DBCache: Database Caching For Web Application Servers*

Mehmet Altinel[1] Qiong Luo[2] Sailesh Krishnamurthy[3] C. Mohan[1] Hamid Pirahesh[1]
Bruce G. Lindsay[1] Honguk Woo[4] Larry Brown[5]

Contact email: {mohan,maltinel}@almaden.ibm.com

1. INTRODUCTION

Many e-Business applications today are being developed and deployed on multi-tier environments involving browser-based clients, web application servers and backend databases. The dynamic nature of these applications necessitates generating web pages on-demand, making middle-tier database caching an effective approach to achieve high scalability and performance [3]. In the DBCache project, we are incorporating a database cache feature in DB2 UDB by modifying the engine code and leveraging existing federated database functionality. This allows us to take advantage of DB2's sophisticated distributed query processing power for database caching. As a result, the user queries can be executed at either the local database cache or the remote backend server, or more importantly, the query can be partitioned and then distributed to both databases for cost optimum execution.

DBCache also includes a cache initialization component that takes a backend database schema and SQL queries in the workload, and generates a middle-tier database schema for the cache. We have implemented an initial prototype of the system that supports table level caching. As DB2's functionality is extended, we will be able to support subtable level caching, XML data caching and caching of execution results of web services.

2. OVERVIEW of DBCache

In this section, we present the main features and highlights of DBCache. The detailed description can be found in [2]. Here, we first explain the DBCache setup process and then give a brief summary of the run-time environment.

DBCache Setup:

The cache initialization tool creates a local database so that application programs can *transparently* use DBCache without any change. The backend tables to be cached are created locally in the cache database as ordinary tables and all other tables are represented in the schema with the *nickname* facility of the DB2's federated functionality. Nicknames are references to remote tables which can be used in a federated query as if they are local. The names of the cached tables and nicknames are created to be identical to their counterparts at the backend database to provide transparent execution of user application queries.

To populate the cached tables initially and to keep them up-to-date later, DBCache relies on DPropR [1] utility which is IBM's asynchronous data replication tool for relational data. The cache initialization tool configures the replication subscriptions for the cached tables so that when the capture and apply programs of DPropR start running, the cached tables are loaded with the data from their counterparts in the backend database and are synchronously updated at the specified interval.

Runtime Environment:

We have implemented a Query Router inside the DB2 engine by utilizing the *passthru* facility in DB2's federated functionality. Passthru allows federated database users to open a session directly to a remote server, bypassing the federated server. The query router uses the passthru commands to direct the queries to either the backend database or local cache database.

Two factors affect the Query Router decisions: (1) SQL statement type, whether it is an IUD (Insert/Update/Delete) query, and (2) currency settings of the cache database, indicating the user's tolerance to out-of-date data. When a DB2 instance is put into the DBCache mode, we capture the input SQL statements in a shallow parser to detect their types. If the query is read-only and the currency setting allows reading stale data, the query is executed locally. In this case, if the query involves nicknames, then existing federated query processing takes over and distributed query plans are selected based on costing information of both local and remote data sources. Furthermore, sometimes we need to use the cache database as a stand-alone database (e.g., for administration). For this purpose we implemented a modified version of the passthru command that allows targeting all the operations to the cache database without considering any routing decisions. This is especially required by the DPropR's apply program as it propagates all the updates to the cache database.

3. SYSTEM DEMONSTRATION

We will demonstrate DBCache using an e-Commerce benchmark called ECDW (Electronic Commerce Division Workload), which is developed and used internally by the WebSphere Commerce Performance group at IBM Toronto Lab. By simulating a range of e-Business applications using the benchmark, we will show the performance gain achieved by DBCache. With the help of dynamic monitoring tools, we will present under-the-hood mechanisms and techniques of DBCache. We will also demonstrate the cache initialization tool and show how it can be used to set up an effective database cache.

4. REFERENCES

[1] IBM DB2 Data Propagator, http://www-4.ibm.com/software/data/DpropR/

[2] Q. Luo, S. Krishnamurthy, C.Mohan, H. Woo, H. Pirahesh, B. G. Lindsay, J. F. Naughton, "Middle-tier database caching for e-Business", ACM SIGMOD'02, Madison, WI, June, 2002.

[3] C. Mohan, "Tutorial: Caching technologies for Web Applications", VLDB'01, Rome, Italy, September, 2001, http://www.almaden.ibm.com/u/mohan/Caching_VLDB2001.pdf

* Work done at IBM Almaden Research Center
[1] IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120
[2] Comp. Sci. Dept, Univ. of Wisconsin-Madison, Madison, WI 53706
[3] Comp. Sci. Div, Dept EECS, UC Berkeley, Berkeley, CA 94720
[4] Dept of Comp. Sci., University of Texas-Austin, Austin, TX 78712
[5] IBM Database Tech. Inst., 11400 Burnet Road, Austin, TX 78758

ACM SIGMOD'2002, June 4-6, Madison, Wisconsin, USA