1.background

Nowadays, many cloud databases adopt a storage-computing separation structure, such asAmazon Aurora,HuaWei TaurusThe computing nodes run the database program, and the storage nodes loadNFSThe network file system exposes the file system interface for computing nodes to use. In this architecture where storage and computing are decoupled, the network becomes a new bottleneck.Buffer PoolThe write amplification caused by frequent write operations in the database has aggravated the severity of the problem. How to optimize network bandwidth and reduce data transmission between nodes is a new challenge facing the database field.

1.1 Buffer PoolWith write amplification

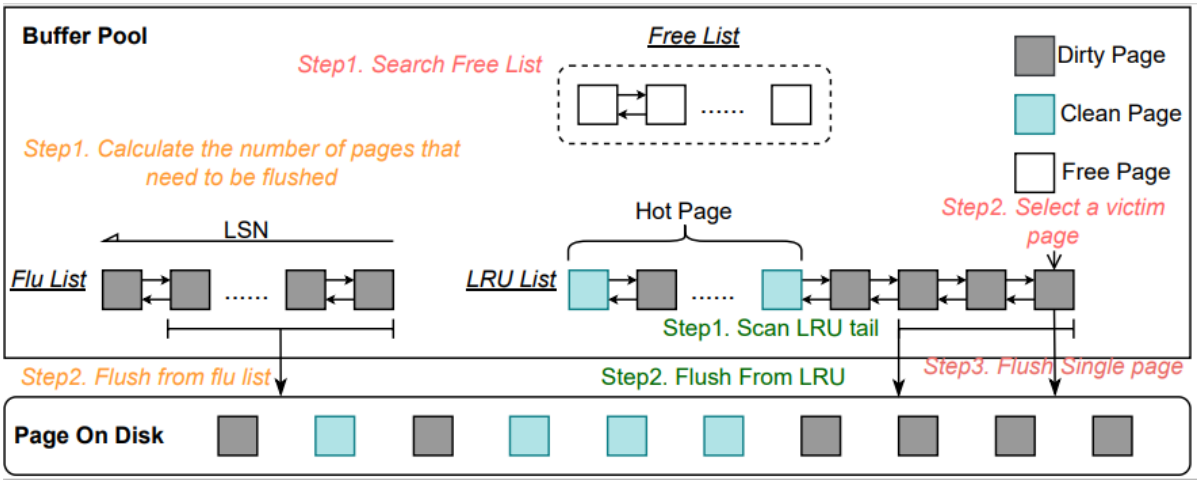
Buffer PoolMostOLTPAn important component of the database used to improve database read performance.InnoDBData is organized into pages and persisted on disk. To alleviate the huge difference in access speed between memory and disk, some data pages are cached in memory and Buffer Pool ManagerThese buffer pages are shared and accessed by foreground threads.Buffer Pool ManagerThe goal is to cache frequently used pages in memory for as long as possible, while rarely accessed pages are quickly eliminated. In order to maximize the spatial locality of data access, improve theBuffer PoolThe hit rate,Buffer Pool ManagerThe data page is buffered in frames (Buffer Frame) in the form of a linked list and use a cache replacement algorithm (such asLRU,Clocketc.) to manage the buffer frame list.InnoDBUsedLRUA variation of the algorithm that keeps the cache hit rate at90%about.

During database operation,Buffer PoolDirty pages are constantly written to disk. In general,Buffer PoolThere are three different types of write operations.

（1) Since the cache size is limited, in some casesBuffer Pool ManagerSome dirty pages must be written back to disk; (2) In order to prevent dirty page write-back operations from affecting foreground user threads, many databases use a pre-refresh strategy;

（3) Database system throughCheckpointMechanism CleanupWAL（Write Ahead Log) space,CheckpointWill cause a large number of dirty pages to be written.

Next, we will introduce these three different types of page write operations, as shown in the figure below.1The process of three types of page write operations is shown.



picture1. Buffer PoolWrite Operation

The data page in memory is called a buffer frame. The buffer frame not only maintains a copy of the disk page, but also maintains the metadata information required by the disk page in memory, such as reference counts. The buffer frame has three states in memory:

（1） FreeThe buffer frame is free and has no stored disk pages.

（2） CleanThe buffer frame stores a copy of the disk page, but the page has not been modified by any transaction.3

） DirtyThe buffer frame stores a copy of the disk page that has been modified by at least one transaction.

1.1.1 BUF_SINGLE_PAGE

Buffer Pool ManagerOrganize the free buffer frames into a linked list (Free List), when the page requested by the current transaction encountersCache Miss WhenFree ListAre there any idle buffer frames in***Step 1***). Otherwise, whenBuffer PoolWhen there are no free buffer frames available, it is necessary to scan from back to front according to the cache replacement algorithmLRU ListFind the statusCleanBuffered frames (***Step 2***). Once a state is encounteredCleanIf the buffer frame is set, the disk page can be read into the buffer frame. Otherwise,LRUThe tail dirty pages are flushed to disk synchronously

（***Step 3***) to get a free frame to use, and then read the missing page into the free frame.

This operation is calledBUF_SINGLE_PAGE,whenBuffer PoolFull and no status isCleanIdle frame, it must comply with Read-After-Write(RAW)This is an expensive operation, and the foreground transaction must pause and wait for the dirty page synchronization write to complete before issuing a read request.Buffer PoolThe utilization rate is very low. When the foreground transaction is busy, each page read will triggerRAWThis will cause serious performance degradation and is unacceptable for the database system.

picture1The red text showsBUF_SINGLE_PAGEprocess.

1.1.2 BUF_LRU_PAGE

To alleviateRAWThe impact caused by this is that modern databases usually usePreflushPre-refresh mechanism, the database system background will haveDirty Page WriterThread, this page refresh method is calledBUF_LRU_PAGE.InnoDBuseRedo Log（WAL) To ensure the persistence of committed transactions, use Undo LogTo ensure the atomicity of uncommitted transactions, the refresh of data pages can be done bySteal,No-Forcestrategy, data pages can be written to disk at any time during transaction execution without affecting the databaseACIDFeatures. The background thread periodicallyLRUTail of linked list (***Step 1***), using asynchronousIOInterface such asLinux AIOWrite a certain number of dirty pages to disk (***Step 2***), thereby reducing the triggering of foreground transactionsRAWThe number of operations.

picture1The green text showsBUF_LRU_PAGEprocess.

