Memcachedb: The Complete Guide

Steve Chu
stvchu@gmail.com

ICRD-Web@Sina

March 12, 2008

Part I

Getting Started

What is Memcachedb?

What is Memcachedb?

"Memcachedb is a distributed key-value storage system designed for persistent."

A complete memcached, but

- *NOT* a cache solution
  Memcached is good enough for cache.
- *NO* expiration
  For memcache protocol compatible, still reserved, but we do nothing.
- Totally for persistent
  Transaction, replication, we do our best to achieve persistent.

Why Memcachedb?

Why Memcachedb?(1/2)

We have MySQL, we have PostgreSQL, we have a lot of RDBMSs, but why we need Memcachedb?

- RDBMS is slow
  All they have a complicated SQL engine on top of storage. Our data requires to be stored and retrieved damnable fast.
- Not concurrent well
  When thousands of clients, millions of requests happens...
- But the data we wanna store is very small size!
  Cost is high if we use RDBMS.

Why Memcachedb?(2/2)

Many critical infrastructure services need fast, reliable data storage and retrieval, but do not need the flexibility of dynamic SQL queries.

- Index, Counter, Flags
- Identity Management(Account, Profile, User config info, Score)
- Messaging
- Personal domain name
- meta data of distributed system
- Other non-relatonal data
- ...

Memcachedb Features

Memcachedb Features

- High performance read/write for a key-value based object
  Rapid set/get for a key-value based object, not relational. Benchmark will tell you the true later.
- High reliable persistent storage with transaction
  Transaction is used to make your data more reliable.
- High availability data storage with replication
  Replication rocks! Achieve your HA, spread your read, make your transaction durable!
- Memcache protocol compatibility
  Lots of Memcached Client APIs can be used for Memcachedb, almost in any language, Perl, C, Python, Java, ...

Supported Commands

Standard Memcache Commands

'get'	Retrieval of one or multiple items
'set'	"Store this data"
'add'	"Store this data, but only if the server *doesn't* already hold data for this key"
'replace'	"Store this data, but only if the server *does* already hold data for this key"
'delete'	deletes one item based a key
'incr/decr'	Increment or decrement a numeric value. It's atomic!
'stats'	shows the status of current deamon. 'stats', 'stats malloc', 'stats maps'

Private Commands

'db_checkpoint'	does a checkpoint manuanlly.
'db_archive'	removes log files that are no longer needed.
'stats bdb'	shows the status of BerkeleyDB.
'rep_ismaster'	shows whether the site is a master.
'rep_whoismaster'	shows which site is a master.
'rep_set_priority'	sets the priority of a site for electing in replication.
'rep_set_ack_policy'	sets ACK policy of the replication.
'rep_set_ack_timeout'	sets ACK timeout value of the replication .
'rep_set_bulk'	Enable bulk transfer or not in replication.
'rep_set_request'	sets the minimum and maximum number of missing log records that a client waits before requesting retransmission.
'stats rep'	shows the status of Replication.

Benchmark

Environment

- Box: Dell 2950III
- OS: Linux CentOS 5
- Version: memcachedb-1.0.0-beta
- Client API: libmemcached

Non-thread Version

key: 16 value: 100B, 8 concurrents, every process does 2,000,000 set/get

```
memcachedb -d -r -u root -H /data1/mdbtest/ -N -v
```

- Write

No.	1	2	3	4	5	6	7	8	avg.
Cost(s)	807	835	840	853	859	857	865	868	848

*2000000 * 8 / 848 = 18868 w/s*

- Read

No.	1	2	3	4	5	6	7	8	avg.
Cost(s)	354	354	359	358	357	364	363	365	360

*2000000 * 8 / 360 = 44444 r/s*

Thread Version

key: 16 value: 100B, 8 concurrents, every process does 2,000,000 set/get

```
memcachedb -d -r -u root -H /data1/mdbtest/ -N -t 4 -v
```

