Computing and Informatics, Vol. 21, 2002, 347–362

MINING LARGE DATA SETS ON GRIDS: ISSUES AND PROSPECTS

David Skillicorn

School of Computing
Queen's University, Kingston, Canada
e-mail: skill@cs.queensu.ca

Domenico Talia

DEIS, Università della Calabria
Rende, Italy
e-mail: talia@deis.unical.it

Revised manuscript received 12 November 2002

Abstract. When data mining and knowledge discovery techniques must be used to analyze large amounts of data, high-performance parallel and distributed computers can help to provide better computational performance and, as a consequence, deeper and more meaningful results. Recently *grids*, composed of large-scale, geographically distributed platforms working together, have emerged as effective architectures for high-performance decentralized computation. It is natural to consider grids as tools for distributed data-intensive applications such as data mining, but the underlying patterns of computation and data movement in such applications are different from those of more conventional high-performance computation. These differences require a different kind of grid, or at least a grid with significantly different emphases. This paper discusses the main issues, requirements, and design approaches for the implementation of grid-based knowledge discovery systems. Furthermore, some prospects and promising research directions in datacentric and knowledge-discovery oriented grids are outlined.

Keywords: Grid computing, data mining, distributed knowlege discovery, datacentric models, high-performance computing, data-intensive systems.

1 INTRODUCTION

In the sixteenth century the Italian philosopher Bernardino Telesio, in his *De natura rerum juxta propria principia* (On the nature of things according to their own principles), urging the importance of scientific knowledge based upon experience and experiment, wrote that "Nature is information". A few years later, the English philosopher Francis Bacon, in proposing the application of the inductive method of modern science, in his famous remark recognized that "Knowledge is power" (*Nam et ipsa scientia potestas est*). After five centuries, the technology of information processing has given new meanings and new horizons to these two remarks. Computer science tools and systems allow us to represent many phenomena as collections of information. Much of this information is captured and stored, stimulating interest in automatic techniques and strategies for extracting useful knowledge from these large repositories of data.

This trend towards capturing large amounts of data seems to have been driven by two factors: the cost of storage media has been dropping rapidly throughout the lifetime of computer technology; and more and more interactions are mediated by computers, making it trivial to make a record of them. The amount of data stored is doubling every nine months, and this trend shows no sign of slackening.

We are much better at storing data than at making use of it. It is easy to believe that there are interesting nuggets of knowledge hidden inside large data repositories able to improve everything from transport scheduling to customer relationships and weather prediction. Extracting such knowledge from data has proven more difficult — some estimate that only 5–10 % of data has ever been examined, even superficially. If it is true that almost all data can produce useful knowledge, then there is hardly an organization in the world that cannot benefit from data mining and knowledge discovery; so this is an important problem.

Extracting knowledge from data is a demanding task: it requires large amounts of computation; it requires moving entire datasets, at least between static storage and processor(s); and it often requires resource sharing across substantial distance and organizational boundaries. These requirements are not easily met by today's web and grid infrastructures [10].

Although there are many different varieties of grids, most aim to provide computational performance to a user. Each user submits a job and the grid finds appropriate resources, in principle anywhere, to complete it. In contrast, grids used for data-intensive applications are constrained in where they execute jobs by the location of the data. This changes the entire resource discovery and allocation problem. Furthermore, the range of knowledge discovery computations is actually quite small, creating the interesting possibility of 'caching' both the execution plans of computations and the results of such computations for others to use.

This paper discusses the main issues, requirements, and design approaches for the implementation of grid-based knowledge discovery systems. Furthermore, some prospects and promising research directions in datacentric and knowledge-discovery oriented grids are outlined. The rest of the paper is organized as follows. Section 2

discusses the main design issues in distributed data mining. Sections 3 and 4 present two models designed for distributed data mining on grids: the Knowledge Grid and the Datacentric Grid. Section 5 discusses related work, and Section 6 concludes the paper.

2 DESIGN ISSUES

We have already commented that datasets stored online are increasing in size and number. Much of this data is stored automatically, for example every purchase at a store, every mobile phone call, or every image taken by a satellite, and this affects how and where such datasets are stored. Although small datasets can be moved around, there are several reasons why large datasets should be regarded as effectively *immovable*:

1. **Pragmatics**: although disks are cheap, very few sites can afford to keep enough 'empty' space to handle the sudden and unpredictable arrival of a large (say, terabyte) dataset.
2. **Performance**: some parts of the world are connected by fat pipes, but the overall bandwidth between arbitrary sites is much smaller than these would suggest — there are many bottlenecks, notably between ISP domains, where bandwidths remain ludicrously small. So bandwidth will remain a significant limiting factor in practice, even though peak bandwidths increase and prices decrease.

