Data Intensive Computing

Rob Allan
Computational Science and Engineering Department,
STFC Daresbury Laboratory, Daresbury, Warrington WA4 4AD

Contact e-Mail: r.j.allan@dl.ac.uk

***** D R A F T *****

Abstract

This report looks at a variety of requirements for and approaches to data intensive computing. It also considers novel ways to extract information from large data sets, particularly if this can be carried out "in situ" avoiding the need for lengthy file transfers.

1 Introduction

An introduction to data intensive computing is given in Wikipedia `http://en.wikipedia.org/wiki/Data_Intensive_Computing`. There, it is defined as a class of parallel computing applications which use a data parallel approach to processing large volumes of data typically terabytes or petabytes in size. They devote most of their processing time to I/O and manipulation of data rather than computation. This is orthogonal to what we are familiar with from modelling and simulation on high performance computing systems.

Data intensive computing refers to capturing, managing, analysing and understanding data at volumes and rates that push the frontiers of current technologies. The challenge is to provide the hardware architectures and related software systems and techniques which are capable of transforming ultra-large data into valuable information and ultimately knowledge.

The National Science Foundation has noted that data intensive computing requires a "fundamentally different set of principles" to other computing approaches. They funded a programme to seek "increased understanding of the capabilities and limitations of data intensive computing". The key focus areas are:

- Enhamcements to parallel programming to address the processing of data;
- Programming abstractions including models, languages, and algorithms which allow a natural expression of parallel data processing;
- Design of data intensive computing platforms to provide high levels of reliability, efficiency, availability and scalability;
- Identifying applications that can exploit this computing paradigm and determining how research should evolve to make best use of such applications.

Pacific Northwest National Labs, responding to the NSF call, defined data intensive computing as "capturing, managing, analyzing, and understanding data at volumes and rates that push the frontiers of current technologies". They believe that to address the rapidly growing data volumes and complexity requires "epochal advances in software, hardware and algorithm development" which can scale readily with size of the data and provide effective and timely analysis and processing results.

There are several important common characteristics of data intensive computing systems that distinguish them from other forms of computing.

1. The principle of co-locating the data and applications or algorithms. To achieve high performance in data intensive computing, it is important to minimise movement of data. For this reason it is useful to execute applications on the nodes where the data resides. High bandwidth and low latency networking using technologies, such as InfiniBand which enables RDMA, allow data to be stored in a separate nearby repository and provide performance comparable to data on local disk.
2. The programming model used. Typical data intensive computing applications are expressed in terms of high level operations on data, and the runtime system transparently controls the

scheduling, execution, load balancing, communications and movement of computation and data across the distributed computing cluster. The programming abstraction and language tools allow the processing to be expressed in terms of data flows and transformations incorporating new programming languages and shared libraries of common data manipulation algorithms such as sorting. A database is often used as optimisations are well known.

3. A focus on reliability and availability. Data intensive computing systems must be designed to be fault tolerant. This typically involves redundant copies of data on disk, storage of intermediate processing results on disk, automatic detection of node or processing failures and selective rollback or re-computation of results. Database technologies are also used for this purpose.

4. The inherent scalability of the underlying hardware and software architecture. Data intensive computing systems can typically be scaled in a linear fashion to accommodate virtually any amount of data, or to meet time critical performance requirements by simply adding additional processing and storage nodes. The number of nodes and processing tasks assigned for a specific application can be variable or fixed depending on the hardware, software, communications and distributed file system architecture.

Note that these criteria implicitly preclude in-memory solutions (but see more below).

A variety of system architectures have been implemented for data intensive computing and large scale data analysis applications including parallel and distributed relational database management systems which have been available to run on clusters for more than two decades. This assumes the data is structured and can be mapped onto a set of connected tables for manipulation. However most data growth is with data in un-structured ore semi-structured form and new processing paradigms with more flexible models are needed. Emerging solutions include the MapReduce architecture pioneered by Google [7] and now available in an open source implementation called Apache Hadoop, see [39, 2] (note Dryad/ LINQ has been used on Windows platforms).

1.1 Amdahl's Laws

Amdahl's Laws are as follows.

1. Amdahl's parallelism law: If a computation has a serial part which takes time S to execute and a parallel component which takes time P/N to execute on N processors, then the speedup on N is (S+P)/(S+P/N). The maximim speedup is therefore (S+P)/S.

2. Amdahl's balanced system law: A system needs one **bit** of I/O per second per instruction per second.

3. Amdahl's memory law: alpha=1: that is the MB/MIPS ratio, in a balanced system is 1. That is one **byte** of memory per cpu instruction cycle.

4. Amdahl's I/O law: Programs do one I/O per 50,000 instructions

Looking at Amdahl's balanced system law, we typically see the following.

- The Amdahl number for super-computers running high performance computing applications ranges around 10^{-5};
- Computation heavy data intensive simulations tend to use hardware with Amdahl number around 10^{-3};
- Data intensive analytical applications are in the range of 10^{-1} or even 1.

1.2 The GrayWulf

We should be aware that with multi-level caching, a much lower I/O to MIPS ratio coupled with a large enough memory can still provide satisfactory performance [33]. This is however only true if the problem fits in memory.

Typically, cache based systems with branch prediction and speculation hardware in CPU are not useful for applications which require streaming data. Here the order of magnitude mis-match between disk memory and CPU bandwidth hampers the processing. A more balanced system is required. Both FPGAs and GPUs have been investigated for this purpose. SSD devices might also be used to reduce the bandwidth to disk and they significantly reduce the seek time for un-structured data. Some data intensive benchmarks were created to test different architectures by Gokhale *et al.* [10].

Special purpose computers based on Amdahl's balanced system law are built to process large volumes of data that will not fit entirely in memory. The original machine designed by Jim Gray of Microsoft is now known as the GrayWulf [33]. This is typically a good solution for university groups who host their own large data sets (it is difficult to move over 1 TB of data over the internet) but who could maintain a reasonably compact commodity cluster solution.

The goals of the GrayWulf project were as follows.

- support analysis of PetaByte data sets;
- provide very high bandwidth sequential access to data;
- support a variety of access patterns (see below);
- support a set of simple tools for database design;
- support a set of tools for fast ingest of data.

It is found, for instance by Gokhale *et al.* [10] that using special purpose hardware can remove the CPU bottlneck and ultimately the system becomes limited by bandwidth to disk.

Some examples of machines based on this concept are given below.

