Memcached

Group - 12
Prerna Mandal
Parikshit Tiwari
Ajinkya Gaikwad
Amarjit Kumar Singh

Memcached

What is Memcached?

Memcached is a …

High performance

Distributed

In-memory

Key-Value Store

Who developed Memcached?

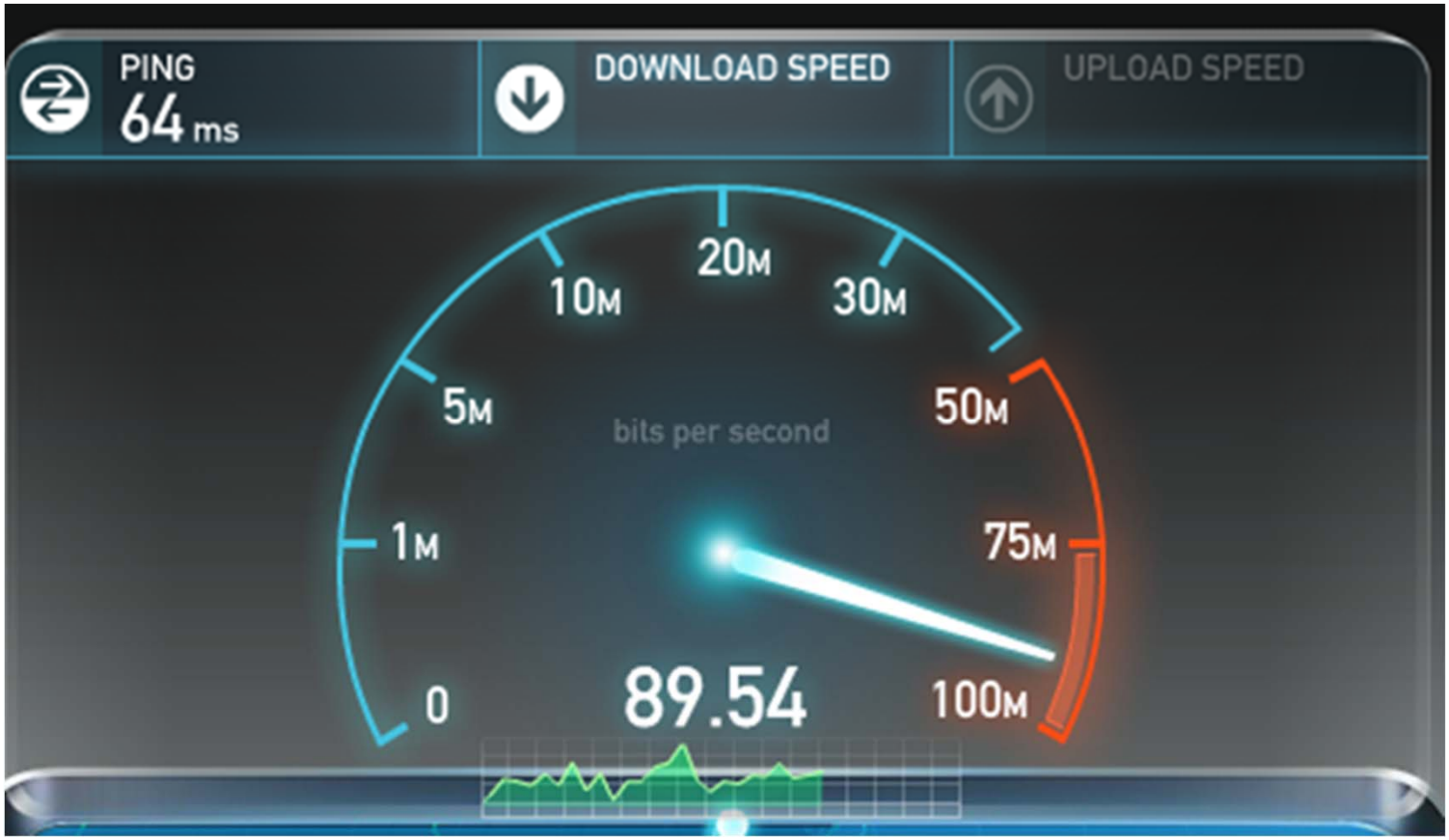
Developed by Brad Fitzpatrick

In 2003 for the website LiveJournal

Originally written in Perl

Later rewritten in C

WHY?


