storm is designed as a large scale processing application and it's architecture may make it too complex to set up for smaller scale projects.

Category / Criteria *2) Variety:*
Does the solution support or provide a mechanism to support, the format your data is currently in and the format you require for output? If support is not built in, how complex is it to implement this support? If you data structure changes, how complex is it to adjust the solution to take this into account?
Evaluation
Storm claims it is capable of integration and communication with any programming language. Using either a "Thrift definition" to types and structures between languages, or alternately communication via a JSON interface on stdin/stdout[91]. It appears that all integration would on some level be required to be custom built. The support for a wide range of programming languages and different platforms is an indication that although you must construct the integration with Twitter Storm, there are many options available to suit a wide variety of skills. Largely meaning the complexity of integration is dependant on the system it is being integrated with and the skills of those performing the task. Twitter Storm operates on 'Tuples' of data, that is a named list of values. It supports any data structure or type as its job is to pass the values through the Streams to your Bolts where the processing occurs.

Category / Criteria *3) Variability:*
Given the current scope of your problem requirements, does this solution provide sufficient coverage? If the solution does not provide sufficient coverage - Is it possible to alter/update/extend the solution so that it does? When considering the possible future scope of your problem (future research, or expanding business analytics), does the solution provide sufficient coverage?
Evaluation
This solution is designed for unbounded data, that is data with no conceivable or known upper limit, and the requirement for near real-time result access. So it is suited for projects where there is a constant stream of near or refreshed data that requires computation. Or when this data flow is to be subject to large numbers of queries or requests from external systems[93]. Such as a recommendation engine that uses several streams of data from different input sources to provide the latest possible result. Any integration with Twitter Storm is almost guaranteed to be custom so any coverage is expected to be developed. Extension and expansion of the capabilities of a Twitter Storm implementation would be equivalent to any large scale application.

Category / Criteria *4) Velocity:*
What levels of velocity of data is supported by the solution?
Evaluation
Twitter Storm is designed as an extremely high velocity system. Responding to requests for results in near real-time. Depending on the computations that are being performed Twitter Storm is capable of responding fast enough to handle requests from a web service or similar API.

Category / Criteria *5) People:*
Required Skills/Experience. Recruiting talent/finding knowledge.
Evaluation
Twitter Storm is a Java application that requires several other libraries and systems to operate properly. These range from the communication and message passing libraries (ZeroMQ[112]) to the queuing and monitoring libraries that are used for coordination of the different bolts and streams, such as Kestrel[114] or RabbitMQ[113]. As well as Apache Zookeeper for the distributed coordination of tasks. Additionally if your system is not built using Java you must be aware of or sufficiently skilled to learn how to integrate your language to the Twitter Storm API.

Category / Criteria *6) Infrastructure:*
How is the deployment and management of the infrastructure to be managed with regard to the chosen solution? Do you have room for experimentation/testing/growth over time?
Evaluation
Twitter Storm is hardware agnostic and doesn't require any specific hardware configuration to operate beyond a distributed cluster of machines. Twitter Storm provides functionality to run as a single-node, or as any size cluster you configure. Additionally it is an open source project so community involvement is encouraged with fixing problems, improving documentation, or creating new functionality.

Category / Criteria *2) Variety:*
Does the solution support or provide a mechanism to support, the format your data is currently in and the format you require for output? If support is not built in, how complex is it to implement this support? If you data structure changes, how complex is it to adjust the solution to take this into account?
Evaluation
As it is a programming library for C++ it is up to you to provide the integration with your other systems. MLPACK is not a standalone machine learning solution, it is a library that provides functionality for others to implement in their application.. Any integration will require C++ programming knowledge in order to create or implement the required support. This depends largely on how MLPACK is integrated into your solution and what data is required by MLPACK. Implementing additional algorithms could be complex depending on your integration and the algorithm you're trying to implement, but it is highly subjective.