1.2.1 JHU-1

The original GrayWulf at Johns Hopkins University has 416 CPUs capable of 1,107 GI/s with a total memory of 1,152 GB. Disk I/O performance was a total of 70 GB/s from just over 1PB of storage.

This results in an Amdahl memory number of 1.04 and I/O number of 0.506. The modular structure of this solution is shown in the following table.

No. Servers	CPU ea	Memory ea	Disk ea	Interconnect
Wide area interconnect, 10Gb/s FC				
Tier 1				
2	16 core Dell R900	128GB	11.25TB 1x MD1000 as below	InfiniBand QLogic 20Gb/s
Tier 2				
4	16 core Dell R900	64GB	33.75TB 3x MD1000 as below	InfiniBand QLogic 20Gb/s
Tier 3				
40	8 core Dell 2950 2.66 GHz	16GB	22.5TB 2x MD1000 SAS total 30x 750GB 7,200RPM SATA discs	InfiniBand QLogic 20Gb/s

Further details are given in [33].

This is basically a distributed SQL server cloud. Applications are run on the Tier-1 servers which are also used for remote access and data transfer.

1.2.2 JHU-2

A second example of a more energy efficient solution has been built using Atom/ Ion technology.

No. Servers	CPU ea	Memory ea	Disk ea	Interconnect
Tier 1				
36	2x Atom + 16x GPU Zotac N330 Intel Atom/ nVidia Ion 1.6GHz	4GB	1x 120GB SSD + 2x 1TB Samsung F1	

In this case, the Ion GPUs are used typically for data mining over astronomy image data. They can be used for other user-defined functions executed as out of process SQL procedures.

1.2.3 EDIM-1

The Edinburgh Data Intensive Machine 1 was built recently also using Atom technology.

No. Servers	CPU ea	Memory ea	Disk ea	Interconnect
Tier 1				
120	2x Atom + 16x GPU	4GB	1x 256MB SDD	

	Zotac N330 Intel Atom/ nVidia Ion 1.6GHz	+ 3x 2TB HDD		

1.2.4 Gordon

The Gordon system at SDSC was launched on 6/12/2011. At the time of writing it is the world's largest flash memory based computer. It was developed over a period of two years based on the Dash prototype and supported by an NSF Track 2 grant of by a $20M. Gordon thus represents the first really big purpose built super-computer for data intensive applications. One aim was to pioneer the technology for the wider enterprise market.

The intention of SDSC and the NSF is to draw in data intensive science codes that have never had a platform this size to push the envelope. This is particularly true of in genomics, an application set that was foremost in the minds of the system engineers when the machine was being designed. Genomics is the classic "big data" science problem, and is the one most frequently cited in HPC circles as suffering from the data deluge crisis. Other application areas like graph problems, geophysics, financial market analytics, and data mining are also expected to be important domains for Gordon.

The system is an evolution of the Appro HPC cluster, using the third generation Xtreme-X architecture and Intel's SandyBridge Xeon E5 CPUs prior to GA. There are 1,024 dual socket, nodes with 64GB of DDR3 memory giving a peak performance of 280 Tflop/s. The system has over 300 TB of Intel solid state "flash" disks, spread over 64 I/O nodes.

The aggregate IO/s performance of the machine is therefore impressive which on all 64 I/O nodes achieves a peak output of 36M IO/s.

Operating system is based on ScaleMP's virtual SMP (vSMP) technology. It allows users to run large memory applications on what they call a "super-node" – an aggregation of 32 Gordon servers and two I/O servers, providing access to 512 cores, 2 TB of RAM and 9.6 TB of SSD. To a program running on a supernode, the hardware behaves as a big cache coherent server. The system can be partitioned into 32 super-nodes. This is currently the largest system deployed with its technology. The system is connected to a disk sub-system via NFS and Lustre, currently 150TB but planned to rise to 4PB by July 2010.

The flash device being employed is Intel's new iSolid-State Drive 710. The 710 uses Intel's High Endurance Technology (HET), which is Intel's version of enterprise multi-level cell (eMLC) flash memory. Like eMLC, the HET flash features the performance and resiliency of single level cell (SLC) flash, but at a much lower cost. SDSC also developed its own flash device drivers to maximize performance of the SSD gear.

Inserting this much flash memory into a supercomputer had never been attempted before, and this aspect was probably the biggest risk for the project. When they began the Gordon effort two years ago, flash memory was just starting to make its way into enterprise storage and was an expensive and un-proven technology.

The system is currently undergoing acceptance testing and is expected to be available for production use dedicated to XSEDE users (the evolution of TeraGrid) on 1/1/2012.

No. Servers	CPU ea	Memory ea	Disk ea	Interconnect
SuperNode				
32	512	2TB vSMP	9.6TB	QDR IB
Tier 1				
1,024 dual socket	16 core SandyBridge 2.6GHz	64GB DDR3		QDR IB 3D torus
I/O Node				
64 dual socket	12 Westmere 2.67GHz	48GB DDR3	4.8TB Intel iSSD 710	

This gives and Ahmdahl memory number of 1.53, but the total bandwidth to SSD is not known.

1.3 Graph500 Benchmark

The Graph500 benchmark is intended to guide the design of hardware architectures and software systems to support data intensive applications of which graph based algorithms are a core part. The benchmark currently includes: (i) concurrent search; (ii) optimisation (single source shortest path); and (iii) edge oriented (maximal independent set). The output is a metric of TEPS, traversed edges per second. Results are typically in the millions or billions.

The Web site `http://www.graph500.org/` contains further information and provides the source code.

2 Software Approaches

We now consider software for data intensive systems. Much of this discussion is based on [31]. We first consider the use of a database approach for structured data sets. A high level language based on SQL is typically used to encode required operations which can be uploaded by users.

Most scientific data analyses (part of Information Lifecycle Management) are performed in hierarchical steps. During the first pass, a subset of the data is extracted by either filtering on certain attributes (typically removing erroneous or unwanted data) or extracting a vertical subset of the columns. In the next step, data are usually transformed or aggregated in some way. In more complex datasets, these patterns are often accompanied by complex joins among multiple attributes, such as external calibrations or extracting and analysing different parts of a gene sequence. For very large datasets, the most efficient way to perform most of these computations is clearly to move the analysis functions as close to the data as possible. Many of the patterns that have been identified are easily expressed by a set oriented, declarative language whose execution can benefit enormously from cost based query optimisation, automatic parallelism and indexing.

