Adaptive Query Processing and the Grid: Opportunities and Challenges

Anastasios Gounaris Norman W. Paton Rizos Sakellariou Alvaro A.A. Fernandes

Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK
E-mail: {gounaris,norm,rizos,alvaro}@cs.man.ac.uk

Abstract

Grid technologies have been developed in response to an increase in demand for computing applications designed to yield the benefits from collaboration, data sharing and sophisticated interaction of autonomous and geographically dispersed resources. Distributed Query Processing (DQP) is an appealing solution for expressing and efficiently evaluating requests across Grid resources. In this paper: (i) we identify parts of the Grid infrastructure that facilitate, and open new directions for, query processing over grid-enabled heterogeneous and autonomous databases, stressing the need for Adaptive Query Processing (AQP); (ii) we discuss some basic challenges arising from the new opportunities and outline the unsuitability for use in a Grid setting and narrow specialisation of existing proposals for AQP; and (iii) we suggest a generic adaptivity framework as a promising way forward.

1. Novel Opportunities

Grid technologies have been developed in response to an increase in demands for computing applications designed to yield the benefits from collaboration, data sharing and sophisticated interaction of autonomous and geographically dispersed resources. Indeed, the Grid is an infrastructure and a set of protocols that enable the integrated, collaborative use of high-end computers, networks, databases, and scientific instruments owned and managed by multiple organizations, referred to virtual organisations [6]. Grid computing, in contrast with traditional distributed computing, focuses on large-scale resource sharing (i.e. not primarily file exchange as on the web, but rather direct access to computers, software, data and other resources) for innovative applications and in some cases, high performance. As such, it is not supported by today's Internet and Web infrastructures. Typical Grid applications include autonomous bioinformatics labs across the world sharing their simulation tools, experimental results and databases, as well as the use of the donated spare computer time of thousands of PCs connected to the Internet in order to identify molecules which might inhibit the growth of various types of cancer cells.

Peer-to-peer, rather than being a competing paradigm to Grid computing, can be deemed as an alternative and complementary approach toward the organisation of dynamic computational communities, the interests of which "are likely to grow closer to Grid computing over time" [5].

The benefits accruing from the combination of database and grid technologies have been recognised [20], and although Grid middleware platforms and toolkits do not yet provide built-in support for database operations, specific initiatives have been taken to this end, e.g., the OGSA-DAI (http://www.ogsa-dai.org.uk/) project, which has developed data access technology for Grid-enabling database systems, and OGSA-DQP (http://www.ogsa-dai.org.uk/dqp/), which has developed a service-based query processor for OGSA-DAI wrapped databases [1]. These query technologies can provide effective declarative support for combining data access with analysis, and are inherently well suited for intensive applications, as they implicitly provide for parallelism.

Distributed query processing (DQP) is an appealing solution for a broad range of Grid applications due to its: (i) declarative, as opposed to imperative, manner for expressing potentially complex computations that integrate independent data resources and analysis tools, which are currently either not feasible, or must be carried out using non-database technologies; (ii) implicit provision of parallelism that makes efficient task execution more likely; and (iii), well-established self-scheduling mechanisms for executing the subtasks of a query plan after this has been constructed and shipped for evaluation. Existing non-Grid-enabled distributed and federated database solutions allow data from individual data repositories, with (e.g., [17]) or without (e.g., [7]) some measure of central control, to be combined for data integration purposes. Nevertheless, they lack the parallel infrastructures that are required to perform complex computations on large amounts of data, despite the fact that parallel query processing has become a mature technology. Query processing on the Grid can overcome this limitation because of the following key differences from tradi-

tional DQP over heterogeneous and potentially autonomous databases:

- The Grid provides for systematic access to remote data and computational resources addressing the security, authentication and authorisation problems involved [6], and, as such, the Grid enables remote data sources to be used not only for data retrieval tasks, but also for computational ones as well.
- The Grid provides mechanisms for dynamic resource discovery, allocation and monitoring [3].
- The Grid provides mechanisms for monitoring network connections [21], which is essential for a query engine to efficiently execute queries in wide-area environments.
- The Grid conforms to (currently evolving) standards (http://www.gridforum.org/6_DATA/dais.htm) and there exist publicly available reference implementations (http://www.ogsa-dai.org.uk) for uniform Grid-enabled access to commercial Object-Relational and XML databases.