1.1.3 BUF_FLUSH_PAGE

Database systems use write-ahead logs (WAL) Stores any changes made by the transaction to the database to ensure the durability of the transaction. When the database crashes, only theWAL, apply it to the database system, and you can restore the state before the crash. Usually the recovery time is proportional to the log size. In order to keep the recovery time short, you need to execute it regularly.CheckpointForcefully flush dirty pages to truncate the log. This page flushing method is calledBUF_FLUSH_PAGE.

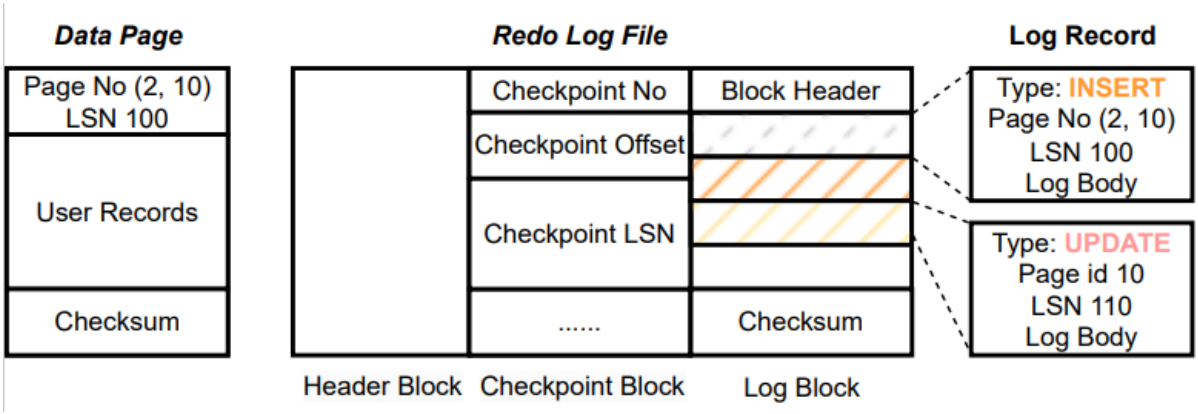
InnoDBThe dirty buffer frame isLSNOrganized from large to smallFLU List, when the background thread executesCheckpointWhen the log is generated, the number of logs to be generated will be calculated based on the current log generation speed and the remaining log capacity.FLU ListNumber of dirty buffer frames flushedN（***Step 1***), in general, the faster the log is generated,NThe larger the value of . Then the dirty buffer frame is flushed to disk (***Step 2***).

picture1The orange text showsBUF_FLUSH_PAGEprocess.

1.2 Redo LogWith crash recovery

becauseBuffer PoolWith the existence of transactions, the modification of disk data by transactions may lag behind the memory. If the database process or machine crashes, the memory data will be lost. Most modern databases useARIESThe algorithm or its variants ensure the durability and atomicity of transactions, and the database strictly adheres toWrite Ahead Logging（WAL) rule, all changes made by a transaction to a data page need to be recorded in the write-ahead log before being reflected on disk, and the log must be written before the correspondingPageWhen a failure occurs and causes memory data to be lost, the database can be restarted by replaying the pre-written log.PageRestore to the state before the crash.InnoDBThe write-ahead log inRedo Log.

As shown below2As shown,InnoDBTakenPhysiological LoggingWays to organizeRedo Log,byPageRecord logs for units, and the logs recordPageLogical changes made. All logs areLog Sequence Number（LSN) to uniquely identify the database.PageThe header information will record the last modification log of the page.LSN. Depending on the modification operations performed by the firm,Log RecordThere are different types ofPageofUser RecordWhen a record is inserted intoINSERTType of log, while updatingPageThe corresponding records inUPDATEType of log.Log BodyThe part stores the data needed for log replay. The content stored in this part varies according to the log type. In order to achieve the expected crash recovery time, the database system background will continuously performCheckpoint,1)fromFlu ListRefresh a certain number of pages, we assume that the largest of these pagesLSNforCheckpoint LSN;2）Checkpoint LSNWill be recordedLogIn the document, this indicates that Checkpoint LSNAll previous logs can be truncated, and their corresponding modifications have been persisted to disk.



picture2. Redo LogStructural diagram

When the database is restarted, it scansRedo LogPerform crash recovery, which is divided into two stages.

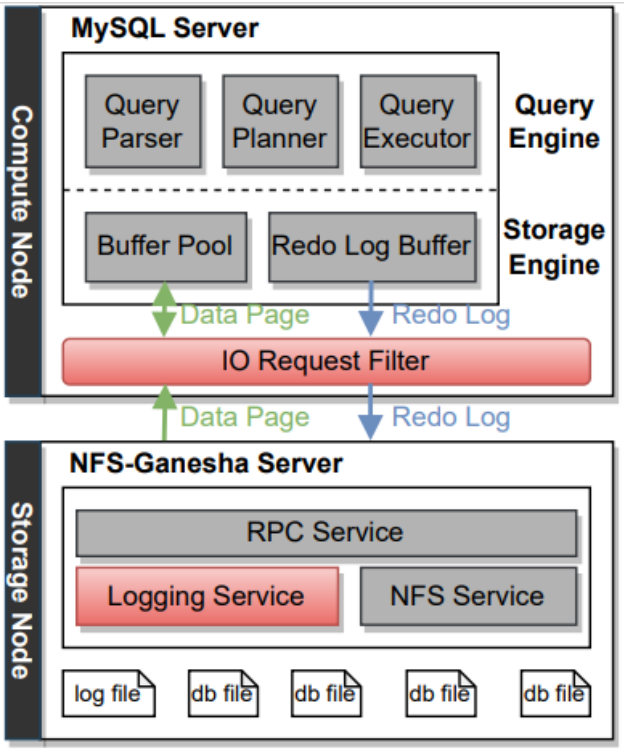
1.2.1 Redostage

This phase is responsible for restoring the database to the state before the crash. This phase does not distinguish between the committed states of transactions. Modifications made by both committed and uncommitted transactions will be restored.Checkpoint LSNThe changes corresponding to the previous logs have been persisted, so this stage will onlyCheckpoint LSNStart ScanRedo Log,ComparePage LSNand currentLog RecordofLSN, only when Page LSNThe log is reapplied to the page only when the size is smaller.

1.2.2 Undostage

InnoDBThe changes made by uncommitted transactions are stored inUndoofPagemiddle,Undo PageAlso affectedRedo Logof protection, so inRedo The stage willUndo PageRestore to the state before the crash.UndoThe stage will useUndo Pageto roll back transactions that were still active at the time of the crash.