PING
64 ms
DOWNLOAD SPEED
UPLOAD SPEED
1M
5M
10M
20M
30M
50M
75M
100M
0
bits per second
89.54

Before Memcached…

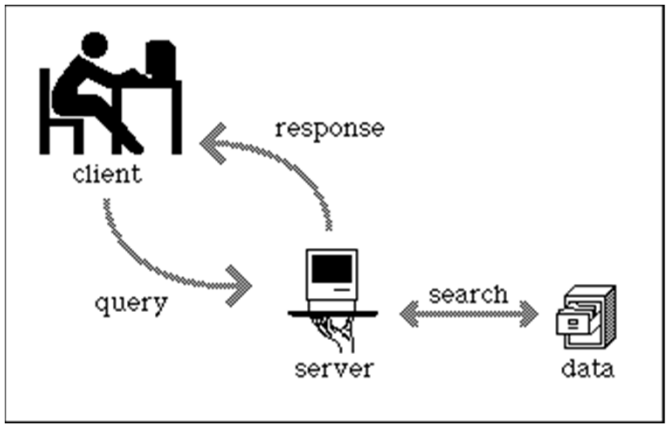
All data is retrieved from databases

Disk accesses are slow

Memory access are faster

Potential solution is caching

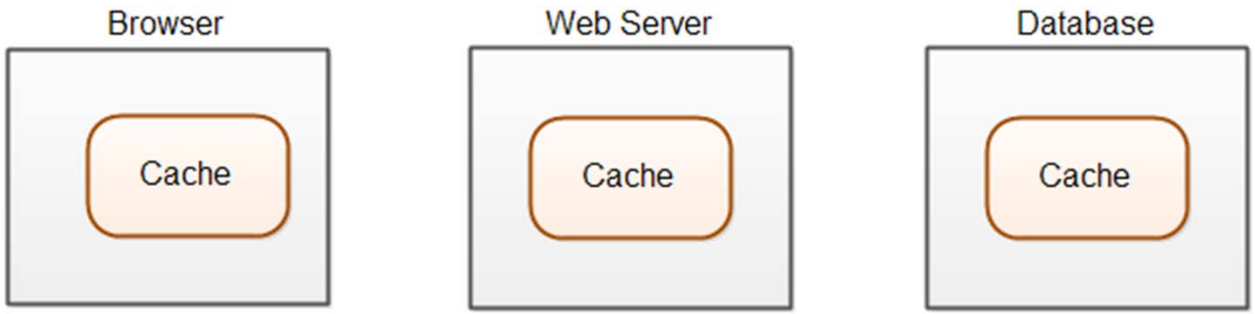
But where to cache data?



Caching Levels

A cache is typically stored in memory.

Caching can occur at different levels in a software system.