A significant similarity to traditional DQP however is the need for adaptivity during query execution [11]: the success and endurance of database technology is partially due to the optimisers' ability to choose efficient ways to evaluate the plan that corresponds to the declarative query provided by the user. The optimiser's decisions are based on data properties, such as cardinalities and predicate selectivities, and on environmental conditions, such as network speed and machine load. In both Grid-enabled and non-Grid-enabled DQP over heterogeneous and autonomous sources, information about data properties is likely to be unavailable, inaccurate or incomplete, since the environment is highly volatile and unpredictable. In fact, in the Grid, the execution environment and the set of participating resources is expected to be constructed on-the-fly either per query or per session [1].

The remainder of the paper is structured as follows. Having presented the aspects of Grid computing that facilitate DQP, Section 2 discusses the novel challenges met in this new environment, focusing on the adaptivity issues. Section 3 suggests some new approaches to such challenges. Section 4 concludes the paper.

2. Novel Challenges

2.1. Adaptivity at all levels

Without loss of generality, Figure 1 shows the typical architecture of a DQP optimiser, which firstly constructs a centralised query plan, and then, a parallelised one, according to the widely adopted 2-phase optimisation approach [15]. The characteristics of the Grid environment discussed in the previous section have an impact on all the components in Figure 1, in contrast with the cases addressed by existing adaptive techniques [11, 9]. The quality of the decisions of the optimiser components is basically controlled by the quality of their input information rather than by the policy they implement, which in real cases are typically well established and validated. Examples of issues arising in Grid environments include:

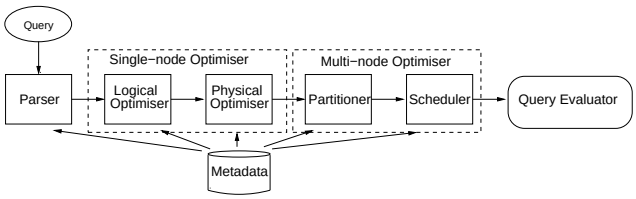

Figure 1. The components of a typical distributed query processor following the 2-phase-optimisation approach.

- *Logical Optimiser:* Typical decisions at this stage of query compilation and optimisation include the order of the joins and the shape of the query plan. Such decisions are affected mostly by the sizes of the input and intermediate data. For the former, accurate statistics about the data stores need to be available, and for the latter, information about the predicate and the filter selectivities are required. Due to the expected lack of such accurate information, it is unlikely that the initial decisions by this component would be near optimal.
- *Physical Optimiser:* Mapping a logical algebraic query plan to a physical one involves the mapping of one logical operator to one of its potentially many physical implementations. For example, a logical join can appear in a query execution plan as a (blocked) nested loop, a hash join, a sort-merge join, a pipelined hash join, and so on. Knowing the specific, individual physical characteristics of computational resources, such as the amount of the available memory, and properties such as ordered attributes, is crucial for ensuring good performance.
- *Partitioner:* The partitioner and the scheduler are commonly used in query evaluators for parallel architectures, but sometimes are omitted in wide-area query engines, which tend to perform all the computation at a central place, using remote machines only to provide data over the network. The partitioner's responsibility is to split the physical query plan into subplans (i.e.,

subsets of physical operators), which can be evaluated at different places according to the capabilities of the available resources in the execution environment.

- *Scheduler:* The scheduler assigns a subplan defined by the partitioner to at least one physical machine. If the execution mechanism does not provide implicit mechanisms for defining the order of operator execution within a subplan (e.g., it does not follow the iterator execution model [10]), the scheduler needs to make these additional decisions as well. Its policies are based mostly on the properties of the physical resources.