G. MLPACK - Scalable C++ Machine Learning Library

Category / Criteria *1) Volume:*
Is there evidence to indicate that this solution can handle the scale of data you require? This is taking into account the ability to handle scaling up to massive scale, as well as handling smaller scale issues.
Evaluation
MLPACK is a scalable and efficient machine learning library. It provides abstractions over data structures and algorithms that simplify the creation of larger data mining applications. Additionally the MLPACK libraries are capable of operating on datasets of any size as it simply a library of efficient data structures and pre-built algorithms[94]. Internally, not all of the classes with MLPACK are considered thread-safe, so their inclusion in large scale systems must be done so with care. A more likely use case for MLPACK is to create the standalone applications that are to be distributed, allowing for the creation of tightly packed, efficient applications designed to operate on subsets of the greater dataset.

Category / Criteria *3) Variability:*
Given the current scope of your problem requirements, does this solution provide sufficient coverage? If the solution does not provide sufficient coverage - Is it possible to alter/update/extend the solution so that it does? When considering the possible future scope of your problem (future research, or expanding business analytics), does the solution provide sufficient coverage?
Evaluation
MLPACK contains a comprehensive list of machine learning algorithms[95], as well as support for efficient matrix manipulations[96]. With the caveat that for the sake of efficiency all matrices are stored in column major format, this is contrary to normal operations and may complicate some integrations[92]. MLPACK is open source, and all of the underlying source code is available online. So modifications and extensions to the library are limited only by time and means. As MLPACK is open source software there is plenty of support for those that want to become involved in extending and improving the library. Either with fixing bugs or adding algorithms.

Category / Criteria *4) Velocity:*
What levels of velocity of data is supported by the solution?
Evaluation
The paper on MLPACK provides benchmarks based on computing the Tree-based, k-nearest-neighbours search and classifier algorithm on trees from 1,000 x 10 nodes, to 1,000,000 x 10. The compute times for MLPACK were 0.078sec and 1423.949sec respectively. The closest contender was "sklean" which completed these two benchmarks in 0.341sec and 1550.72sec. The other libraries failed to complete the benchmarks at the higher amount of nodes, or took over 9000 seconds to complete. With this sort of efficiency it is reasonable to assume that the velocity of the library is largely dependant on the nature of its implementation and which algorithms are in use. MLPACK is suited for both high and low velocity environments.

Category / Criteria *5) People:*
Required Skills/Experience and Recruiting talent/finding knowledge.
Evaluation
A reasonable level of C++ knowledge is required to ensure the efficiency of the library is properly utilised. A strong knowledge of the algorithms that it implements is not as vital a requirement, but should not be ignored.

Category / Criteria *6) Infrastructure:*
The infrastructure requirements for MLPACK are entirely dependent on the implementation and how it is used. Suffice to say that when targeting large or massive scale datasets you will need hardware that is fitting for such requirements as MLPACK is not likely the only component of such a system.

H. GraphLab / GraphChi

Category / Criteria *1) Volume:*
Is there evidence to indicate that this solution can handle the scale of data you require? This is taking into account the ability to handle scaling up to massive scale, as well as handling smaller scale issues.
Evaluation
It is difficult to locate suitable benchmarks that demonstrate the ability for GraphLab to scale. However GraphLab claims to be able to scale from a laptop to a Hadoop cluster (and they claim complete compatibility with the Hadoop running environment)[98]. GraphChi runs only on a single machine and is capable of handling both small and massive scale graphs.

Category / Criteria *2) Variety:*
Does the solution support or provide a mechanism to support, the format your data is currently in and the format you require for output? If support is not built in, how complex is it to implement this support? If you data structure changes, how complex is it to adjust the solution to take this into account?
Evaluation
GraphLab mainly supports a specific list of file formats[99]. It can also support any arbitrary user data provided it complies with the GraphLab "Serializable" structures 96. Additionally GraphLab supports the HDFS file system and thus is compatible with existing Hadoop installations. Potentially allowing for integration with larger distributed applications[100]. GraphChi requires that data be provided in C structs that utilise properties that are of a constant size[101]. GraphChi requires you to construct the interface between your data and the processing component. The specific file formats themselves are contained within in-built libraries, however the source code is available should any modifications need to be made. Given GraphLab's support for using any arbitrary user data structures it does not appear likely that this would prove to be an issue. This depends on the scope of your implementation.