2.Solution



picture3. logpushdownOverall structure

MySQLAs a commonly used relational database management system, it plays an important role in cloud database systems. However, while the storage-computing separation architecture improves scalability and availability, it introduces the problem of network bottlenecks. In addition,Buffer PoolandWALThe mechanism leads to serious write amplification, which has a significant impact on the performance of the entire system. To address these challenges, we propose a non-intrusive improvement solution to reduce networkIOAnd reduce write amplification.

TraditionalMySQLThe log system requires that all data modification operations be persisted toRedo Logand flush it to disk. We noticed that we only needed to ensureRedo LogBased on this observation, we propose a non-intrusive solution that bypassesBuffer Poolmiddle Data PageThe down-brush operation can reduce the networkIO, and use the storage nodeCPUResources implement log replay to reduce the impact of write amplification.

Our non-invasive approach offers the following advantages:

- AvoidMySQLSource code modification: Our solution does not require modificationMySQLThis means that no additional risk and complexity are introduced. It can be used with existingMySQLDeployment is compatible and easy to implement and maintain.
- Reduce networkIO: By swiping down onlyRedo Logand bypassData PageBy refreshing the network, we can greatly reduce theIOThis is particularly important in the storage-computing separation architecture, which can significantly reduce the consumption of network transmission bandwidth and improve the performance and response speed of the entire system.
- Take full advantage of storage nodesCPUResources: Our solution relies on the storage nodesCPUThe system uses the resources to replay logs instead of relying on the speed of network transmission. This allows log replay to be completed quickly, avoiding delays in reading and waiting, and further improving system performance and availability.

Through this non-invasive improvement solution, we can solve the problem ofMySQLThe network bottleneck and write amplification problem of the log system. Our approach brings better performance and scalability to the cloud database system, providing users with more efficient and stable data management services.

We are based onMySQL 5.7.30andNFS-GaneshaA prototype system was implemented, calledLogPushDown,picture3Shown LogPushDownthe overall structure.

NFS-GaneshaIt is a widely used program that runs in user mode.NFSservice, which implements the completeNFSProtocol, which can provide a file system interface to the outside world.NFSService, it is more flexible and more conducive to the development of prototype systems.

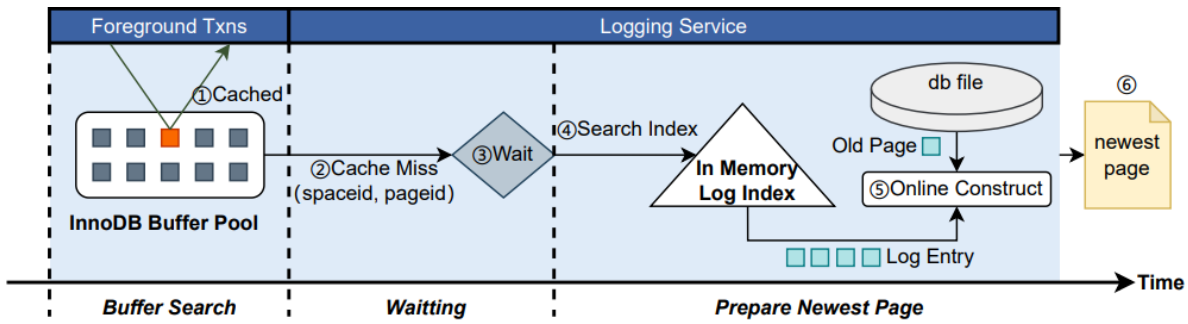
Although the prototype system we implemented is specific toMySQLdatabase, but consideringRedo LogThis is the solution adopted by most disk-oriented database systems, so we believe that this method is universal and can be transplanted to other database systems.

IO Request Filter.To bypassBuffer PoolThe refresh operation of the data page is incorrect.MySQLTo prevent any impact on applications, we implemented a pluggableIORequest filter,MySQLThis component is unaware of theMySQLIssuedData PageWrite requests are filtered out, keeping onlyRedo LogWrite request.

Logging Service.whenBuffer PoolWhen the cache misses, the correspondingData Page,Logging Service WillRedo Log Manage and replay logs regularly to ensureData Page ReadRequest to read the latest version of the page.

2.1Data page read path

Below4It shows the path of the foreground transaction when requesting the data page. It can be divided intoBuffer Search,Waiting, Prepare Newest PageThree stages.



picture4.Data page read path

Buffer Search.ifBuffer PoolTarget in cachePage, it will be returned to the foreground transaction immediately (①), otherwise it will generate Cache Miss,Buffer PoolThe corresponding page will be requested from the storage server (②).

Waiting.Data PageThe read request will be stored in the nodeLogging Servicedeal with.Logging ServicesA memory log index is maintained, and there will beLog ParserThe thread continuously parses the log file and updates the index. The parsing of the log file may lag behind the speed of log file generation.Data PageWhen a read request comes,Log ParserThe latest log files may not be parsed and updated to the index structure in time, and the unparsed logs may containPageRelated logs. We need to record the latest log generated when the request arrivesLSN, then spin waitlog parserAnalyze the latest log (③).

Prepare Newest Page.Logging ServiceThere will be correspondingApply WorkerThe thread periodically performs log replay operations. Apply WorkerThe thread will select some logs from the log index and replay themData PageThe replayed logs will be deleted from the log index.Data PageWhen a read request comes in, the log index is retrieved.PageIf the relevant log (④) cannot be retrieved, it proves that PageHas beenApply WorkerThe thread is updated to the latest version. At this time, no other operations are required, just return the latest page (⑥). Otherwise, we need to replay the log online to build the latest version.Page （⑤）

3.Implementation details

3.1Non-invasiveIOFilters

MySQLAllIORequests are made usingPOSIXStandard file system call interface, such asopen,pread,pwriteEtc. These system calls are usuallylibcIn the realization, LD_PRELOADThe mechanism allows the specified shared library to take precedence over the system default library. Therefore, when a function in a program is called, the dynamic linker will first look for the corresponding function implementation in the preloaded shared library, and if it is found, it will use the preloaded function implementation. Therefore, we willIOThe filter is implemented as a shared librarylibcatcher,libcatcherThe same name is implemented in pread,pwriteWait for system call, replacelibcDefault system call behavior.

Specifically,libcatcherThe following system calls are intercepted:open,open64,pread,pread64,pwrite,pwrite64, close.

Of course, in some cases we still want to uselibcsystem call,dlsymThe function can help accomplish this task. libcatcherThe library is loaded usingdlsymexistlibcFind the address of the system call with the specified name in the library, register it, and use it later.