Existing solutions for adaptive query processing (AQP) can address only partially the above issues. They can compensate for inaccurate or unavailable data properties (e.g., [2, 14]), bursty data retrieval rates from remote sources (e.g., [12]), and provision of prioritised results as early as possible (e.g., [18]) but they also suffer from the following major limitations, which prohibit their widespread usage:

- They are too specific in terms of the problem they address and are designed in isolation [13]. As such, they cannot easily be combined to meet the broader adaptivity demands of query processing on the Grid.
- They focus on centralised, single-node query processing and do not yet provide robust mechanisms for responding to changes in the pool of available resources, even when the data are initially stored remotely, whereas the Grid provides novel opportunities to benefit from the multiple forms of parallelism (i.e., independent, pipelined, and intra-operator) over many resources.

Efficient query processing on the Grid needs not only to be adaptive, but also to address, in a unifying framework, the cases mentioned above, i.e., both for single-node and multi-node query processing.

2.2. Harnessing the available power

It is perhaps worth mentioning that the selection and scheduling of the resources that will participate in query evaluation from an unlimited and heterogeneous pool is an open issue, even in its static form. Generic Grid schedulers, like Condor [19], support DAGs that can represent query plans but they do not provide for pipelined or partitioned parallelism. Existing scheduling algorithms for distributed databases either support limited partitioned parallelism if all the participating machines have the same capabilities (e.g., [4]), or no partitioned parallelism at all (e.g., [16]). Thus, they are inappropriate for intensive query applications and unable to harvest the benefits of the typically heterogeneous resources that a Grid makes available to its users (note also that a distributed database is not necessarily heterogeneous and autonomous).

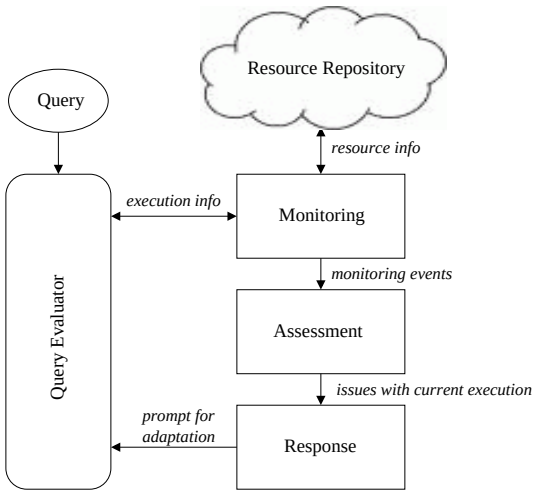

Figure 2. The monitoring, assessment and response phases of AQP.

3. A Roadmap for an Adaptivity framework

3.1. Database architecture in a service-based world

Section 2 underlined the need for a unifying and comprehensive adaptivity framework. Before elaborating on this issue, it is important to answer the question as to what architectural modifications are needed for query processing over Grid resources. The prominence of Web and Grid services as a promising architectural paradigm for wide-area computing has an impact on DQP. Service-based applications are characterised by well-defined interfaces and mechanisms for registering their properties and capabilities, for querying such registries and for binding to remote services. [1] describes an architecture for service-based DQP, in which not only the databases manifest themselves as services, but the whole system is itself a service and is accessible in the same way as any other service. The benefits of such an approach include the capability to dynamically discover relevant data stores and computational resources that are capable of evaluating the query.

3.2. Divide-and-Conquer

It has already been reported that, so far, efforts in AQP have resulted in a collection of isolated techniques rather than in a generic framework, as comprehensive as in other areas of database research [13]. A query processing system

is defined in [11] to be adaptive if it receives information from its environment and determines its behaviour according to that information in an iterative manner, i.e., if there is a feedback loop between the environment and the behaviour of the query processing system. Although such a feedback loop may only be completed between query executions, the most challenging case is where the feedback loop produces effects during the execution of the query. A slightly finer-grained analysis of this loop leads to the identification of three semantically distinct phases, namely *monitoring*, *assessment* and *response*. The execution of a plan and the execution environment itself are monitored, then an assessment is made relating to the progress of the execution, depending on which a response may be carried out that affects the continuing evaluation of the query. The response may be fine grained (e.g., directing the next tuple to a particular node) or coarse grained (e.g., rerunning the optimiser over some or all of the query).