- Write

No.	1	2	3	4	5	6	7	8	avg.
Cost(s)	663	669	680	680	684	683	687	686	679

*2000000 * 8 / 679 = 23564 w/s*

- Read

No.	1	2	3	4	5	6	7	8	avg.
Cost(s)	245	249	250	248	248	249	251	250	249

*2000000 * 8 / 249 = 64257 r/s*

Part II

MDB In Action

Installation

Dependencies

libevent	An event notification library that provides a mechanism to execute a callback function when a specific event occurs on a file descriptor or after a timeout has been reached. Now it supports /dev/poll, kqueue(2), event ports, select(2), poll(2) and epoll(4). `http://www.monkey.org/~provos/libevent/`
BerkeleyDB	The industry-leading open source, embeddable database engine that provides developers with fast, reliable, local persistence with zero administration. `http://www.oracle.com/technology/products/berkeley-db/db/index.html`

Installing libevent

```
~ % tar zvxf libevent-1.3e.tar.gz
```

Installing libevent

```
~ % tar zvxf libevent-1.3e.tar.gz
~ % cd libevent-1.3e
```

Installing libevent

```
~ % tar zvxf libevent-1.3e.tar.gz
~ % cd libevent-1.3e
~/libevent-1.3e % ./configure
...
```

Installing libevent

```
~ % tar zvxf libevent-1.3e.tar.gz
~ % cd libevent-1.3e
~/libevent-1.3e % ./configure
...
~/libevent-1.3e % make
...
```

Installing libevent

```
~ % tar zvxf libevent-1.3e.tar.gz
~ % cd libevent-1.3e
~/libevent-1.3e % ./configure
...
~/libevent-1.3e % make
...
~/libevent-1.3e % su
Password:
```

Installing libevent

```
~ % tar zvxf libevent-1.3e.tar.gz
~ % cd libevent-1.3e
~/libevent-1.3e % ./configure
...
~/libevent-1.3e % make
...
~/libevent-1.3e % su
Password:
/home/sc/libevent-1.3e # make install
...
```

Installing libevent

```
~ % tar zvxf libevent-1.3e.tar.gz
~ % cd libevent-1.3e
~/libevent-1.3e % ./configure
...
~/libevent-1.3e % make
...
~/libevent-1.3e % su
Password:
/home/sc/libevent-1.3e # make install
...
/home/sc/libevent-1.3e # exit
```

Installing BerkeleyDB

```
~ % tar zvxf db-4.6.21.tar.gz
```

Installing BerkeleyDB

```
~ % tar zvxf db-4.6.21.tar.gz
~ % cd db-4.6.21
```

Installing BerkeleyDB

```
~ % tar zvxf db-4.6.21.tar.gz
~ % cd db-4.6.21
~ % cd build_unix
```

Installing BerkeleyDB

```
~ % tar zvxf db-4.6.21.tar.gz
~ % cd db-4.6.21
~ % cd build_unix
~/db-4.6.21/build_unix % ../dist/configure
...
```

Installing BerkeleyDB

```
~ % tar zvxf db-4.6.21.tar.gz
~ % cd db-4.6.21
~ % cd build_unix
~/db-4.6.21/build_unix % ../dist/configure
...
~/db-4.6.21/build_unix % make
...
```

Installing BerkeleyDB

```
~ % tar zvxf db-4.6.21.tar.gz
~ % cd db-4.6.21
~ % cd build_unix
~/db-4.6.21/build_unix % ../dist/configure
...
~/db-4.6.21/build_unix % make
...
~/db-4.6.21/build_unix % su
Password:
```

Installing BerkeleyDB

```
~ % tar zvxf db-4.6.21.tar.gz
~ % cd db-4.6.21
~ % cd build_unix
~/db-4.6.21/build_unix % ../dist/configure
...
~/db-4.6.21/build_unix % make
...
~/db-4.6.21/build_unix % su
Password:
/home/sc/db-4.6.21/build_unix # make install
...
```

