Zephyr: Database Live Migration in Shared Nothing Databases for Elastic Cloud Platform:

Tawatchai Siripanya

Seminar on Data Management:
No-/ New-SQL techniques and systems
S 19562
Instructor: Prof.Dr.-Ing. Heinz F. Schweppe
July 31, 2012
Computer Science Division
Free University of Berlin, Berlin
tsiripanya@googlemail.com

Abstract. Elastic load balancing is believed to be the key idea of controlling resources and costs in elastic could platforms [1]. The database system can be scaled up when the load is high, and be scaled down during low load to save costs. The paper presents Zephyr, the technique that can minimize service interruption and with no downtime during the database migration.

1 Introduction

When we think about using online application such as in Facebook.com, we can see that there are many applications we can use with Facebook. We can feel that the numbers of online application are growing. When there are new interesting applications coming, we want to use and try it. After that, if there are not interesting anymore or the trail period has expired and we do not want to pay for it, we just unsubscribe and do not use it. On the other hand, when we were the online application providers, we must be able to control the resources essential for the online applications such as databases because most applications today, the database is mandatory. Also, the database is not for free, and of course there are many open source databases available, but still we have to pay for it such as the operational costs. Therefore, as providers on one side, we must be able to make the databases or services available for all users (or tenants) when the numbers of users increase. On the other side, when the numbers of users decrease, we must decrease the number of resources such as database nodes to reduce the costs. We called this as elasticity; the application can scale up for high load, and scale down for low load. Now, when talking about scaling up and scaling down in the databases system for the elastic could platform or the cloud environments, where providers pay for the infrastructure per usages or pay per use. The providers can expand the infrastructure and migrate databases to the new infrastructures

(nodes) when high load to accomplish the service level agreements (SLA). They can also shrink the number of nodes to save the costs.

The paper aims to introduce **Zephyr [1]**, a technique for database live migration with no downtime independent on the database size.

The remainder of the paper is structured as follows. Section 2 provides background information. It describes the basic terminology for database migration such as why we have to migrate databases. Section 3 presents the Zephyr technique and describes the limitations of Zephyr. Section 4 evaluates the Zephyr by comparing it to the stop and copy technique and gives the results of the experiment from [1]. Finally, Section 5 concludes the paper.

2 Database Live Migration

In this section we will give a short description about database live migration include telling what database live migration is, explaining why we have to do database live migration, giving information what to keep in mind while doing live migration, and giving techniques of database live migration.

2.1 What is database live migration?

When talking about data migration, we can think of copying data e.g. from our personal computer (PC) at home to our notebook. Also, talking about database migration, we can think of copying databases from a database server to a new database server. We called database live migration, when during the migration the source database server and the destination database server can service transactions.

2.2 Why do we need database live migration?

In daily business world, we can assume that our database system can be executed 24 hours per day without downtime. When, business transactions have been interrupted, the business might get lesser benefits or damage the business reputation. So, business transactions shall not interrupted, therefore the migration of databases shall not have downtime and without interruptions. At the provider's point of view, we have to accomplish the SLA while occurrences of high workload and the tenant should not be noticed, for this reason; we have to migrate data from the node with high workload to a new node.

2.3 What to keep in mind when doing database live migration?

As mentioned earlier, there is no perfect database system in the real world. The things we have to keep in mind, when we dealing with database live migration can be described in the following: firstly, the costs of migrant that might associate with the SLA. These costs can be for instance; service interruption, migration overhead, and

additional data transferred. Due to the page limitation, the paper will not further describe details of the mentioned costs, also the correctness and fault tolerance of Zephyr have to be skipped, for further reading can be found in [1], and [3].

2.4 What are techniques for database live migration?

There are many techniques available for database migration such as *stop and copy*; this technique, the source stops services of updates, checkpoints the states, then, the destination loads the checkpoints. And, finally, the services are started at the destination. This technique is however expensive because of the long period of services interruption or downtime during migration. This paper will introduce the other technique which is called "ZEPHYR". This technique allows both source and destination database server service transaction concurrently. This technique is explained in the next sections.

Now, before we describe more details about live migration in cloud computing environment, we have to understand the fundamental terms that will be used throughout this paper.