   As distributed computations reach global scale, the latency required to access remote data is almost entirely time of flight (and so not susceptible to technological improvement). This latency cannot ultimately be hidden from end users. Furthermore, data has an inertia-like property — it is relatively easy to store, and easy to keep moving once in a transmission medium; but the transitions between these two states are expensive in energy, time, and complexity.
3. **Politics**: Many countries have privacy legislation that prevents data movement across their borders; hence data collected about individuals cannot be moved, even if it were technically feasible. There are also social barriers to data movement; for example, owners are often willing to make datasets public but not to have them mirrored.

This does not, of course, mean that datasets can never be moved. What it does suggest is that datasets should not be moved arbitrarily and extemporaneously. Many grid designs assume, perhaps implicitly, that processor cycles are the limiting resource. For data-intensive applications, the bandwidth required to move data is a much more critical resource, suggesting that *computations should be moved to data*, rather than the traditional processor-centric view in which it is data that moves.

This new scenario, in which data is immovable and computations move is more like a converse to distributed computing than to sequential computing. Some

datasets might be naturally stored in a single place; however, some of the pressures that make moving data hard also make it likely that single (logical) datasets will be stored in pieces that may be widely distributed. For example, data collected in different countries may not be merged because of privacy restrictions, and data collected by different parts of an organization may not be centralized. Applications that use such datasets must deal with their distributed nature.

Some of the implications that follow from this view of data as both central to knowledge discovery and other data-intensive applications and physically distributed are:

1. Algorithms for knowledge discovery must be able to make significant progress towards a global goal using only the data stored in each piece of a dataset. In other words, we need new algorithms that can extract local knowledge that can somehow be fitted together to make global knowledge without too much redundancy and/or contradiction. There are connections between algorithms that can work with distributed data and online algorithms, stream algorithms, and some parallel algorithms, but this new class also has some differences.
2. Programming models must be redesigned to allow for the fact that data is arranged in some predetermined way (based on how and where it is collected) and computations must be arranged to match. Many approaches to parallel programming assume that data can be partitioned equally across processors, for example, and this will no longer be true.
3. The nodes of a knowledge discovery grid must combine large computational power with data storage so that the 'pipe' from storage to processor(s) is both fat and short. Such nodes hardly exist in today's grids (although see the Grid Datafarm architecture [21]), but they will be necessary because neither clusters nor network-attached storage, by themselves, are effective for data-intensive applications.

There are also implications for the design of knowledge discovery grids that arise from properties of typical applications:

1. The knowledge, in the form of models and statistics, that is extracted from datasets is both costly and potentially reusable. It is reasonable to consider caching such results, either for others to use directly or to act as the starting place for higher-order knowledge discovery. This new opportunity arises because, comparatively speaking, there are only a few different techniques for knowledge extraction from data. This creates issues about representing what can be found at a given data repository — either complex indexes must be made available and updated regularly, or programs must be prepared to use data representations at different levels of abstraction (and hence different representations) depending on what they find when they arrive at a repository.
2. Knowledge discovery applications are characteristically iterative in the sense that a user's first computation may be followed by a sequence of quite similar computations, perhaps on slightly different data, or with slightly different

parameters. There is therefore a stronger advantage to be gained by caching execution plans and strategies than for other kinds of computations.

3. Many datasets will be public, or at least widely visible. Since computations take place at the site that holds the data, it will be hard to prevent others from observing what is being computed. For some applications, even the fact that certain data are being examined may create security problems.

Consider the following example, which illustrates a typical knowledge extraction situation and some of the problems it creates. Suppose we wish to determine the most popular DVD rental in a large city in a particular week. We assume that each rental store keeps a list of its own popular rentals in decreasing order.

The processor-centric solution to this problem is to have each store send its entire list of rentals and their frequency to some central site. The frequencies are then summed across stores and sorted into decreasing order. The DVD at the top of this combined list is the most popular.

The problems with this approach are obvious: it requires a lot of data to be gathered; and, while stores from the same chain might be persuaded to send their results to a central place, it is highly unlikely that stores from different chains would do so for fear of revealing too much about their operations.

