Buddy Instance - A Mechanism for Increasing Availability in Shared-Disk Clusters

Anjan Kumar Amirishetty, Yunrui Li, Tolga Yurek, Mahesh Girkar, Wilson Chan, Graham Ivey, Vsevolod Panteleenko, Ken Wong

Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065, U.S.A.

{Anjan.Kumar.Amirishetty}@oracle.com

ABSTRACT

Oracle's Real Application Cluster (RAC) allows multiple database instances to run on different server nodes in a cluster against a shared set of data files. A critical aspect of an Oracle RAC system is that of *instance recovery*. When a node suffers from a hardware failure, or a database instance suffers from a software failure, instance recovery is performed by a surviving instance to ensure that the database remains in a consistent state. High-availability comes from the surviving database instances, each running on a surviving node, that are still able to provide database services. During instance recovery, the set of database resources that are in need of recovery must be identified and then repaired. Until such time as the identification of these resources has been done, Oracle needs to block any requests by database clients to all database resources. The whole database appears to be frozen during this time, a period that is called application brown-out. In the interests of availability it is therefore important that instance recovery endeavors to keep this period of identification as short as possible. In doing so, not only is the brown-out period reduced, but also the overall time to make available those resources that need repair, is reduced.

This paper describes the use of a *Buddy Instance*, a mechanism that significantly reduces the brown-out time and therefore also, the duration of instance recovery. Each database instance has a buddy database instance whose purpose is to construct in-memory metadata that describes the resources needing recovery, on a continuous basis at run-time. In the event of node or instance failure, the buddy instance for the failed instance uses the in-memory metadata in performing instance recovery. The buddy instance mechanism for single instance failures is available in the 12.2 release of Oracle Database. Performance results show a significant reduction in brown-out time and also in overall instance recovery time.

Categories and Subject Descriptors

H.2.4 **[Database Management Systems]:** Database transaction processing→ Database recovery, C.4 [**Performance of Systems**]: reliability availability and serviceability

General Terms

Algorithms, Design, Performance

Keywords

Database, Real Application Cluster, Recovery, Availability

1. INTRODUCTION

Oracle RAC [1] transparently extends database applications from single-node systems to multi-node systems which share the disks that provide storage for the database. The database spans multiple hardware systems yet appears as a single unified database to the application. An *instance* is a collection of processes and memory accessing a set of data files. Single-instance Oracle databases have a one-to-one relationship between the database and the instance. Oracle RAC environments, however, have a one-to-many relationship between the database and instances.

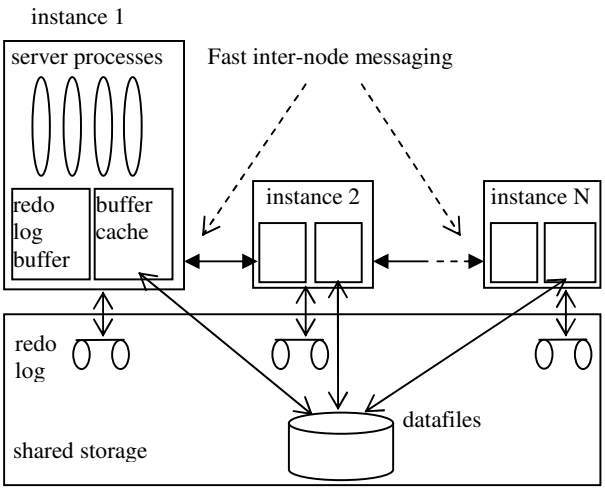

Figure 1. Real Application Cluster Architecture

An interconnect serves as the communication path between each node in the cluster database. Each Oracle instance uses the interconnect to exchange messages that synchronize each instance's use of shared resources.

RAC is so called since it transparently allows any database application to run on a cluster without requiring any application changes. RAC improves application performance since the application is executed in parallel across multiple systems. RAC also improves availability since the application is available as long as at least one of the cluster nodes is alive.

Each database instance in RAC has its own redo log. The redo log is a set of files that records all the changes to the database that have been made by the instance.

An Oracle RAC database is a shared everything database. All data files and redo log files must reside on cluster-aware shared disks so that all the instances can access these storage components.