Category / Criteria *3) Variability:*
Given the current scope of your problem requirements, does this solution provide sufficient coverage? If the solution does not provide sufficient coverage - Is it possible to alter/update/extend the solution so that it does? When considering the possible future scope of your problem (future research, or expanding business analytics), does the solution provide sufficient coverage?
Evaluation
GraphLab is capable of solving extremely complex graph processing problems at high speed. It is capable of running in single node, or distributed modes. However it's design is batch oriented and thus may not be suitable for certain requirements. GraphLab must be implemented as part of an application so any scheduling or other long running aspects must be custom built. As GraphLab is an API, it is capable of becoming part of any larger application that can leverage C++, thus greatly increasing the extensibility and problems that can be solved. As requirements change, the application that contains the GraphLab implementation will grow and change like any other application. Only experimentation and testing with respect to the problem at hand will demonstrate if GraphLab is capable of supporting your requirements.

Category / Criteria *4) Velocity:*
What levels of velocity of data is supported by the solution?
Evaluation
It is difficult to find conclusive benchmarks for GraphLab, however this is not entirely unexpected as the performance of GraphLab can be so greatly affected by how it is implemented. There were some initial benchmarking discussions occurring on this page: IntelLabs report on GraphLab vs. Mahout[102]. However there was a lot of contention around the specifics of the test and whether the frameworks were compared on even terms. The discussion is no longer active.

Category / Criteria *5) People:*
Required Skills/Experience and Recruiting talent/finding knowledge.
Evaluation
Implementing a solution in GraphLab requires skills in both C++ and graph theory that are commensurate with the complexity of the problem being solved. Implementing a solution in GraphChi requires knowledge of graph theory, and either C++ or Java in a similar manner as GraphChi.

Category / Criteria *6) Infrastructure:*
How is the deployment and management of the infrastructure to be managed with regard to the chosen solution? Do you have room for experimentation/testing/growth over time?
Evaluation
The deployment and management of infrastructure that is used to host the application containing the GraphLab API must meet the requirements for the installation and running of GraphLab applications. There are multiple tutorials on the GraphLab site that demonstrate the procedure for establishing suitable environments on either a Amazon EC2 Cluster, a alternate distributed environment, or on a single node multiple core machine[103]. GraphChi is designed to operate on a single machine only, this machine can be desktop grade. The benchmarks provided on the website were from a Mac Mini[104]. GraphLab (GraphChi only runs on a single node) can run on a multiple core machine as a 'single node'. Thus allowing for testing and experimentation with the API[98]. GraphLab and GraphChi are open source, so contributions from the community are encouraged[105]. An application containing a GraphLab/GraphChi component can be expected to grow and change with needs and requirements as per any application.

IV. Conclusion(s)

The various solutions, frameworks, libraries, and applications described above are but a fraction of what is available under the heading of big data. However they represent what can be considered to be 'state of the art' solutions in their respective areas. The qualifier of "in their respective areas" is important because when analysing each of the candidates it quickly becomes apparent that each has been created with a specific problem subset in mind.

Hadoop, CUDA, and Twitter Storm are capable of processing tabular data at massive scale. The Bulk Synchronous Parallel Processing implementations and GraphLab/Chi are useful when your data possesses a high density of relationships.

The Hadoop file system HDFS allows for the efficient, distributed, and fault-tolerant storage of massive scale tabular data, As well as being the basis for a large number of other processing solutions, Mahout,Giraph, and GraphLab to name a few.

There are also the graph database solutions that are extremely efficient at storing highly structured data. Graph databases also provide many methods for graph specific queries that would bring a traditional database to its knees.

The machine learning and processing libraries Mahout and MLPACK provide hand-tuned and highly efficient bundles of algorithm implementations that are designed to be integrated with a larger system for various data mining needs. Mahout is written in Java and integrates tightly with Hadoop, but is also capable of integrating with any Java application. MLPACK is written in C++ and allows for extremely fine grained control of its implementation. Neither is reportedly less effective than the other as the algorithms are battle tested and debugged by the community that surrounds both libraries.

Hadoop and GraphLab are primarily batch processing environments, data is received or updated and the respective tasks are triggered. These make them suited towards medium to lower velocity environments. Twitter Storm can be configured to operate as an extremely high velocity processing engine that responds with the results of an 'always on' computation whenever the data is updated or a request is received.