A datacentric approach to this problem would be to send a person (a computation) to each store to process their information locally and then report back enough information to enable the global most-popular DVD to be discovered. This is not so easy — it is not enough to ask each store for its most popular rental, for the most popular overall may not have been most popular at many stores (in the limit, all but one). Bringing back some prefix of the most popular rentals will suffice, although it is hard to determine how large this prefix should be. Either a probabilistic algorithm can be used [18] or some communication between the people at each store can eliminate rentals that cannot possibly be most popular overall. This 'algorithm' requires much less data to be moved and is naturally parallel, since much of the work is done at each store.

An alternate datacentric approach, resembling mobile agents, would be for one person to visit each store in turn and build the global result incrementally from the information available at each one. This requires even less communication, but takes longer since it is sequential. There is an interesting tradeoff here between time and communication.

In the next two sections, we examine two systems for grids directed at knowledge discovery. The first, the Knowledge Grid, builds directly on existing grid technologies, but specializes them for data-intensive applications. The second, the Datacentric Grid, is more radical and is designed around immovable data. Both systems are preliminary, but they illustrate the range of design decisions.

3 THE KNOWLEDGE GRID

The *Knowledge Grid* architecture [3, 4], is built on top of a computational grid that provides dependable, consistent, and pervasive access to high-end computational resources. This architecture uses the basic grid services, for instance the Globus services, and defines two additional layers that implement a set of distributed knowledge-discovery services on world wide connected computers, where each node can be either a sequential or a parallel machine. The Knowledge Grid enables collaboration between scientists who must mine data that are stored in different sites as well as analysts who need to use a knowledge-management system that operates on data warehouses located in the different parts or sites of organizations.

The Knowledge Grid attempts to overcome the difficulties of wide area, multi-site operation by exploiting the underlying grid infrastructure that provides basic services such as communication, authentication, resource management, and information. To this end, the Knowledge Grid architecture is organized so that more specialized data mining tools are compatible with lower-level grid mechanisms and also with the Globus Data Grid services. This approach benefits from "standard" grid services that are increasingly utilized and offers an open Parallel and Distributed Knowledge Discovery (PDKD) architecture that can be configured on top of grid middleware in a simple way.

The Knowledge Grid services are organized in two hierarchical layers: the **Core K-Grid layer** and the **High level K-Grid layer**. The former refers to knowledge services directly implemented on the top of generic grid services, the latter is used to describe, develop and execute knowledge discovery computations over the Knowledge Grid (see Figure 1).

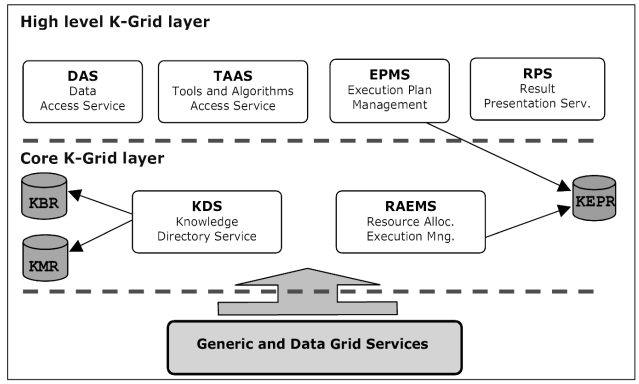

Fig. 1. Layers and components of the Knowledge Grid architecture

The Core K-Grid layer supports the definition, composition, and execution of a knowledge discovery computation over the grid. Its main goals are the management of all metadata describing characteristics of data sources, third-party data-mining

tools, and data-management and data-visualization tools and algorithms. Moreover, this layer coordinates the execution of each knowledge discovery computation, attempting to match each application's requirements to the available grid resources. This layer implements the following basic services:

- the Knowledge Directory Service (KDS) responsible for maintaining a description of all the data and tools used in the Knowledge Grid;
- the Resource Allocation and Execution Management services (RAEMS) used to find a mapping between an execution plan and available resources, with the goal of satisfying requirements (computing power, storage, memory, database, network bandwidth and latency) and constraints.