In Oracle RAC, Cache Fusion [3] allows the data blocks to be shipped directly between Oracle instances through fast inter-node messaging, without requiring expensive disk I/O. Oracle instances therefore directly share the contents of their volatile buffer caches [8], resulting in a *shared-cache* clustered database architecture.

When some but not all instances of an Oracle RAC database fail, instance recovery is performed automatically by a surviving instance in the cluster. Instance recovery ensures that the database is in a consistent state after such a failure.

Instance recovery is done in two phases. The first phase, *cache recovery* or *rolling forward* [7], involves reapplying (or rolling forward) all necessary changes recorded in the redo log to the data blocks of data files. After cache recovery, data files could contain the changes of transactions that had not yet been committed at the time of failure.

The second phase of instance recovery, *transaction recovery* or *rolling back* [7], uses changes recorded in the undo segment to roll back uncommitted changes in data blocks. After transaction recovery, data files reflect a transactionally-consistent image of the database at the time of failure.

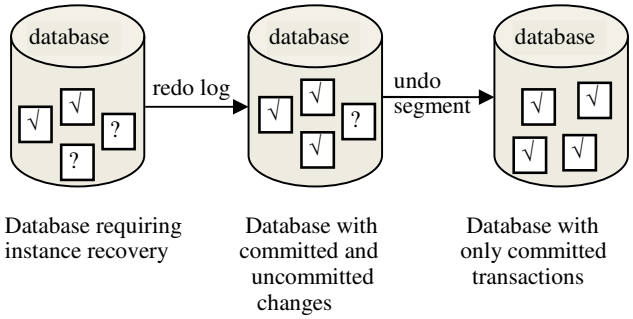

Figure 2. Recovery Phases: Cache Recovery and Transaction Recovery

Cache recovery must scan the redo log of each failed instance to recover the data blocks that were lost when these instances failed. Cache recovery scans the redo log in two passes [2]. The first pass constructs the metadata that is subsequently used by the second pass to speed-up recovery. Section 2 discusses the details of cache recovery in more detail.

The buddy instance mechanism potentially eliminates the first pass of cache recovery, thereby improving the performance of instance recovery.

Each instance in the cluster becomes a *protected instance* when another instance is designated to serve it as its *buddy instance*. As the protected instance records changes to the database in its redo log at runtime, its buddy instance proactively scans its log to build the metadata that the second pass can make use of if a failure were to happen at this moment. When the protected instance fails, its buddy instance performs cache recovery for the failed instance and uses the metadata it has accumulated to shortcut this process.

The rest of this paper is organized as follows. First, the motivation behind this new technique is described. After this, the existing two-pass recovery scheme is outlined. Then, the buddy instance mechanism that optimizes the two-pass recovery scheme for single instance failure is detailed. Following this there is a discussion on extending the buddy instance mechanism to multi-instance failure. Finally a performance study is tabled and related work is looked at.

2. MOTIVATING USE CASE

A major e-retail customer of the Oracle RAC database has been impacted by application brown-out that happens during instance recovery. The vast majority of these failures were single instance failures. A mechanism was needed to improve availability by reducing the length of brown-out. It was clear that this use case was not a specific one and that any improvements made, would benefit the majority of customers using RAC.

As the number of nodes increase in Oracle RAC database, probability of node failure increases and there is a need to perform instance recovery in a seamless manner, without affecting the database throughput.

The result was the buddy instance functionality, made available in the 12.2 release of Oracle Database.

3. CACHE RECOVERY

Each Oracle RAC instance is configured with its own cache of disk buffers which together, form a global buffer cache. In order to maintain cache coherency across this, global resource control is needed. The Global Cache Service (GCS) [3] tracks and maintains the locations and access modes of all data blocks in the global cache thereby maintaining the consistency of the database at the cluster level. Database blocks accessed concurrently by cluster instances have corresponding GCS resources to ensure the same data block is not updated without coordination across different instances.

GCS adopts a distributed architecture. Each instance shares the responsibility of managing a subset of the global cache. GCS maintains the status of global cache resources to ensure the overall consistency of database. When one or more instances fail, Oracle needs to rebuild the global cache resource information. Only the cache resources that reside on or are mastered by the GCS on the failed instances, need to be rebuilt or re-mastered.

