Scaling up Data Mining Techniques to Large Datasets Using Parallel and Distributed Processing

Frederic Stahl and Mohamed Medhat Gaber and Max Bramer

Abstract Advances in hardware and software technology enable us to collect, store and distribute large quantities of data on a very large scale. Automatically discovering and extracting hidden knowledge in the form of patterns from these large data volumes is known as data mining. Data mining technology is not only a part of business intelligence, but is also used in many other application areas such as research, marketing and financial analytics. For example medical scientists can use patterns extracted from historic patient data in order to determine if a new patient is likely to respond positively to a particular treatment or not; marketing analysts can use extracted patterns from customer data for future advertisement campaigns; finance experts have an interest in patterns that forecast the development of certain stock market shares for investment recommendations. However, extracting knowledge in the form of patterns from massive data volumes imposes a number of computational challenges in terms of processing time, memory, bandwidth and power consumption. These challenges have led to the development of parallel and distributed data analysis approaches and the utilisation of Grid and Cloud computing. This chapter gives an overview of parallel and distributed computing approaches and how they can be used to scale up data mining to large datasets.

Frederic Stahl
Bournemouth University, The School of Design, Engineering & Computing, Poole House, Talbot Campus, Poole, Dorset, BH12 5BB, United Kingdom, e-mail: fstahl@bournemouth.ac.uk

Mohamed Medhat Gaber
University of Portsmouth, School of Computing, Lion Terrace, Portsmouth, Hants, PO1 3HE, United Kingdom e-mail: Mohamed.Gaber@port.ac.uk

Max Bramer
University of Portsmouth,School of Computing, Lion Terrace, Portsmouth, Hants, PO1 3HE, United Kingdom e-mail: Max.Bramer@port.ac.uk

1 Performance Challenges in Data Mining

There is a substantial commercial interest in developing and improving business intelligence and data mining applications in order to extract useful information in the form of patterns from very large data volumes. Computer systems capture our lives in the form of credit card transactions; loyalty reward systems record our shopping habits; CCTV cameras, GPS systems embedded in our smartphones and navigation systems record our movement and whereabouts; and the world wide web records our data through applications such as facebook, email, twitter and blogs [52]. In [22] the authors estimated that in the year 2020 the size of our digital universe will be 44 times as big as it was in the year 2009. Advances in storage technology make it possible to store all these data volumes at a very low cost, hence the ubiquitous challenge in data mining is the scalability of data mining techniques to these large data volumes. Many areas in science are confronted with the problem of scalability of data mining techniques also. For example in cosmology, researchers store terabytes of image data in massive databases such as in the Sloan digital sky survey [51, 49]. The bioinformatics community is just starting to be able to store and analyse molecular dynamics simulation data which can easily comprise hundreds of gigabytes for only one simulation experiment [3]; also in bioinformatics the human genome project stores the entire genetic blueprint of our bodies [35]. In the business area large and complex databases are reported, for example Amazon's two largest databases combine 42 terabytes of data and AT&T's largest database comprises 312 terabytes [34]. Loosely speaking, the scientific and business worlds are confronted with very large data bases storing information. In order to extract meaningful patterns, analysts need to apply data mining techniques. Hence scalable data mining technologies are required.

A further complication to the mining of these massive amounts of data is the fact that organisations often store their data in geographically distributed locations in order to overcome bandwidth problems when transferring such large amounts of data to a central data repository. This generates further problems such as heterogeneous data base schemas and confidentiality issues when dealing with sensitive data. Some research has been conducted in order to overcome these bandwidth constraints such as in the DataMiningGrid.org project [48]. One of their approaches is to deploy individual data mining close to the data sources in a dynamic way and execute them remotely rather than downloading myriads of data. However this chapter is about scaling data mining algorithms to deal with large datasets rather than dealing with geographically distributed data. For a comprehensive reading list about Distributed Data Mining approaches that can be used to mine geographically distributed data sources the reader is referred to [4].

The rest of this chapter is organised as follows. Section 2 highlights parallel and distributed data mining approaches to tackling the problem of scalability of data mining techniques. Section 3 discusses approaches to scaling up data stream mining techniques in resource constraint environments. Section 4 provides a summary of successful applications, available open and commercial parallel data mining sys-

tems are discussed. This chapter closes with a discussion of remaining challenges in scaling up data mining to large datasets in Section 5.

2 Parallel and Distributed Data Mining Approaches and Frameworks

Parallel and distributed data mining approaches have been proposed in the past in order to tackle the challenge of scalability to large data sources. Whereas parallel data mining clearly refers to the parallelisation of a data mining task by executing data mining tasks concurrently, the term distributed data mining is used ambiguously in the data mining literature. Often the term 'distributed' is associated with data mining of geographically distributed datasets and is not necessarily concerned with the computational scalability. The parallelisation of a data mining task often follows a data parallel approach as the computational workload of data mining tasks is usually directly dependent on the amount of data that needs to be processed [43]. In data parallelisation the data is partitioned into smaller subsets and distributed to multiple processors on which the data mining tasks are executed concurrently. Unless stated otherwise this chapter uses both the terms parallel data mining and distributed data mining to refer to *data parallelism*.