The High-level K-Grid layer implements services used to compose, validate, and execute a knowledge discovery computation. Moreover, the layer offers services to store and analyze the knowledge obtained as result of knowledge discovery computations. The main services are:

- the Data Access Services (DAS) that are responsible for the search, selection (Data Search Services), extraction, transformation and delivery (Data Extraction Services) of data to be mined;
- the Tools and Algorithms Access Services (TAAS) that are responsible for the search, selection, and downloading of data-mining tools and algorithms;
- the Execution Plan Management Services (EPMS) that handle execution plans as an abstract description of a knowledge discovery grid application. An execution plan is a graph describing: the interaction and data flows between data sources, extraction tools, DM tools, visualization tools; and storing of knowledge results in the Knowledge Base Repository;
- the Results Presentation Service (RPS) that specifies how to generate, present and visualize the knowledge discovery results (rules, associations, models, classification, etc.).

Recently VEGA [5], a visual tool set that implements the main steps of application composition and execution of the Knowledge Grid, has been implemented. A user can compose a data-mining application as a set of workspaces where the steps of a data-mining process can be defined in terms of computing nodes, datasets and data-mining algorithms. The tool set generates an XML specification of the defined execution plan and the Globus RSL code that will run the mining application on the grid. VEGA embodies an implementation of the Knowledge Directory Service and the Knowledge Metadata Repository of the Core K-Grid layer, and the Data Access Service of the High level K-Grid layer [4].

In the Knowledge Grid, metadata describing relevant objects for knowledge discovery computations, such as data sources and data mining software, are represented by XML documents in a local repository (KMR), and their availability is indicated by publishing entries in the Directory Information Tree maintained by a LDAP server,

which is provided by the Grid Information Service (GIS) of the Globus Toolkit. The main attributes of the LDAP entries specify the location of the repositories containing the XML metadata, whereas the XML documents maintain more specific information for the effective use of resources. The basic tools of the DAS services have been implemented to find, retrieve and select metadata about knowledge discovery objects on the grid on the basis of different search parameters and selection filters. Moreover, execution plans are modelled as graphs, where nodes represent computational elements (data sources, software programs, results, etc.) and arcs represent basic operations (data movements, data filtering, program execution, etc.). Work on considering different network parameters, such as topology, bandwidth and latency, for knowledge discovery program execution optimization is ongoing.

The Knowledge Grid represents a step in the direction of studying the unification of knowledge discovery and computational grid technologies and defining integrating environments for distributed data mining and knowledge discovery based on grid services. The definition of such architectures will accelerate progress on very large-scale geographically distributed data mining by enabling the integration of currently disjoint approaches and revealing technology gaps that require further research and development.

4 THE DATACENTRIC GRID

The *Datacentric Grid* consists of four kinds of entities:

1. **Data/Compute Servers** (DCS), the nodes at which data-intensive computation takes place. Each DCS has three parts:

   - A compute engine;
   - A data repository;
   - A metadata tree.

   When an application builds a model from a base dataset and caches it for subsequent use, the model becomes a new dataset which could be used by other applications. For example, extracting a sample from a dataset is a particular (simple) kind of data mining application. Such a sample may be worth keeping, particularly if it is carefully chosen and/or much smaller than the base dataset. Hence each dataset acquires a tree of models above it, each of which could be regarded by other applications as the root of its own tree. The metadata tree keeps track of the properties of each of these models and their relationships. The compute engine and data repository might be implemented as a cluster backed by a multiported RAID using today's technology (but there are opportunities to build interesting new architectures to support datacentric grids as well).
2. **Grid Support Nodes** (GSN), the nodes that maintain information about the grid as a whole. GSNs hold two kinds of information:

- A directory of Data/Compute Servers, describing the static properties of their compute engines and the dynamic information about their usage into the future. As well, they contain information about what datasets and models each DCS holds.
- An execution plan cache, containing recent execution plans parameterized by their properties as tasks and also be their achieved performance. Note that execution planning in a datacentric grid is simpler than in a computational grid. A normal computation could possibly execute on a huge number of different compute servers. A datacentric application can only execute on the computer servers local to the data it needs. On the other hand, because the models stored in a particular data repository change dynamically, an application must be prepared to start at different places in the model hierarchy, depending on what it finds when it reaches a given DCS. In the most fortunate situation, the model it needs has already been computed; if not, it may have to compute it from a larger model/dataset. Even though many knowledge discovery applications are long running, it may not be possible to keep information about each data repository current in each GSN.