2.1 RDBMS

These features were recognised by Jim Gray and his collaborators at Microsoft and sometimes referred to as "Gray's Laws". They have shown in several projects that existing relational database technologies can be successfully applied in this context [31]. It is also possible to integrate complex functions written in a procedural or other languages (such as C or R) as an extension of the underlying database engine, as exemplified by the work at Johns Hopkins University.

In recent years, MapReduce has become a popular distributed data analysis and computing paradigm [7], see below. The principles behind it resemble the distributed grouping and aggregation patters that already existed in parallel relational database systems. Such capabilities are now being referred to as "MapReduce in the database". Benchmarks to compare different approaches are being developed [24].

It was found by Szalay *et al.* that data operations performed by users of the Sloan Digital Survey database [33] exhibit a 1/f power law distribution. Most queries are very simple single row lookups based on simple indices such as celestial position (high frequency, low volume). A small number of analyses did not use pre-computed indices, and required lengthy sequential scans through the data possibly combined with merge-join operations. These can take over an hour on a multi-terabyte database. Other access patterns involved multiple simple scripts to extract a series of results, a kind of database "crawler". Application of such scripts might be combined into a more complex workflow which can be executed from a batch queue.

For efficient analysis of structured data. it is important to start with an appropriate schema (for metadata) and database design (table layout and connectivity, data ordering, etc.). It is notoriously hard to modify these on an *ad hoc* basis. Given these requirements it is then possible to provide a set of sample SQL scripts which might meet the requirements of most users. The database design should aim to accommodate these scripts in the simplest way. The scripts can be extended to do more complex analyses.

To facilitate access and reduce I/O requirements the data must be factored in different ways which reflect requirements, for instance it could be divided in space or time into blocks which can be read in and processed in parallel. Typically this will result in a hierarchical database architecture. This is explained in further detail with examples from the Pan-STARRS and SLOAN DB in [31].

Management of the database and servers, in particular to ensure fault tolerance, is also described in [31].

A number of management tasks can be codified into workflows as follows.

Load Data: Data streaming from sensors and instruments or simulations may need to be assimilated into existing databases. The load workflow does the initial data format conversion from the incoming data format into the database schema. This may involve data translation, verifying structure consistency, checking quality, mapping the incoming data structure to the database schema, determining optimised placement of the "load databases", performing the actual load and running post-load validations.

Merge Data: The merge workflow ingests the incoming data in the load databases into the "cold" databases and re-organises the incremented database in an efficient manner. If multiple databases

are to be loaded, there may be additional scheduling required to do the updates concurrently and expose a consistent data release version to the users.

Data Replication: The data replication workflow creates copies of databases over the network in an efficient manner. This is necessary to ensure availability of adequate copies of the databases, replicating databases upon updates or failures. The workflow coordinates with job schedulers to acquire locks on "hot" and "warm" databases, determines data placement, makes optimal use of I/O and network bandwidth and ensures the databases are not corrupted during copy.

Crawler: Complete scans of the available data are required to check for silent data corruption, or to update object data statistics. These workflows need to be designed as minimally intrusive background tasks on live systems. A common pattern for these crawlers is to "bucket" portions of the large data space and apply the validation or statistic on that bucket. This allows the tasks to be performed incrementally, checkpoint, recover from faults at the last processed bucket, balance the load on the machine and enable parallelisation.

Fault Recovery: Recovering from hardware or software faults is a key task. Fault recovery requires complex coordination between several software components and must be fully automated. Fault recovery can be modelled using state diagrams that can then be implemented as workflows. These may be associated as fault handlers for other workflows or triggered by a monitoring event. Occasionally, a fault can be recovered in a shorter time by allowing existing tasks to proceed to completion; hence look ahead planning is required.

2.2 MapReduce

Some authors, e.g. [19] have claimed that MapReduce is the most successful abstration to date to go beyond the classic von Neumann model of computing.

MapReduce is a software framework introduced by Google in 2004 to support distributed computing on large data sets on loosely coupled clusters of computers. The framework is inspired by the map and reduce functions commonly used in functional programming, but in a modified form. MapReduce libraries have now been written in C++, C#, Erlang, Java, OCaml, Perl, Python, PHP, Ruby, F#, R and other languages.

Such abstractions manage complexity by hiding details and presenting well defined behaviours to users. They make certain tasks easier but others more difficult, and sometimes impossible. It may be only the first in a new class of programming models that will allow us to more effectively organise computations at a massive scale, and extensions are already being suggested.

MapReduce frameworks were developed for commercial information retrieval, which is perhaps currently the world's most demanding data analysis problem. Exploiting commercial approaches offers a good chance that one can achieve a high quality robust solution. MapReduce now has both commercial and open source implementations. Qiu *el.* [27] looked at MapReduce and MPI, and showed how to analyse biological samples with modest numbers of sequence on a modern multi-core cluster. To support iterative operations they evaluated the open source Java i-MapReduce [43] software which uses the Apache Hadoop, see [39, 2]. Another iterative MapReduce implementation is Twister [8].

Iterative data intensive computation is pervasive in many applications such as data mining or social network analysis. These typically involve massive data sets containing at least millions or even billions

of records. MapReduce in its original form lacks built in support for iterative processes that require parsing data sets repeatedly and doing bookkeeping. Besides specifying MapReduce jobs, users have to write a driver program that submits multiple jobs and performs convergence or other testing at the client. In a system such as i-MapReduce, users specify the iterative computation with the separated map and reduce functions Ths system provides the support of automatic iterative processing without the need of a user-defined driver program. As well as making it easier to use, this can improve the performance by: (1) reducing the overhead of creating new MapReduce jobs repeatedly; (2) eliminating the shuffling of static data; and (3) allowing asynchronous execution of map tasks.

Whilst specially optimised block based file systems such as Hadoop FS have been developed to support MapReduce [2], there are also versions implemented in MPI such as the library from Sandia National Labs [26, 25] and Indiana [14]. Interfaces from Sandia are available for C, C++, Python and OINK (c.f. PIG, the scripting language for HADOOP). It has for instance been used to parallelise BLAST and SOM algorithms [34]. Another is YAMML, a Google code prototype.

There are several database like implementations on top of Hadoop, for instance Hive which offers an SQL like interface.

Geoffrey Fox (Indiana University) has compared MPI with iterative MapReduce. He noted that MapReduce is more dynamic and fault tolerant than MPI and it is simpler and easier to use. Both Fox [8] and Chen and Scholosser [6] noted that it however requires some extensions and an iteration framework to make it useful for a wider range of applications. Some applications listed in papers on MapReduce are as follows.

