Modular Adaptive Query Processing for Service-Based Grids

Anastasios Gounaris, Norman W. Paton, Rizos Sakellariou and Alvaro A.A. Fernandes
School of Computer Science,
University of Manchester, UK
Email: (gounaris,norm,rizos,alvaro)@cs.man.ac.uk

Jim Smith, Paul Watson
School of Computer Science,
University of Newcastle-upon-Tyne, UK
Email: (Jim.Smith,Paul.Watson)@ncl.ac.uk

Abstract— **Distributed and heterogeneous environments present significant challenges to complex software systems, which must operate in the context of continuously changing loads, with partial or out-of-date information on resource capabilities. A distributed query processor (DQP) can be used to access and integrate data from distributed sources, as well as for combining data access with data analysis. However, in heterogeneous environments, statically constructed query plans may commit a query evaluator to following significantly suboptimal strategies. As such, there is considerable interest in using adaptive query processors (AQPs) in such settings to provide self-optimizing behaviour. However, with many possible adaptive strategies available, it is important that AQPs can be constructed in a systematic and efficient manner. This paper outlines an approach to the development of AQPs in which adaptive behaviour is implemented using cooperating monitoring, assessment and response components. It is shown how this decomposition has been applied in the development of an adaptive DQP system for service-based grids, which reallocates load at query runtime, thereby supporting self-optimization.**

I. Introduction

Distributed and heterogeneous environments present significant challenges to complex software systems, which must operate in the context of continuously changing loads, with partial or out-of-date information on resource capabilities. A distributed query processor (DQP) can be used to access and integrate data from distributed sources, as well as for combining data access with data analysis. However, in heterogeneous environments, statically constructed query plans may commit a query evaluator to following significantly suboptimal strategies. As such, there is considerable interest in using adaptive query processors (AQPs) in such settings to provide self-optimizing behaviour.

However, with many possible adaptive strategies available [2], it is important that AQPs can be constructed in a systematic and efficient manner. Proposals exist for addressing the problems of fluctuations in memory, bursty arrival rates of remote data, inaccurate data statistics, changing machine load, changing network load, and variable machine availability. However, although adaptive techniques cover such a broad range, their designs are often tailored to specific needs, narrowly specialised, and cannot be easily combined [5]. Thus AQP systems are characterised by the lack of shared abstractions and architectures, and do not conform to any common framework that provides built-in support for interactivity and interconnectivity.

This short note makes two contributions. Firstly, it proposes an architectural framework for AQP that is capable of accommodating various kinds of self-optimizing behaviour. Secondly, it describes how the framework has been implemented as an extension to the OGSA-DQP service-based distributed query processor [1] [1].

II. The Monitoring-Assessment-Response Framework

A query processing system is adaptive if it receives information from its environment, analyses this information and revises its behaviour dynamically in an iterative manner, i.e., if there is a feedback loop between the environment and the behaviour of the query processing system. A finer-grained analysis of the feedback loop leads to the identification of three semantically distinct phases, namely:

- *monitoring*, which is concerned with the collection of feedback, both on the query execution itself and on the resources available;
- *assessment*, which deals with the evaluation of the collected feedback and the establishment of potential opportunities for improvement, or problems with the ongoing execution, and
- *response*, which plans and enacts the decisions made.

Figure 1 depicts the components of the framework and shows how these cooperate to cause an adaptation. The framework is generic in the sense of being both adaptivity environment independent, and technique independent, whilst being capable of capturing many existing AQP techniques. It makes no assumptions as to the number of adaptivity components cooperating to achieve an adaptation, or their specific interconnections. The construction of general frameworks for identifying or composing generic and reusable techniques for monitoring, assessment or response has the following key advantages.

1) It allows component reuse and enables the assembling of different combinations, thus covering a wider spectrum

[1]Available from http://www.ogsadai.org.uk/dqp/.