2.1 Multiprocessor Computer Architectures

In order to execute data parallel algorithms a multiprocessor architecture is needed. The basic idea in data parallelism is to distribute or assign the workload to several processing units in the form of subsets of the dataset. There are two relevant multiprocessor architectures that are suited for this purpose, *tightly-coupled* architectures and *loosely-coupled* architectures. However hybrids between the two architectures are possible.

(a) A *loosely-coupled* architecture comprises multiple standalone computers. Each computer comprises one processing unit and its local private memory. This architecture requires data distribution and accumulation mechanisms, and a communication network. An implementation of this architecture is also often referred to as *'Massively Parallel Processors'* (MPP). MPPs are illustrated in Figure 1(a).
(b) A *tightly-coupled* architecture consists of multiple processors that share a common memory using a shared bus system. No data distribution is required as the processors do not have a private memory. An implementation of this architecture is also often referred to as *'Shared memory MultiProcessor machines'* (SMP). SMPs are illustrated in Figure 1(b).

(c) A hybrid approach between both architectures is possible by building a loosely-coupled system out of 'Shared memory Multiprocessor machines'. SMP/MPP hybrids are illustrated in Figure 1(c).

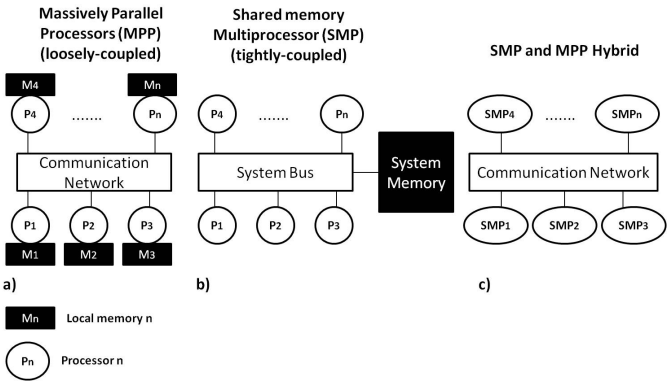

Fig. 1 Multiprocessor architectures. (a) Loosely-coupled multiprocessor architecture. (b) Tightly-coupled multiprocessor architecture. (c) A hybrid of loosely- and tightly-coupled architectures.

A loosely coupled system uses a communication network in order to communicate data, whereas a tightly-coupled system uses the system bus. Regarding tightly coupled systems there is a bottleneck with the number of processors the system can support in the context of data mining applications. This is because the more processors share the system bus, the less bandwidth is available per processor. A loosely coupled system does not have this bottleneck as the processors do not share a common system bus. A further advantage of a loosely-coupled system is that the application components are hosted on different computers distributed in a network. This makes loosely-coupled systems more robust to hardware failures compared with tightly coupled systems as a failing processing node will not cause the entire application to fail. In a tightly-coupled system a failing processor usually represents a single point of failure. However a disadvantage of a loosely-coupled system is that it requires communication and collaboration between its computing nodes which introduces an additional overhead for the application. An advantage of a tightly-coupled system over a loosely-coupled system is that it is usually more efficient at processing data as it avoids data replication and does not need to transfer information between processing units. Yet loosely-coupled systems can be obtained at a relatively low cost compared with tightly coupled systems, if standard workstations are used as processing units. This allows modest sized organisations which cannot afford a SMP to harvest the computational power and memory storage of their whole local area network in order to execute parallel data mining tasks. Also a loosely-coupled system can be upgraded gradually by simply replacing old workstations with newer ones, whereas a tightly-coupled system needs to be replaced in its entirety.

Hybrid architectures of loosely-coupled SMPs are becoming common and most workstations nowadays can be seen as small MPPs as they utilise multicore processor technology. However, research in the area of data mining has just begun to utilise such hybrid systems efficiently. The data mining community is called on to investigate the advantages of such hybrids, their suitability and benefits for scaling up data mining systems to large data volumes.

2.2 Parallel Predictive Algorithms

One of the most important data mining tasks is classification rule induction, which can be categorised into 'divide and conquer' and 'separate and conquer' methods [52]. 'Divide and conquer' generates a decision tree by recursively partitioning the training data according to the variables and classifications, aiming to optimise a chosen metric such as the entropy [36]. 'Separate and conquer' generates a ruleset by specialising a general rule for a certain target class on the training data. After each rule induced the subset of the training examples that is covered by the rules induced so far is deleted and the next rule is induced until all training examples are covered by the ruleset. Examples of 'separate and conquer' classifiers are [14, 13] and of the Prism family of algorithms are [10, 6, 5].

Most recent research does not focus on the parallelisation of decision tree induction, but rather on concurrent execution of whole data mining tasks, we highlight general parallel decision tree induction approaches in the section as they give a valuable insight into issues that may occur when parallelising data mining tasks. There are two principal ways of parallelising decision tree classifiers: the *synchronous tree construction* approach highlighted in Figure 2a and the *partitioned tree construction* approach highlighted in Figure 2b [41].

In the'synchronous tree construction' the training dataset is initially distributed between n processors. During the tree induction each processor holds an exact copy of the tree in its memory (assuming a MPP machine has been used). The processors cooperate in expanding the same tree node by gathering statistics of their portion of the data and sharing these statistics through communication. Eventually each processor will perform the same tree node expansion independently on their copy of the tree. Some of the most well-known parallel tree classifiers are based on 'synchronous tree construction' with vertical partitioning [39]. In the 'Partitioned Tree Construction' different processors work on different parts of the tree and the training data. Initially only one processor is assigned to expand the root node. The resulting child nodes are then assigned to different processors, each independently expanding the subtree of its child node. This is done recursively until all processors are assigned to different subtrees.