K-means clustering: divides data points into a pre-defined number of clusters based on mean "distance"

Matrix multiplication: using traditional row and column decomposition. Maps columns onto processors then iterestes over rows and reduces the partial blocks into row blocks for output

Sequence comparison: Smith Waterman Gotoh pair wise distance algorithm

CAP3: sequence assembly program

Network analysis: MDS (multi-dimensional scaling) compared to GTM (Generative Topographic Mapping) using GraphViz for visualisation of output graphs. These are examples of non-linear dimensionality reduction applied to interpret large data sets.

PhyloD: A computational biology code from Microsoft. PhyloD is a statistical tool that can identify HIV mutations that defeat the function of the HLA proteins in certain patients.

HEP: data analysis originally coded in ROOT and CINT.

PageRank: effectively a recursive Markov Chain algorithm which iterates to convergence

Graph search: a breadth first search algorithm by Cailin uses iterative MapReduce

Word count: simple counting algorithm adds up occurrences of words in files

Mesh refinement: using hierarchically tiled arrays (HTA), e.g. with htalib [5]

Machine learning: e.g. classification algorithms, image recognition, ranking

BLAST: BLAST and SOM algorithms have been implemented using both Twister and MPI-MapReduce [34]

SOM: Self Organising Map, for dimensionality reduction and clustering or classification used in bio-informatics, see above

ABMS: Agent Based Modelling and Simulation has been implemented by Wang *et al.* [38] using their own BRACE framework and BRASIL scripting language.

MD: there has been at least one attempt to implement a Molecular Dynamics algorithm using Hadoop [12]. Other uses of MapReduce are in trajectory analysis.

For more information about some of these examples see `http://www.iterativemapreduce.org/samples.html`.

We note that applications written in languages other than Java can also be used on Hadoop with the Pipes library [39].

2.3 Dryad

Dryad [15] is a Microsoft Research project developing an execution environment for data parallel applications expressed as directed acyclic graphs (DAGs). In general this ongoing programme of work is investigating programming models.

2.4 Pregel

Pregel [20] is another technology developed by Google to mine information from graphs. Examples include road networks (maps), the Internet, etc. Pregel extracts information about the nodes and vertices. This is an inherently iterative algorithm with a similar approach to BSP [37].

3 Examples

Some examples of data intensive work in the UK and elsewhere which could presumably benefit from access to a larger facility and some joint development work.

3.1 Life Sciences: Biology and Bio-informatics

In biology, an obvious set of examples is represented by the so called next generation sequencing methods for nucleic acids. Recent reviews of these are given by [21, 30]. Big databases with fast access and information visualisation look like being important for this area of biology.

Modelling biological networks is inherently data driven and data intensive. The combinatorial nature of this type of modelling, however, requires new methods capable of dealing with the enormous size and irregularity of the search. Searching via "back tracking" is one possible solution that avoids exhaustive

searches by constraining the search space to the sub-space of feasible solutions. Despite its wide use in many combinatorial optimisation problems, there are currently few parallel implementations of backtracking capable of effectively dealing with the memory intensive nature of the process and the extremely un-balanced loads present.

Some other topics include: bio-medical studies in gene selection and filtering; feature selection algorithms for mining high dimensional DNA micro-array data; random matrix theory to analyse biological data.

Bio-informatics in this context is primarily concerned with sequence studies. Qiu *et al.* [27] provide a detailed description of current approaches to this.

3.2 Geo-science

Some topics include: processing extreme scale datasets in the geo-sciences; geo-spatial data management, e.g. with TerraFly (from NASA); parallel earthquake simulations on large scale multi-core super-computers.

3.3 Social Science and Security

One application is to discovering relevant entities in large scale social information systems. This typically involves graph based network analysis. A breadth first search will compute all nodes in the tree which can be accessed from a given node and gives the shortest path between them. This is done by processing an adjacency list iteratively down each connected node.

The adjacency list can be repressented as a sparse adjacency matrix. In Cailin's method, colouring is used to identify if a node is yet to be explored, is being explored or its depth has been found. This illustrates a problem with the basic Hadoop implementation. Since the algorithm is iterative, a global condition must be set to ensure termination when there are no more links still being explored. There is no support in the Hadoop framework to do this. See `http://www.johnandcailin.com/blog/cailin/breadth-first-graph-search-using-iterative-map-reduce-algorithm`.

3.4 Evolutionary Biology

Example of such work is from Prof. Mark Pagel FRS, University of Reading. `http://www.evolution.reading.ac.uk/`.

His group builds statistical models to examine the evolutionary processes imprinted in human behaviour, from genomics to the emergence of complex systems, to culture. Their latest work examines the parallels between linguistic and biological evolution by applying methods of phylogenetics, the study of evolutionary relatedness among groups, essentially viewing language as a culturally transmitted replicator with many of the same properties we find in genes. They are looking for patterns in the rates of evolution of language elements, and hoping to find the social factors that influence trends of language evolution.

Software and projects are based on Bayesian analysis and visualisation.

3.5 Astronomy, Astrophysics and Cosmology

Examples are from Jeremy Yates, DIRAC Consortium, see `http://www.ucl.ac.uk/star/research/miracle/dirac`.

Researchers at 27 HEIs in the UK are studying the properties of sub-atomic particles, cosmology and the physics of the early universe, the formation of large scale structure and galaxies, the properties of galaxies, the formation of stars and planets, astro-chemistry and the inter-stellar medium, the properties of exo-planets and planetary atmospheres, the physics of the sun and solar system, the properties of nano-particles, molecules and atoms in astrophysical conditions.

Software and projects include: AstroGrid; COSMOS; VIRGO; Miracle.

Elsewhere, projects like the Large Synoptic Sky Telescope, LSST, will begin collecting data in 2012 at a rate of approx. 500 MB/s which will need to be processed in real time. SWarp is used in the processing pipeline and is based on a Lanczos re-sampling filter [4]. This normalises the images to the sky and makes it possible to recognise new or changed features by comparing images.

3.6 ADMIRE

ADMIRE, Advanced Data Mining and Integration Research for Europe, is an EU project which includes EPCC and NeSC, see `http://www.admire-project.eu/`.

The main application areas addressed are: gene annotation, flood forecasting using data mining on observed data linked to simulations, analytical CRM from call centre data, search for quasars using digital telescope data, analysis of seismology databases.