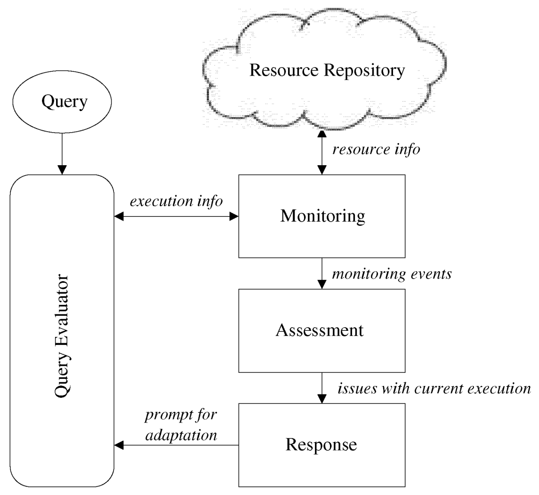

Fig. 1. The monitoring, assessment and response phases of AQP.

of capabilities. For example, a particular approach to monitoring can be used with different forms of assessment and response, or different categories of response can be made in the light of a single approach to monitoring and assessment.

2) The adoption of a generic framework makes the design activity more systematic. Developers can focus on the internal logic of the components without worrying about how the others are implemented internally, although the developer of the internal logic does have to think about the local and remote interfaces.
3) By focusing on the interfaces of cohesive, decoupled modules, the design of the framework conforms to service-oriented architectures, which are suitable for advanced, wide-area applications.

III. Instantiating the Framework

The framework has been instantiated to support several different forms of adaptation, including adaptive load balancing (ALB), for which the algorithm is described in more detail in [4]. In this setting, the framework has been instantiated as follows, with all components implemented as web services that implement a publish-subscribe interface:

- *Monitoring:* Each query plan fragment, of which there may be many running in parallel, acts as a monitoring component, using self-monitoring operators, as described in [3]. For ALB, each component is capable of generating events that describe the rate at which tuples are produced and the amount of time for which the evaluator is idle.
- *Assessment:* Each collection C of query plan fragments that are running a part of a query plan in parallel using partitioned parallelism is associated with an assessment component. The assessment component of C subscribes to the monitoring components of the plan fragments in C to obtain details on their throughput and idle time. The assessment component uses that information to detect load imbalance, and is capable of generating events that describe where imbalance has been detected.
- *Response:* Each query is associated with one or more response components, which subscribe to assessment components for which they may be able to implement an adaptation. For ALB, a response component subscribes to assessment components that are detecting imbalance, and where the imbalance is determined to require resolution, redistributes workload and associated state across the fragments, using the algorithm described in [4].

IV. Conclusions

This short note argues that AQP stands to benefit from a behavioural decomposition, in which loosely coupled complements, which implement publish-subscribe interfaces, can be combined in flexible ways to support different kinds of adaptation. The the resulting framework has been described, and illustrated through its instantiation for adaptive load balancing.

Acknowledgement

This work is supported by the EPSRC, through Grant GR/R51797/01, and through the UK e-Science Programme DAIT Project. We are pleased to acknowledge their support.

References

[1] M. 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*, pages 467–482, 2003.
[2] S. Babu and P. Bizarro. Adaptive query processing in the looking glass. In *CIDR*, pages 238–249, 2005.
[3] A. Gounaris, N. W. Paton, A. A. A. Fernandes, and Rizos Sakellariou. Self monitoring query execution for adaptive query processing. *Data and Knowledge Engineering*, 51(3):325–348, 2004.
[4] A. Gounaris, N. W. Paton, R. Sakellariou, and A. A. A. Fernandes. Adapting to changing resource performance in grid query processing. In *1st Int. Workshop on Data Management in Grids*, pages 30–44. Springer-Verlag, 2005.
[5] Z. Ives, A. Halevy, and D. Weld. Adapting to source properties in processing data integration queries. In *ACM SIGMOD*, pages 395–406, 2004.