The advantage of the 'Synchronous tree construction' is that there is no communication of training data. However, the disadvantage of 'Synchronous tree construction is that the communication of statistics increases as the tree grows. Also the workload between the processors is changing during the tree induction and may

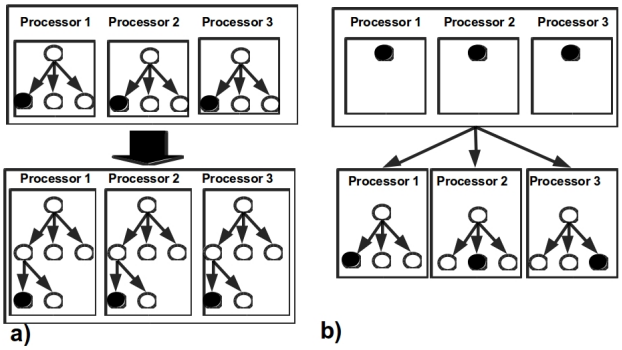

Fig. 2 Parallel decision tree induction approaches. The data is evenly distributed between all processors. (a) shows the synchronous tree construction approach. All processors keep the same decision tree in memory and cooperate on expanding the same tree node. (b) shows the partitioned tree construction approach. Different processors are assigned on to different subtrees (when possible) and expand these subtrees simultaneously and independently.

cause workload imbalances [41]. The advantage of 'Partitioned Tree Construction' is that as each processor works independently no communication is needed. However, the disadvantage of the 'Partitioned Tree Construction' approach is that initially a single processor has the entire workload [41]. Taking the aforementioned advantages of both approaches and minimizing the disadvantages has resulted in a hybrid approach [41]. This approach starts with the 'synchronous tree construction' until the communication overhead becomes too high and then switches to the 'partitioned tree construction', which removes the entire communication.

This illustration of different parallel decision tree based approaches shows the difficulties encountered when parallelising data mining algorithms, i.e. communication overheads, synchronisation overheads and workload imbalances. This has led to the development of frameworks that support the parallelisation of whole families of algorithms. The remainder of this section highlights frameworks and infrastructure technologies to assist with the parallelisation of data mining algorithms.

2.3 Parallel Formulations of Separate and Conquer Classification Rule Induction using PMCRI

The Parallel Modular Classification Rule Induction (PMCRI) framework [44, 46] is a framework for parallelising algorithms of the Prism family whose members follow the 'separate and conquer' approach.

Figure 3 outlines PMCRI's basic architecture, which is based on a distributed blackboard system. A blackboard system is often illustrated using the metaphor of several experts, with expertise in different domains, that are gathered around a physical blackboard. The experts can solve a problem they have in common by using

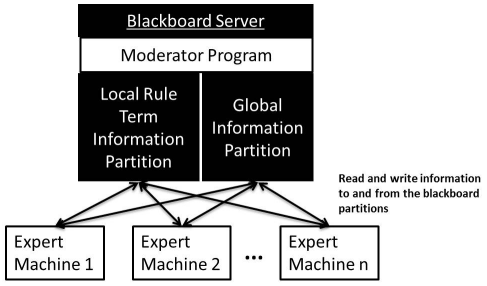

Fig. 3 The PMCRI framework based on a distributed blackboard architecture.

their own expert knowledge and information written on the blackboard in order to infer new knowledge and information. Experts share information by writing it on the blackboard. The software implementation of a blackboard system can be realised by a client server architecture [32]. Experts are client machines whose expertise is represented by the portion of the data they hold in memory and the blackboard is a communication server. PMCRI divides the blackboard into several logical partitions that have different meanings to the experts. PMCRI's blackboard is divided into two logical partitions: one to submit local rule term information and one to retrieve global information about the whole algorithm's status. PMCRI partitions the data vertically (according to the features) and distributes the subsets evenly amongst the expert machines. The experts induce each rule concurrently by inducing rule terms that are the 'best' terms in the local feature subspace to further specialise a rule. The experts use the blackboard in order to communicate which rule term is globally the best one and to assembly the final rule. Figure 3 also highlights a moderator program. The moderator is also implemented in the form of an expert machine. It coordinates the rule induction schedule and thus represents the underlying Prism algorithm. Just replacing the moderator by a different one allows one member of the Prism family to be changed to another.

2.4 The MapReduce Paradigm for Parallelisation Data Mining

Google's *MapReduce* paradigm of parallel programming [16] provides a means to simplify the development of parallel data mining techniques offering load balancing and fault tolerance. The actual source code regarding parallelisation and data communication is hidden from the programmer by limiting the parallel programming model to only the *map* and the *reduce* functions. Data mining applications parallelised using MapReduce make use of the Google File System (GFS) [23] which provides a means of storing data in a distributed manner and redundantly over a network of commodity workstations. Google's MapReduce is a proprietary software. However, Hadoop provides an open source implementation of Google's MapReduce paradigm based on its Hadoop Distributed File System (HDFS) which is Hadoop's

implementation of GFS [25]. MapReduce splits an application into smaller parts called *Mappers*. Each mapper can be processed by any of the workstations in the nodes in the cluster. A high reliability of the application is provided by the framework's ability to recover a failed mapper. Intermediate results produced by these mappers are then combined by one or more *Reducer* nodes.