3. **User Support Nodes** (USN), which carry out execution planning, and also hold the results of computations for users who may not want to access them immediately. USNs are separated from the sites that act as user interfaces to the Datacentric Grid because both the descriptions of datacentric tasks and their results can be small in size. Hence, it is possible to interact with the Datacentric Grid using thin pipes. USNs act partly as proxies for users.
4. **User Access Points** (UAP), which allow users to create descriptions of computations and see their results. We have already commented that only relatively few techniques for model building are known, perhaps a few hundred, so it is plausible that datacentric applications could be described using a query language syntax. Such applications do not require a large device so that they might be able to be run from handhelds. Similarly, the results of many datacentric computations are quite small (even though they may have required huge computations on a global scale) so it is plausible to display them on a simple device. In this setting it is natural to expect computations to run detached from uses; hence the existence of USNs to interact with the Datacentric Grid when users are not connected.

In summary, the differences between the Datacentric Grid and more conventional computational grids are twofold: the execution planning functionality is both simpler with respect to cycles and more complex with respect to data; and there is a much greater emphasis on caching and reusing both results and execution strategies.

5 RELATED WORK

A number of projects are built on the assumption that available bandwidth will grow as rapidly as online datasets. These projects are building *data grids*, which add to computational grids the functionality to move large datasets from place to place on demand. Example projects of this class are the EU DataGrid project, the Grid DataFarm [21], and the TeraGrid [22], a collaboration between four U.S. high-performance computing centers built around 40 Gbps links.

There are two difficulties with this approach:

1. Even if some regions of the Internet are well connected, sites outside them see much lower effective bandwidths — often orders of magnitude lower. A new data divide may be created between those with access to large datasets and those without.
2. Bandwidth may not be the limiting factor — it may be the allocation of storage space for transient, but large, datasets. Even if storage space becomes cheap, the machinery and overheads to capture large datasets arriving dynamically may still create significant overheads.

Thus data grid approaches work well at present, but they may not be scalable, either with size or time.

As suggested in recent work [1, 12] and by significant recent research projects, such as UK e-Science programme, the creation of datacentric and knowledge grids on the top of computational grid middleware is an important enabling condition for new grid applications in many strategic areas. Emerging architectures and models for data mining on grids can be roughly classified as: domain-specific systems such as ADaM and the AstroGrid project; and generic architectures such as the Knowledge Grid, the Datacentric Grid, and Discovery Net.

The TeraGrid project is building a powerful grid infrastructure, connecting four main sites in USA (San Diego Supercomputer Center, National Center for Supercomputing Applications, Caltech and Argonne National Lab), that will provide access to tera-scale amounts of data [22]. The most challenging application on the TeraGrid will be the synthesis of knowledge from data. The development of knowledge-synthesis tools and services will enable the TeraGrid to operate as a knowledge grid.

The ADaM (Algorithm Development and Mining) systems is an agent-based data mining framework developed at the University of Alabama in Huntsville used to mine in parallel hydrology data from four sites [11]. The system includes a mining engine and a daemon-controlled database. The database contains information about the data to be mined including its type and its location. A user provides the mining engine with a mining plan (i.e. the sequential list of mining operations that are to be performed along with any parameters that may be required for each mining operation). The mining engine consults the database in order to find out where the data to be mined is stored and then applies the mining plan to the set of data that

has been identified in the database. In the grid version of ADaM, the database and its associated daemon reside on a processor distinct from that of the mining engine. The data is managed at multiple sites by SRB/MCAT and GridFTP.

The Discovery Net is an EPSRC project launched at Imperial College, UK [6]. Its main goal is to design, develop and implement a generic architecture to support realtime processing, interaction, integration, visualization and mining of massive amounts of time-critical data generated by high-throughput devices. The knowledge discovery process will be applied to raw and processed data from biotechnology, pharmacogenomic, remote sensing and renewable energy data. The DNET architecture aims to develop Large-scale Dynamic Realtime Decision support, used in renewable energy and remote sensing applications.

The National Center for Data Mining (NCDM) at the University of Illinois at Chicago (UIC) is developing several testbeds for knowledge discovery over grids [23]. The Terra Wide Data Mining Testbed is an infrastructure built for the remote analysis, distributed mining, and real time exploration of scientific, engineering, business, and other complex data. The Terra testbeds are currently developed for Climate data, Health Care Data Astronomical Data and High Energy Physics.

The Terabyte Challenge Testbed consists of ten sites distributed over three continents connected by high-performance links. It has been instrumented for network measurements and provides a platform for experimental discovery of scientific, engineering, business, and e-business data. The testbed includes a variety of distributed data-mining applications, including the analysis of climate data, astronomical data, network data, web data, and business data.