In existing AQP proposals, the monitoring, assessment and response phases are not normally addressed as standalone topics. Rather, individual techniques tend to group together an approach to monitoring, a means of assessment, and a form of response. Consequently, to date no general framework has been constructed for identifying or composing generic and reusable techniques for monitoring, assessment or response. For example, one could envisage a particular approach to monitoring being used with different forms of assessment and response, or different categories of response being made in the light of a single approach to monitoring and assessment. In [8] it is discussed how dynamically gathered execution information about the actual selectivity of an operator can be used either to re-route tuples through joins, or to reconstruct a query execution plan for the remainder of the query, or to build more accurate predictions of the query completion time. By decoupling the generation of selectivity information and its usage, it becomes possible to use a single monitoring mechanism for all the above techniques. This is true for assessment and response, as well. A form of response can be decoupled from the problem it tries to tackle. For example, reoptimising the query plan may be used for various reasons, including the unavailability of accurate statistics at compile time, non-responding remote data sources and unexpected memory shortage.

Figure 2 shows a diagram of what we envisage as a generic framework for adaptive query processing, i.e., a basis for constructing, explaining and comparing adaptive techniques. The execution engine generates information about the state, quality and progress of the evaluation of a query plan. Also, the resource repository provides up-to-date information about the registered resources. Based on such monitoring information, events are generated. The assessment phase evaluates these events in order to verify whether they denote changes in the values of interesting properties and whether such changes are an issue for the current execution, in the sense of something that may compromise its optimality. Once an issue has been identified, the system tries to identify potential ways to respond. If such ways are found, the execution engine is notified accordingly and its behaviour changes as a result.

The above model of AQP is simple, yet powerful and comprehensive, as it can describe and combine existing proposals. Other key benefits that have not been previously mentioned or implied include:

- *Conformance to the service-based computing paradigm:* the framework proposed does not depend on, but conforms to, a service-oriented architecture as it can be implemented by individual components for each distinct phase that are not necessarily tightly coupled with other components, can be enhanced and modified separately, only expose an interface, and communicate primarily by exchanging messages.
- *Suitability for multi-node executions:* there is no restriction on the number of instances and the locality of each of the components comprising an AQP system. For instance, monitoring components can reside with evaluators, or be placed centrally, or form a hierarchy of monitors. As Figure 2 imposes no restrictions on the cases supported, the framework is suitable for multi-node query evaluation and adaptivity control.
- *Comprehensiveness:* it can cover many approaches with respect to
  - *architecture:* AQP can be achieved by the evaluator calling back the static query optimiser, or the evaluator calling a different central mechanism, or the evaluator being self-adaptive. Still, in all these cases there is a need for monitoring, assessment, and response, and thus, the framework can be applied.
  - *plan annotations:* AQP may, but need not, depend on annotations of query plans with performance expectations in order to operate, thereby enabling it to cover a broader range of cases, and not to require any modifications to the static optimiser with respect to this particular issue.
  - *proactive vs reactive adaptivity techniques:* the framework, in general, can realise and implement any existing proposals as it is orthogonal to them. It does not define any specific purposes for plan alteration, kinds of monitoring information, etc.

Implementing the framework: Below, we briefly present some insights for implementing the framework.

- Monitoring: creating monitoring information about the execution of a query plan can be achieved (i) by the incorporation of new components in the evaluator, (ii) by incorporation of dedicated operators in the plan, and (iii) by planting specific probes in the operators, which yields good results [8].
- Assessment: without loss of generality most of the existing approaches can be expressed, and evaluated, as Event-Condition-Action (ECA) rules.
- Response: defining response messages on the basis of the impact of adaptation on the current execution (e.g., operator reordering, operator replacement, etc), rather than on the basis of its purpose and the problems it addresses (e.g., memory limitations, fluctuating data arrival rate, etc) can lead to a small, concrete set of response message types.