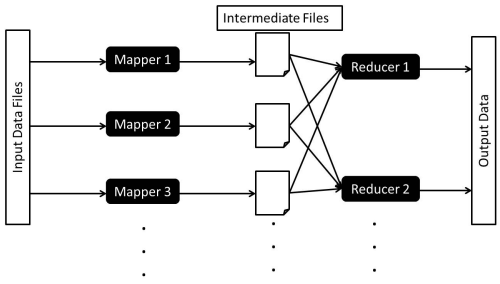

Fig. 4 A typical setup of a Hadoop computing cluster. A physical node in the computer can execute more than one Mapper and Reducer.

A typical Hadoop computing cluster is highlighted in Figure 4. Large amounts of data are processed by splitting this data into smaller portions and storing it redundantly in the cluster using HDFS. Then the smaller portions of the data are loaded into the mapper machines and processed using a user defined function. The results computed by the mappers (intermediate files) are passed on the reducers where they are combined. The programmer is only required to implement the processing functions used by the mappers and the combining function by the reducers.

MapReduce's relevance in the data mining community has been demonstrated in countless projects. For example Google reported having utilised MapReduce in at least 900 projects [16]. However, this was in 2008 and it is very likely that there are many more projects by now. For example Google developed MiniHash clustering using the MapReduce paradigm in order to generate personalised recommendations for users of Google News [15]. Google also makes use of MapReduce to cluster billions of images in order to find new duplicates [30]. The authors of [33] used MapReduce to create the Parallel Learner for Assembling Numerous Ensemble Trees (PLANET) system. The PLANET system provides a scalable parallel decision tree classifier, regression trees and also parallel ensemble learners.

A data mining approach that lends itself to parallelisation using MapReduce is *ensemble learning*. In ensemble learning multiple models are generated from subsamples or bags of the same data and combined in order to achieve a better predictive performance. Ensemble learners lend themselves to parallelisation as the multiple models can potentially be generated concurrently using multiple processors. Recently published work that aims to parallelise ensemble learners is reported in: [33, 53, 2, 45]. The authors of [12] adapted MapReduce in order to parallelise several learning algorithms: amongst others algorithms such as k-means, logistic regression, naive bayes, support vector machines etc. Also the partitioned tree con-

struction approach as highlighted in Section 2.2 could be parallelised using MapReduce by assigning the construction of different subtrees to different mappers.

However as pointed out by [8], MapReduce is designed for use in a dedicated cluster assuming that each node is equally powerful, reliable and only dedicated to processing. The authors of [8] further propose to remove some of these assumptions and create a more 'grid like' version of MapReduce.

2.5 Grid and Cloud Computing for Parallel Data Mining

The grid and cloud computing paradigm has emerged as an attractive computing infrastructure for implementing complex and computationally costly applications. The difference between grid and cloud computing is often blurred in the literature. In grid computing users consume but also share computing resources whereas cloud computing is a rather commercial paradigm where computing resources are offered through services on demand against payment.

The computing grid is often described with the analogy of an electrical power grid that transparently provides electricity to the end user. According to this analogy the computing grid provides computing power in terms of CPU time, memory and storage. The grid aims to mass computing resources whilst hiding their specifications. Thus it provides a consistent interface for the end user to access high performance and/or high throughput computation [31]. A grid is organised in geographically distributed virtual organisations. Virtual organisations are collections of computational resources in terms of processing and storage. In general computing grids are constructed using a set of grid software libraries, the middleware, whose service-oriented architecture provides services to access virtual organisations for software applications that want to make use of the grid. An example for a widely used open source grid middleware is the Globus Toolkit [24].

Data mining in a grid environment is a special form of distributed data mining [47] as it is stimulated by sharing of resources using local and wide area networks [29]. Several projects concerned with high performance data mining using grid environments have been followed up in the past couple of years, such as the *DataMiningGrid.org* project which aims to integrate a wide variety of of data mining applications and different application scenarios into a single grid framework [47, 48], or the gridminer project [7] which aims to integrate the entire knowledge discovery process in a service-oriented grid application.

Cloud computing refers to the on demand delivery of applications and hardware resources over the internet in the form of services. Three types of service can be accessed using a cloud infrastructure. Both grid computing and cloud computing aim to provide access to large computing and storage resources. However, the cloud additionally uses visualisation in order to provide access to the computing resources and thereby conceals physical heterogeneity, geographical distribution, and faults [37]. Compared with the cloud the grid provides services that enable the collaborative sharing of distributed computing resources. In this sense the grid is comple-

mentary yet independent from cloud computing [37]. In other words, Grid computing aims to solve computational problems whereas cloud computing provides software and computing services on demand. Because of this on demand principle cloud computing is usually used in the private sector whereas grid computing is used more in public sector research projects. Probably the best known commercial cloud system is *Amazon Web Services* [1], comprising Amazon's *S3* (Simple Storage Service) providing data storage, and Amazon's *EC2* service providing on demand computing capacity. However, the disadvantage of grid and cloud computing is that consumers give away partial control of security management to the grid partners or cloud service providers. For a discussion of this topic the reader is referred to [42].