Finally, the Global Discovery Network is a collaboration between the National Center for Data Mining (Laboratory for Advanced Computing) and the Discovery Net project (mentioned above). The new Global Discovery Network will link the Discovery Net to the Terra Wide Data Mining Testbed to create a combined global testbed with a critical mass of data.

In summary, most of these emerging data and knowledge grids are aimed at specific application domains. A few generalize to multiple domains but usually only scientific ones in which, for example, security issues are not considered critical.

Other research work is directed at providing interfaces to large static datasets so that they can be more readily accessed remotely. Few of these provide the functionality of a datacentric grid node of the kind we have suggested, but many provide some aspects of it.

All grid approaches share a common problem of how to allow a computation sent by A to be executed on a system owned by B without compromising either A's result or B's system. Authentication of users is technically feasible, but protecting against malicious users or malicious hosts is much more difficult. Sites that hold data repositories have a slightly simpler problem to solve, since the kinds of programs they are asked to run may be more stylized and chosen from a smaller repertoire than in other kinds of grids. In particular, if the required computation can be described by giving a set of parameters, rather than executable code, then a web services interface can be used. Such an interface can range from a web form (in the ordinary http sense)

up to a coordination language interface (in the style of Netsolve, for example). In fact, a web search engine can be regarded as a simple datacentric node, with the query as a small program, since the cycles used and the data accessed both reside on the search engine site. Some examples of these kinds of interfaces are Espresso [20], which makes climate modelling data available via a web browser interface, and the Distributed Computational Laboratory at Georgia Tech [16], There are also a large number of grid portal projects (see `www.gridcomputing.com` for an up-to-date list).

Another piece of related work is the PMML (Predictive Model Markup Language) [7], which makes it possible to describe both data and models within a single XML-based framework. PMML goes some way towards defining how the results of data-mining operations could be labelled, for example.

6 RESEARCH PROSPECTS

Grids for data-intensive applications raise research questions that either do not occur in computational grids or occur in quite different ways. Here are some of the major ones:

- Which applications can be decomposed in a way that permits useful progress on partial (local) data as well as composition of partial results into global results? There is a large class of computations that might be called *reductive*, in which the global answer is easily computed from answers obtained from local pieces of a dataset. The simplest example is to compute the sum of an attribute across an entire distributed dataset. Applications such as these are the target of the DataCutter system [2]. It is more surprising that many other results can also be obtained by local computations supplemented by relatively small volumes of communication between each site. In the easiest scenario, each site contains information about different objects, but all share the same attributes. Then, for example, many kinds of neural networks can be trained by training identical networks on each local piece of the dataset using batch learning, and then exchanging the errors vectors and summing them to give an overall result [17]. The information being communicated consists of sets of weights, which are orders of magnitude smaller than the dataset itself. A harder scenario is when each site contains only some of the attributes of each object. Kargupta *et al.* have shown that it is still often possible to learn useful information locally by finding a basis such that local information requires only a few non-zero coefficients to represent it with respect to the basis. This has been done successfully for Fourier bases [13], and wavelets [15]; and also by Yang and Skillicorn for SVD [19].
- How can the security of applications be preserved, so that it is not possible to determine either the results of asking certain questions about a dataset, nor even that those questions are being asked? Many existing grid systems assume either that the grid is internal to a single organization (e.g. Entropia), or involve computations that are essentially in the public domain such as scientific research. Hence, the threat model is of unauthorized use rather than subversion or theft of

ideas. Data-intensive applications such as data mining, while relevant to scientific inquiry, are more likely to involve enterprise-critical computations, and so the security problem is more difficult. Furthermore, many interesting questions will require data from inside and outside an organization, forcing consideration of a world that is at least partly open. In such enterprise-critical applications, traffic analysis, that is knowing what knowledge is being sought, may also be an important threat.