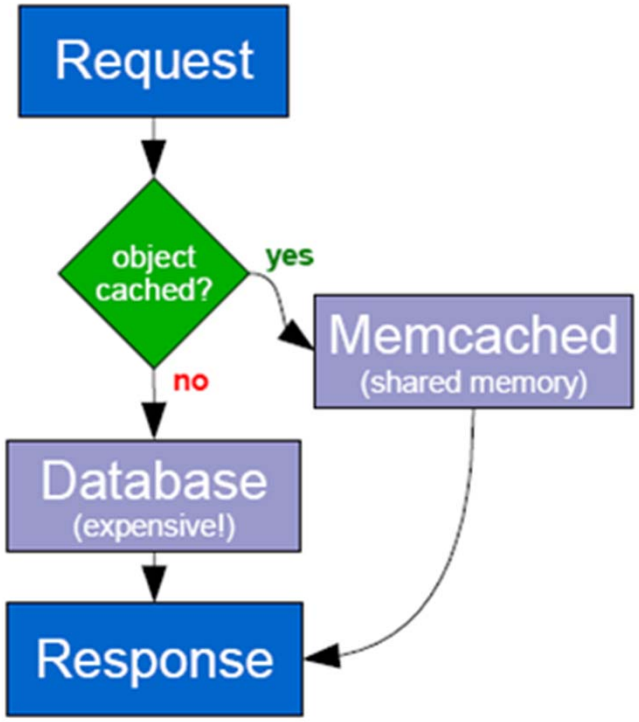
Disadvantages of Traditional Caching

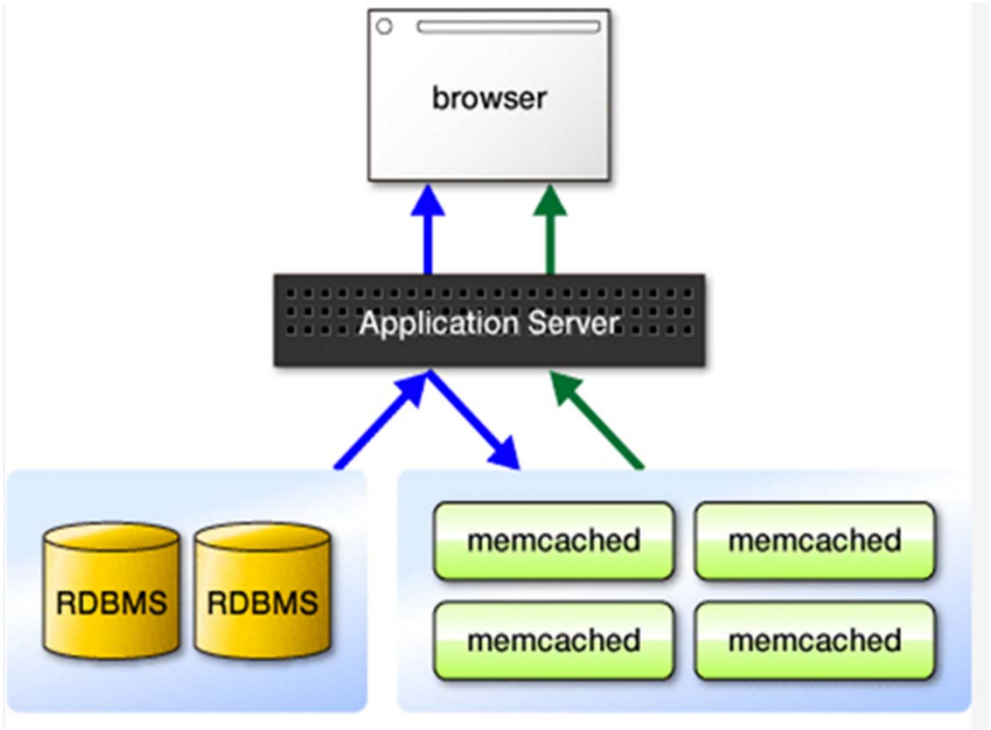
Single machine can cache only on its machine

To increase cache capacity server machines need to be upgraded

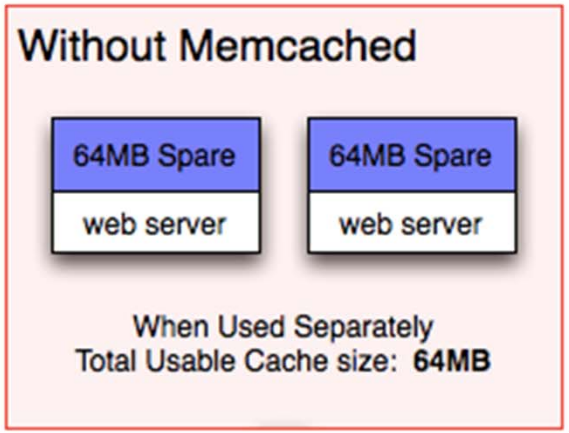
The complete network memory is not utilized