One of the deliverables of the project is DISPEL, a parallel process engineering language.

The ADMIRE software stack is open source, see `http://www.admire-project.eu/` and `http://sf.net/projects/admire/`.

3.7 AI and Computational Linguistics

Language processing is becoming increasingly important, for instance in analysis of text on the Web. This is not confined to the English language. It requires processing large document streams.

Some work at University of Bristol, group won the 2009 Morpho challenge for machine learning. Other work by Prof. Paul Watry, University of Liverpool, see `http://www.liv.ac.uk/english/staff/paulwatry.htm`.

Current focus is on developing and implementing technologies in the areas of computational linguistics and textual analysis. Current work in understanding the set of relationships that exist between human

and machine defined semantics and exploring the application of semantic grid technologies to improve discovery and generate knowledge. Related work in the analysis of electronic document structure, data and text mining, natural language processing, corpus linguistic tools (information retrieval), annotation and visualisation technologies, persistent archives and new media.

Software and projects from the Liverpool group include: Cheshire III Digital Library software; iRODS; SHAMAN; PrestoPrime. See also `http://www.nactem.ac.uk/`.

One processing techique is to use n-grams consisting of a sequence of `n` characters or words. The algorithm extracts the `t` most frequent n-grams found. See Wikipedia `http://en.wikipedia.org/wiki/N-gram`. Document profiles can be compared, e.g. to determine their similarity or what language they are written in by closeness to the language training set profile. Languages can be modelled statistically as n-grams, based on Shannons' theories of information. These are sometimes Markov models. FPGAs can be used for this purpose, e.g. via a parallel Bloom filter as a probabilistic test of set membership [10]. N-grams are also used in gene sequence analysis.

Graph analysis is another application described in [10].

There is a book on text processing with MapReduce [19].

3.8 Data Analytics

This typically refers to what is done with customer derived commercial data using SQL or a combination of SQL and MapReduce. It can yield information used in: predictive and granular forecasting; trend analysis and modelling; credit and risk management; fraud detection; cross platform advert and event attribution; cross platform media affinity analysis; merchandising and packaging optimisation; service personalisation; graph analysis; consumer segmentation; consumer buying patterns and behaviour; click stream analysis; compliance and regulatory reporting.

Sometimes the acronym OLAP is used for On Line Analytic Processing. It can be found that an RDBMS approach does not scale to the size requires for this kind of semi-structured data, and solutions like Hadoop Hive and PIG are used instead.

4 Data Exploration

This section discusses some other "non-traditional" ways to analyse large data sets resulting from numerical modelling and simulation.

4.1 Visualisation

Visualisation typically uses large volumes of data. Rendering is carried out to view and study parts of the data, e.g. to produce surface or volume images of multi-dimensional data. It is usual to want to scan through this data interactively, viewing successive parts, for instance rendering slices through a volume to explore the interior structure. Other activities are to map functions using artefacts such

as streamlines or vectors (hedgehog plots). It is also likely that these will be composed together as a "movie". Some examples of work performed on HECToR are given by Turner and Leaver [36].

The visualisation workflow is thus to extract a part of the data and then manipulate and render it. In terms of what is commonly referred to as the standard visualisation pipeline the steps are: (i) read and filter data; (ii) map; (iii); render; and (iv) display. This involves a combination of data extraction, computation and rendering which is greatly facilitated by an appropriate dataset format. For instance slicing multi-dimensional meteorological data is facilitated by array structured formats such as GRIB. Software is often build from modules, each of which is responsible for one of the pipeline steps (e.g. parallel rendering) and may produce intermediate file output requiring high bandwidth i/o.

A certain amount of filtering can be applied to reduce the volume of data required in this process. Visualisation doesn't always need 64-bit, so reduction is possible. It may also be possible to extract a sub-set of data to manipulate if the full volume is not required for rendering. This can help if the data needs to be migrated to the visualisation system. An alternative is to do remote visualisation but rendering with data *in situ*.

Many visualisation software packages exist which are intended to be used with data in one or more of the available scientific formats discussed in [2]. Here are pointers to some lists of information about this software. An overview can be found on Wikipedia at `http://en.wikipedia.org/wiki/Visualization_computer_graphics`. Brief descriptions and pointers to software that can be used with netCDF is at `http://www.unidata.ucar.edu/packages/netcdf/utilities.html`. A page of links to many scientific visualisation and graphics software packages is at `http://static.msi.umn.edu/user_support/scivis/scivis-list.html`.

4.2 Server Side Visualisation

The primary techniques for building visualisation servers are as follows.

Parallel Architecture: uses multiple graphics cards for one application, sharing the load. These graphics accelerators could be attached to a single system, e.g. with large memory and multiple CPUs to render large data sets. A cluster with graphics accelerators can form a parallel rendering system reducing data available on local disks.

Parallel Decomposition: splits the problem across multiple graphics cards in the application's data space or in display space.

Open Software Architecture: addresses server side rendering at middleware, API, or application levels.

Software such as Chromium, FieldView, ParaView, OpenSceneGraph, DMX, VisIt and SciRun can also do parallel rendering so it is important to be able to manipulate the data inside a cluster to enable this, for instance re-ordering the data from the original application to an order suitable for the rendering package. This could be done in MapReduce. Table 4.2 lists properties of some well known packages.