2.5 Multitenancy Models

In general, there are three multitennacy models include *shared table*, *shared process*, and *shared machine* [1]. We can describe the three models in the following:

Shared table model

The example for the share table model can be such as; the applications provided by Saleforce.com. These applications are typically called Software as a Service or in short SaaS. The data of tenants are stored in a big table. Each tenant is isolated from each other using row isolation level. This technique maximizes resource sharing, however the schema of each tenant has to be identical [1].

Shared process model

This model allows tenants to have different schema. Also, it supplies better isolation than the shared table model in the same database process [1]. But, it cannot supplement virtual machines (VM) migration to move individual tenants from a shared database process [1].

Shared Machine

This model uses VM to isolate tenants from each other. That is, each tenant is installed in different VM. Using this technique can leverage VM migration techniques for elastic load balancing [1]. But, by doing so, it is inefficient for resource sharing between tenants [2].

3 ZEPHYR Technique

3.1 What is ZEPHYR

ZEPHYR is a framework developed by the University of California, Santa Barbara, USA in 2011. It aims to avoid the downtime during the database migration that allows both source database and the destination database nodes execute transactions concurrently. It uses page ownership principle, index wireframe, and based on the share process model [1].

3.2 How does ZEPHYR work?

ZEPHYR aims to avoid service interruptions by increasing service availability allowing both the source and the destination concurrently serves services. Unlike stop and copy technique which migrates data one time as a whole, ZEPHYR breaks down the migration into collection of phases. It starts to transfer minimal information from the source to the destination called "wireframe" consists of database schema, metadata, user authentication, and index wireframe. This information allows the destination to serve services or to execute transactions. We called this period or phase "Init Mode". During migration the index structures are made immutable; index structures are not allowed to be changed at both source and destination; that means any update that will change the index structures will be aborted. Then, the source will tell the router to direct any new transactions to arrive at the destination. In other word, the source will serve only the existing transactions, and the destination will serve the new coming transactions. By doing so, the source and the destination are allowed to execute transactions concurrently. ZEPHYR introduces the concept of *page owner ship* that is; at first the source owns all the pages, the destination will pull the data from the page that it not owned from the source for the new transactions on demand. Then, the source will transfer the page ownership to the destination. Once the data has been pulled by the destination, it is not allowed to pull back. There is no synchronization between the source and the destination at this point; this allows minimizing the overhead during this period. The system controller is used to determine the destination for migration. We called the period where the source and the destination are sharing the ownership of database being migrated " Dual Mode". When there are no more transactions in the source, the source will initiate transfer of the exclusive ownership to the destination using a handshake [1] between both two nodes, and then both nodes (the source and the destination) enter the "Finish Mode". In the finish mode, only the destination serves transactions and it does not yet has the ownership of all the database pages, therefore the source has to push the remaining database pages to the destination so that the destination is the sole owner of the database pages. The source initiates the termination of migration, when all database pages have been moved to the destination and the operation can be switched back to the normal operation mode. At this period, the handshake between the source and the destination is invoked again. Also, the source will notify the system controller of the termination. The figure 1

illustrates the timeline of different phases during migration and the figure 2 shows the ownership transfer of database pages during the migration.

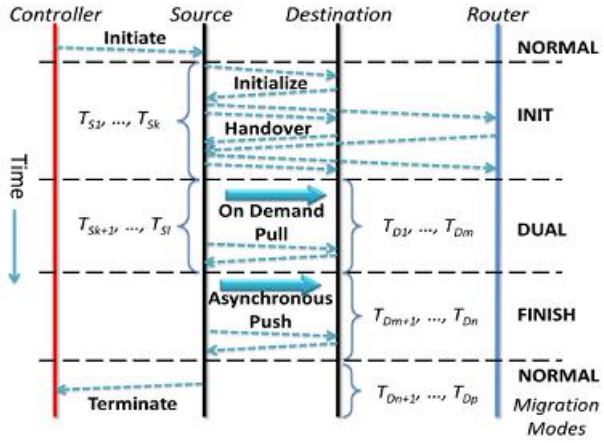

Fig. 1. Timeline during the migration [1]