- What is the best programming model for data-intensive applications? We have already noted that many data-intensive applications involve a relatively small set of model-building techniques. This suggests that perhaps a query language or high-level scripting language might be appropriate.
- How do applications find datasets and/or models? Nodes in computational grids have relatively few wares to offer: processor cycles of certain speeds, numbers of processors, intracluster bandwidth, and so on. In contrast, nodes on datacentric grids contain data with many possible descriptors: what the objects are, what the attributes are, how it was generated, when it was updated, and so on. Furthermore, if the results of computations are cached, this information changes rapidly. Data-intensive applications may find it difficult to discover which site(s) hold the datasets they wish to use, and may have to be prepared to handle different versions of them, discovering which one only when their code arrives at the site on which it will execute.
- What is the best architecture for nodes that must supply both large datasets and high-performance computing cycles? Clusters have revolutionized the provision of computing cycles, making the slice price for a large parallel system very close to the single processor price. A similar revolution has not taken place for data storage. Databases greatly increase the storage cost per byte above the cost for raw disks. RAIDS reduce the storage cost somewhat but do not cheaply allow channels to many processors.
- How can ontologies help and to improve knowledge discovery on grids? A knowledge-discovery process that includes grid resources needs to manage distributed objects such as datasets, mining algorithms, computing nodes, and data models. The definition and deployment of ontologies that integrate metadata services in supporting identification, search, and discovery of objects involved in knowledge-discovery applications will improve the description of a grid environment and help users in developing applications. This approach may represent a step towards a Semantic Grid able to support knowledge-management processes.

7 CONCLUSION

Extracting knowledge from data is a demanding task: it requires large amounts of computation and it needs moving entire datasets, at least between static storage and processor(s). In a distributed setting it requires resource sharing across substantial

distance and organizational boundaries. These requirements are not easily met by today's web and grid infrastructures.

In particular, grids for data-intensive and knowledge discovery applications raise design and implementation issues that either do not occur in computational grids or occur in quite different ways. However, despite the limited efforts devoted to knowledge-extraction grids, it is natural to consider grids as tools for distributed data-intensive applications such as data mining.

This paper has discussed the main issues, requirements, and design approaches for the implementation of grid-based knowledge-discovery systems. Two specific grid-based architectures, the Knowledge Grid and the Datacentric Grid, which address several issues in knowledge-extraction grids, have been presented. The Knowledge Grid is designed to run in the context of today's Grid environments. It is described in terms of services, without necessarily committing itself to where these services will run, and what underlying architecture they will use. The Knowledge Grid also assumes that some datasets can be moved (which is a reasonable near-term assumption) and that underlying grid services such as GridFTP will be available to handle the required transfers. The Datacentric Grid is more radical in the sense that it assumes the existence of particular architectures (for example, co-located clusters and data repositories) that are not common today. Hence its design is expressed in terms of entities which have an assumed architectural substrate. It is intended for the medium term, a time when the flexibility to move datasets around will be reduced by lack of on-demand bandwidth and storage.

Clearly the Knowledge Grid architecture could eventually be implemented on a Datacentric Grid. When this happens, some simplification of the design of the Knowledge Grid may be possible because the middleware required, at present, to interface to existing Grids will not be necessary. The Knowledge Grid and the Datacentric Grid can be viewed as extensions of Data Grids to deal, first, with data-access driven applications such as data mining, and then subsequently to deal with the problems of large, immovable data.

Beside the presentation and discussion of the Knowledge Grid and the Datacentric Grid, the paper has also sketched some prospects and promising research directions in datacentric and knowledge-discovery oriented grids. As a general outcome of our work, we conclude that the development of grid models and architectures specifically designed for the analysis of very large datasets is raising new challenges and will significantly improve our ability to retrieve and process huge volumes of data available across a large number of geographically dispersed repositories. This approach will allow grid users to capture, enrich, classify and structure knowledge about scientific experiments or provided by virtual organizations and remote research teams.

REFERENCES

[1] Berman F.: From TeraGrid to Knowledge Grid. Communications of the ACM, Vol. 44, 2001, No. 11, pp. 27–28.

[2] Beynon, M.—Ferreira, R.—Kurc, T.—Sussman, A.—Saltz, J.: DataCutter: Middleware for Filtering Very Large Scientific Datasets on Archival Storage Systems. IEEE Symposium on Mass Storage Systems, 2000, pp. 119–134.

[3] Cannataro, M.—Talia, D.: Parallel and Distributed Knowledge Discovery on the Grid: A Reference Architecture. Proc. of the 4th International Conference on Algorithms and Architectures for Parallel Computing (ICA3PP), Hong Kong, World Scientific Publ., 2000, pp. 662–673.

[4] Cannataro, M.—Talia, D.—Trunfio, P.: Knowledge Grid: High Performance Knowledge Discovery Services on the Grid. Proc. GRID 2001, Vol. LNCS 2242, Springer-Verlag, 2001, pp. 38–50.