3.1 Checkpointing

Cache recovery uses checkpoints to determine the set of changes that must be applied to the data files. A checkpoint represents the point at which all changes to the database have been made persistent.

Each instance in Oracle has its own redo log which is effectively, an ever-growing list of redo records generated by an instance [4]. The position of each record in the redo log may be identified by its *redo byte address* (RBA) [4]. The location of the checkpoint is identified using the checkpoint RBA. This is the position in the redo log of an instance at which all changes to data blocks made by that instance are known to be on disk. Hence, recovery for that instance needs to recover only those data blocks whose redo records occur between the checkpoint RBA and the end of the log [4].

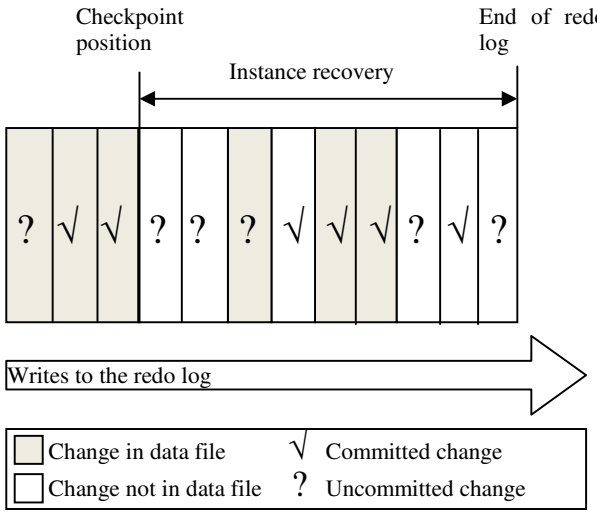

Figure 3. Checkpoint Position in Redo Log

Cache recovery must scan the redo log of each failed instance and apply the changes that occur between the marker for the last checkpoint and the end of the redo log.

3.1.1 Instance Checkpoint

The low RBA for a disk buffer is the RBA corresponding to the first (in-memory) modification of the data block. Oracle Database maintains a buffer checkpoint queue [4] which contains modified buffers linked in ascending order of their *low RBA*. Each buffer header contains the value of the low RBA associated with the buffer; this value is set when the buffer is first modified. A buffer that contains a yet-to-be-changed block does not have a low RBA in its buffer header and is not linked on the checkpoint queue. After a changed buffer is written, it is unlinked from its checkpoint queue.

As buffers from the head of the queue are written to disk, the instance checkpoint (lowest low-RBA of the modified buffers) will keep advancing [4]. This lowest low-RBA is referred to as the current position of the *instance checkpoint* for the instance. The instance checkpoint advances the database checkpoint RBA as a lightweight background activity.

3.1.2 Database Checkpoint

Each change in Oracle is associated with a time, known as the *system change number* (SCN). Instance checkpoint is also associated with a SCN. The *database checkpoint* in RAC is the instance checkpoint that has the lowest checkpoint SCN of all the instances.

3.2 Two-pass Recovery Scheme

Instance recovery for all failed instances is triggered automatically on a surviving instance. Oracle uses a two-pass database recovery scheme [2] to recover the changes to data blocks that were lost on the failed instances. The first pass scans the redo logs of each failed instance to decide the data blocks that need to be recovered. This list of blocks is referred to as the *recovery set*. The second pass applies redo from the redo logs to the blocks in the recovery set.

A *Block Written Record* (BWR) is recorded in the redo log whenever an instance writes a block to a data file. When the first pass encounters a BWR, the corresponding data block entry in the recovery set is removed because it is known that at this point in time, the block changes have been made persistent on disk. BWRs allow instance recovery to avoid unnecessary reads of data blocks that were not modified between being written to disk and the point at which instance failure occurred. BWRs ensure that the recovery set constructed by the first pass is much smaller than the total number of blocks that were actually modified on the failed instances.