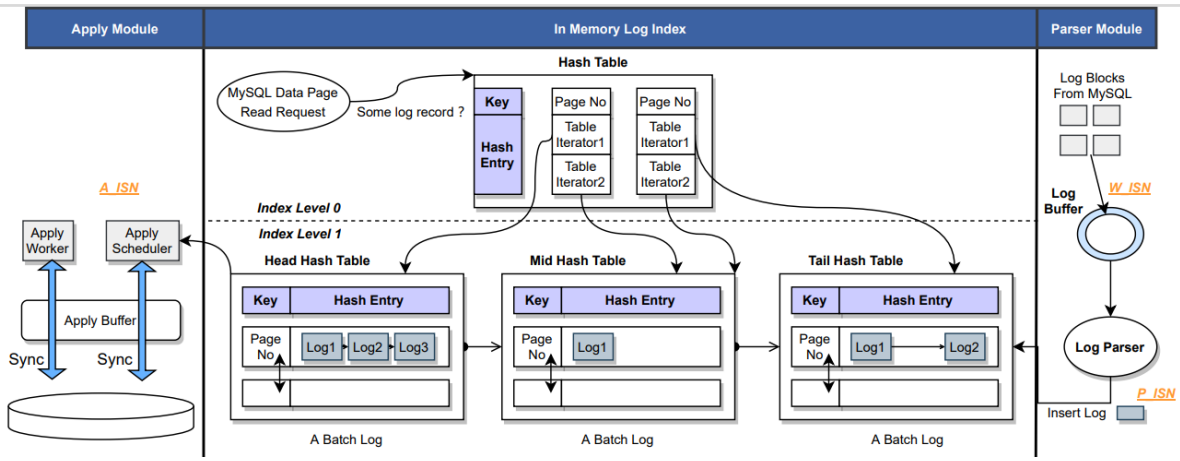
MySQLmiddleRedo LogThe file has a specific file name, and the format of the database file is *.ibd, therefore, inopenIn the system call, different files can be distinguished according to the file name. If it is a database file (ibd), we add its file descriptor to the filter list, so that, inpwriteThe file descriptor filter table can be searched in the system call to directly filter the write operation of the database file.

This method does not requireMySQLTo make any changes, justMySQLAt startup, viaLD_PRELOADSystem variable assignment libcatcherLocation, such as:

```
LD_PRELOAD=./libcatcher.so ./mysqld
```

This allows for seamless integration of filter components.

3.2 Logging Service



picture5. Logging service

3.2.1 Parser Module

whenMySQLWhen writing logs to a storage node, they are not only written to the log file on disk, but also to the log file in memory.Log BufferA copy is cached in .Log BufferIt is a circular buffer that only caches the latest unplayed logs. Its size isBatch SizeThrough experiments, we found that in the scenario of storage and computing separation, due to the influence of network bandwidth, lock contention, etc.MySQLTransaction system handles transaction generationRedo LogThe speed is always slower than the speed of log playback. In our design, since the background thread will batch (Batch Size) PlaybackRedo Log, so in the worst case, there will be a batch of unfilledBatch SizeA log is receiving write requests, and a batch is full Batch SizeThe log is being replayed, so we willRing BufferThe size is set to2timesBatch Sizeis reasonable. In the future, if computing nodes use more powerfulCPU, while using higher bandwidth network technologies, such asRDMA, there may beRedo LogThe generation speed is faster than the log playback speed. As the system runs, the logs that are not played back areRedo LogWill gradually accumulate, on the one handLog BufferDynamic expansion may be required.Redo Logwill accumulate infinitely, requiring a large amount of memory space. On the other hand, the unplayed data accumulated in the storage nodeRedo LogThe more,MySQLReading data pages requiresOnline ConstructTherefore, users need to giveLog BufferSet a reasonable upper limit whenRedo LogThe accumulation reaches the threshold and must be paused MySQLWriteRedo Log.

It should be noted that in order to avoidRedo LogPart of the write (Partial Write)question,MySQLbyBlock（512Byte)Redo Log,andLog EntryThe size of is arbitrary, as shown in the figure2middleRedo LogThe file structure is shown below, exceptLog RecordIn addition, Log BlockSome additional metadata is stored in the header and footer.Ring BufferOnly theLog Record, and does not store this metadata.Log RecordNot enough to fill the entireLog Block, the rest will be0Byte padding.MySQLTwo consecutive timesRedo LogIn a write request, there may be twoLog BlockThe situation of repeated writing, so we need to solve two problems,1)rightLog BlockRemove the metadata at the beginning and end.2) and accurately extractLog RecordThe incremental part of .

We use three pointers to manageLog BufferSpace:W_ISN,P_ISN,A_ISN,ISN（internal sequence number) is an integer that increases globally monotonically according to the number of bytes.

W_ISNWe maintain aW_ISNTo indicate where the next write request should be written.NBytes in sizeRedo Log Record When writing,W_ISNWill grow accordinglyNbytes.

P_ISNFrom the figure5Be able to see.Log ParserWill continue to parse the log, each parsingNBytes of log,P_ISNWill grow accordinglyN bytes.P_ISN = W_ISNhourLog ParserWill pause parsing and waitLog BufferWrite a new log in.

A_ISNFrom the figure5Be able to see.Apply WorkerThe logs will be played back continuously. Once a batch of logs are played back,A_ISNWill move forward Batch SizeTo prevent unplayed logs from being overwritten by newly written logs, only whenW_ISN - A_ISN < Log Buffer SizeLog writing is allowed only when

MySQLThere are twoRedo LogFiles, which form a log file group, are used in a circular manner.Log BufferIt is much smaller than the log file group, so the third problem needs to be solved.3) How toLog BlocksExtracted fromLog RecordMap to a smaller rangeLog BufferThe code snippet shows how toRedo Log blocksWrite toLog Bufferprocess.

3.2.2 In Memory Log Index

As shown5As shown, we use the hash table asRedo LogMaintains a two-level memory index to speed upRedo LogWe will first1The layer index is organized into a hash table array. The hash table isPage Nois the key and the value is aLog Record's linked list.

Each hash table in the first-level index has three states:Open,Close,Applying.