Table 5: Some Visualisation Packages

Package Name	Comments/ File Formats	Web site
Amira (licensed)	*, Amira, Analyze, various images, netCDF, DXF, PS, HTML, Hoc, Icol, Interfile, Matlab, Nifti, Open Inventor, PSI, Ply, raw binary, SGI RGB, STL, SWC, Tecplot, VRML, Vevo	http://www.amiravis.com
AtomEye	PDB, standard CFG and extended CFG. Provides scripts to convert between PDB and CFG as well as for converting VASP files into CFG	http://mt.seas.upenn.edu/Archive/Graphics/A/
Avizo (licensed)	* see http://www.uni-ulm.de/fileadmin/website_uni_ulm/kiz/it/software-betriebssysteme/software-liste/Visualisierungssoftware/Avizo/VSG_Avizo_FileFormats.pdf	http://www.vsg3d.com
AVS/ Express (licensed)	** netCDF, etc. see http://personal.cscs.ch/~mvalle/AVS/Reader_Status_List.html	http://www.avs.com
Chimera	see http://www.cgl.ucsf.edu/chimera/docs/UsersGuide/filetypes.html	http://www.cgl.ucsf.edu/chimera/
Chromium	parallel rendering	http://chromium.sourceforge.net
DMX	multi-display X	http://dmx.sourceforge.net/
EnSight (free or licensed)	CFD and multi-physics, netCDF, etc. see User Manual Chapter 11	http://www.ensight.com
Environmental Workbench	netCDF, ...	http://www.ssesco.com/files/ewb.html
IDL	netCDF, ...	http://www.rsinc.com/idl/index.cfm
FieldView	Plot3D, FV-UNS, etc.	http://www.ilight.com/products.php
OpenSceneGraph		http://www.openscenegraph.org/projects/osg
OpenDX	**, CDF, netCDF, HDF-4, HDF-5, DX, various images, mesh, ...	http://www.opendx.org
ParaView	netCDF, HDF-5, ParaView, EnSight, VTK, Exodus, XDMF, Plot3d, SpyPlot, DEM, VRML, Ply, PDB, XMol, Stereo Lithography, Gaussian Cube, AVS, Meta Image, Facet, various images, SAF, LS-Dyna, raw binary	http://www.itk.org/Wiki/ParaView
PV-Wave (licensed)	netCDF, ...	http://www.vni.com/products/wave/index.html
SciRun	**, Nimrod, ...	http://www.sci.utah.edu

VisIt	Analyze, Ansys, BOV, Boxlib, CGNS, Chombo, CTRL, Curve2D, Ensight, Enzo, Exodus, FITS, FLASH, Fluent, FVCOM, GGCM, GIS, H5Nimrod, H5Part, various images, ITAPS, MFIX, MM5, NASTRAN, Nek5000, netCDF, OpenFOAM, PATRAN, Plot3d, Point3D, PDB, SAMRAI, Silo, Spheral, STL, TecPlot, VASP, Vis5D, VTK, Wavefront OBJ, Xmdv, XDMF, ZeusMP (HDF-4)	`https://wci.llnl.gov/codes/visit`
VMD	molecular coordinates, dynamics and fields, see `http://www.ks.uiuc.edu/Research/vmd/plugins/molfile/`	`http://www.ks.uiuc.edu/Research/vmd/`

* Amira also has a number of "packs" available, such as Molecular Pack, Mesh Pack, VR Pack, etc. which support additional application specific file formats.

* Avizo is also packaged in six editions with extension modules: Standard; Earth; Wind; Fire and Green.

** OpenDX, SciRun and AVS can be extended by adding special purpose file readers into the pipeline.

For more activities involving scientific visualisation in the UK, see the JISC funded VizNet project `http://www.viznet.ac.uk`.

4.3 Remote Visualisation

Some of the packages noted above can operate in client-server mode, this permits remote visualisation so that the data does not have to be moved. Instead, a visualisation server is installed on a system wih good connection to the data store and a client allows the user to control the images.

The primary techniques to provide easy, high performance access from remote networked clients are as follows.

Transmitting images: rather than graphical data, from servers to clients.

Compression: compressing images on the server and de-compressing them on the client to avoid the need for high network bandwidth. This generally increases performance but can make the CPU a bottleneck.

Widely used packages which operate in this way include the following.

FieldView: FieldView from Intelligent Light is a proprietary system similar to ParaView described below. See `http://www.ilight.com/products.php`. It is used as a post-processor for CFD

applications such as Fluent. FieldView will run in parallel and client-server mode and can be used for exploration of large data sets in an immersive VR environment.

IBM DCV: Deep Computing Visualisation. The DCV client can be freely installed and is used with a modified version of RealVNC to display 3D images from the server via a compressed OpenGL graphics stream. The DCV server can also be freely installed for academic use. DCV is now supported by NICE and used in the EnginFrame portal, see `http://www.nice-software.com/web/nice/products/dcv`.

ParaView: ParaView is an open source, multi-platform application designed to visualise data sets of various sizes. The goals of the ParaView project include supporting distributed computational models to process large data sets. It has an open, flexible, and intuitive user interface. Furthermore, ParaView is built on an extensible architecture based on open standards. It runs on distributed and shared memory parallel as well as single processor systems and has been tested on Windows, Linux, Mac OS-X, IBM BlueGene, Cray XT3 and various Unix workstations and clusters. Under the hood, ParaView uses the Visualization Toolkit (VTK) as the data processing and rendering engine and has a user interface written using the Qt cross platform application framework. See `http://www.paraview.org/Wiki/ParaView`.

SGI RemoteVUE: Part of the VUE family of products which includes PowerVUE. No longer available after SGI merged with Rackable Systems in 2009.

TurboVNC: Virtual Network Computing with accelerated JPEG compression. See `http://www.virtualgl.org/About/TurboVNC`.

VirtualGL: which interposes on un-modified Unix 3D applications, re-directing its 3D commands onto a server side 3D graphics accelerator, reading back rendered images, and converting the application images into a video stream with which remote clients can interact to view and control the 3D application in real time. The video stream is normally compressed using server specific optimisation. See `http://www.virtualgl.org/`.

VisIT: VisIT from LLNL is an open source package for parallel and distributed visualisation. This enables it to handle extremely large data sets interactively. VisIt's rendering and data processing capabilities are split into viewer and engine components that may be distributed across multiple machines. The Viewer is responsible for rendering and is typically run on a local desktop or visualisation server so that it can leverage the extremely powerful graphics cards that have been developed in the last few years. The Engine is responsible for the bulk of the data processing and i/o. It typically runs on a remote server where the data is located. This eliminates the need to move the data and makes high end compute and i/o resources available. The engine can be run serially or in parallel on the remote system. See `https://wci.llnl.gov/codes/visit`.

4.4 Data Mining and Feature Recognition

Techniques similar to visualisation also apply to processes of data mining and feature recognition, some examples of which were mentioned above. Sub-sets of data can be used and processes can be applied *in situ*. Example, looking for phase transitions in aluminium flourides [35]. Data Mining is sometimes referred to as KDD, Knowledge Discovery through Data.

Data mining is a process for extracting patterns from data. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection and is also used in scientific discovery. In recent years, data mining has been widely used in areas of science and engineering including bioinformatics, genetics, medicine, education, electrical power engineering, plasma physics experiments and simulations, remote sensing imagery, video surveillance, climate simulations, astronomy and fluid mixing experiments and simulations. An introduction is provided in a paper by Ramakrishnan and Grama [28] and a more recent review is provided by Kamath [16].