The first and second passes both start at the lowest checkpoint SCN of all failed instances. The redo records of all the failed instances are merged in SCN order. In both passes, Oracle scans the redo until the end of all redo logs for all the failed instances, proceeding through as many log files as necessary to complete cache recovery and roll forward the database to the state it was in at the time of instance failure. Because changes to blocks in the undo segment are recorded in the redo log, rolling forward the redo log also regenerates the corresponding undo blocks that contain a record of changes that need to be undone when transaction recovery is run to roll back incomplete transactions.

Oracle can initiate the first pass of the recovery process concurrently with the GCS rebuild process. After the first pass completes, the database is made available for service to applications for all but the data blocks impacted by the failure [2] (that is, for all data blocks but those in the recovery set). The buddy instance mechanism can potentially eliminate the first pass thereby making the database available for service almost immediately.

4. BUDDY INSTANCE MECHANISM TO HANDLE SINGLE INSTANCE FAILURES

Under the buddy instance mechanism, each RAC instance becomes a protected instance by virtue of having a designated instance to serve as its buddy. In a two instance RAC database shown in Figure 4, instance-1 is designated to serve instance-2 as its buddy instance and instance-2 is designated to serve instance-1 as its buddy instance. This designation of buddy instances is referred to as *buddy instance map*. This is also called as *one-on-one buddy instance map* as each protected instance has one designated buddy instance.

As changes to the database are recorded at run-time by an instance in its redo log, a server process in its buddy instance continuously scans that redo to construct the recovery set. The server process starts at the checkpoint RBA (referred to as the s*tart RBA*) and scans the redo till the end of the log. By default, the rate at which this server process scans the redo is adjusted to the ongoing redo generation rate. This can be overridden by using the *_buddy_instance_num_read_buffers* parameter to establish a constant redo scan rate. This parameter dictates the number of buffers, each of size 4MB, which will be read and processed approximately every three seconds. If the value of the parameter is low, the server process scans less aggressively. This results in less load on the system but has the disadvantage that the amount of work required by the first pass of instance recovery may increase.

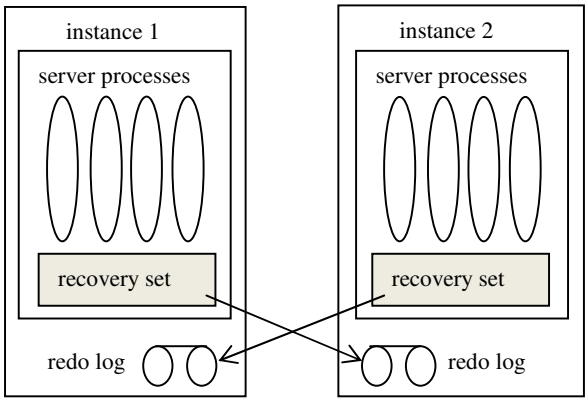

Figure 4. Buddy instance map in two instance RAC database

At regular intervals, the server process estimates the amount of time required for the first pass of instance recovery, if a crash were to happen at that time. If that amount of time is less than the value of the *_buddy_instance_scan_phase_threshold* parameter (which has a default value of 3 seconds), the redo is not scanned.

For each block in the recovery set, Oracle maintains the *last RBA* which refers to the RBA of the last redo record that changed or created that block. If defined, this RBA must be between the start RBA and the end of the log.

4.1 Recovery Set Pruning

As the checkpoint of an instance progresses, its buddy instance must advance its start RBA to that of the instance's checkpoint RBA. After advancing this, the recovery set can be pruned by removing blocks which have a last RBA that is less than the new start RBA.

4.2 RAC Membership Changes

Oracle RAC Database maintains the buddy instance map and automatically updates it as and when instances join or leave the cluster. Instances can be added or taken out of an Oracle RAC system without shutting the database down. When an instance joins or leaves the cluster, the buddy instance map must be dynamically adjusted to reflect the new cluster configuration.

When instance-3 joins the Oracle RAC system shown in Figure 4, the buddy instance map is updated to that shown in Figure 5. Here, instance-1 is designated to serve instance-2 as its buddy instance, instance-2 is designated to serve instance-3 as its buddy instance and instance-3 is designated to serve instance-1 as its buddy instance.

In the similar way, when instance-3 leaves the Oracle RAC system shown in Figure 5, the buddy instance map is updated to that shown in Figure 4.