Scaling up data mining is not only restricted to problems that deal with large datasets. Performing data mining tasks on resource-constrained devices like smartphones and small sensing devices stimulates the need for new strategies for scaling up data mining techniques. The following section is devoted to the discussion of strategies of dealing with this problem. In fact, this problem represents the other side of the coin, traditional scaling up of data mining techniques look at having high peformance hardware and large data sets, while scaling up data mining techniques for resource-constrained devices look at having possibly smaller data sets to be analysed using restricted processing capabilities.

3 Approaches to Scaling up Data Stream Mining in Resource Constrained Environments

We witness the era of handheld devices and small sensors performing tasks that in the near past required high performance systems. Data streams in and/or produced on such small devices can serve a number of extremely important applications in areas such as astronomy, stock market analysis and national security, to name a few. The following two important facts require the processing of data streams to be performed locally on-board small devices with low computational power. We shall refer to such devices in this chapter as resource-constrained environments.

1. It has been proven experimentally that local data processing is an energy efficient alternative to sending the data streams to a computational service with high power like the cloud [19]; and
2. current computational capabilities of resource-constrained environments do allow the performance of complex tasks, like data mining [11].

Despite the continuous advances in the computational capabilities of resource constrained devices, the demand of having increasingly complex computational tasks performed has also been continuous. Thus, we encounter what we can refer to as *relative resource constraints*, i.e. the advances in the hardware technologies fall short of addressing the demands of current application needs. The application needs are in fact coupled with the rise of large amounts of streaming data, which in turn led to the *big data* phenomenon. To address the issue of *relative resource*

constraints, adapting the process to resource availability is needed. The *Algorithm Granularity* approach [18] is a generic framework that is able to adapt any data stream mining to data rate and resource availability. The following subsection gives an overview of the approach.

3.1 Algorithm Granularity Approach Overview

Algorithm Granularity approach has been introduced by Gaber with a comprehensive treatment of the subject reported in [17]. The approach relies on the concept of *resource consumption patterns*. Resource consumption patterns represent the change in resource consumption over a period of time, referred to as a time frame. The algorithm can change its settings from its three entry points: input, output, and processing. This could be applied to any data stream processing technique. However, the *Algorithm Granularity* approach was specifically designed for adapting data stream mining techniques. The three categories of settings of a mining algorithm are changed over time to cope with the availability of resources and current data rate. Definitions of these settings are as follows:

Algorithm Input Granularity (AIG) represents the process of changing the data rates that feed the algorithm. Examples of this include sampling, load shedding, and creating data synopsis. This is a common solution in many data stream mining techniques.

Algorithm Output Granularity (AOG) is the process of changing the output size of the algorithm in order to preserve the limited memory space. In the case of data mining, we refer to this output as the number of knowledge structures. For example the number of clusters or rules. The output size could be changed also using the level of output granularity which means the less detailed the output, the higher the granularity and vice versa.

Algorithm Processing Granularity (APG) is the process of changing the algorithm parameters in order to consume less processing power. Randomisation and approximation techniques represent the potential solution strategies in this category.

It has be noted that there is a collective interaction among the above three categories. *AIG* mainly affects the data rate and it is associated with bandwidth consumption and battery life. Batteries tend to be drained rapidly when continuously sending or receiving data streams. On the other hand, *AOG* is associated with memory and *APG* is associated with processing power. The more memory is consumed, this implies that more knowledge structures are resident in the memory, and thus *AOG* is the suitable strategy. When the algorithm falls short of processing the incoming streams, algorithmic approximation and randomisation are used to speed the process up. Thus, *APG* appears as the suitable strategy. Figure 5 [21] shows the interaction among the three strategies and associated computational resources.

However, the change in any of the three affects the other resources. For example, approximation could be used to address the problem of high data rate by having less processing time per data record. It is important to note that the process of enabling

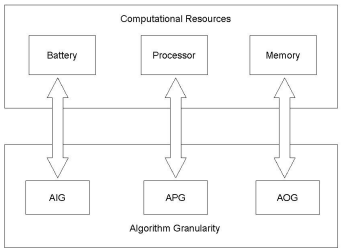

Fig. 5 The Effect of Algorithm Granularity on Computational Resources

resource awareness should be very lightweight in order to be feasible in a streaming environment characterised by its scarcity of resources.

3.2 Formalisation of Algorithm Granularity

The Algorithm Granularity requires continuous monitoring of the computational resources. This is done over fixed time intervals/frames that are denoted as TF. According to this periodic resource monitoring, the mining algorithm changes its parameters/settings to cope with the current consumption patterns of resources. These parameters are AIG, APG and AOG settings discussed briefly in the previous section. It has to be noted that setting the value of TF is a critical parameter for the success of the running technique. The higher the TF is, the lower the adaptation overhead will be, but at the expense of risking a high consumption of resources during the long time frame, causing the run-out of one or more of the computational resources.

The use of *Algorithm Granularity* as a general approach for mining data streams will require us to provide some formal definitions and notations. The following are definitions and notation that we will use in our discussion.