A project called Sapphire at LLNL is carrying out research to develop scalable algorithms for interactive exploration of large, complex, multi-dimensional scientific data sets. By applying and extending ideas from data mining, image and video processing, statistics and pattern recognition, they are developing a new generation of computational tools and techniques that are being used to improve the way in which scientists extract useful information from data. See `https://computation.llnl.gov/casc/sapphire/research.html` (last updated Mar'2009).

An interesting application is to code vaildation through comparison of simulations to experiment, for instance in studies of the Richtmyer-Meshkov instability, see [17] and `https://computation.llnl.gov/casc/sapphire/jacobsdata/jacobsdata.html`. This could become increasingly important as the complexity of the model increases, e.g. in fluids and plasmas, climate modelling, agent based modelling, etc.

Methodologies applied include Bayesian analysis, regression analysis, and self organising feature maps (SOMs or Kohonen maps). In addition to some standardisation efforts, freely available open source software have been developed, including Orange, RapidMiner, Weka, KNIME, and packages in the GNU R Project which include data mining processes. Most of these systems are able to import and export models in PMML (Predictive Model Markup Language) which provides a standard way to represent data mining models so that they can be shared between different statistical applications. PMML is an XML based language developed by the Data Mining Group (DMG), an independent group composed of many companies interested in data mining. PMML version 4.0 was released in June 2009, see `http://www.dmg.org`.

A useful places to look at activities around statisitical analysis and data mining are the communities using the Analytic Bridge Web 2.0 tool `http://www.analyticbridge.com/` and KDD Nuggets `http://www.kdnuggets.com/`.

4.5 Computational Steering

Computational steering is a potentially valuable procedure for scientific investigation in which the parameters of a running program can be altered interactively and the results visualised. As an investigative paradigm its history goes back more than ten years. With the advent of Grid computing, the range of problems that can be tackled interactively has widened, although some implementations have been very complicated. Applications specialists now see the prospect of accomplishing "real" science using computational steering whilst, at the same time, computing researchers are working to improve the supporting infrastructure of middleware, tool kits and environments that makes this possible.

EPSRC funded a collaboration network to develop and promote computational steering techniques from 2006-2008, see `http://compusteer.dcs.hull.ac.uk/index.php`.

Computational steering typically requires remote visualisation, see above. Some packages available include the following.

CSE: Computational Steering Environment, Centre for Mathematics and Computer Science, Netherlands, `http://www.cwi.nl/projects/cse/cse.html`;

CUMULVS: Oak Ridge National Laboratory, `http://www.csm.ornl.gov/cs/cumulvs.html`;

gViz: University of Leeds, `http://www.comp.leeds.ac.uk/vvr/gviz`;

Progress and Magellan: Georgia Institute of Technology

RealityGrid: University of Manchester, `http://www.realitygrid.org`;

SciRun: University of Utah, `http://software.sci.utah.edu/scirun.html`;

VASE: Vizualisation and Application Steering Environment, University of Illinois,

VIPER: University of Munich

A survey and comparison is presented in [22].

5 Acknowledgements

This work was partly funded by EPSRC through the SLA with Daresbury Laboratory.

The author would like to thank the following people for input:

Srikanth Nagella (RAL), Martin Turner (Manchester), Paul Watry (Liverpool),

References

[1] Y. Ahmad, R. Burns, M. Kazhdan, C. Meneveau, A. Szalay and A. Terzis *Scientific Data Management at the Johns Hopkins Institute for Data Intensive Engineering and Science.* In SIGREC (2010) `http://idies.jhu.edu`

[2] R.J. Allan *Management and Analysis of Large Research Data Sets* DL Technical Report (2011) draft See `http://www.grids.ac.uk/NWGrid/LargeData/`

[3] G. Bell, T. Hey and A. Szalay *Beyond the data deluge* Science 323 (2009) 1297-8

[4] E. Bertin *SWarp v2.17.0, Users Guide* Institut d'Astrophysique et Observatoire de Paris (2008) `http://terapix.iap.fr/IMG/pdf/swarp.pdf`

[5] G. Bikshandi, J. Guo, C. von Praun, G. Tanase, B.B. Fraguela, M.J. Garzarán , D. Padua and L. Rauchwerger *Design and Use of htalib a Library for Hierarchically Tiled Arrays*

[6] S. Chen and S.W. Schlosser *MapReduce meets wider varieties of Applications* Intel IRP-TR-08-05

[7] J. Dean and S. Ghemawat *MapReduce: Simplified Data Processing on Large Clusters* OSDI (2004) DOI: 10.1145/1327452.1327492.

[8] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, G. Fox *Twister: A Runtime for Iterative MapReduce* 1st Int. Workshop on MapReduce and its Applications at HPDC2010 (2010)

Twister Iterative MapReduce Pervasive Technology Institute, Indiana University. Web site: `http://www.iterativemapreduce.org/`

[9] B. Furht and A. Escalante (Eds.) *Handbook of Data Intensive Computing* (Springer, 2011) 962pp. ISBN: 978-1461414148

[10] M. Gokhale, J. Cohen, A. Yoo, W.M. Miller, A. Jacob, C. Ulmer and R. Pearce *Hardware Technologies for high performance Data Intensive Computing* IEEE Computer Society (2008) 0018-9162/08 32-48

[11] I. Gorton, P. Greenfield, A. Szalay and R. Williams *Computing in the 21st Century* IEEE Computer 41:4 (2008)30-32. `http://www.pnl.gov/science/images/highlights/computing/dic_special.pdfData-Intensive`

[12] C. He *Molecular Dynamics Simulation based on Hadoop MapReduce* M.Sc. Thesis (University of Nebraska, Lincoln, 2011)

[13] T. Hey, S. Tansley and K. Tolle (eds.) *The Fourth Paradigm: Data-Intensive Scientific Discovery* (Microsoft Research, Redmond, 2009) ISBN 978-0-9825442-0-4

[14] T. Hoeffer, A. Lumsdaine and J. Dongarra *Towards Efficient MapReduce Using MPI* Lecture Notes in Computer Science Vol.5759. M. Ropo, J. Westerholm, J. Dongarra (Eds.) (Springer, 2009) 24049

[15] M. Isard, M. Budiu, Y. Yu, A. Birrell and D. Fetterly *Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks* European Conference on Computer Systems (EuroSys) (Lisbon, 21-23/3/2007)

[16] C. Kamath *Scientific Data Mining: a Practical Perspective* SIAM (2009) 286pp ISBN 978-0898716757