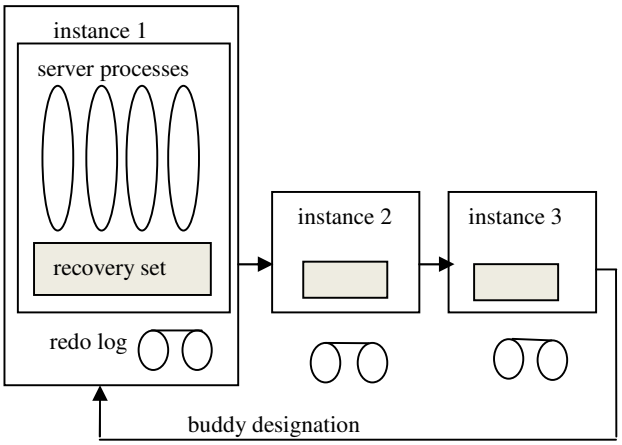

Figure 5. Buddy instance map in three instance RAC database to handle single instance failure

4.3 Handshake with Instance Recovery

When an instance fails, its buddy instance is asked to perform instance recovery of the failed instance. In the first pass of instance recovery, the buddy instance uses the recovery set that was constructed during run-time. If the checkpoint RBA of the recovery set that was constructed during runtime is behind the checkpoint RBA as determined by instance recovery, the buddy instance prunes the recovery set using the checkpoint RBA for instance recovery. If the buddy instance had not scanned till the end of the redo log prior to instance failure, it will do so during the first pass of instance recovery.

GCS can make the global cache available to surviving instances as soon as the recovery set is constructed by the first pass of instance recovery. Since Oracle is expected to spend significantly less in the first pass, the availability of the database is significantly increased by using the buddy instance mechanism.

5. EXTENDING BUDDY INSTANCE MECHANISM TO HANDLE MULTI-INSTANCE FAILURES

The buddy instance mechanism of Oracle RAC currently handles single instance failure only. This section presents a possible implementation of the buddy instance mechanism for multi-instance failures. This does not constitute a commitment by Oracle to deliver any code or functionality and should not be relied upon in making purchasing decisions.

One possible implementation to handle multi-instance failure is to give each protected RAC instance more than one buddy instance each continuously scanning the redo log of the protected instance during runtime to construct its recovery set. The number of instances that serve as buddies determines the degree of multi-instance failure that can be handled while fully getting the benefit of the buddy instance mechanism. In a RAC of n-instances, each instance can have up to (n-1) buddies.

The recommended buddies for an instance depend on statistics such as number of instances that previously failed together and which instances failed together. It is possible for Oracle Database to recommend the buddies for each of the RAC instances based on the statistics that were collected during previous instance failures.

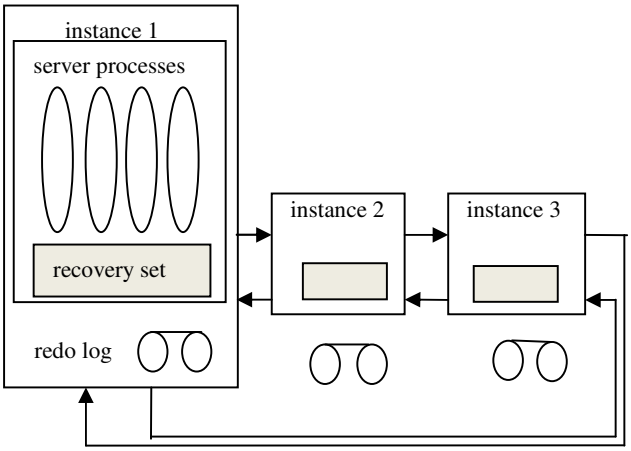

Figure 6. Buddy instance map in three instance RAC database to handle multi-instance failure

Figure 6 shows an example of a three instance Oracle RAC system in which a single instance is protected by two buddies. This system can tolerate up to two instances failing simultaneously, while still taking advantage of the buddy instance mechanism.

5.1 Recovery Set Pruning

For each protected RAC instance, all its buddy instances must perform the steps detailed in Section 4.1.

As the checkpoint of an instance progresses, all its buddy instances must advance the start RBA to that of the instance's checkpoint RBA and prune its recovery set by removing the blocks which have a last RBA that is less than the new start RBA.