Installing BerkeleyDB

```
~ % tar zvxf db-4.6.21.tar.gz
~ % cd db-4.6.21
~ % cd build_unix
~/db-4.6.21/build_unix % ../dist/configure
...
~/db-4.6.21/build_unix % make
...
~/db-4.6.21/build_unix % su
Password:
/home/sc/db-4.6.21/build_unix # make install
...
/home/sc/db-4.6.21/build_unix # exit
```

Installing Memcachedb

```
~ % tar zvxf memcachedb-1.0.3-beta.tar.gz
```

Installing Memcachedb

```
~ % tar zvxf memcachedb-1.0.3-beta.tar.gz
~ % cd memcachedb-1.0.3-beta
```

Installing Memcachedb

```
~ % tar zvxf memcachedb-1.0.3-beta.tar.gz
~ % cd memcachedb-1.0.3-beta
~/memcachedb-1.0.3-beta % ./configure #--enable-threads if
you wanna thread version
...
```

Installing Memcachedb

```
~ % tar zvxf memcachedb-1.0.3-beta.tar.gz
~ % cd memcachedb-1.0.3-beta
~/memcachedb-1.0.3-beta % ./configure #--enable-threads if
you wanna thread version
...
~/memcachedb-1.0.3-beta % make
...
```

Installing Memcachedb

```
~ % tar zvxf memcachedb-1.0.3-beta.tar.gz
~ % cd memcachedb-1.0.3-beta
~/memcachedb-1.0.3-beta % ./configure #--enable-threads if
you wanna thread version
...
~/memcachedb-1.0.3-beta % make
...
~/memcachedb-1.0.3-beta % su
Password:
```

Installing Memcachedb

```
~ % tar zvxf memcachedb-1.0.3-beta.tar.gz
~ % cd memcachedb-1.0.3-beta
~/memcachedb-1.0.3-beta % ./configure #--enable-threads if
you wanna thread version
...
~/memcachedb-1.0.3-beta % make
...
~/memcachedb-1.0.3-beta % su
Password:
/home/sc/memcachedb-1.0.3-beta # make install
...
```

Installing Memcachedb

```
~ % tar zvxf memcachedb-1.0.3-beta.tar.gz
~ % cd memcachedb-1.0.3-beta
~/memcachedb-1.0.3-beta % ./configure #--enable-threads if
you wanna thread version
...
~/memcachedb-1.0.3-beta % make
...
~/memcachedb-1.0.3-beta % su
Password:
/home/sc/memcachedb-1.0.3-beta # make install
...
/home/sc/memcachedb-1.0.3-beta # exit
```

Running Options Explained

Deamon Options

'-p <num>'	TCP port number to listen on (default: 21201)
'-l <ip_addr>'	interface to listen on, default is INDRR_ANY
'-d'	run as a daemon
'-r'	maximize core file limit
'-u <username>'	assume identity of <username> (only when run as root)
'-c <num>'	max simultaneous connections, default is 1024
'-b <num>'	max item buffer size in bytes, default is 1KB
'-v'	verbose (print errors/warnings while in event loop)
'-vv'	very verbose (also print client commands/reponses)
'-P <file>'	save PID in <file>, only used with -d option

BekerleyDB Options

'-m <num>'	in-memmory cache size of BerkeleyDB in megabytes, default is 64MB
'-f <file>'	filename of database, default is /data1/memcachedb/default.db
'-H <dir>'	env home of database, default is /data1/memcachedb
'-L <num>'	log buffer size in kbytes, default is 32KB
'-C <num>'	do checkpoint every XX seconds, 0 for disable, default is 60s
'-D <num>'	do deadlock detecting every XXX millisecond, 0 for disable, default is 100ms
'-N'	enable DB_TXN_NOSYNC to gain big performance improved, default is off

Managing Daemon

Before start..