Nearly all of the solutions presented here are open sourced under various licences so they can be investigated, extended, and improved to suit various requirements.

So which solution is right for your needs? The answer is simply: "That depends…".

It depends on how your data is structured. It depends on the specific problem you're trying to solve. It depends on the hardware you have available. It depends on the experience, knowledge, and skills that are available. It depends on the amount of time you have. It depends on the algorithm(s) that you need to utilise. It depends if you need to respond to immediate requests for results or scheduled batches. It depends if you have static or shifting data. It depends if you're utilising existing storage or need a new system.

There is no simple answer, the field of 'big data' is vast, complex, intricate, and highly varied and will remain so as problems and the data behind them are investigated, solved,

discovered, created, cleaned, parsed, distributed, deleted, and condensed.

The unique aspects of what you are trying to achieve will determine the composition of your solution.

Perhaps Hadoop and Mahout can be applied to the massive amount of saved user comments, parsing the individual threads of discussion for common patterns. Whilst a GraphChi implementation investigates the relationships between the users posting the comments, the articles they are commenting on, and the times they commented.

It is reasonable to say that the breadth of solutions available will maintain pace with the depth of problems for the foreseeable future. Consequently it is vital that you examine your own goals and ruthlessly parse the available options with an eye for what you are trying to achieve, but also what you might wish to achieve in the future.

V. Bibliography

[1] "Welcome to Apache Hadoop!." 2007. 29 Jul. 2013 <http://hadoop.apache.org/>

[2] "Parallel Programming and Computing Platform | CUDA | NVIDIA ..." 2011. 29 Jul. 2013 <http://www.nvidia.fr/object/cuda_home_new.html>

[3] "Storm, distributed and fault-tolerant realtime computation." 2012. 31 Jul. 2013 <http://storm-project.net/>

[4] Gerbessiotis, Alexandros V, and Leslie G Valiant. "Direct bulk-synchronous parallel algorithms." Journal of parallel and distributed computing 22.2 (1994): 251-267.

[5] Low, Yucheng et al. "Graphlab: A new framework for parallel machine learning." arXiv preprint arXiv:1006.4990 (2010).

[6] "Neo4j - The World's Leading Graph Database." 2007. 29 Jul. 2013 <http://www.neo4j.org/>

[7] "thinkaurelius/titan GitHub." 2012. 29 Jul. 2013 <https://github.com/thinkaurelius/titan>

[8] Curtin, Ryan R et al. "MLPACK: A scalable C++ machine learning library." Journal of Machine Learning Research 14 (2013): 801-805.

[9] "Welcome to Apache Hadoop!." 2007. 29 Jul. 2013 <http://hadoop.apache.org/>

[10] "YARN - Apache Hadoop." 2012. 29 Jul. 2013 <http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html>

[11] "MapReduce Tutorial - Apache Hadoop - The Apache Software ..." 2012. 29 Jul. 2013 <http://hadoop.apache.org/docs/stable/mapred_tutorial.html>

[12] "PoweredBy - Hadoop Wiki - General Wiki." 2008. 29 Jul. 2013 <http://wiki.apache.org/hadoop/PoweredBy>

[13] "Developer Community - Cloudera." 2012. 29 Jul. 2013 <http://www.cloudera.com/content/cloudera/en/developer-community.html>

[14] "Hortonworks. We Do Hadoop." 2011. 29 Jul. 2013 <http://hortonworks.com/>

[15] "Newest 'hadoop' Questions - Stack Overflow." 2008. 29 Jul. 2013 <http://stackoverflow.com/questions/tagged/hadoop>

[16] "HBase - Apache HBase Home." 2010. 29 Jul. 2013 <http://hbase.apache.org/>

[17] "Apache Mahout: Scalable machine learning and data mining." 2010. 29 Jul. 2013 <http://mahout.apache.org/>

[18] "Books Tutorials and Talks - Apache Mahout - Apache Software ..." 2010. 29 Jul. 2013 <https://cwiki.apache.org/confluence/display/MAHOUT/Books+Tutorials+and+Talks>