5.2 RAC Membership Changes

This is an extension of the RAC membership changes described in Section 4.2, where instances can be added or taken out of an Oracle RAC system without shutting the database down.

When an instance joins or leaves the cluster configuration, the buddy map needs to be updated based on the statistics that were collected during previous instance failures with the same cluster configuration.

5.3 Handshake with Instance Recovery

When one or more instances fail, instance recovery is performed by a surviving buddy instance which has the recovery set for the most number of failed instances. This instance is designated the *recovery instance.* Below is the sequence of events that are performed on recovery instance during the first pass of recovery.

1. If the recovery instance is the buddy of a failed instance and it did not scan the redo till the end of the log prior to the failure, the recovery instance needs to do so now.
2. If the recovery instance does not have the recovery set for a specific failed instance, it needs to receive the recovery set from the buddy of that specific instance (if there is one). If that buddy instance had not scanned the redo till the end of the log prior to the failure, the recovery instance needs to do so now.
3. If a failed instance does not have a surviving buddy instance, the recovery instance needs to scan the entire redo of that failed instance from the checkpoint of the failed instance to the end of the log.
4. The recovery instance needs to merge the recovery sets of all the failed instances.

In the Oracle RAC system shown in Figure 6, if both instance-1 and instance-2 fail, instance recovery needs to be performed by instance-3 which has the recovery set for both of the failed instances.

6. PERFORMANCE STUDY

Experiments were conducted to measure the impact on database throughput and also to measure the acceleration in instance recovery.

6.1 Impact on Run Time Performance

Experiments were conducted using a TPC-C workload to evaluate the impact on database throughput with the buddy instance mechanism enabled.

6.1.1 Hardware Setup

The study was conducted using an Oracle Exadata Database Machine [6] with an InfiniBand cluster interconnection for Oracle RAC servers. An X3-2 RAC configuration was used, comprising two database server nodes, each equipped with 32, 8-core Intel Xeon processors running at 2.9 GHz and 128 GB of memory.

6.1.2 TPC-C Workload

TPC-C benchmark is the industry standard for evaluating the performance of OLTP systems [5]. The setup consisted of 1000 warehouses and 512 clients.

CPU utilization at 91%, was not affected when using the buddy instance mechanism. Not surprisingly however, the number of reads of the redo log increased.

Figure 7 shows the impact on database throughput for a fully-cached TPC-C workload (where the buffer cache is sized large enough to accommodate the entire database), and a partially-cached TPC-C workload (where the buffer cache is sized at approximately 20% of the database size). For a fully-cached TPC-C workload, no impact was observed on database throughput

when the buddy instance mechanism was enabled. Due to the nature of the workload, around 1% variation is expected across different runs. The impact on database throughput was less than 2% for the partially cached TPC-C workload.

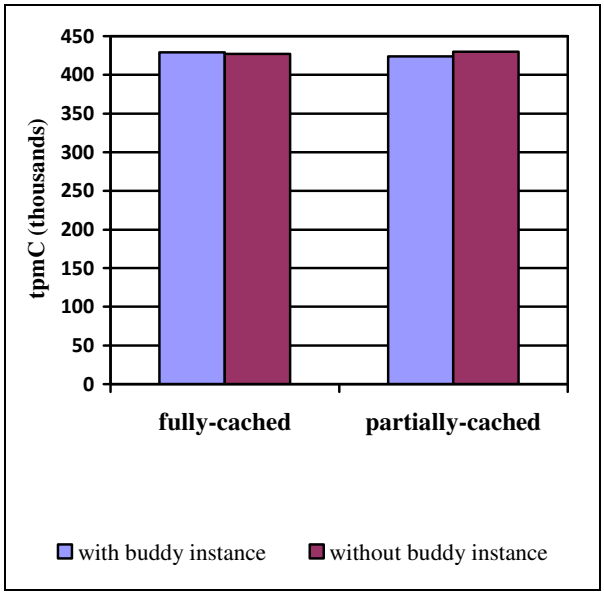

Figure 7. Impact on database throughput (tpmC) for a TPC-C workload

6.2 Acceleration in Instance Recovery

6.2.1 Hardware Setup

This study was also conducted using an Oracle Exadata Database Machine [6] with an InfiniBand cluster interconnection for Oracle RAC servers. An X2-8 RAC configuration was used, comprising two database server nodes, each equipped with 8, 12-core Intel Xeon processors running at 2.40 GHz, 2 TB of memory and 14 shared storage servers amounting to 200 TB total storage capacity over a Direct-to-Wire 3 x 36 port QDR (40 Gb/sec) InfiniBand interconnect.