Please take care this two options, a lot of mistakes have been made due to this:

`-b <num>` max item buffer size in bytes, default is 1KB. '-b option' determines MAX size of item can be stored. Just choose a suitable size. Following this formula:
item_buffer_size(-b) = *key_size* + *data_size* + *37(Max)*

`-N` enable DB_TXN_NOSYNC to gain big performance improved, default is off. By using '-N' option, 'ACID' in transaction will lose 'D'. The data in transaction log buffer may be gone when the machine loses power(So we need replication).

How to start a deamon?

Non-replication:

```
memcachedb -p21201 -d -r -u root -f 21201.db -H /data1/demo
-N -P /data1/logs/21201.pid
```

How to stop a deamon?

Just kill it:

```
kill `cat /data1/logs/21201.pid`
```

When the deamon recives a signal of SIGTERM/SIGQUIT/SIGINT, it will do a checkpoint instantly and close the db and env resource normally. So don't be afraid, just kill it!

Commands Using telnet

set/get/delete a Item

```
~ % telnet 127.0.0.1 21201
```

set/get/delete a Item

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

set/get/delete a Item

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
set test 0 0 4
```

set/get/delete a Item

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
set test 0 0 4
1234
```

set/get/delete a Item

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
set test 0 0 4
1234
STORED
```

set/get/delete a Item

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
set test 0 0 4
1234
STORED
get test
```

set/get/delete a Item

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
set test 0 0 4
1234
STORED
get test
VALUE test 0 4
1234
END
```

set/get/delete a Item

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
set test 0 0 4
1234
STORED
get test
VALUE test 0 4
1234
END
delete test
```

set/get/delete a Item

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
set test 0 0 4
1234
STORED
get test
VALUE test 0 4
1234
END
delete test
DELETED
```

stats

```
~ % telnet 127.0.0.1 21201
```

stats

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

stats

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
stats
```

stats

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
stats
STAT pid 18547
STAT uptime 41385
STAT rusage_user 0.084005
STAT rusage_system 0.804050
STAT curr_connections 1
...
STAT bytes_read 5347
STAT bytes_written 122797
STAT threads 1
END
```

stats bdb

```
~ % telnet 127.0.0.1 21201
```

stats bdb

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

stats bdb