[5] Cannataro, M.—Congiusta, A.—Talia, D.—Trunfio P.: A Data Mining Toolset for Distributed High-performance Platforms. Proc. Conf. Data Mining 2002, Wessex Inst. Press, 2002, pp. 41–50.

[6] Curcin, V. et al.: Discovery Net: Towards a Grid of Knowledge Discovery. Proc. Conf. on Knowledge Discovery in Databases, 2002.

[7] Data Mining Group: `www.dmg.org`.

[8] Fayyad, U. M.—Piatesky-Shapiro, G.—Smyth P.: From Data Mining to Knowledge Discovery: an Overview. In Advances in Knowledge Discovery and Data Mining, AAAI MIT Press, 1996, pp. 1–34.

[9] Foster, I.—Kesselman C.: Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputing Applications, Vol. 11, 1997, pp. 115–128.

[10] Foster, I.—Kesselman C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publ., 1999.

[11] Hinke, T.—Novonty J.: Data Mining on NASA's Information Power Grid. Proc. Ninth IEEE International Symposium on High Performance Distributed Computing, 2000.

[12] Johnston, W. E.: Computational and Data Grids in Large-Scale Science and Engineering. Future Generation Computer Systems, Vol. 18, 2002, No. 8, pp. 1085–1100.

[13] Kargupta, H.—Park B. H.: Mining Decision Trees from Data Streams in a Mobile Environment. Proc. ICDM, 2001, pp. 281–288.

[14] Kargupta, H.—Chan P. (Eds.): Advances in Distributed and Parallel Knowledge Discovery. AAAI/MIT Press, 2000.

[15] Hershberger, D.—Kargupta, H.: Distributed Multivariate Regression Using Wavelet-Based Collective Data Mining. Journal of Parallel and Distributed Computing, Vol. 61, 2001, No. 3, pp. 372–400.

[16] Plale, B.—Elling, V.—Eisenhauer, G.—Schwan, K.—King D.—Martin, V.: Realizing Distributed Computational Laboratories. Int. J. Parallel and Distributed Systems and Networks, 1999.

[17] Rogers, R.—Skillicorn D.: Using the BSP Cost Model to Optimize Parallel Neural Network Training. Future Generation Computer Systems, Vol. 14, 1998, pp. 409–424.

[18] Skillicorn, D.: Parallel Frequent Set Counting. Parallel Computing, Vol. 28, 2002, pp. 815–825.

[19] Skillicorn, D.—Yang X.: High-Performance Singular Value Decomposition. In Data Mining for Scientific and Engineering Applications, Kluwer, 2002, pp. 401–424.

[20] Taylor, J.—Dvorak M.—Mickelson S.: Developing Grid Based Infrastructure for Climate Modeling, Climate and Global Change Report ANL/CGC-010-0102, January 2002.

[21] Tatebe, O.—Morita, Y.—Matsuoka, S.—Soda, N.—Sekiguchi, S.: Grid Datafarm Architecture for Petascale Data Intensive Computing. VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research, Moscow, June 2002.

[22] Teragrid: `www.teragrid.org`.

[23] National Center for Data Mining: Laboratory for Advanced Computing at the University of Illinois at Chicago, `http://www.ncdm.uic.edu/testbeds.htm`, 2002.

David Skillicorn received his Ph.D. from the University of Manitoba. After a brief time at Dalhousie University, he moved to Queen's where he is now a professor. His research interests are in high-performance computing, where he has published extensively on parallel programming models, and non-numeric computation, where he has worked on structured text and data mining. His interests at the moment centre on migrating strategies and techniques for parallel data mining to more distributed settings.

Domenico Talia is a professor of computer science at the Faculty of Engineering at the University of Calabria, Italy. Talia received the Laurea degree in physics at University of Calabria. From 1985 to 1996 he was a researcher at CRAI (Consortium for Research and Applications of Informatics) and from 1997 to 2001 he was a senior researcher at the ISI-CNR — Institute of Systems Analysis and Information Technology of the Italian National Research Council. His main research interests include parallel computation, parallel data mining, grid computing, cellular automata, and computational science. He is a member of the Editorial Boards of the IEEE Computer Society Press, the Future Generation Computer Systems journal, the Parallel and Distributed Practices journal, and the Information journal. He is also a member of the Advisory Board of Euro-Par and a Distinguished Speaker in the IEEE Computer Society Distinguished Visitors Program. He published three books and more than 120 papers in international journals and conference proceedings. He is member of the ACM and the IEEE Computer Society.