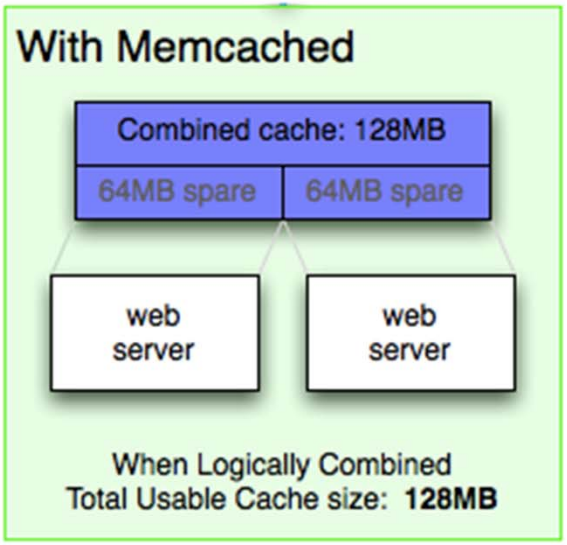
Alleviating Database Load



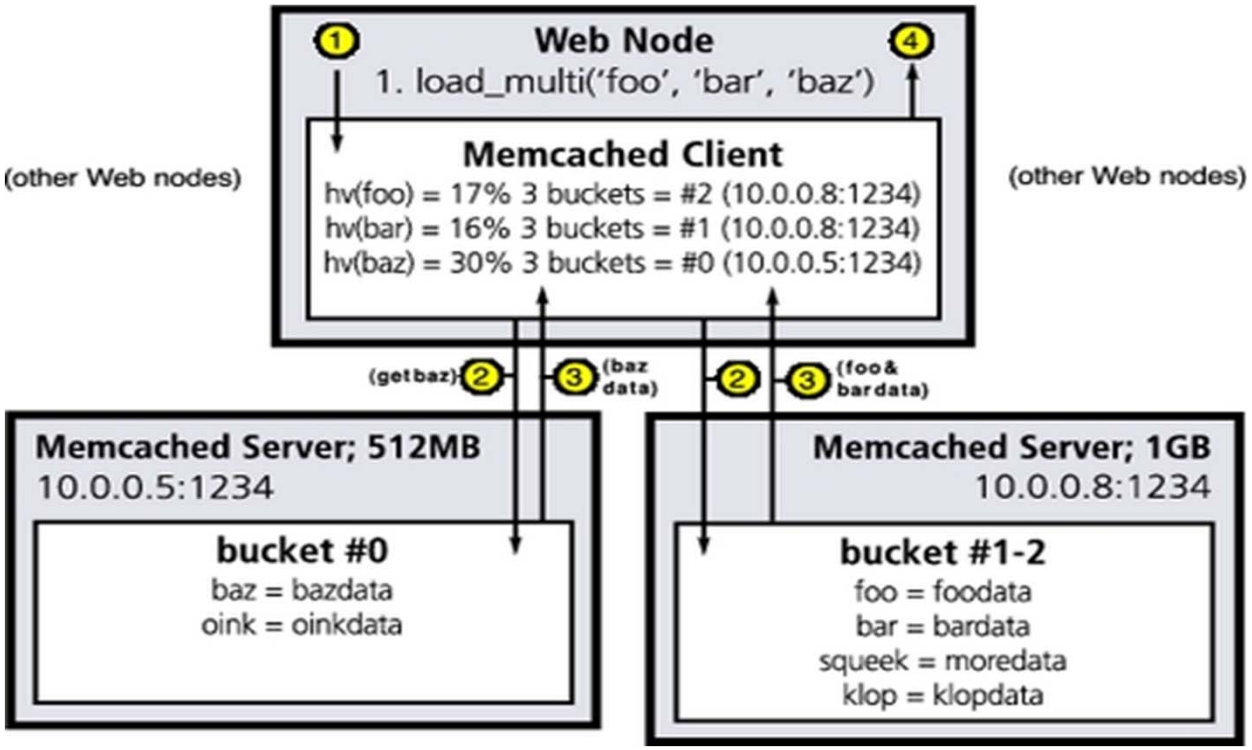


Spare Memory Utilization





Memcached is Distributed



Architecture
and
Process Flow

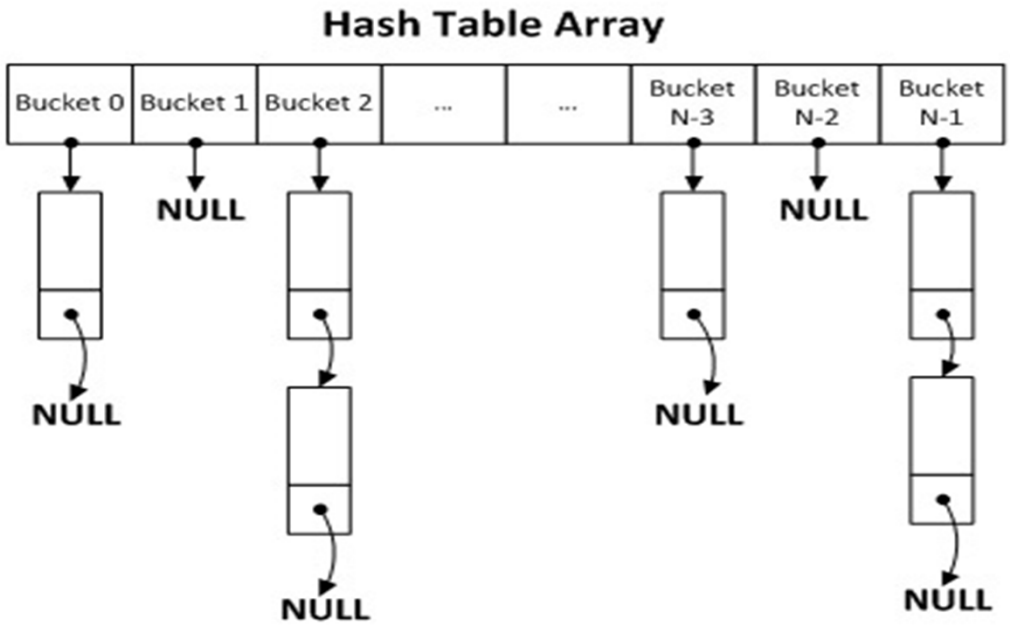
Architecture

- Key Value store