```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
stats bdb
```

stats bdb

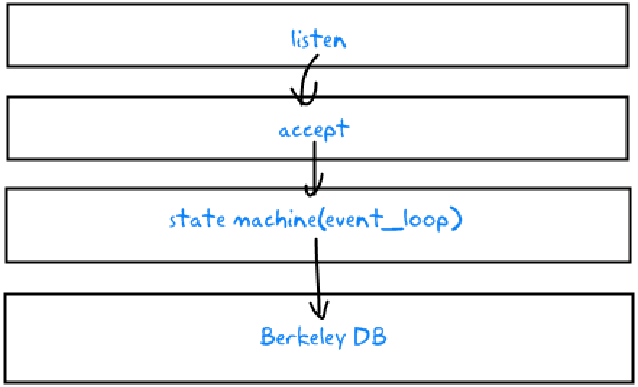
```
~ % telnet 127.0.0.1 21201
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
stats bdb
STAT cache_size 67108864
STAT txn_lg_bsize 32768
STAT txn_nosync 1
STAT dldetect_val 100000
STAT chkpoint_val 60
END
```

Part III

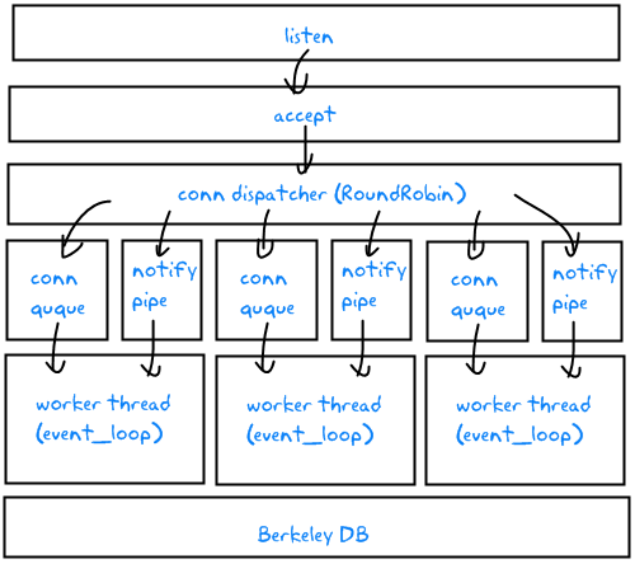
Internals

The Big Picture

Nonthread Version

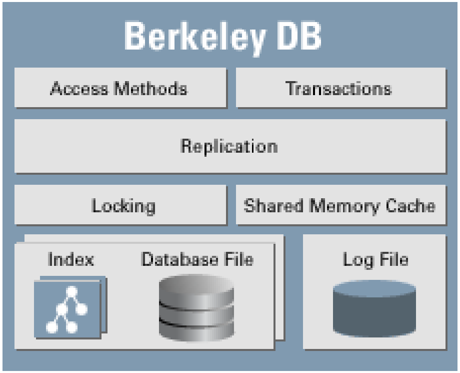


Thread Version


listen
accept
conn dispatcher (RoundRobin)
conn quque
notify pipe
conn quque
notify pipe
conn quque
notify pipe
worker thread (event_loop)
worker thread (event_loop)
worker thread (event_loop)
Berkeley DB

The Backend: BerkeleyDB

http://www.oracle.com/technology/products/berkeley-db/db/index.html



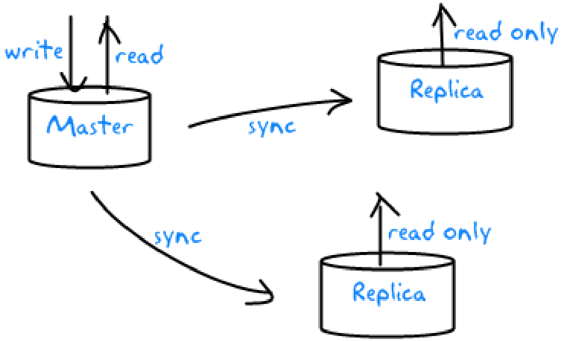
Part IV

Replication

Overview

Replication Model

Consistency is an important issue that every engineer must resolve when designing a distributed system. The BerkeleyDB replication framework resolves this by following a `single master, multiple replica` model.



Replication Benefits

- Improve application reliability
  By spreading your data across multiple machines, you can ensure that your application's data continues to be available even in the event of a hardware failure on any given machine in the replication group.
- Improve read performance
  By using replication you can spread data reads across multiple machines on your network.
- Improve transactional commit performance and data durability guarantee
  Replication allows you to avoid this disk I/O and still maintain a degree of durability by `committing to the network`. So we can use '-N' option for better performance but never lose durability(The D of ACID).

Replication Patterns

ACK Policy(1/2)

Messaging is the key facility that implements replication. How to process a message influences your data reliability and performance. Now we go deep into these policies:

'DB_REPMGR_ACKS_ALL' The master should wait until all replication clients have acknowledged each permanent replication message.

'DB_REPMGR_ACKS_ALL_PEERS' The master should wait until all electable peers have acknowledged each permanent replication message (where "electable peer" means a client capable of being subsequently elected master of the replication group).

'DB_REPMGR_ACKS_NONE' The master should not wait for any client replication message acknowledgments.

'DB_REPMGR_ACKS_ONE' The master should wait until at least one client site has acknowledged each permanent replication message.

ACK Policy(2/2)

'DB_REPMGR_ACKS_ONE_PEER' The master should wait until at least one electable peer has acknowledged each permanent replication message (where "electable peer" means a client capable of being subsequently elected master of the replication group).