[17] C. Kamath and T. Nguyen *Feature Extraction from Simulations and Experiments: Preliminary Results Using a Fluid Mix Problem* Technical report UCRL-TR-208853 (Lawrence Livermore National Laboratory, Jan'2005) `https://computation.llnl.gov/casc/sapphire/pubs/UCRL-TR-208853.pdf`

[18] R.T. Kouzes, G.A. Anderson, S.T. Elbert, I. Gorton and D.K. Gracio *The Changing Paradigm of Data-Intensive Computing* Computer 42:1 (2009) 26-3 `http://www.computer.org/portal/web/csdl/doi/10.1109/MC.2009.26`

[19] J. Lin and C. Dyer *Data Intensive Text Processing with MapReduce* Morgan and Claypool (Synthesis Lectures on Human Language Technologies) (2010) 178pp ISBN: 978-1608453429

[20] G. Malewicz, M.H. Austern, A.J.C. Bik, J.C. Dehnert, I. Horn, N. Leiser and G. Czajkowski *Pregel: a system for large-scale graph processing* In Proc. 28th ACM symposium on Principles of distributed computing (PODC '09) (ACM, New York, 2009) 6-6 DOI: 10.1145/1582716.1582723

[21] E.R. Mardis *Next generation DNA sequencing methods* Annual Rev. Genomics Hum. Genet. 9 (2008) 387-402

[22] J.D. Mulder, J.J. van Wijk and R. van Liere *A Survey of Computational Steering Environments* Elsevier Science (July 1998)

[23] B.-S. Park, N.F. Samatova, T. Karpinets, A. Jallouk, S. Molony, S. Horton and S. Arcangeli *Data-driven, data-intensive computing for modelling and analysis of biological networks: application to bioethanol production* J. Phys.: Conf. Ser. 78 (2007) DOI:10.1088/1742-6596/78/1/012061

[24] A. Pavlo, E. Paulson, A. Rasin, D.J. Abadi, D.J. Dewitt, S. Madden and M. Stonebraker *A Comparison of Approaches to Large-Scale Data Analysis* Proc. 35th SIGMOD Int. Conf. on Management of Data (2009). DOI: 10.1145/1559845.1559865. `http://www.cse.nd.edu/~dthain/courses/cse40771/spring2010/benchmarks-sigmod09.pdf`

[25] S. J. Plimpton and K. D. Devine, *MapReduce in MPI for Large-Scale Graph Algorithms* accepted for a special issue of Parallel Computing (2011)

[26] S. Plimpton and K. Devine *MapReduce-MPI Library* Sandia National Labs. Web site: `http://www.sandia.gov/~sjplimp/mapreduce.html`

[27] J. Qiu, J. Ekanayake, T. Gunarathne, J. Youl Choi, S.-H. Bae, Y. Ruan, S. Ekanayake, S. Wu, S. Beason, G. Fox, M. Rho and H. Tang *Data Intensive Computing for Bioinformatics* In "Data Intensive Distributed Computing: Challenges for large scale information management" ed. T. Kosar (IGI Global, 2012) 351pp. DOI: 10.4018/978-1-61520-971-2; ISBN: 978-1-61520-971-2

[28] N. Ramakrishnan and A.Y. Grama *Mining Scientific Data* Adv. Computers 55 (2001) 119-69

[29] P. Sethia *High Performance Multi-Agent System based Simulations* M.Sc. Thesis (IIIT, Hyderabad, June 2011)

[30] J. Shendure and H. Ji *Next generation DNA sequencing* Nat. Biotechnol. 26:10 (2008) 1135-45.

[31] Y. Simmhan, R. Barga, C. van Ingen, M. Nieto-SantiSteban, L. Dobos, N. Li, M. Shipway, A.S. Szalay, S. Werner, J. Heasley *GrayWulf: Scalable Software Architecture for Data Intensive Computing* HICSS'09. 42nd Hawaii International Conference on System Sciences (5-8/1/2009) 1-10 ISBN: 978-0-7695-3450-3 DOI: 10.1109/HICSS.2009.235

[32] A.S. Szalay and J. Gray *Science in an Exponential World* Nature 440 (2006) 23-24

[33] A.S. Szalay, G. Bell, J. Vandenberg, A. Wonders, R. Burns, D. Fay, J. Heasley, T. Hey, M. Nieto-SantiSteban, A. Thakar, C. van Ingen and R. Wilton. *GrayWulf: Scalable Clustered Architecture for Data Intensive Computing* HICSS'09. 42nd Hawaii International Conference on System Sciences (2009) 1-10. DOI: 10.1109/HICSS.2009.234

[34] S.-J. Sul and A. Tovchigrechko *Parallelizing BLAST and SOM algorithms with MapReduce-MPI library* IEEE International Parallel and Distributed Processing HICOMB Symposium (2011)

[35] J.M.H. Thomas, J. Kewley, R.J. Allan, J.M. Rintelman, P. Sherwood, C.L. Bailey, A. Wander, B.G. Searle, N.M. Harrison, S. Mukhopadyhay, A. Trevin, G.R. Darling and A.I. Cooper *Experiences with Different Middleware Solutions on the NW-GRID* Proc. UK e-Science All Hands Conference 2007 (EPSRC, 2007)

[36] M. Turner and G. Leaver *Remote Scientific Visualization for Large Datasets* EG-UK Theory and Practice of Computer Graphics (Elsevier, 2010)

[37] L.G. Valiant *A bridging model for parallel computation* Communications of the ACM 33:8 (1990)

[38] G. Wang, M.V. Salles, B. Sowell, X. Wang, T. Cao, A. Demers, J. Gehrke and W. White *Behavioural Simulations in MapReduce* Proc. 36th Int. Conf. on Very Large Data Bases 3:1 (VLDB Endowment, 2010) ISSN: 2150-8097/10/09

[39] T. White *Hadoop, the definitive Guide* (O'Reilly and Yahoo! Press, 2009, 2nd edn. 2010) ISBN: 978-1-449-38973-4

[40] SciDAC Scientific Data management Center. `https://sdm.lbl.gov/sdmcenter`

[41] *Data Intensive Computing by NSF* Data Intensive Computing (2009) `http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503324&org=IIS`

[42] *Data Intensive Computing by PNNL* Data Intensive Computing (2008) `http://www.cs.cmu.edu/~bryant/presentations/DISC-concept.ppt`

[43] i-MapReduce Web site: `http://code.google.com/p/i-mapreduce/wiki/iMapReduceIntroduction`