The hash table at the end of the queue isOpenstate,Log ParserSingle-threaded sequential parsingLog BufferAccording to the log inLog RecordIn Page Nofield, will belong to aPageThe log is added to the hash table at the end of the queue.Log ParserThere are two main reasons for designing it as a single thread instead of using multithreading for acceleration: first,MySQLofRedo LogThere is no field in theLog Recordlength, so for aLog RecordFor example, all fields must be parsed sequentially to know the nextLog RecordThe location to which it belongs; secondly, to ensure the correctness of log playback,LSNSmall logs need to be replayed first, so theLog RecordThe linked list must beLSNArranged in order, multiple threads parse outLog RecordNo guaranteeLSNMonotonically increasing order. For these two reasons,Log ParserMust be designed to be single-threaded. OpenThe total number of log bytes stored in the hash table of the state reaches a certain threshold (Batch Size) will be converted toCloseStatus, no longer receivingLog ParserWrite request.Log ParserAn empty hash table will be reinitialized and added to the end of the hash queue to receive new write requests. The hash table at the head of the queue isApplyingstatus, where the logs will beApply ModuleConsume.

exist3.2.1In the previous section, we mentioned that when the log generation rate is greater than the log playback rate, logs will accumulate, which will inevitably cause the hash queue of the first-level index to be too long.MySQLSend a page read request,Logging ServiceNeed to be doneOnline ConstructWhen the first hash table is scanned, all hash tables in the hash queue need to be scanned. In the worst case, there are no related logs in all hash tables, which means this is an invalid scan. To speed up this process, we build an additional hash table as the first0Layer index, the hash table isPage NoThe key is the value, which is a hash table pointer array pointing to the1The layer index stores thePage NoThe hash table of the relevant logs. The data page read request only needs to query the hash table to find the corresponding log.1The position in the layer index greatly improves the query efficiency.

3.2.3 Apply Module

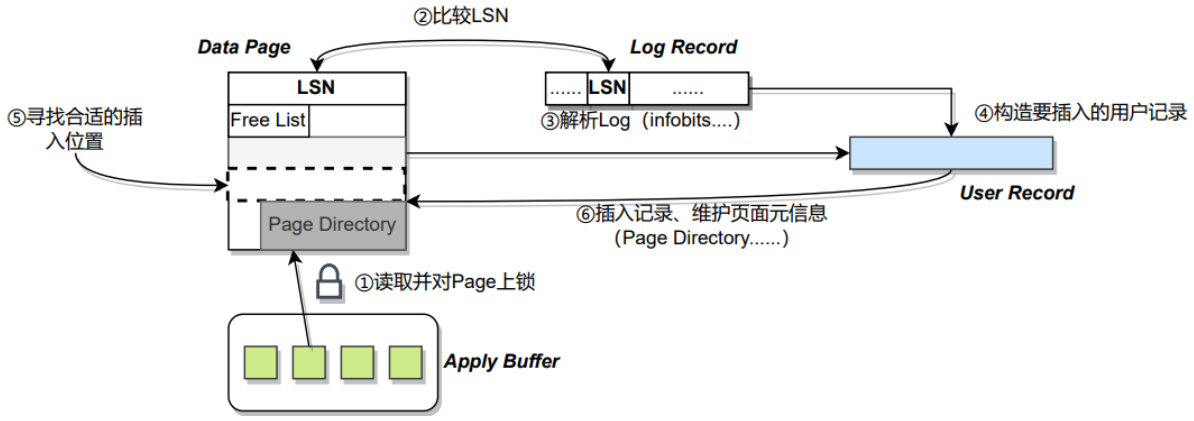
Apply ModuleProvides a "best effort" service that continuously replays logs from the index in the background in an effort to build the latest version of the page. MySQLThere are as many as61There are three types of logs. According to the type of logs acting on the target page, they can be divided into6Types, as shown in the following table1As shown. Among them, the effect isRTree pages and logs acting on compressed pages are optional.RThe tree isMySQLThe data type used to represent spatial data in . This type of log is generated only when spatial index is enabled. Similarly, logs for compressed pages are generated only when page compression is enabled.MySQLUse a separateUndoType of page to storeUndoLogs also need to record the corresponding types of logs to ensureUndo Page persistence.MySQLCorrespondingCompactandRedundantThere are two row formats to format the row records in the user table. Different row formats require different information to be recorded during crash recovery and playback, so different types of logs are required.MySQL 5.0Start by usingCompactWhen creating a file or modifying the file name, the file system meta information (inode), in order to prevent the loss of metadata due to power failure, the corresponding logs must also be recorded.

作用于文件信息	作用于 B+ 树页面		作用于 R 树页面	作用于 Undo 页面	作用于压缩页面	其它
FILE_NAME、	Compact 行格式	Redundant 行格式	PAGE_CREATE_RTREE、	UNDO_INSERT、	ZIP_PAGE_COMPRESS、	TRUNCATE、
FILE_RENAME、	COMP_REC_INSERT、	REC_INSERT、	PAGE_COMP_CREATE_RTREE	UNDO_HDR_REUSE、	ZIP_PAGE_WRITE_HDR、	INDEX_LOAD、
FILE_CREATE	COMP_PAGE_REORGANIZE	LIST_END_DELETE		UNDO_HDR_CREATE	ZIP_PAGE_REORGANIZE	WRITE_STRING

surface1. Redo Logtype

Each type of log has different playback logic, and it is very laborious to fully support the playback of all log types.TPC-C, SysbenchThe workload generatedRedo LogWe found that it will only affect the file information andB+Tree Page (Compactrow format), which acts on UndoPages and some other logs, in which theB+The proportion of logs of tree page type has reached95%, so we only implementB+Tree page type logs and some other necessary types of logs, a total of17ForUndoPages and other unsupported log types, we choose to comply withMySQLThe original logic,IO Filter The swipe down operation of the corresponding page is allowed in the component.

As shown below6WeB+Insert a record into the treeCOMP_REC_INSERTThis article takes the log of the type as an example to explain how to perform log playback.