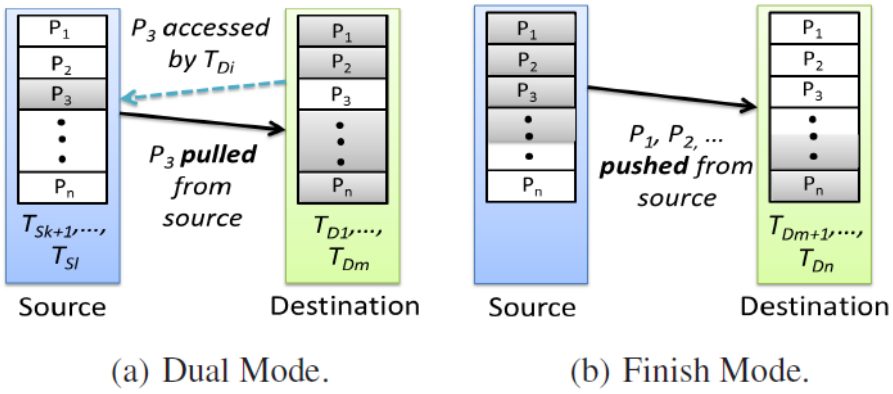

Fig. 2. The ownership transfer of the database pages [1]

3.3 What are limitations of ZEPHYR

Zephyr is not designed to support large tenants as described in [1]. Also, it relies heavily on the index structures that must be made immutable during migration. Furthermore, Zephyr is not off-the-shelf software that can be used right away, that means extra implementations and modifications are required. Further information about the implementation of Zephyr as described in [1] can be found only a short detail.

4 ZEPHYR Evaluation

We can describe the experimental evaluation of Zephyr in the following [1]: Zephyr has been compared with the off-the-shelf stop and copy technique described in the earlier section. There was two database nodes with 2.40GHz Intel Core 2 Quad processor, the main memory 8 GB, Hard disk drive 7200 RPM SATA with 32MB Cache, the operating system 64-bit Ubuntu Server Edition with Java 1.6, the network connection: via a gigabit switch, the workload was using load Generator & transactions from Yahoo! cloud serving benchmark (YCSB), the database size of (page size 16KB and cache size: 32MB) with 100,000 rows. And, the total size of the database was about 250MB.The significant founds are in the following: (1) Using Zephyr technique requires 10 to 18 seconds to finish the migration, also both the source and destination nodes have no downtime compare to stop and copy technique which requires shorter migration time but with the 3-8 seconds downtime [1]. (2) The stop and copy technique required longer time of unavailability as the size of the increases, on the other hand, for Zephyr required longer time to switch to the finish mode, but it does not result unavailability, also the size of the database has no effect to the number of failed operations [1]. Generally speaking, Zephyr requires longer migration, but it has no downtime. Furthermore, with Zephyr, the size of the database has no impact to the number of failed operations.

5 Conclusion and Perspectives

In this paper, we have described the database live migration, the reason why the database has to be migrated to a new node; one reason is to accomplish SLA, to make all tenants happy. Further, we have to consider migration costs include service interruption, migration overhead, and additional data transferred. There are two techniques have been introduced; the stop and copy and Zephyr techniques. The first technique, all update operations have to be aborted during migration which causes long service interruption, whereas Zephyr does not cause service interruptions during migration. We have found some limitations of Zephyr and have given them in short details. The experiment has shown Zephyr requires longer migration than the stop and copy technique, but it has no downtime. Furthermore, with Zephyr, the size of the database has no impact to the number of failed operations.

We believe that the limitations shall be improved, and more features shall be added in the future. Finally, we hope for more research in this area.

6 References

1. Ellmore, A. et al.: Zephyr: Live Migration in Shared Nothing Databases for Elastic Cloud Platforms, ACM SIGMOD '11, 2011.
2. C. Curino, E. Jones, R. Popa, N. Malviya, E. Wu, S. Madden,H. Balakrishnan, and N. Zeldovich. Relational Cloud: A Database Service for the Cloud. In CIDR, pages 235–240, 2011.
3. Barker, S. et al.: “Cut Me Some Slack”: Latency-Aware Live Migration for Databases, EDBT 2012.