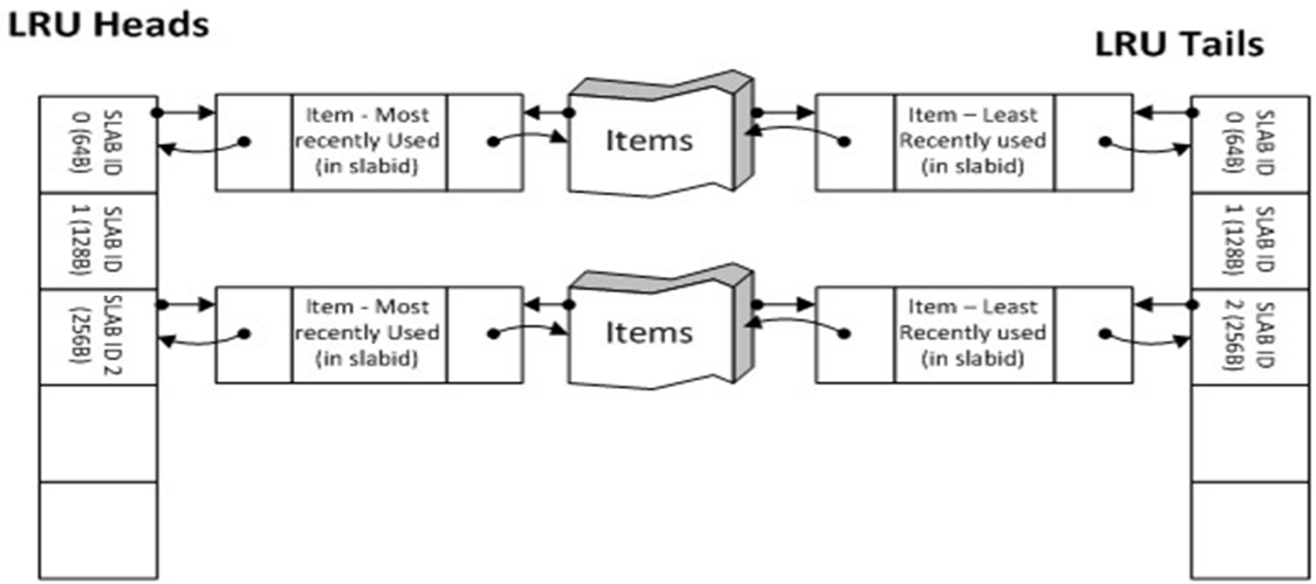
Data structures used-

- Hash table
- LRU(Least Recently Used) List
- Cache item
- Slab allocator

Hash table


Hash Table Array
Bucket 0
Bucket 1
Bucket 2
...
...
Bucket N-3
Bucket N-2
Bucket N-1
NULL
NULL
NULL
NULL
NULL
NULL