R: set of computational resources $R = \{r_1, r_2, \ldots, r_n\}$
TF: time interval for resource monitoring and adaptation.
ALT: application lifetime.
ALT': time left to last the application lifetime.
$NoF(r_i)$: number of time frames to consume the resources r_i, assuming that the consumption pattern of r_i will follow the same pattern of the last time frame.
$AGP(r_i)$: algorithm granularity parameter that affects the resource r_i.

According to the above, the main rule to be used to use the algorithm granularity approach is as follows:

IF $\frac{ALT'}{TF} < NoF(r_i)$
THEN SET $AGP(r_i)+$

ELSE SET $AGP(r_i)-$

Where $AGP(r_i)+$ achieves higher accuracy at the expense of higher consumption of the resource r_i, and $AGP(r_i)-$ achieves lower accuracy at the advantage of lower consumption of the resource r_i. For example, when dealing with clustering, it is computationally cheaper to allow incoming data instances in the stream to join an existing cluster with randomisation applied to which cluster the data instance would join. Ideally, the point should join a cluster that has sufficient proximity or a new cluster should be created to accommodate the new instance. This strategy has been applied to a technique by Gaber and Yu [20] termed *RA-Cluster*.

This simplified rule could take different forms according to the monitored resource and the algorithm granularity parameter applied to control the consumption of this resource. The *Algorithm Granularity* approach has been successfully applied to a number of data stream mining techniques. These techniques have been packaged in a java-based toolkit, coined *Open Mobile Miner* [28].

This section gives an overview of the *Algorithm Granularity* approach. For a comprehensive treatment of the subject area, the reader is referred to [17]. For interested readers in the data stream mining area including this approach, Gama's textbook [21] is our suggested source.

4 Software Tools and Applications

This section highlights some of the readily available parallel and distributed mining tools, highlights several successful applications. The success in merging data mining with distributed computing technologies has lead to several distributed and parallel commercial as well as freely available data mining tools. Probably the most popular commercial system is Amazon's cloud [1], as mentioned in Section 2.5, it provides data storage, data analytics tools and on demand computing capacity. Amazon's cloud can only be remotely accessed whereas SAS's 'Analytics Infrastructure' can be deployed on site as well being used as a cloud service. SAS's 'Analytics Infrastructure' can be deployed on SMP or on MPP parallel architectures and comprises three means to scale up data analysis to large datasets. These are grid computing, moving computation close to the data in order to avoid data movement, and in memory analytics in order to reduce data access time to disk storage. A further commectial product is Microsoft's SQL Server which offers a scalable data warehouse implementation that can be hosted on MPP architectures. However, there are also free products such as the well known WEKA data mining software which allows parallel cross-validation calculations [9]. The Hadoop infrastructure [25] highlighted in Section 2.4 is a freely available technology, still it is widely used in commercial business intelligence application.

With the rapid development of parallel and distributed data mining technology, several successful applications have been reported. For example Google's PLANET system [33] mentioned in Section 2.4 has been applied in the domain of 'compu-

tational advertising'. Two of the applications of the 'DataMiningGrid.org' project mentioned in Section 2.5 are in the automotive industry [47] for text mining as well as in bioinformatics for the distributed storage and analysis of very large quantities of Molecular Dynamics simulation data [50]. Parallel processing capabilities of grid architectures are the method of choice for the analysis of very large astronomical datasets such as in [54]. Probably the most popular application of grid computing is the SETI@home project. The project's goal is to detect intelligent extraterrestrial life through the analysis of massive radio telescope data [38].

5 Conclusions and Future Directions

This chapter presented techniques that can be and are used to scale up data mining tasks to large quantities of data. The approaches and systems highlighted in the previous sections along with already available commercial as well as free tools show the prosperity of this field of research. However, despite the recent successes in the scalability of data mining techniques, it remains an active research area, with unresolved issues and distinctly new innovative approaches.

Apart from the obvious improvements on hardware such as the usage of Graphics Processing Units (GPUs) [26], or better parallelisation frameworks and algorithms, the movement of data and the cost is an issue. Cloud computing is often presented as the method of choice if you urgently need a large amount of computing power to analyse a large amount of data. However, bandwidth is often the bottleneck as cloud systems are usually remotely located. In cases where the data becomes very large, local processing is still needed. A further open issue on the usage of the cloud and parallel data mining approaches is the cost/benefit relationship, which is hardly explored in the literature. Usage of cloud services or in-house parallel computing facilities is an investment decision that is only justified if the financial benefit of analysing large quantities of data outweighs the associated computational and hardware costs in financial terms. A further issue of grid and cloud computing that needs to be addressed more thoroughly is the security of confidential data, as outsourcing the data analysis also relies on trusting the grid / cloud service provider.

Whereas this chapter mainly discussed scalability issues associated with the size of the data, the type of method chosen may also inflict a computational bottleneck. For example the computational INtelligence platform For Evolving and Robust predictive systems (INFER) [55] provides a complex environment that automatically adapts to concept changes in the data. It basically evolves by training many different data mining models and returns the 'fittest model' either autonomously or with user assistance. Rather than the size of the training data, the training as such and the adaptation of many models in a concurrent way imposes a considerable computational bottleneck. Hence loosely coupled as well as tightly coupled parallelisation needs to be considered for improving the scalability of such systems.

The trend towards multi-core processors in standard workstations needs to be explored, as networks of such workstations are what we described as a hybrid ar-

chitecture between loosely and tightly coupled architectures. Existing systems such as Hadoop rely on the workstations' operating system to balance the workload efficiently among the available cores.