'DB_REPMGR_ACKS_QUORUM' The master should wait until it has received acknowledgements from the minimum number of electable peers sufficient to ensure that the effect of the permanent record remains durable if an election is held (where "electable peer" means a client capable of being subsequently elected master of the replication group). This is the default acknowledgement policy.

Note: The current implementation requires all sites in a replication group configure the same acknowledgement policy.

Performance vs Data Reliability

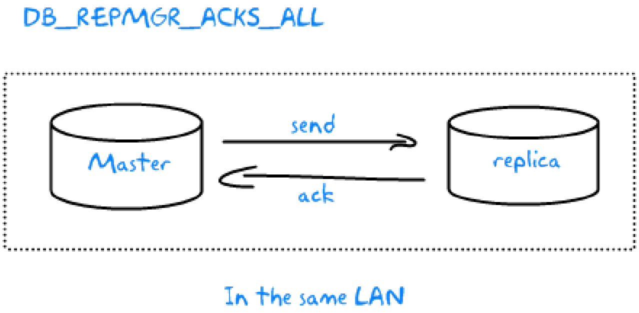
'ACK_ALL'	More data reliability, but poor performance due to the blocked thread waiting for ack(the thread can not continue to write).
'ACK_NONE'	Better performance, but may cause reliable problem because of the unstable network between a master and replica(the data of repllica may be out-of-date).

So we must do a tradeoff:
Let Replica who in the same LAN with Master do the reliable thing, and let the site far from the Master recieves replication message with ACK_NONE.

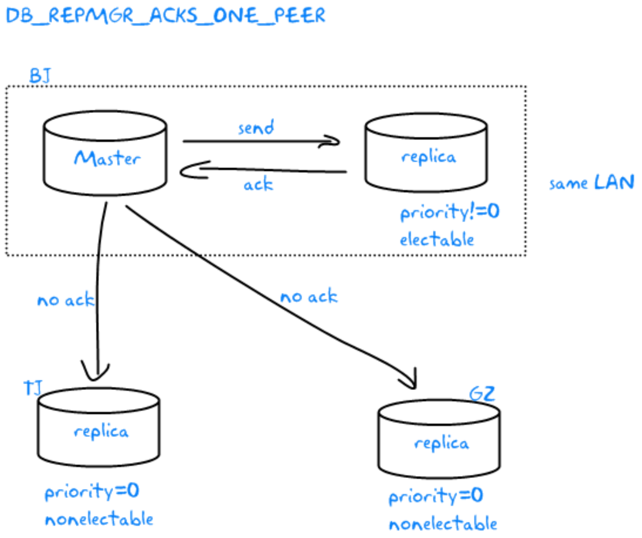
How ACK_NONE Replicas catch up with Master

- Restart your replica daemon, and force a replica sync with master. Not that flexible..
- Set a minor number of missing log records that a client waits before requesting retransmission.
  A replication client checks the log sequence number of each incoming log record, and can detect gaps in the sequence. If some log records are lost due to network problems, then when later log records arrive the client detects the missing records. The client waits for some number of out-of-sequence log records before issuing the request for retransmission.

Replication over LAN



Replication over WAN



Replication Howto

Design your deployment

Your deployment based the replication pattern you choose, and try to think about these:

- Network QoS
- How large your dataset
- Proportion of read/write
- Throughput of read/write
- The replication group size
- ...

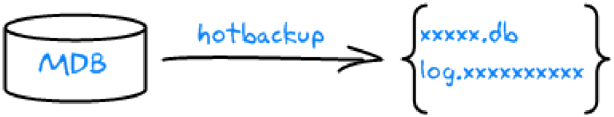
Find out:

- How many sites? Over LAN or WAN?
- Which ACK policy to take?
- Which is electable or not?
- ...

Prepare your dataset

If your initial dataset is empty, then go to next step.. otherwise follow this:

- Initialize your data into a Master site.
- Do a hotbackup of your master environment and compress all data into a package.
- Drag the package to where replica locates, decompress, and go to next step.



Start and Configure the Daemon(1/4)

Replication Options:

'-R'	identifies the host and port used by this site (required).
'-O'	identifies another site participating in this replication group
'-M/-S'	start as a master or slave

Start as a master:

```
memcachedb -p21201 -d -r -u root -f 21201.db -H /data1/demo
-N -P /data1/logs/21201.pid -R 127.0.0.1:31201 -M
```

Start as a slave:

```
memcachedb -p21202 -d -r -u root -f 21202.db -H /data1/demo
-N -P /data1/logs/21202.pid -R 127.0.0.1:31202 -O
127.0.0.1:31201 -S
```

Start and Configure the Daemon(2/4)

Besides running replication options, there are private commands available to configure the current site:

'`stats rep`' shows the status of Replication.

'`rep_set_priority`' sets the priority of a site for electing in replication.

'`rep_set_request`' sets the minimum and maximum number of missing log records that a client waits before requesting retransmission.

'`rep_set_bulk`' Enable bulk transfer or not in replication.

'`rep_set_ack_timeout`' sets ACK timeout value of the replication .

'`rep_set_ack_policy`' sets ACK policy of the replication.

DB_REPMGR_ACKS_ALL	1
DB_REPMGR_ACKS_ALL_PEERS	2
DB_REPMGR_ACKS_NONE	3
DB_REPMGR_ACKS_ONE	4
DB_REPMGR_ACKS_ONE_PEER	5
DB_REPMGR_ACKS_QUORUM	6

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
```

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
rep_set_priority 0
```

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
rep_set_priority 0
0
```

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
rep_set_priority 0
0
rep_set_ack_policy 5
```

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
rep_set_priority 0
0
rep_set_ack_policy 5
5
```

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
rep_set_priority 0
0
rep_set_ack_policy 5
5
rep_set_ack_timeout 50000
```

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
rep_set_priority 0
0
rep_set_ack_policy 5
5
rep_set_ack_timeout 50000
50000
```

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
rep_set_priority 0
0
rep_set_ack_policy 5
5
rep_set_ack_timeout 50000
50000
rep_set_request 2 4
```

Start and Configure the Daemon(3/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
rep_set_priority 0
0
rep_set_ack_policy 5
5
rep_set_ack_timeout 50000
50000
rep_set_request 2 4
2/4
```

Start and Configure the Daemon(4/4)

```
~ % telnet 127.0.0.1 21202
```

Start and Configure the Daemon(4/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

Start and Configure the Daemon(4/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
stats rep
```

Start and Configure the Daemon(4/4)

```
~ % telnet 127.0.0.1 21202
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
stats rep
STAT rep_whoismaster 127.0.0.1:31201
STAT rep_localhp 127.0.0.1:31202
STAT rep_ismaster REP_FALSE
STAT rep_priority 0
STAT rep_ack_policy 5
STAT rep_ack_timeout 50000
STAT rep_bulk 1
STAT rep_request 2/4
STAT rep_next_lsn 25/6752622
END
```

Part V

Managing and Monitoring

Managing DB Files

Home Environment

> *In order to transaction protect your database operations, you must use an environment.*
>
> *An environment, represents an encapsulation of one or more databases and any associated log and region files.*

There are three types of files in a environment:

'`Database files`' the exact files that store your data.

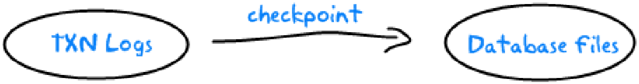
'`Log files`' all your transcations you commit first come into logs.

'`Region files`' files that back the share memory region using mmap().

Checkpoint(1/2)

- When databases are modified (that is, a transaction is committed), the modifications are recorded in DB's logs
- But the increased logs make recovery take too long time.
- Log files also have more unnecessary data than database file.
- Log files make catastrophic recovery possible.

So here comes Checkpoint:



Checkpoint(2/2)

The checkpoint:

- Flushes dirty pages from the in-memory cache.
- Writes a checkpoint record.
- Flushes the log.
- Writes a list of open databases.

How to do checkpoint in Memcachedb:

- Run checkpoint periodically with '-C' option
- The private command: 'db_checkpoint'

Backup Procedures

Hot Backup

