Available online at www.sciencedirect.com
ScienceDirect
Information Systems 32 (2007) 320–343
www.elsevier.com/locate/infosys

The leganet system: Freshness-aware transaction routing in a database cluster

Stéphane Gançarski[a], Hubert Naacke[a], Esther Pacitti[b], Patrick Valduriez[b,*]

[a]*LIP6, University Paris 6, France*
[b]*INRIA and LINA, University of Nantes, France*

Received 17 February 2004; received in revised form 14 September 2005; accepted 20 September 2005

Recommended by: B. Kemme

Abstract

We consider the use of a database cluster for Application Service Provider (ASP). In the ASP context, applications and databases can be update-intensive and must remain autonomous. In this paper, we describe the Leganet system which performs freshness-aware transaction routing in a database cluster. We use multi-master replication and relaxed replica freshness to increase load balancing. Our transaction routing takes into account freshness requirements of queries at the relation level and uses a cost function that takes into account the cluster load and the cost to refresh replicas to the required level. We implemented the Leganet prototype on an 11-node Linux cluster running Oracle8i. Using experimentation and emulation up to 128 nodes, our validation based on the TPC-C benchmark demonstrates the performance benefits of our approach.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Database cluster; Transaction routing; Load balancing; Replication; Freshness; Performance

1. Introduction

Database clusters now provide a cost-effective alternative to parallel database systems. A *database cluster* [1] is a cluster of PC servers, each running an off-the-shelf DBMS. A major difference with parallel database systems implemented on PC clusters [2], e.g., Oracle Real Application Cluster, is the use of a "black-box" DBMS at each node which avoids expensive data migration. However, since the DBMS source code is not necessarily available and cannot be changed or extended to be "cluster-aware", additional capabilities like parallel query processing must be implemented via middleware. Database clusters make new businesses like Application Service Provider (ASP) economically viable. In the ASP model, customers' applications and databases (including data and DBMS) are hosted at the provider site and need be available, typically through the Internet, as efficiently as if they were local to the customer site. Thus, the challenge for a provider is to fully exploit the

*Corresponding author. Tel.: +33 2 51 12 58 24; fax: +33 2 51 12 58 97.
E-mail addresses: Stephane.Gancarski@lip6.fr (S. Gançarski), Hubert.Naacke@lip6.fr (H. Naacke), Esther.Pacitti@univ-nantes.fr (E. Pacitti), Patrick.Valduriez@inria.fr (P. Valduriez).

0306-4379/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2005.09.004

cluster's parallelism and load balancing capabilities to obtain a good cost/performance ratio. The typical solution to obtain good load balancing in a database cluster is to replicate applications and data at different nodes so that users can be served by any of the nodes depending on the current load. This also provides high-availability since, in the event of a node failure, other nodes can still do the work. This solution has been successfully used by Web search engines using high-volume server farms (e.g., Google). However, Web search engines are typically read-intensive which makes it easier to exploit parallelism.

In the ASP context, the problem is far more difficult. First, applications and databases must remain *autonomous*, i.e., remain unchanged when moved to the provider site's cluster and remain under the control of the customers as if they were local, using the same DBMS. Preserving autonomy is critical to avoid the high costs and problems associated with code modification. Second, applications can be update-intensive and the use of replication can create consistency problems [3,4]. For instance, two users at different nodes could generate conflicting updates to the same data, thereby producing an inconsistent database. This is because consistency control is done at each node through its local DBMS. The main solution readily available to enforce global consistency is to use a parallel database system such as Oracle Real Application Cluster or DB2 Parallel Edition. If the customer's DBMS is from a different vendor, this requires heavy migration (for rewriting customer applications and converting databases). Furthermore, this hurts the autonomy of applications and databases which must be under the control of the parallel database system.

In this paper, we describe a new solution for routing transactions in a database cluster which addresses these problems. This work has been done in the context of the Leg@Net project sponsored by the RNTL[1] whose objective was to demonstrate the viability of the ASP model for legacy (pharmacy) applications in France. Our solution exploits a replicated database organization. The main idea is to allow the system administrator to control the tradeoff between database consistency and performance when placing applications and databases onto cluster nodes. Databases and applications are replicated at multiple nodes to increase access performance. Application requirements are captured (at compile time) and stored in a shared directory used (at run time) to allocate cluster nodes to user requests. Depending on the users' requirements, we can control database consistency at the cluster level. For instance, if an application is read-only or the required consistency is weak, then it is easy to execute multiple requests in parallel at different nodes. But if an application is update-intensive and requires strong consistency (e.g., integrity constraint satisfaction), an extreme solution is to run it at a single node and trade performance for consistency.

There are important cases where consistency can be relaxed. With lazy replication [5], transactions can be locally committed and different replicas may get different values. Replica divergence remains until reconciliation. Meanwhile, the divergence must be controlled for at least two reasons. First, since synchronization consists in producing a single history from several diverging ones, the higher the divergence is, the more difficult the reconciliation. The second reason is that read-only applications may tolerate reading inconsistent data. In this case, inconsistency reflects a divergence between the values actually read and the values that should have been read in ACID mode.

In most approaches (including ours), consistency reduces to freshness: update transactions are globally serialized over the different cluster nodes, so that whenever a query is sent to a given node, it reads a consistent state of the database. In this paper, global consistency is achieved by ensuring that conflicting transactions are executed at each node in the same relative order. However, the consistent state may not be the latest one, since update transactions may be running at other nodes. Then, the *data freshness* of a node reflects the difference between the data state of the node and the state it would have if all the running transactions had already been applied to that node.

In this paper, we describe the design and implementation of the Leganet system which performs freshness-aware transaction routing in a database cluster. We use multi-master replication and relaxed replica freshness to increase load balancing. The Leganet architecture, initially proposed in [6], preserves database and application autonomy using non-intrusive techniques that work independently of any DBMS. The main

[1] www.industrie.gouv.fr/rntl/AAP2001/Fiches_Resume/LEG@NET.htm, between LIP6, Prologue Software and ASP-Line.

دریافت فوری ← متن کامل مقاله

ISIArticles
مرجع مقالات تخصصی ایران

✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
✓ امکان دانلود نسخه ترجمه شده مقالات
✓ پذیرش سفارش ترجمه تخصصی
✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
✓ دانلود فوری مقاله پس از پرداخت آنلاین
✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات