

Automatic Coordination and Configuration of Net Data Management Applications

David Maier

Department of Computer Science and Engineering
Oregon Graduate Institute
maier@cse.ogi.edu

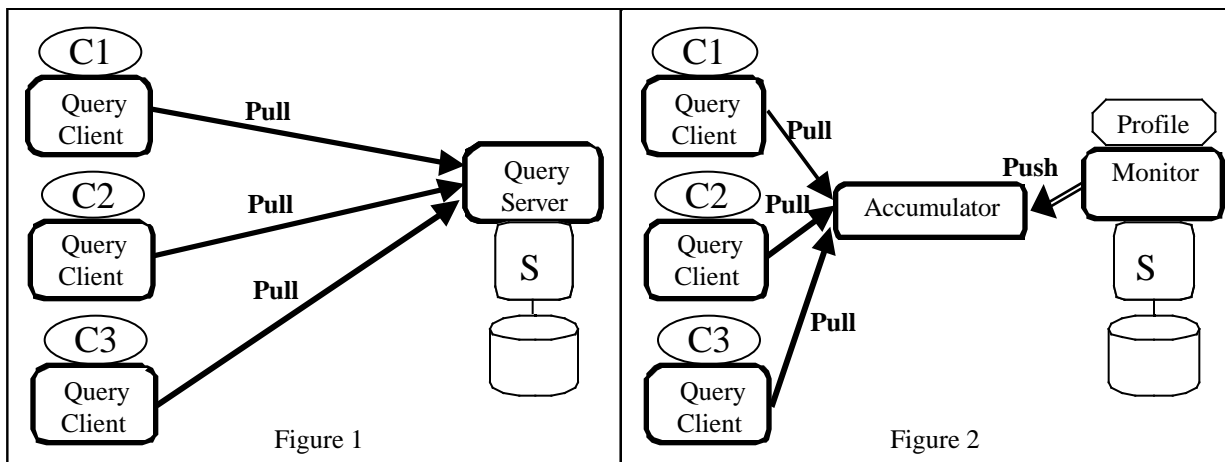
1 Introduction

The NIAGARA project (<http://www.cs.wisc.edu/niagara/>) at the University of Wisconsin and Oregon Graduate Institute is investigating *Net Data Management* (NDM): infrastructure for data-intensive, network-centric applications. We hope to do for such applications what relational database technology has done for business data processing applications: enormously reduce the coding effort to produce such applications, while improving performance, scalability and reliability. While conventional DBMS technology has focused on *data storage*, NDMs must concentrate on *data movement*. We imagine net data managers consisting of a variety of configurable components, from which net-centric applications can be assembled rapidly. Such components include **alterters** that can monitor thousands or millions of conditions over multiple data sources, **accumulators** that efficiently manage high-turnover data sets, **semantic routers** that direct information based on its content and **stream-based query processors**.

2 Scaling NDM Systems

One of our prime concerns is scalability, to cope with the number of information sources on the Internet, the number of potential clients and worldwide geographic distribution. Obviously, we are considering high-performance versions of NDM components, using indexing, optimization and parallelism—techniques that have served well in current DBMS technology. These techniques, however, target improvements in size and speed within individual applications. We see the potential in NDM applications for great efficiency gains if we can coordinate across multiple applications, to share processing work and data streams.

As a simple example, consider a data source *S* that is a catalog for a wide range of formed metals used in manufacturing (sheets, rods, bars of steel, brass, copper, etc.). Source *S* contains up-to-the-minute pricing and availability information. Figure 1 shows several applications that access *S* remotely, each periodically querying for information on products of interest. (Thus, their information is obtained in *pull* mode.) Suppose that the client-sides of these applications are geographically close (say within different departments of the same company) and that there is significant overlap in the information they access (say stainless steel rods and plates). We would reduce load on the server and bandwidth requirements with the architecture shown in Figure 2. A monitor at source *S* uses a common interest profile for the applications to push relevant updates to a local accumulator, from which the client sides pull the subset they are interested in.



Obviously, if someone was aware of the sharing between applications, the architecture of Figure 2 could be constructed manually. However, we would like to automatically enable shifts in processing strategy, to accommodate changes of interest, new applications being added, variations in update rates, and so forth. To enable such a coordination and configuration facility, NDM components need to externally reflect what they are doing in a way that commonalities can be detected. We would like to support a **coordinator** “meta-component” that can talk to regular NDM components about their current tasks, look for common processing work and data movement, and then reconfigure, retask and add or move components to improve overall efficiency.

3 Reflecting Component Tasks

A key question in supporting a coordinator is how components should express their current tasks to it. In the realm of relational databases, high-level descriptions of queries have been used in situations requiring automatic detection of commonalities. For example, a relational algebra (functional) representation is generally used for common sub-expression detection, while query containment algorithms typically start with a relational calculus (logical) form. While I believe a high-level representation is needed, relational algebra or calculus is unlikely to be adequate for NDM coordination. They work with too limited a data model—flat tabular data. We want to handle a much larger range of data types in NDMs, to reflect the wide variety of information forms seen on the Internet. The NIAGARA project currently uses XML [<http://www.w3.org/XML/>] as our external data model for components. While query languages for XML do exist, they don’t yet have the formal underpinnings of the relational model.

While being able to say what data an NDM component is interested in on behalf of an application is essential, it is not the whole story. We need to have some means to describe data sources, in addition to which subsets of their information are of interest. URLs aren’t the final word, as they don’t expose situations where the same information is available from different sources. To illustrate a further issue, consider Figure 3. Here, two of the applications are now being served completely in push mode. The choice to make this change could be based on the known behavior of the different applications. For example, the two applications might only ever retrieve any given information item once, while the first application could retrieve the same item several times. In order for a coordinator to automatically transition between these two architectures, it needs to know the different access regimens the NDM components follow in regard to the data they are interested in. The components need to externalize future plans. (Such information can also be used to determine a data-dropping policy for the accumulator.) Previous work on *microlanguages* (<http://www.cse.ogi.edu/sysl/projects/microfeedback/>), used for applications to inform system services of their upcoming data needs, may be relevant here.

In short, the challenge is to make NDM components that can reflect both interest and intent.