[19] "Graph database - Wikipedia, the free encyclopedia." 2009. 29 Jul. 2013 <http://en.wikipedia.org/wiki/Graph_database>

[20] "Performance of Graph vs. Relational Databases | Architects Zone." 2013. 29 Jul. 2013 <http://architects.dzone.com/articles/performance-graph-vs>

[21] "The Software Abstractions Blog: Big Data and Graph Databases." 2013. 30 Jul. 2013 <http://blog.softwareabstractions.com/the_software_abstractions/2013/06/big-data-and-graph-databases.html>

[22] "Getting Started thinkaurelius/faunus Wiki GitHub." 2012. 29 Jul. 2013 <https://github.com/thinkaurelius/faunus/wiki/Getting-Started>

[23] "Giraph - Welcome To Apache Giraph!." 2012. 29 Jul. 2013 <http://giraph.apache.org/>

[24] Seo, Sangwon et al. "Hama: An efficient matrix computation with the mapreduce framework." Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on 30 Nov. 2010: 721-726.

[25] "Hama - The Apache Software Foundation!." 2012. 31 Jul. 2013 <http://hama.apache.org/>

[26] Valiant, LG. "A bridging model for parallel computation - ACM Digital Library." 1990. <http://dl.acm.org/citation.cfm?id=79181>

[27] "Parallel Scientific Computation - Oxford University Press." 2009. 5 Aug. 2013 <http://ukcatalogue.oup.com/product/9780198529392.do>

[28] "GeForce GTX 780 | Specifications | GeForce." 2013. 29 Jul. 2013 <http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-780/specifications>

[29] "What is CUDA | NVIDIA Developer Zone." 2011. 29 Jul. 2013 <https://developer.nvidia.com/what-cuda>

[30] Kouzinopoulos, C. S., & Margaritis, K. G. (2009, September). String Matching on a multicore GPU using CUDA. In Informatics, 2009. PCI'09. 13th Panhellenic Conference on (pp. 14-18). IEEE.

[31] "JAVA vs C++ Benchmark - Metabolomics Fiehn Lab - UC Davis." 2006. 29 Jul. 2013 <http://fiehnlab.ucdavis.edu/staff/kind/Collector/Benchmark/JAVA_Benchmark/>

[32] "Java vs. C++: The Performance Showdown, Page 2 - Developer.com." 2010. 29 Jul. 2013 <http://www.developer.com/java/article.php/10922_3856906_2/Java-vs-C-The-Performance-Showdown.htm>

[33] Kahan, William et al. "How Java's floating-point hurts everyone everywhere." Talk given at the ACM 1998 Workshop on Java for High-Performance Network Computing (http://www. cs. uscb. edu/conferences/wkahan/JAVAhurt. pdf) 1 Mar. 1998.

[34] "OpenCL - The open standard for parallel programming of ..." 2008. 9 Aug. 2013 <http://www.khronos.org/opencl/>

[35] "nathanmarz/storm GitHub." 2011. 31 Jul. 2013 <https://github.com/nathanmarz/storm>

[36] "Common patterns nathanmarz/storm Wiki GitHub." 2011. 31 Jul. 2013 <https://github.com/nathanmarz/storm/wiki/Common-patterns>

[37] "LAPACK — Linear Algebra PACKage - Netlib." 31 Jul. 2013 <http://www.netlib.org/lapack/>

[38] Sanderson, Conrad. "Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments." Sep. 2010.

[39] Curtin, Ryan R et al. "MLPACK: A scalable C++ machine learning library." Journal of Machine Learning Research 14 (2013): 801-805.

[40] "GraphLab - The Abstraction." 2012. 26 Aug. 2013 <http://graphlab.org/home/abstraction/>

[41] Han, Wook-Shin et al. "TurboGraph: a fast parallel graph engine handling billion-scale graphs in a single PC." Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining 11 Aug. 2013: 77-85.

[42] Kyrola, Aapo, Guy Blelloch, and Carlos Guestrin. "GraphChi: Large-scale graph computation on just a PC." Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI) 8 Oct. 2012: 31-46.

[43] "Vertex Centric Indices thinkaurelius/titan Wiki GitHub." 2013. 3 Sep. 2013 <https://github.com/thinkaurelius/titan/wiki/Vertex-Centric-Indices>

[44] Zikopoulos, Paul, and Chris Eaton. Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media, 2011.