```
/usr/local/BerkeleyDB.4.6/bin/db_hotbackup [-c] -h home -b
backup_dir
```

DB files has a high compression ratio, and use *gzip* and *tar* to archive.

Recovery Procedures

- Normal Recovery
  Put database files and log files since the last checkpoint(or a hotbackup copy) into a empty home environemt, Start the deamon on this environment, the deamon will do recovery automatically. Also it can be done by using standalone utility 'db_recover':

```
/usr/local/BerkeleyDB.4.6/bin/db_recover -f -h home
```

- Catastrophic Recovery
  Put all your database files and all log files(since environment created) into a empty home environment, Use standalone utility 'db_recover' with option '-c':

```
/usr/local/BerkeleyDB.4.6/bin/db_recover -cf -h home
```

Removing Log Files

Log Files in which transactions have been checkpointed into database file can be removed directly or archived to offline storage devices. You can remove this log files by

- using private command: 'db_archive'
- using standalone utility 'db_archive' with option '-d':

```
/usr/local/BerkeleyDB.4.6/bin/db_archive -d -h home
```

Warning: log file removal is likely to make catastrophic recovery impossible. If the data is very important(I mean if the data is lost, you are over.), you'd better archive them to offline storage devices instead of removal.

Monitoring

Overview

There are four ways now available for monitoring Memcachedb:

- the raw 'stats' commands: 'stats', 'stats bdb', 'stats rep'
- the standalone utility 'db_stat'
- the coming along monitoring tool 'tools/mdbtop.py'
- writing your own monitoring tool using 'tools/memcache.py'

'stats' command

There are a few of 'stats' commands in Memcachedb:

`'stats'` shows the status of current deamon.

`'stats bdb'` shows the status of BerkeleyDB.

`'stats rep'` shows the status of Replication.

Just telnet in, and use them. Some of Memcached Client APIs support 'stats' command, but some not. If you find no one can do it, have a try of patched 'memcache.py' in 'tools/' of distribution.

db_stat

BerkeleyDB distribution has a standalone utility 'db_stat' that can help us to get statistics for Berkeley DB environments.

'-c'	Display locking subsystem statistics
'-l'	Display logging subsystem statistics
'-m'	Display cache statistics
'-r'	Display replication statistics
'-t'	Display transaction subsystem statistics
	...

See 'docs/utility/db_stat.html' in BerkeleyDB distribution for more info.
Warning: '-d' option which displays database statistics for the specified file will be very expensive(that requires traversing the database), do not use it on a high traffic Memcachedb daemon.

mdbtop.py

'mdbtop.py' is a monitoring tool coming along with distribution. It is built on the patched 'memcache.py' Python API.

```
./mdbtop.py <mdbtop.cfg> [rep]
```

'mdbtop.cfg' is the configure file that mdbtop.py uses:

mdbtop.cfg

```
[Server]
server1 = 127.0.0.1:21201
server2 = 127.0.0.1:21202
[View]
interval = 1
```

[rep] option is for only replication syncing monitoring.

memcache.py

The patched 'memcache.py' now has all private commands supported and you can write your own monitoring tools depending on these APIs:

memcache.py

```
import memcache
mc = memcache.Client(['127.0.0.1:21202'], debug=0)
mc.db_archive()
mc.db_checkpoint()
mc.rep_ismaster()
mc.rep_whoismaster()
mc.rep_set_priority(100)
mc.rep_set_ack_policy(5)
mc.rep_set_ack_timeout(20000)
mc.rep_set_request(4, 16)
mc.disconnect_all()
```

Part VI

The End

Project info

Homepage: `http://memcachedb.org`
Mailing list: `http://groups.google.com/group/memcachedb`

- To subscribe to maillist, send email to memcachedb-subscribe@googlegroups.com
- To post to maillist, send email to memcachedb@googlegroups.com
- To unsubscribe from maillist, send email to memcachedb-unsubscribe@googlegroups.com

Please report your bugs and issues to the Maillist.

Q/A

Any question?