LRU List


LRU Heads
LRU Tails
SLAB ID 0 (64B)
SLAB ID 1 (128B)
SLAB ID 2 (256B)
Item - Most recently Used (in slabid)
Items
Item – Least Recently used (in slabid)
Item - Most recently Used (in slabid)
Items
Item – Least Recently used (in slabid)
SLAB ID 0 (64B)
SLAB ID 1 (128B)
SLAB ID 2 (256B)

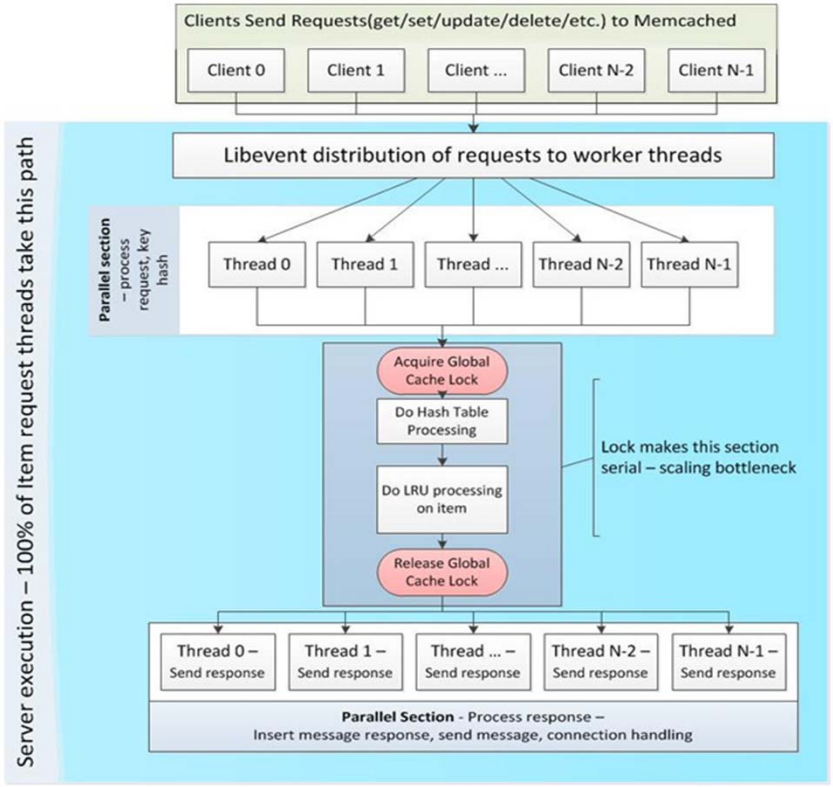
Cache item

- Pointers
- Key and Value
- Thread Counter
- Status

Basic Operations

- GET
- STORE
- DELETE

Process Flow



Memcached

Operations, Query Languages and Applications

Operations and Protocols Supported

- Protocol - TCP / UDP
- Data - Text / Unstructured
- Keys - Text string to uniquely identify the data
- Commands/Operations
    - Storage Commands
    - Retrieval commands

Storage commands

Command:

- <command name> <key> <exptime> <bytes>

"set", "add", "replace", "append" or "prepend"

- CAS "Check and Set"

Reply:

STORED NOT_STORED EXISTS NOT_FOUND

Retrieval commands

Command:

get/gets <key>*

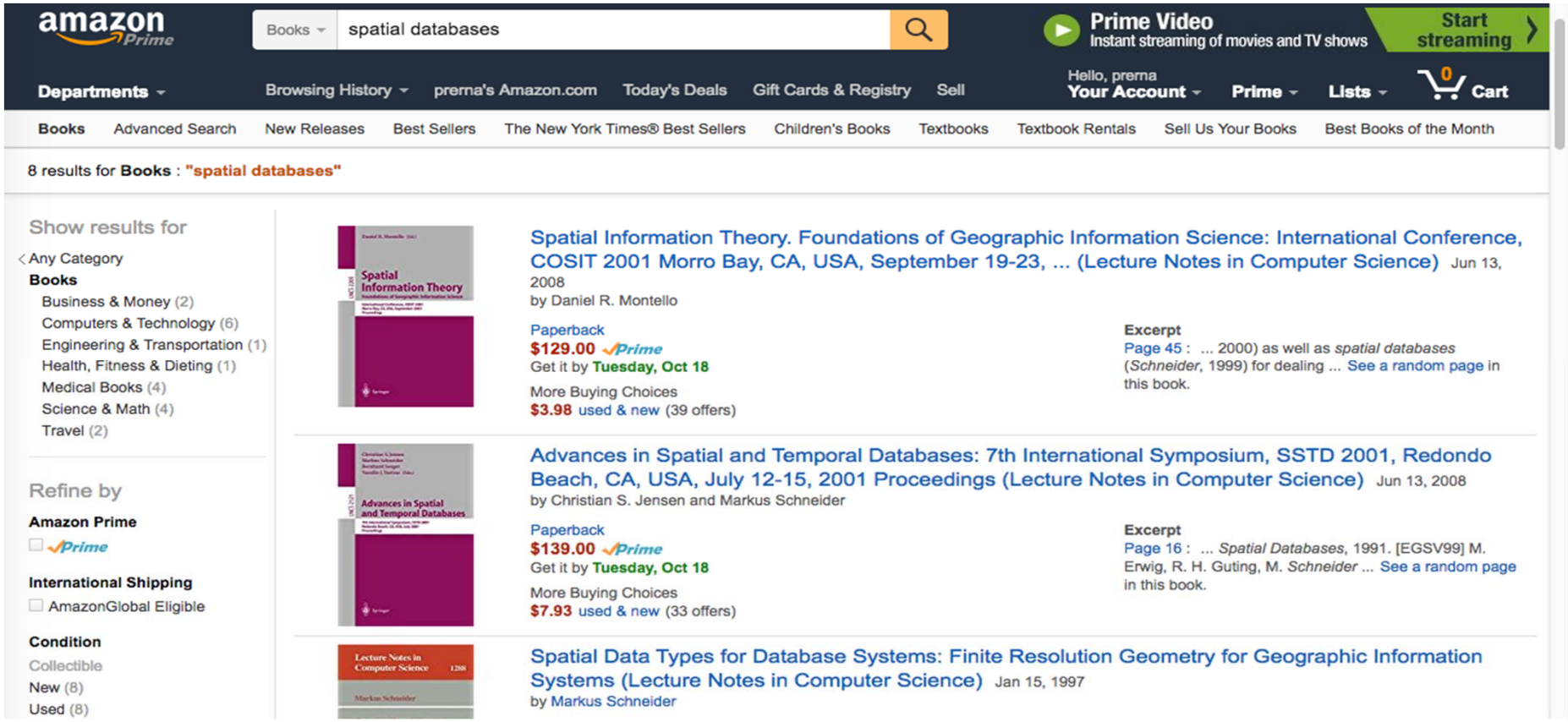
Reply:

VALUE <key> <flags> <bytes> [<cas unique>] <data block>

Other Commands

- Flush_all
- Stats
- Version
- Touch
- LRU_crawler
- Quit

Turbocharge Your Website With Memcached



Connecting to our Cache Server

```
// Connection constants
define('MEMCACHED_HOST', '127.0.0.1');
define('MEMCACHED_PORT', '11211');

// Connection creation
$memcache = new Memcache;
$cacheAvailable = $memcache->connect(MEMCACHED_HOST,
MEMCACHED_PORT);
```

Storing Data in our Cache

```
$sql = "INSERT INTO products (id, name, description, price) VALUES
($id, '$name', '$description', $price)";
$querySuccess = mysql_query($sql, $db);
```