[45] Philip Russom. "The Three Vs of Big Data Analytics: VELOCITY – TDWI -The Data ..." 2011. 1 Aug. 2013 <http://tdwi.org/blogs/philip-russom/2011/06/three-vs-of-big-data-analytics-3-data-velocity.aspx>

[46] "The Art of Big Data - SlideShare." 2011. 1 Aug. 2013 <http://www.slideshare.net/ksankar/the-art-of-big-data>

[47] "InputFormat (Apache Hadoop Main 2.0.5-alpha API)." 2012. 1 Aug. 2013 <http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/InputFormat.html>

[48] "OutputFormat (Apache Hadoop Main 2.0.5-alpha API)." 2012. 1 Aug. 2013 <http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/OutputFormat.html>

[49] "Has anyone used Graph-based Databases (http://neo4j.org/)? [closed]." 2009. 1 Aug. 2013 <http://stackoverflow.com/questions/1000162/has-anyone-used-graph-based-databases-http-neo4j-org>

[50] Irani, Zahir. "Information systems evaluation: navigating through the problem domain." Information & Management 40.1 (2002): 11-24.

[51] "Oracle: Big Data for the Enterprise (PDF)." 2011. 2 Aug. 2013 <http://www.oracle.com/us/products/database/big-data-for-enterprise-519135.pdf>

[52] Schaeffer, Satu Elisa. "Graph clustering." Computer Science Review 1.1 (2007): 27-64.

[53] Ho, Yu-Chi. "Heuristics, rules of thumb, and the 80/20 proposition." Automatic Control, IEEE Transactions on 39.5 (1994): 1025-1027.

[54] "sjmackenzie/phoebus_core GitHub." 2010. 5 Aug. 2013 <https://github.com/sjmackenzie/phoebus_core>

[55] "HDFS: Facebook has the world's largest Hadoop cluster!." 2010. 6 Aug. 2013 <http://hadoopblog.blogspot.fr/2010/05/facebook-has-worlds-largest-hadoop.html>

[56] "The Small Files Problem | Apache Hadoop for the ... - Cloudera Blog." 2012. 6 Aug. 2013 <http://blog.cloudera.com/blog/2009/02/the-small-files-problem/>

[57] “Input Formats | Hadoop: The Definitive Guide | MapReduce ... - Inkling.” 2012. 6 Aug. 2013 <https://www.inkling.com/read/hadoop-definitive-guide-tom-white-3rd/chapter-7/input-formats>

[58] “Hadoop: RecordReader and FileInputFormat | Hadoopi.” 2013. 6 Aug. 2013 <http://hadoopi.wordpress.com/2013/05/27/understand-recordreader-inputsplit/>

[59] “InputFormat | Hadoopi.” 2013. 6 Aug. 2013 <http://hadoopi.wordpress.com/category/inputformat/>

[60] “Programming Community Index - TIOBE.com.” 2008. 6 Aug. 2013 <http://www.tiobe.com/content/paperinfo/tpci/index.html>

[61] “Apache Hadoop 2.0.5-alpha - Apache Hadoop.” 2012. 6 Aug. 2013 <http://hadoop.apache.org/docs/current/>

[62] “Pivotal HD | GoPivotal.” 2013. 6 Aug. 2013 <http://gopivotal.com/pivotal-products/pivotal-data-fabric/pivotal-hd>

[63] “Hadoop: “It’s damn hard to use” — Tech News and Analysis - GigaOM.” 2013. 6 Aug. 2013 <http://gigaom.com/2013/03/21/hadoop-its-damn-hard-to-use/>

[64] “java - Using Apache Mahout with Ruby on Rails - Stack Overflow.” 2010. 7 Aug. 2013 <http://stackoverflow.com/questions/3223390/using-apache-mahout-with-ruby-on-rails>

[65] “Overview - Apache Mahout - Apache Software Foundation.” 2010. 7 Aug. 2013 <https://cwiki.apache.org/confluence/display/MAHOUT/Overview>