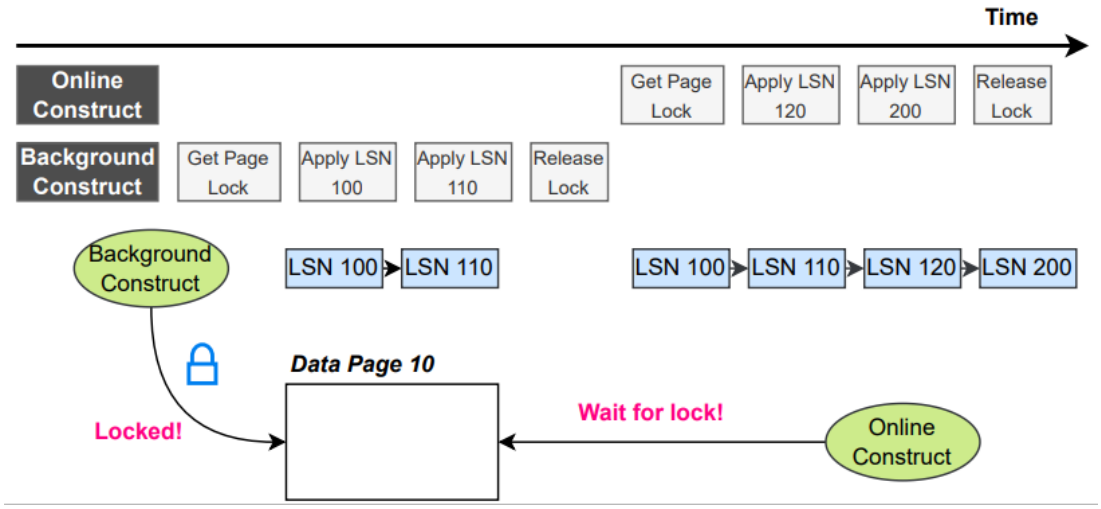


picture6. COMP_REC_INSERTLog playback process

fromApply BufferRead the page and lock it (①). Log playback requires that the page must be in the correct state, so a log playback operation must be atomic. Any thread that wants to play back a page must first obtain the page lock before proceeding.Redo Logof LSN andData PageofLSN(②), ifRedo Log LSNLess thanData Page LSN, which proves that the data page has been replayed to a newer version by other threads. This log replay can be skipped directly.

As shown below7, imagine a scenario where the background playback thread startsLevel 1 In Memory Log IndexofHead Hash TableExtracted from LSNfor100and110Two logs, the foreground process from the entireIn Memory Log IndexExtracted fromLSNfor100,110, 120,200of four logs, all of which they want toPage 10To replay the log, they need to compete for the page lock. Assuming that the background thread first competes for the lock, it replays the log.100,110After that, the lock is released, and the foreground thread can obtain the lock of the page.LSNDiscovery, Log100,110It will skip these two logs and only replay the log120and200.

In order to minimizeLog RecordThe size ofCOMP_REC_INSERTThis type of log will only record the inconsistent parts with the previous user record, which needs to be parsedLog Record(③), get the previous user record inData PagePosition in, mergeLogand the data of the previous user record to construct a complete user record (④), and then you need toPageApply for a space in (⑤) (may be reused)Free Listspace), copy the newly constructed user record to the specified space (⑥), and at the same time, maintain the meta information in the page, the most important of which is to update the pageLSN.



picture7. Online ConstructandBackground ConstructConflict

Only by speeding up the background log playback rate,MySQLofBuffer PoolproduceCache MissBefore reading the data page, replay the logs related to the data page as early as possible to reduceOnline ConstructThe number of log entries that need to be replayed during the process can avoid page read requests pausing for too long, thus affecting transaction throughput and latency.Apply ModuleDesigned as aSchedulerand multipleWorkerThe multi-threaded model.Head Hash TableTransformed intoCloseWhen the state is reached, a round of background log replay will be started immediately.SchedulerDistribute the logs in the hash table evenly to several buckets.Worker, logs in the same hash bucket will only be assigned to oneWorker, to reduce contention between threads.WorkerAfter completing the log playback of a page, the page needs to be written back to disk immediately.SchedulerAfter the tasks are assigned, it will also becomeWorker Perform log playback and wait for allworkerAfter completing the work and becoming idle,SchedulerWill move forwardA_ISN.

4.1 MySQLend

This article is based onMySQL 5.7.30To build, you first need to buildMySQLenvironment.

1.First install the dependencies:

```
sudoapt install pkg-config libssl-dev bison-y
```

2.Configuration GenerationMakefile:

```
# Enter the project root directory
mkdircmake-build-release cmake
-S. \
- Bcmake-build-release \
-G"Unix Makefiles"\
-DCMAKE_BUILD_TYPE=Release \
-DCMAKE_INSTALL_PREFIX=path/to/mysql \
-DMYSQL_DATADIR=path/to/mysql/data \
-DMYSQL_UNIX_ADDR=path/to/mysql/data/mysql.sock \
-DSYSCONFDIR=path/to/mysql/data \
-DWITH_DEBUG=ON \
-DWITH_BOOST=path/to/mysql-code/boost \
-DMYSQL_MAINTAINER_MODE=OFF
```

3.Compile and generate executable files:

```
# - - - j 16Specifies the number of threads to use when compiling the project.
cmake--buildcmake-build-release--targetall-- -j16
```

4.At startupMySQLBefore that, you need to initialize the data directory and setrootAccount permissions and password:

```
# Create the data directory before initializationMySQLUser groups and
MySQLuser sudogroupadd mysql
sudouseradd-r -gmysql-s/bin/false mysql
```

5.initializationmysqldData directory, enter the project root directory, and perform the following operations:

```
cd./cmake-build-release/sql
  .mysqld--basedir=path/to/mysql--datadir=path/to/mysql/data--lower_case_table_names=0--user=
mysql--innodb-flush-method=O_DIRECT-- innodb_flush_log_at_trx_commit=1--innodb_log_file_size=
2G-- innodb_change_buffering=none--default-storage-engine=InnoDB--default-tmpstorage-engine=
InnoDB--disabled_storage_engines=MyISAM--innodb-checksumalgorithm=none--
innodb_log_checksums=OFF--innodb_doublewrite=0--initializeinsecure
```

6.Solve startup `Error` :

`MySQL` Dependency when printing error logs `errmsg.sys` File, compile `MySQL` Afterwards, will be `cmake-build debug/sql/share` Generate various language versions in the folder `errmsg.sys` file, copy the file to `/home/lemon/mysql/share` Folder.

```
mkdirpath/to/mysql/share/
cp/path/to/mysql-code/cmake-build-release/sql/share/english/errmsg.sys mysql/share/          path/to/
```

7.Add read, write and execute permissions to the data directory:

```
chmod-R777path/to/mysql/data
```

8.start upServerendmysqld:

```
 .mysqld--basedir=path/to/mysql--datadir=path/to/mysql/data--
socket=path/to/mysql/data/mysql.sock--lower_case_table_names=0--user=mysql--innodb-flush-method
=O_DIRECT--innodb_flush_log_at_trx_commit=1-- innodb_log_file_size=1G--innodb_change_buffering=
none--default-storageengine=InnoDB--default-tmp-storage-engine=InnoDB--disabled_storage_engines=
MyISAM--innodb-checksum-algorithm=none-- innodb_log_checksums=OFF--innodb_doublewrite=0
```

9.start upClientendmysqlReviserootUser password, enter the project root and record it, and perform the following operations:

```
/path/to/mysql-code/cmake-build-release/client/mysql-uroot -h127.0.0.1-P3306 -p
```

10.No need to enter a password, just press Enter to startMySQLClient loginMySQLAfter that, modify the code on the `root` User password client side and close it.ServerDuanheClientend:

```
ALTER USER'root'@'localhost'IDENTIFIED WITH mysql_native_password BY"root"; FLUSH privileges;

shutdown;
exit;
```

4.2Preparing test data

bysysbenchThis article takes the benchmark test tool as an example to introduce how to generate test data.