4. Conclusions

Grid technologies open new directions for DQP, as they provide solutions for problems such as security, authorisation, authentication, resource discovery, etc. However, the volatility, multiple ownership and heterogeneity of the environment necessitate the development of a comprehensive framework that can cover, generalise, combine and extend adaptive proposals to date. To this end, three phases have been identified in AQP, viz., monitoring, assessment and response, which form the foundations of our proposal for such an adaptivity framework. Studying these phases separately yields generality, substitutability, and reusability across different techniques.

Acknowledgement: Research on AQP at Manchester is supported by the DIM Programme of the UK EPSRC, through Grant GR/R51797/01. We are pleased to acknowledge their support. We are also grateful to our co-workers in this area at Newcastle, namely Paul Watson, Jim Smith and Arijit Mukherjee, and to M. Nedim Alpdemir.

References

[1] N. Alpdemir, A. Mukherjee, N. W. Paton, P. Watson, A. A. A. Fernandes, A. Gounaris, and J. Smith. Service-based distributed querying on the grid. In *Proc. of ICSOC*, LNCS, pages 467–482. Springer, 2003.

[2] R. Avnur and J. Hellerstein. Eddies: continuously adaptive query processing. In *Proc. of ACM SIGMOD 2000*, pages 261–272, 2000.

[3] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *10th IEEE Symp. On High Performance Distributed Computing*, 2001.

[4] R. Epstein, M. Stonebraker, and E. Wong. Distributed query processing in a relational data base system. In *Proc. of 1978 ACM SIGMOD.*, pages 169–180. ACM Press, 1978.

[5] I. Foster and A. Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.

[6] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, CA, USA, second edition, 2003.

[7] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widomthers. The TSIMMIS Approach to Mediation: Data Models and Languages. *J. Intelligent Information Systems*, 8(2):117–132, 1997.

[8] A. Gounaris, N. W. Paton, A. A. A. Fernandes, and R. Sakellariou. Self monitoring query execution for adaptive query processing. *To appear in Data and Knowledge Engineering*.

[9] A. Gounaris, N. W. Paton, A. A. A. Fernandes, and R. Sakellariou. Adaptive query processing: A survey. In *19th BNCOD*, pages 11–25. Springer, 2002.

[10] G. Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2):73–170, 1993.

[11] J. Hellerstein, M. Franklin, S. Chandrasekaran, A. Deshpande, K. Hildrum, S. Madden, V. Raman, and M. Shah. Adaptive query processing: Technology in evolution. *IEEE Data Engineering Bulletin*, 23(2):7–18, 2000.

[12] Z. Ives. *Efficient Query Processing for Data Integration*. PhD thesis, University of Washington, 2002.

[13] Z. Ives, A. Halevy, and D. Weld. Adapting to source properties in processing data integration queries. In *Proc. of the 2004 ACM SIGMOD*. ACM Press, 2004.

[14] N. Kabra and D. DeWitt. Efficient mid-query re-optimization of sub-optimal query execution plans. In *Proc. of ACM SIGMOD 1998*, pages 106–117, 1998.

[15] D. Kossmann. The state of the art in distributed query processing. *ACM Computing Surveys*, 32(4):422–469, 2000.

[16] L. F. Mackert and G. M. Lohman. R* optimizer validation and performance evaluation for distributed queries. In W. W. Chu, G. Gardarin, S. Ohsuga, and Y. Kambayashi, editors, *VLDB'86*, pages 149–159. Morgan Kaufmann, 1986.

[17] M. Ozsu and P. Valduriez, editors. *Principles of Distributed Database Systems (Second Edition)*. Prentice-Hall, 1999.

[18] V. Raman, B. Raman, and J. Hellerstein. Online dynamic reordering for interactive data processing. *Proc. of 25th VLDB Conference*, pages 709–720, 1999.

[19] D. Thain, T. Tannenbaum, and M. Livny. Condor and the grid. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc., 2003.

[20] P. Watson. Databases and the grid. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons Inc., 2003.

[21] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, 1999.