[66] “Algorithms - Apache Mahout - Apache Software Foundation.” 2010. 7 Aug. 2013 <https://cwiki.apache.org/confluence/display/MAHOUT/Algorithms>

[67] “How To Become A Committer - Apache Mahout - Apache Software ...” 2010. 7 Aug. 2013 <https://cwiki.apache.org/confluence/display/MAHOUT/How+To+Become+A+Committer>

[68] “Deploying a massively scalable recommender system with Apache ...” 2011. 7 Aug. 2013 <http://ssc.io/deploying-a-massively-scalable-recommender-system-with-apache-mahout/>

[69] “Recommender Documentation - Apache Mahout - Apache Software ...” 2010. 7 Aug. 2013 <https://cwiki.apache.org/confluence/display/MAHOUT/Recommender+Documentation>

[70] “Why becoming a data scientist is NOT actually easier than ... - Medium.” 2013. 7 Aug. 2013 <https://medium.com/cs-math/5b65b548069b>

[71] “A case for Neo4j database - Manning Publications.” 2012. 7 Aug. 2013 <http://www.manning.com/partner/Neo4J_meap_ch01.pdf>

[72] “orientechnologies/orientdb GitHub.” 2013. 7 Aug. 2013 <https://github.com/orientechnologies/orientdb/>

[73] “The Apache Cassandra Project.” 2010. 7 Aug. 2013 <http://cassandra.apache.org/>

[74] “HBase - Apache HBase Home.” 2010. 7 Aug. 2013 <http://hbase.apache.org/>

[75] “GraphDB Comparison orientechnologies/orientdb Wiki GitHub.” 2013. 7 Aug. 2013 <https://github.com/orientechnologies/orientdb/wiki/GraphDB-Comparison>

[76] “TinkerPop Graph Stack thinkaurelius/titan Wiki GitHub.” 2012. 7 Aug. 2013 <https://github.com/thinkaurelius/titan/wiki/TinkerPop-Graph-Stack>

[77] “Blueprints Interface thinkaurelius/titan Wiki GitHub.” 2012. 7 Aug. 2013 <https://github.com/thinkaurelius/titan/wiki/Blueprints-Interface>

[78] “Gremlin Query Language thinkaurelius/titan Wiki GitHub.” 2012. 7 Aug. 2013 <https://github.com/thinkaurelius/titan/wiki/Gremlin-Query-Language>

[79] “Titan Server thinkaurelius/titan Wiki GitHub.” 2012. 7 Aug. 2013 <https://github.com/thinkaurelius/titan/wiki/Titan-Server>

[80] “Rexster Graph Server thinkaurelius/titan Wiki GitHub.” 2012. 7 Aug. 2013 <https://github.com/thinkaurelius/titan/wiki/Rexster-Graph-Server>

[81] Malewicz, Grzegorz et al. “Pregel: a system for large-scale graph processing.” Proceedings of the 2010 ACM SIGMOD International Conference on Management of data 6 Jun. 2010: 135-146.

[82] “OCTO talks ! Introduction to large-scale graph processing.” 2012. 8 Aug. 2013 <http://blog.octo.com/en/introduction-to-large-scale-graph-processing/>

[83] Malewicz, Grzegorz et al. “Pregel: a system for large-scale graph processing.” Proceedings of the 2010 ACM SIGMOD International Conference on Management of data 6 Jun. 2010: 135-146.

[84] Erra, Ugo et al. “An efficient GPU implementation for large scale individual-based simulation of collective behavior.” High Performance Computational Systems Biology, 2009. HIBI’09. International Workshop on 14 Oct. 2009: 51-58.

[85] Nicole Hemsoth. “The GPU”Sweet Spot” for Big Data - Datanami.” 2012. 9 Aug. 2013 <http://www.datanami.com/datanami/2012-09-11/the_gpu_sweet_spot_for_big_data.html>

[86] “GPU and Large Scale Data Mining Azinta Systems Blog.” 2011. 9 Aug. 2013 <http://www.azintablog.com/2010/10/16/gpu-large-scale-data-mining/>

[87] Jon Stokes. “What’s so hard about doing non-graphics ... - Ars Technica.” 2012. 9 Aug. 2013 <http://arstechnica.com/uncategorized/2007/02/8931/>

