Metode avansate în sisteme distribuite (Seria: AAC)

Lab 8: Memcached

For the purpose of communicating with a memcached server we will be using libmemcached (not to be mistaken for libmemcache). It is a client library written in C.

libmemcahced Basics

Including <libmemcached/memcached.h> gives you access to all of libmemcached's functionality. Link your app with -lmemcached.

Most functions return an error code (or provide it via a parameter). Checking errors can be done as follows:

```
memcached_st *memc = ...;
memcached_return_t error = memcached_do_something_fancy();
if (error != MEMCACHED_SUCCESS)
    fprintf(stderr, "memcached_do_something_fancy: %s\n", memcached_strerror(memc
, error));
```

Creating a memcached context

```
memcached_st *memcached_create(memcached_st *ptr)
```

Sample usage:

```
memcached_st *memc = memcached_create(NULL);

/* do something with memc... */
memcached_free(memc);
```

Connecting to a memcached server

```
memcached_return_t memcached_server_add(memcached_st *ptr, const char *hostname,
in_port_t port)
```

Sample usage:

```
error = memcached_server_add(memc, "192.168.1.2", 0 /* use default port*/);
```

Adding a value to a server

```
memcached_return_t memcached_set(memcached_st *ptr, const char *key, size_t key_l
ength, const char *value, size_t value_length, time_t expiration, uint32_t flags)
```

```
memcached_return_t memcached_add(memcached_st *ptr, const char *key, size_t key_l
ength, const char *value, size_t value_length, time_t expiration, uint32_t flags)
```

```
memcached_return_t memcached_replace(memcached_st *ptr, const char *key, size_t k
ey_length, const char *value, size_t value_length, time_t expiration, uint32_t fl
ags)
```

While similar, the functions' behaviour differs depending on wether or not a key-value pair already exists on the server:

	exists	**doesn't exist**
set	replaced	created
add	error	created
replace	replaced	error

Sample usage:

```
error = memcached_set(memc, "foo", strlen("foo"), "bar", strlen("bar"), 0 /* does
 not expire */, 0);
```

Reading a single value

```
char *memcached_get(memcached_st *ptr, const char *key, size_t key_length, size_t
 *value_length, uint32_t *flags, memcached_return_t *error)
```

Sample usage:

```
char *value;
size_t value_length;
uint32_t flags = 0;
memcached_return_t error;

value = memcached_get(memc, "foo", strlen("foo"), &value_length, &flags, &error);

/* do something with the value... */

free(value);
```

Reading multiple values

```
memcached_return_t memcached_mget(memcached_st *ptr, const char * const *keys, co
nst size_t *key_length, size_t number_of_keys)
```

```
char *memcached_fetch(memcached_st *ptr,  char *key,  size_t *key_length,  size_t
 *value_length,  uint32_t *flags,  memcached_return_t *error)
```

Sample usage:

```
memcached_return_t error;
const char *keys[] = {"apple", "orange", "apricot"};
size_t key_lengths[]= {5, 6, 7};
size_t key_count = 3;
uint32_t flags;

char return_key[MEMCACHED_MAX_KEY];
size_t return_key_length;
char *return_value;
size_t return_value_length;

error = memcached_mget(memc, keys, key_lengths, key_count);

while ((return_value = memcached_fetch(memc, return_key, &return_key_length, &ret
urn_value_length, &flags, &error)))
{
    /* do something with the value */
    free(return_value);
}
/* anything aside from MEMCACHED_END and MEMCACHED_NOTFOUND means failure */
```

Flushing all the data from a server

The following command erases all key-value pairs residing on a server:

```
memflush --servers=<server address>
```

TASK 0: Install the following packeges: memcached, libmemcached-dev. Start memcached. Configure the loopback interface as follows:

- Set the MTU to 1500:

```
ifconfig lo mtu 1500
```

- Turn off generic and TCP segmentation offload:

```
ethtool -K lo gso off; ethtool -K lo tso off
```

- Rate-limit it to 100Mb/s and add a 10ms delay (RTT):

```
tc qdisc add dev lo root handle 1: tbf rate 100Mbit burst 10000 latency 10ms
tc qdisc add dev lo parent 1:1 handle 10: netem delay 5ms
```

- Use iperf and ping to make sure everything is ok.

TASK 1: Write a profiling tool for memcached.

- Suggested syntax:

```
./profiler write <key_count> <value_size> # writes key_count values of value_size bytes each
./profiler read <key_count> <batch_size>  # reads key_count values batch_size at a time
```

- The tool should print out the number of milliseconds it took to read/write the values. (Hint: use ftime)
- Use the following naming convention: key_<key number>. The key number must be made up of 6 digits. E. g.: key_000000, key_000001 etc.
- The values should be garbage (i.e. using uninitialized memory is ok).

TASK 2: Plot the following graphs:

- Plot the number of keys written per second as a function of value size. Use the following value sizes: 1B, 10B, 100B, 1KB, 10KB, 100KB, 1MB. (Use logscale on the x axis.)
- Plot the number of keys read per second as a function of value size for various batch sizes. Use the following batch sizes: 1, 5, 10, 20.
- Hints:
  - This can be tedious if done manually. It's a good idea to write scripts.

- The number of keys should vary by value size. Tweak it manually such that each run of the profiler (in write mode) takes roughly 3-4 seconds.

Ultima modificare: marţi, 9 decembrie 2014, 07:32

MENIU PRINCIPAL

Pagina mea

- Pagina principală
- Pagini site
- Profilul meu
- Current course
  - M-2-MAS-AAC
    - Participanţi
    - Badges
    - General
    - 6 octombrie - 12 octombrie
    - 13 octombrie - 19 octombrie
    - 20 octombrie - 26 octombrie
    - 27 octombrie - 2 noiembrie
    - 3 noiembrie - 9 noiembrie
    - 10 noiembrie - 16 noiembrie
    - 17 noiembrie - 23 noiembrie
    - 24 noiembrie - 30 noiembrie
    - 1 decembrie - 7 decembrie
    - 8 decembrie - 14 decembrie
      - Lecture 9: Distributed shared memory
      - **Lab 8: Memcached**
    - 15 decembrie - 21 decembrie
    - 22 decembrie - 28 decembrie
    - 29 decembrie - 4 ianuarie
    - 5 ianuarie - 11 ianuarie
    - 12 ianuarie - 18 ianuarie
    - 19 ianuarie - 25 ianuarie
- Cursurile mele

SETĂRI

- Administrare curs
- Setări profilul meu

OPENSTACK BETA

Create user

Check status

Goto Dashboard
Goto Documentation