RestartMySQL, use the client tool to connect and create a database for testing:

```
CREATE DATABASE sbtest;
```

Then usesysbenchInitialize test data:

```
sysbench - - threads=20 \
- -mysql-host=127.0.0.1 \
- -mysql-port=3306 \
- -mysql-user=root \
- - mysql-password=root /usr/share/sysbench/
oltp_common.lua \
- - tables=40 \
- - table_size=200000 \
prepare
```

4.3 NFS-Serverend

1.First install the dependencies

```
sudoapt install g++libboost-dev cmakemake gitdoxygen sudoapt install build-essential
libglu1-mesa-dev libc6-dev sudoapt install libkrb5-dev libsasl2-modules-gssapi-mit
libkrb5-dev sudoapt install liburcu-dev

sudoapt install libcap-dev libtirpc-dev rpcbind sudoapt-get install
uuid-dev libacl1-dev liblzo2-dev
```

Note: Startup `nfs-ganesha` Before you install `nfs-kernel-server` Otherwise, when you start `nfs-ganesha` When you get When I get the following error, I guess some dependency might be missing? I found that it should be missing `rpcbind` .

```
Cannot register NFS V3 on TCP
```

2.Install `nfs-kernel-server`

```
sudoapt install nfs-kernel-server
sudo/etc/init.d/nfs-kernel-serverstop#closurenfs-kernel-serverBecause we don't need to use it
```

3.ReviseNFS-ServerConfiguration Files:

turn upnfs-ganesha-sql/src/include/applier/config.hFile, modifyLOG_PATH_PREFIX, DATA_FILE_PREFIX,DATA_FILES, PER_FILE_LOG_FILE_SIZEFour variables, among whichDATA_FILESVariables are what you createsysbenchAll table data files, LOG_PATH_PREFIXyesRedo LogThe file path where it is located,DATA_FILE_PREFIX yesMySQLThe main directory of the database table data files,PER_FILE_LOG_FILE_SIZEEveryRedo LogThe size of the log file.

4.Compile the source code:

```
# Enter the project root directory
cd nfs-ganesha-sql
mkdir src/build
cmake -S src -B src/build -DCMAKE_BUILD_TYPE=Release -DUSE_FSAL_VFS=ON - DUSE_9P=OFF
-DUSE_FSAL_LUSTRE=OFF -DUSE_FSAL_GPFS=OFF -DUSE_NLM=OFF -
DUSE_FSAL_CEPH:STRING=OFF - DUSE_FSAL_GLUSTER:STRING=OFF -
DUSE_FSAL_KVSFS:STRING=OFF DUSE_FSAL_NULL:STRING=OFF -
DUSE_FSAL_PROXY_V3:STRING=OFF DUSE_FSAL_MEM:STRING=OFF -
DUSE_FSAL_RGW:STRING=OFF -DUSE_FSAL_XFS:STRING=OFF -DUSE_FSAL_PROXY_V4:STRING=OFF -
cmake --build src/build --parallel 20 -DUSE_GSS=OFF
```

5.start upnfs-server:

Create a configuration file first

```
vim~/ganesha.conf
```

The configuration file content is as follows, you need to modifyPlugins_Dirparameter, so that it points tonfs-serverCompile the generated library directory,PathThe parameter represents the file directory you need to expose to the client.

```
NFS_CORE_PARAM {
    Plugins_Dir =       path/to/nfs-ganesha-sql/src/build/lib;
}
EXPORT
{
      Export_Id =     77;
      Path =    path/to/mysql-Dir;
    Pseudo =    /;

      FSAL {
            Name =  VFS;
      }
      Access_Type    =  RW;
      Disable_ACL    =  true;
      Squash =    No_Root_Squash;
      Protocols     = 3,4;
}
EXPORT_DEFAULTS
      Transports = UDP, TCP;
      SecType = sys;
}
```

no_root_squash：Indicates that when the clientrootLocal access when identity is grantedrootPermissions (default isroot_squash).

root_squash：Indicates that the client usesrootWhen a user accesses the shared directory,rootThe user is mapped to an anonymous user. Create a

directory to store `ganesha` ofpidinformation:

```
sudo mkdir/var/run/ganesha
```

The final privilege escalation startnfsserver:

```
sudopath/to/nfs-ganesha-sql/build/bin/ganesha.nfsd-F -L/dev/stdout-fpath/to/ganesha.conf-NNIV_EVENT
```

- F (foreground),Willganesha.nfsdThe service runs in the foreground, -LSpecify the location where the log will be output.fSpecify the path where the configuration file is located.NSpecify the output log level.

4.4Start and test

1.ClientMySQLOne side needs to be mountedNFSservice, the servermysqlDirectory exposed tomysqlOne side:

```
sudo mount -t nfs4 11.11.11.12:/ path/to/mysql-Dir(Local mount point)
```

2.Reviseio_filterComponent, compile, openlibcacherProject, againcatcher_filter.cppModifications in the filedata_file_setVariable, pointing to allsysbenchGenerate the database table file, and then compile it, the compiled .soDynamic library is called libcatcher_filter.so.

3.start upMySQL,inLD_PRELOADSpecifylibcatcher_filter.soThe location of the dynamic library, --basediranddatadirIt should be the remote directory you mounted.

```
LD_PRELOAD=path/to/libcatcher/libcatcher_filter.so ./mysqld --basedir=path/to/ mysql --datadir=path/
to/mysql/data --
socket=/tmp/mysql.sock --lower_case_table_names=0 --user=mysql --innodb-flushmethod=O_DIRECT --
innodb_flush_log_at_trx_commit=1                                    - - innodb_change_buffering=none
- - default-storage-engine=InnoDB              - - default-tmp-storage-engine=InnoDB              - -
disabled_storage_engines=MyISAM            - - innodb-checksum-algorithm=none --
innodb_log_file_size=2G --innodb_buffer_pool_size=128M --innodb_doublewrite=0 --
innodb_log_checksums=OFF
```