A distinctly different approach to the analysis of large and complex datasets is emerging: 'Visual Analytics' (VA). VA describes the reasoning assisted by graphical visualisations in an interactive way. VA is based on the concept of information visualisation which aims at using the computational power of the human brain to process images and hence gain understanding of the data to be analysed [40]. VA extends this concept by interactively incorporating automatic analysis methods prior to and during the visualisation process [27]. The visual representations are not only used to visualise data and patterns for the user, but also to feed back information from the user to the analysis system. Using the combined computational power of silicon as well as biological hardware may result in well-scaling data analysis systems.

In general, parallel and distributed data analysis is still an open field of research. Past efforts to parallelise data mining techniques have come to fruition, but open issues outlined in this section remain to be addressed.

Acknowledgements The research leading to these results has received funding from the European Commission within the Marie Curie Industry and Academia Partnerships & Pathways (IAPP) programme under grant agreement n^{o} 251617.

References

1. Amazon. Amazon web services, 2012.
2. Justin D. Basilico, M. Arthur Munson, Tamara G. Kolda, Kevin R. Dixon, and W. Philip Kegelmeyer. Comet: A recipe for learning and using large ensembles on massive data. *CoRR*, abs/1103.2068, 2011.
3. Daniel Berrar, Frederic Stahl, C S Goncalves Silva, J R Rodrigues, and R M M Brito. Towards data warehousing and mining of protein unfolding simulation data. *Journal of Clinical Monitoring and Computing*, 19:307–317, 2005.
4. Kanishka Bhaduri, Kamalika Das, Kun Liu, Hillol Kargupta, and Jessica Ryan. Distributed data mining bibliography, 2008.
5. M A Bramer. Automatic induction of classification rules from examples using N-Prism. In *Research and Development in Intelligent Systems XVI*, pages 99–121, Cambridge, 2000. Springer-Verlag.
6. M A Bramer. An information-theoretic approach to the pre-pruning of classification rules. In B Neumann M Musen and R Studer, editors, *Intelligent Information Processing*, pages 201–212. Kluwer, 2002.
7. Peter Brezany, Ivan Janciak, and A. Min Tjoa. *GridMiner: An Advanced Support for E-Science Analytics*, pages 37–55. John Wiley and Sons, Ltd, 2009.
8. K Cardona, J Secretan, M Georgiopoulos, and Anagnostopoulos. A grid based system for data mining using mapreduce. Technical Report, AMALTHEA TR-2007-02, 2007.
9. Sebastian Celis and David R. Musicant. Weka-parallel: Machine learning in parallel. Technical report, Carleton College, CS TR, 2002.
10. J. Cendrowska. PRISM: an algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349–370, 1987.

11. Lorraine Chambers, E. Tromp, M. Pechenizkiy, and Mohamed Gaber. Mobile sentiment analysis. In *Proceedings of the 16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, September 2012.
12. Cheng T. Chu, Sang K. Kim, Yi A. Lin, Yuanyuan Yu, Gary R. Bradski, Andrew Y. Ng, and Kunle Olukotun. Map-Reduce for Machine Learning on Multicore. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *NIPS*, pages 281–288. MIT Press, 2006.
13. P Clark and T Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.
14. William W Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.
15. Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.
16. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.
17. M. Gaber. Data stream mining using granularity-based approach. *Foundations of Computational Intelligence*, 6:47–66, 2009.
18. Mohamed Gaber. Foundations of adaptive data stream mining for mobile and embedded applications. In *Biomedical Engineering Conference, 2008. CIBEC 2008. Cairo International*, pages 1 –6. IEEE, Piscataway, December 2008. DOI: 10.1109/CIBEC.2008.4786099.
19. Mohamed Medhat Gaber, Uwe Röhm, and Karel Herink. An analytical study of central and in-network data processing for wireless sensor networks. *Information Processing Letters*, 110(2):62–70, 2009.
20. Mohamed Medhat Gaber and Philip S. Yu. A holistic approach for resource-aware adaptive data stream mining. *New Generation Comput.*, 25(1):95–115, 2006.
21. J. Gama. *Knowledge discovery from data streams*. Chapman & Hall/CRC, 2010.
22. John Gantz and David Reinsel. The digital universe decade, are you ready? *IDC*, 2009(May):1–16, 2010.
23. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 29–43, New York, NY, USA, 2003. ACM.
24. Globus. The globus toolkit, 2012.
25. Hadoop. Hadoop mapreduce, http://hadoop.apache.org/mapreduce/ 2012.
26. Liheng Jian, Cheng Wang, Ying Liu, Shenshen Liang, Weidong Yi, and Yong Shi. Parallel data mining techniques on graphics processing unit with compute unified device architecture (cuda). *The Journal of Supercomputing*, pages 1–26.
27. Daniel A. Keim, Florian Mansmann, Jorn Schneidewind, and Hartmut Ziegler. Challenges in visual data analysis. In *Proceedings of the conference on Information Visualization*, IV '06, pages 9–16, Washington, DC, USA, 2006. IEEE Computer Society.
28. S. Krishnaswamy, Mohamed Gaber, M. Harbach, C. Hugues, A. Sinha, B. Gillick, P. Haghighi, and A. Zaslavsky. Open mobile miner: a toolkit for mobile data stream mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, June 2009.
29. Anup Kumar, Mehmed Kantardzic, and Samuel Madden. Guest editors' introduction: Distributed data mining–framework and implementations. *IEEE Internet Computing*, 10(4):15–17, July 2006.
30. Ting Liu, Charles Rosenberg, and Henry A. Rowley. Clustering billions of images with large scale nearest neighbor search. In *Proceedings of the Eighth IEEE Workshop on Applications of Computer Vision*, WACV '07, page 28, Washington, DC, USA, 2007. IEEE Computer Society.
31. Ping Luo, Kevin Lü, Zhongzhi Shi, and Qing He. Distributed data mining in grid computing environments. *Future Gener. Comput. Syst.*, 23(1):84–91, January 2007.
32. Lars Nolle, K C P Wong, and Adrian Hopgood. DARBS: a distributed blackboard system. In *Proceedings of the Twenty-first SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, Cambridge, 2001. Springer.