6.2.2 TPC-C Workload

The TPC-C benchmark application was run to generate a workload. The FAST_START_MTTR_TARGET parameter [10] which affects the rate of checkpointing and hence the duration of instance recovery, was set to its default value of "0" in line with what is done on most customer systems. After running the TPC-C benchmark for 27 minutes, an instance was crashed. Figure 8 shows the time taken both by the first pass of instance recovery and the time taken overall by instance recovery. The time taken for instance recovery as a whole has been reduced because of the decrease in the time taken for the first pass.

Without making use of the buddy instance, instance recovery spent 130 seconds in its first pass when the second pass needed to apply 11.5 GB of redo. By comparison, when using the buddy instance mechanism, instance recovery spent only 1 second in its first pass when the second pass needed to apply a similar amount of redo.

Since the first pass correlates with brown-out time, the time that the database was completely unavailable to applications was reduced from 130 seconds to 1 second.

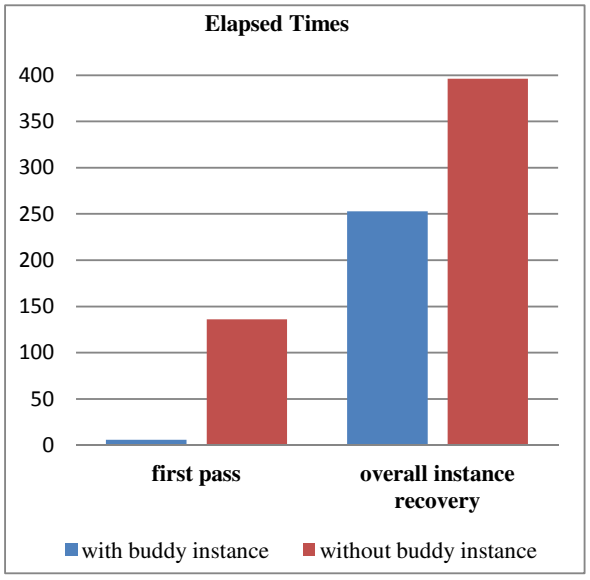

Figure 8. Elapsed times (seconds) for a TPC-C workload

6.2.3 Results for a Commercial Workload

The experiment was repeated using a real customer workload. The database consisted of 31 tables and 35 indexes. The total on-disk size of the database was approximately 200GB. This workload generated redo by having multiple concurrent users repeatedly perform updates and inserts into their own tables. An instance was then crashed. Figure 9 shows the time taken by the first pass and the time taken overall for instance recovery. This experiment had set the FAST_START_MTTR_TARGET [10] parameter to 100.

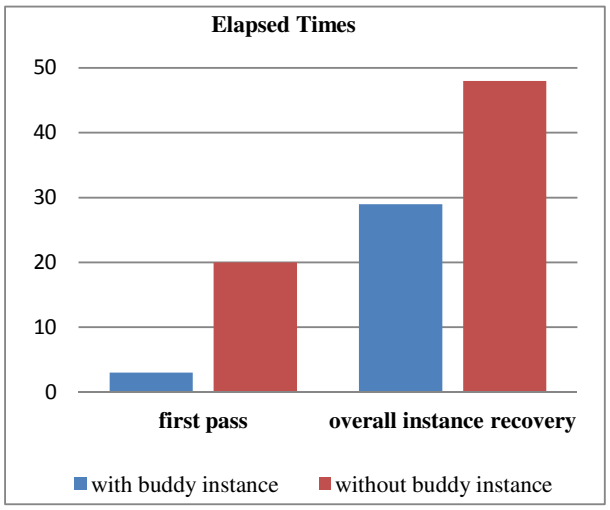

Figure 9. Elapsed times (seconds) for a commercial workload

Without using the buddy instance, instance recovery spent 20 seconds in its first pass when the second pass needed to apply 1.5GB of redo. By comparison, when using the buddy instance mechanism, instance recovery spent only 3 seconds in the first pass when the second pass needed to apply a similar amount of redo.