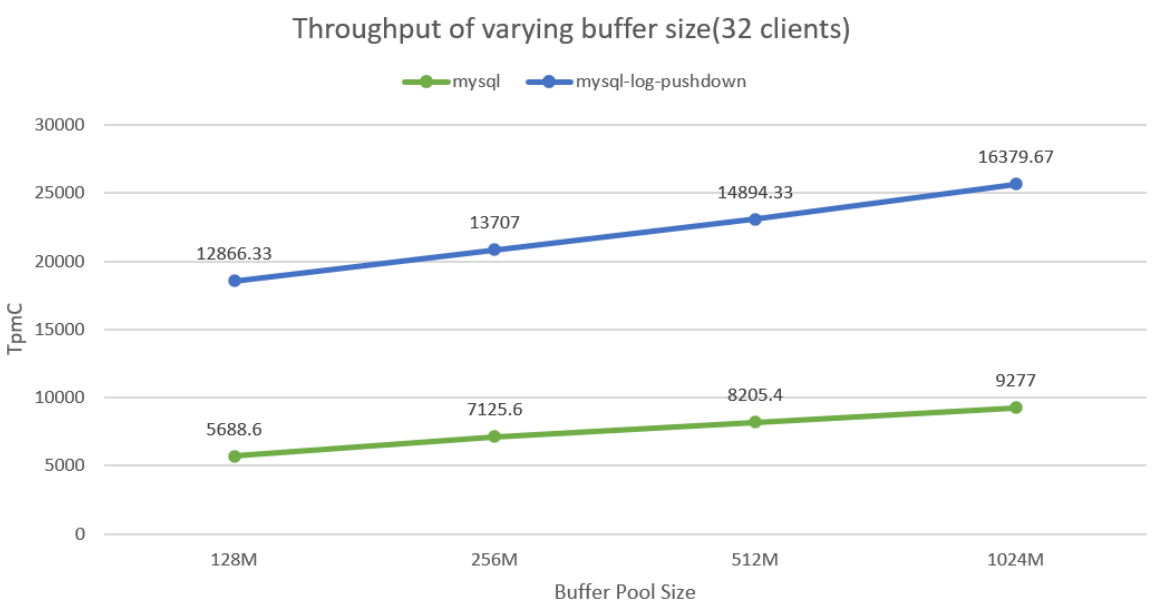
4.start upsysbenchTo test:

```
sysbench    - - threads=8 \
            - - time=180 \
            - - report-interval=10 \
            - -mysql-host=127.0.0.1          \
            - -mysql-port=3306       \
            - -mysql-user=root       \
            - - mysql-password=root /usr/share/sysbench/
            oltp_read_write.lua                                  \
            - - tables=40 \
            - - table_size=2000000 \
            - - mysql-ignore-errors=2013 \
            - - mysql-socket=/tmp/mysql.sock \ run
```

5.Experimental results display

We built a storage and computing separation system to conduct experiments and testLogPushDownThe storage node has the same configuration as the computing node, with an additionalPCIe3.0Samsung980Solid-state drives are used to store database files.RDMANetwork card (still using the normalTCP/IPProtocol) for network communication, and computing nodes runMySQL-ServerandTPC-CWorkload, storage node running NFS-Ganesha.

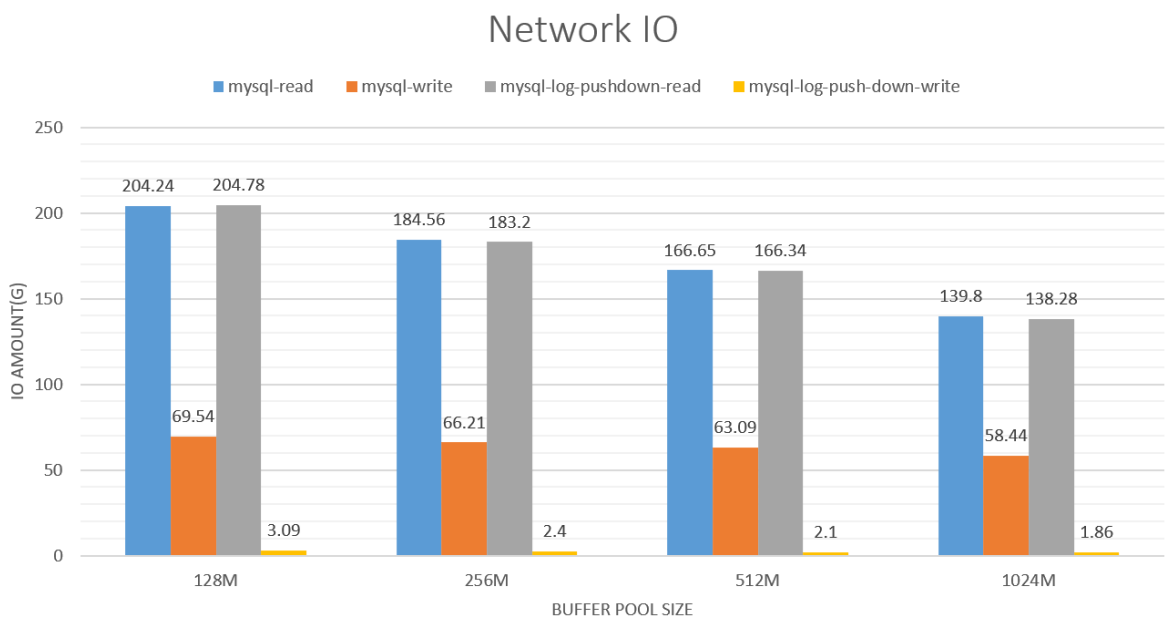
5.1Throughput test



picture8. TPC-cThroughput test

As shown above8As shown, in the fixedMySQLThe number of connections is32In the case of multiple threads, we willMySQLofBuffer Poolfrom128MGradually increase to1024M, the figure above showsTPC-CThroughput comparison, green linemysqlThe blue line represents the throughput in the original scenario. mysql-log-pushdownIt represents the throughput of the solution proposed in this paper. It can be seen that compared with the original scenario, the throughput is increased by nearly two times.

5.2networkIOtest



picture9.networkIOtest

As shown above9As shown, we modifiedTPC-CThe source code is inMySQLThe number of connections is32In the case of multiple threads, make each thread run 3000transactions, after the test we measured the network between the computing nodes and the storage nodesIOAs shown in the figure above, compared with the original scene, the networkIOBasically remain unchanged, but because our solution basically filters out all data page write operations, onlyRedo Log,Undo Logand some other pages are written, so writeIOVery small, compared to the original scene,Buffer Pool Sizefor512MIn the case ofIOThe most decreased79.2 times. When running the same number of transactions, the larger theBuffer PoolAbility to cache more pages, triggeringIOThe number of times will also be less, so asBuffer PoolAs the size increases, the corresponding reading and writingIOThe total amount shows a downward trend.