33. Biswanath Panda, Joshua S. Herbach, Sugato Basu, and Roberto J. Bayardo. Planet: massively parallel learning of tree ensembles with mapreduce. *Proc. VLDB Endow.*, 2:1426–1437, August 2009.
34. Compare Business Products. The 10 largest data bases in the world, 2012.
35. Human Genome Project. Human genome project information, 2012.
36. Ross J Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
37. Thomas Rings, Geoff Caryer, Julian R. Gallop, Jens Grabowski, Tatiana Kovacikova, Stephan Schulz, and Ian Stokes-Rees. Grid and cloud computing: Opportunities for integration with the next generation network. *J. Grid Comput.*, 7(3):375–393, 2009.
38. SETI@home. About seti@home, 2012.
39. John Shafer, Rakesh Agrawal, and Manish Metha. SPRINT: a scalable parallel classifier for data mining. In *Proc. of the 22nd Int'l Conference on Very Large Databases*, pages 544–555. Morgan Kaufmann, 1996.
40. Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–343, Washington, DC, USA, 1996. IEEE Computer Society.
41. A. Sirvastava, E. Han, V. Kumar, and V. Singh. Parallel formulations of Decision-Tree classification algorithms. *Data Mining and Knowledge Discovery*, pages 237–261, 1998.
42. Madhan Kumar Srinivasan, K. Sarukesi, Paul Rodrigues, M. Sai Manoj, and P. Revathy. State-of-the-art cloud computing security taxonomies: a classification of security challenges in the present cloud computing environment. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, ICACCI '12, pages 470–476. ACM, 2012.
43. F. Stahl and M. Bramer. Scaling up classification rule induction through parallel processing. *Knowledge Engineering Review*, in Press.
44. F. Stahl, M. Bramer, and M. Adda. Pmcri: A parallel modular classification rule induction framework. *Machine Learning and Data Mining in Pattern Recognition*, pages 148–162, 2009.
45. Frederic Stahl and Max Bramer. Random prism: An alternative to random forests. In *Thirty-first SGAI International Conference on Artificial Intelligence*, pages 5–18, Cambridge, England, 2011.
46. Frederic Stahl and Max Bramer. Computationally efficient induction of classification rules with the pmcri and j-pmcri frameworks. *Knowledge-Based Systems*, 2012.
47. V Stankovski, M Swain, V Kravtsov, T Niessen, D Wegener, M Rohm, J Trnkoczy, M May, J Franke, A Schuster, and et al. Digging deep into the data mine with datamininggrid, 2008.
48. Vlado Stankovski, Martin Swain, Valentin Kravtsov, Thomas Niessen, Dennis Wegener, Jrg Kindermann, and Werner Dubitzky. Grid-enabling data mining applications with datamininggrid: An architectural perspective. *Future Generation Computer Systems*, 24(4):259 – 279, 2008.
49. Sloan Digital Sky Survey. The sloan digital sky survey, 2012.
50. Martin Swain, Cíndida G. Silva, Nuno Loureiro-Ferreira, Vitaliy Ostropytskyy, João Brito, Olivier Riche, Frederick Stahl, Werner Dubitzky, and Rui M. M. Brito. P-found: Grid-enabling distributed repositories of protein folding and unfolding simulations for data mining. *Future Gener. Comput. Syst.*, 26(3):424–433, 2010.
51. A Szalay. *The Evolving Universe*. ASSL 231, 1998.
52. Ian H Witten and Frank Eibe. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, second edition, 2005.
53. Gongqing Wu, Haiguang Li, Xuegang Hu, Yuanjun Bi, Jing Zhang, and Xindong Wu. Mrec4.5: C4.5 ensemble classification with mapreduce. In *ChinaGrid Annual Conference, 2009. ChinaGrid '09. Fourth*, pages 249 –255, 2009.
54. Qing Zhao, Jizhou Sun, Ce Yu, Jian Xiao, Chenzhou Cui, and Xiao Zhang. Improved parallel processing function for high-performance large-scale astronomical cross-matching. *Transactions of Tianjin University*, 17:62–67, 2011.
55. I. Zliobaite, A. Bifet, Mohamed Gaber, B. Gabrys, J. Gama, L. Minku, and K. Musial. Next challenges for adaptive learning systems. *SIGKDD Explorations Newsletter*, 14(1), 2012.