[88] Jon Stokes. “What’s so hard about doing non-graphics ... - Ars Technica.” 2012. 9 Aug. 2013 <http://arstechnica.com/uncategorized/2007/02/8931/>

[89] “Category: Research :: GPGPU.org.” 2009. 12 Aug. 2013 <http://gpgpu.org/category/research>

[90] “Scalable - About Storm.” 2012. 12 Aug. 2013 <http://storm-project.net/about/scalable.html>

[91] “any programming language - About Storm.” 2012. 12 Aug. 2013 <http://storm-project.net/about/multi-language.html>

[92] “Simple API - About Storm.” 2012. 12 Aug. 2013 <http://storm-project.net/about/simple-api.html>

[93] “Storm, distributed and fault-tolerant realtime computation.” 2012. 12 Aug. 2013 <http://storm-project.net/>

[94] “Simple Sample mlpack Programs.” 2012. 12 Aug. 2013 <http://www.mlpack.org/doxygen.php?doc=sample.html>

[95] “API documentation - MLPACK.” 2012. 12 Aug. 2013 <http://www.mlpack.org/doxygen.php>

[96] “Matrices in mlpack.” 2012. 12 Aug. 2013 <http://www.mlpack.org/doxygen.php?doc=matrices.html>

[97] Curtin, Ryan R et al. “MLPACK: A scalable C++ machine learning library.” Journal of Machine Learning Research 14 (2013): 801-805.

[98] “About | GraphLab.” 2013. 26 Aug. 2013 <http://graphlab.com/about/>

[99] “GraphLab: Distributed Graph-Parallel API: Graph File Formats.” 2012. 26 Aug. 2013 <http://docs.graphlab.org/graph_formats.html>

[100] “GraphLab - The Software.” 2012. 2 Sep. 2013 <http://graphlab.org/home/the-software/>

[101] “Creating GraphChi Applications GraphChi/graphchi-cpp Wiki GitHub.” 2013. 2 Sep. 2013 <https://github.com/GraphChi/graphchi-cpp/wiki/Creating-GraphChi-Applications>

[102] “Intel Labs report on GraphLab vs. Mahout - Large Scale Machine ...” 2013. 2 Sep. 2013 <http://bickson.blogspot.fr/2013/03/intel-labs-report-on-graphlab-vs-mahout.html>

[103] “GraphLab - Tutorials.” 2012. 2 Sep. 2013 <http://graphlab.org/tutorials-2/>

[104] “Apple - Mac mini.” 2012. 2 Sep. 2013 <http://www.apple.com/mac-mini/>

[105] “graphlab-code/graphlab GitHub.” 2013. 2 Sep. 2013 <https://github.com/graphlab-code/graphlab>

[106] “Apache ZooKeeper - Apache Foundation.” 2010 <https://zookeeper.apache.org/>

[107] “Apache Ambari - Apache Foundation.” 2010 <https://incubator.apache.org/ambari/>

[108] “Bridging Model - Wikipedia.” Communications of the ACM - Volume 33, Issue 8, Aug. 1990. Pages 103-111. <http://dl.acm.org/citation.cfm?id=79181>

[109] “Small Files Problem - Cloudera Blog.” 2009. 2 Feb <http://blog.cloudera.com/blog/2009/02/the-small-files-problem/>

[110] “Jcuda - CUDA Bindings for Java” 2012. 8 Feb <http://www.jcuda.org/>

[110] “PyCuda - CUDA Bindings for Python” 2012. 8 Feb <http://thema.tician.de/software/pycuda>

[111] “CUDA Performance Question - StackOverflow” 2012. 3 Apr <http://stackoverflow.com/a/9986748>

[111] “Nvidia CUDA Warps and Occupancy” 2012. 3 Apr <http://on-demand.gputechconf.com/gtc-express/2011/presentations/cuda_webinars_WarpsAndOccupancy.pdf>

[112] “ZeroMQ - Distributed Computing Made Simple” 2013 <http://zeromq.org/>

[113] “RabbitMQ - Messaging that just Works” 2013 <http://www.rabbitmq.com/>

[114] “RabbitMQ - Messaging that just Works” 2013 <http://robey.github.io/kestrel/>