Oracle database became available on surviving instances in 3 seconds versus 20 seconds. In addition, the overall time for instance recovery was reduced from 48 seconds to 29 seconds. This experiment validates the key claims in this paper by reducing both the brown-out time and overall time for instance recovery.

7. RELATED WORK

The fast-start fault recovery [10] technology in Oracle Database allows control over the duration of the roll-forward phase by adaptively varying the rate of checkpointing. This is applicable more to crash recovery which takes place when every RAC instance fails. In Oracle RAC Database, each instance may have differing amounts of workload and different instances may perform checkpoint activity at different rates. The *database checkpoint* in RAC is the instance checkpoint that has the lowest checkpoint SCN of all the instances. For instance recovery, the instance checkpoint determines the set of redo log changes that need to be applied. Unlike fast-start fault recovery which relies on the database checkpoint, the buddy instance mechanism relies on the instance checkpoint and is therefore a more customized solution for instance recovery in Oracle RAC Database. Since checkpoint issues disk writes for data blocks, aggressive checkpoint activity can be detrimental for database throughput. The buddy instance mechanism only issues disk reads for the redo log which do not mandate acquisition of locks as no synchronization is required. The buddy instance mechanism is therefore less intrusive than using fast-start fault recovery. This paper recommends that the buddy instance mechanism be used in conjunction with fast-start fault recovery technology for best results.

Regarding other database vendors that have a shared disk cluster database solution, published documentation for both SAP Sybase ASE Cluster Edition [11] and IBM DB2 pureScale Clustered Database [12] indicates that neither make use of a scheme similar to the buddy instance mechanism.

8. CONCLUSION

The availability of a cluster system can be improved by making use of the buddy instance mechanism[1] which significantly reduces the amount of time the database spends in the first pass of instance recovery. Since the database can be made available as soon as the first pass of instance recovery completes, the availability of the cluster increases significantly. In addition, the overall time taken for instance recovery is reduced.

9. REFERENCES

[1] Oracle Corporation. Oracle 9i Real Application Clusters concepts Release 2 (9.2), Part Number A96597-01.

[2] Building Highly Available Database Servers Using Oracle Real Application Clusters, An Oracle White Paper, May 2002.

[3] Lahari, T., Srihari, V., Chan, W., Macnaughton, N., Chandrasekaran, S. Cache Fusion: Extending Shared-Disk Clusters with Shared Caches. *Proceedings of the VLDB Conference 2001*.

[4] Joshi, A., Bridge, Loaiza, J., W., Lahiri, T. Checkpointing in Oracle. *Proceedings of the 24th VLDB conference 1998*.

[5] The Transaction Processing Council. TPC-C Benchmark. http://www.tpc.org/tpcc/, 2016.

[6] Oracle Exadata Database Machine System Overview 12*c* Release 1 (12.1), Part Number E51953-14, Feb 2017

[7] Oracle Database Concepts 12c Release (12.1), Part Number E41396-13.

[8] Bridge, W., Joshi, A., Keihl, M., Lahiri, T., Loaiza, J. The Oracle Universal Server Buffer Manager. Proceedings of the 23rd VLDB Conference, pp. 590-594, 1997.

[9] Lahari, T., Ganesh, A., Weiss, R., Joshi, A. Fast-Start: quick fault recovery in oracle, Proceedings of the ACM SIGMOD international conference on Management of data, 2001.

[10] Oracle9*i* Database Performance Tuning Guide and Reference Release 2 (9.2), Part Number A96533-02.

[11] Technical Comparison of Oracle Real Application Clusters 11g vs. IBM DB2 v9 for Linux, Unix, and Windows, An Oracle White Paper, December 2009.

[12] Adaptive Server® Enterprise Cluster Edition 15.5 Users Guide, Document ID: DC00768-01-1550-01

[1] The technique presented in the paper has been granted a U.S. patent.