IaaS 環境における Warm-Cache DBMS インスタンス生成手法

福 地 開 帆 † 山 田 浩 史†

1. Introduction

Database management systems (DBMSs) are an essential component of web services and the performance of DBMSs is a dominant factor to the quality of the service. In Infrastructure-as-a-service (IaaS) platforms, users can scale up and down their services to dynamic workloads by adjusting the number of virtual machines running DBMSs (DBMS instances). Some cloud venders offer supports to easily manage DBMS instances. For example, Amazon RDS provides well-configured VMs running a familiar DBMS such as MySQL and PostgreSQL, and enables us to scale up the DBMS for read-heavily workloads such as business analysis by launching one or more DBMS instances with the read-only DB whose contents are the same as one of the already running DBMS.

However, handling burst workloads is still difficult even for read-only requests. The throughput of a DBMS is not high just after launching the DBMS instances due to frequent DBMS's cache misses. When the buffer pool of the DBMS does not have target data, the DBMS has to fetch it from much slower disks than memory. Since disks are basically shared with cloud users' instances and receive a huge amount of requests, warming the DBMS cache takes long time and the I/O performance is not stable. To successfully handle burst requests, users are required to launch redundant DBMS instances and redirect coming requests to the instances to warm the buffer pool of the DBMSs. As a result, users have to perform error-prone and difficult administrative tasks such as an appropriate prediction of redundant DBMS instance number and a careful configuration of load balancers for the request redirection.

To release users from these tasks, this paper presents a hypervisor-level mechanism to achieve high performance of DBMSs just after launching the DBMS instances. Our key insight behind this approach is to warm up new DBMS's buffer pool by leveraging the already-running DBMS's one, instead of issuing disk reads. Specifically, our mechanism tracks DBMS's read requests to disks, memorizes which memory pages contain DB data, and shares pages containing the same DB data with the original DBMS instance. By this page sharing, the new instantiated DBMS fetches requested data without slow disk accesses. Note that we assume that new launched DBMS instances have the same DB as the original running DBMS instance's one like Amazon RDS described above.

2. Approach

This idea is based on hypervisor-level advanced page sharing mechanisms[1),2)]. For example, Satori[2)] provides a copy-on-write sharing disk to VMs, checks block numbers in VMs' read requests to the disk, and share the page if the requested block has been read already and placed into a page of other VMs. They can effectively share pages of *kernel level* buffer cache among VMs, but do not work for DBMS buffer pools since a hypervisor does not have any knowledge about which pages contain DB data and which DB data is contained in a page.

To track memory pages into which DB data read from disks is placed, our mechanism adds to disk read requests *IDs* that are unique for every DB data in a DBMS. For example, the ID of MySQL, which is a relational DBMS, consists of a pair of the space number and the offset. The ID of Riak, which is a key-value store, is the key number. The DBMSs pass IDs to the guest kernel through a new system call in reading data from disks. The guest kernel put the IDs to disk read requests that are intercepted by the hypervisor. Our mechanism manages a map table that associates the IDs with the address of the machine page into which the read data is placed. If the ID of the read request from

† 東京農工大学
Tokyo University of Agriculture and Technology

DBMSs is in the map table, our mechanism map the machine page to the page in the map table in a copy-on-write manner.

3. Current Status

We have implemented a prototype into Xen 4.2.1 and tailored MySQL 5.6.14 and Linux 3.11.5 to the prototype. We conducted a preliminary experiment and its result demonstrates that our approach significantly outperforms the conventional method of generating a read replica.

参 考 文 献

1) K.Miller, F.Franz, M.Rittinghaus, and M.Hillenbrand. XLH: More Effective Memory Deduplication Scanners Through Cross-layer Hints. In *Proc. of the USENIX ATC '13*, pages 279–290.
2) G. Milos, D. G. Murray, S. Hand, and M. A. Fetterman. Satori: Enlightened page sharing. In *Proc. of the USENIX ATC '09*, pages 1–14.