Storing Data in our Cache

```
$sql = "INSERT INTO products (id, name, description, price) VALUES
($id, '$name', '$description', $price)";
$querySuccess = mysql_query($sql, $db);
if ($querySuccess === true)
{
   $key = 'product_' . $id;
   $product = array('id' => $id, 'name' => $name, 'description' =>
$description, 'price' => $price);
   $memcache->set($key, $product);
}
```

Cache Hit vs Cache Miss

- A cache hit is when something is looked up in cache and is found, no disk lookup is required.
- A cache miss is when something is looked up in the cache and is not found, cache did not contain the item being looked up.
- Cache miss further requires disk read and majorly contributed to performance overhead.

Retrieving Data from our Cache

```
  $sql = "SELECT id, name, description, price FROM products WHERE
id = " . $id;
   $queryResource = mysql_query($sql, $db);
   $product = mysql_fetch_assoc($queryResource);
```

```php
$product = null;
if ($cacheAvailable == true)
{
   $key = 'product_' . $id;
   $product = $memcache->get($key);
}
// do we need to access MySQL ?
if (!$product)
{
  $sql = "SELECT id, name, description, price FROM products WHERE
id = " . $id;
   $queryResource = mysql_query($sql, $db);
   $product = mysql_fetch_assoc($queryResource);
}
```

Other Languages Supported

.Net	ColdFusion	Lisp	Python
C	Erlang	Lua	Perl
C++	Java	OCaml	Ruby

Memcached Applications

- Cache everything that is slow to query, fetch, or calculate.
- Specially suited for 1-9-90 rule.
- Majorly social networking and content websites.

Memcached Users

craigslist

WIKIPEDIA
The Free Encyclopedia

Theoretical Limits

- No of Keys:
  - Limited by the memory size
  - When memcached runs out of memory, eviction based on LRU policy of keys
- Memcached operations are almost all O(1)
- Maximum key length = 250 bytes
- Maximum item size = 1MB

Advantages

- Memcached boosts performance
- Memcached is non - blocking
- Cross Platform.
- Cross-DBMS.
- Memcached is free.

Limitation & Caveats

- Memcache is volatile
- Memcache is a limited resource
- Not Transactional
- Cold Cache
- Limitations Due To Scale Out Approach
- Old Data Access

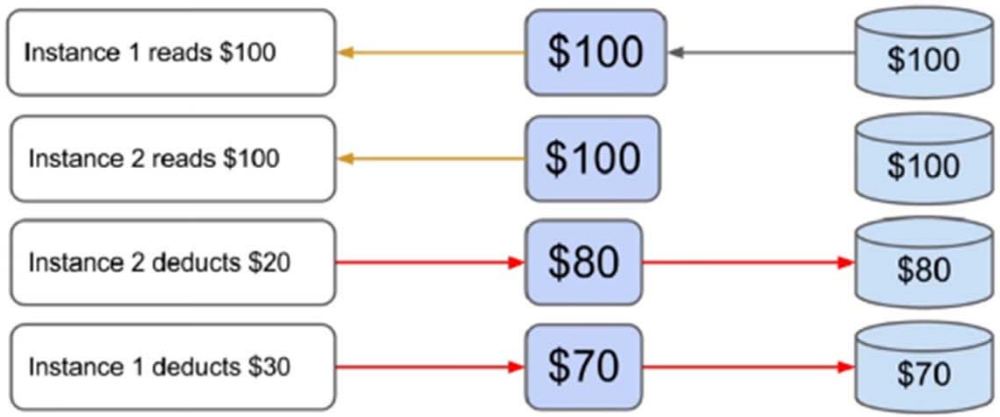
Memcached is volatile

- Entries can be evicted anytime for various reasons:
  - Entries reach expiration.
  - Entry is evicted because memory is full.
  - Memcache server fails.

Memcached is a limited resource

- Only need to cache what is useful and necessary.
- Only gives performance boost.
- Application should function without memcache.

Memcache - not transactional



Make use of getIdentifiable() and putIfUntouched()

Cold Cache

- Sudden failure or offline maintenance causes data loss at memcache server and performance degradation
- Memcached server needs to warm up again.
- All data request needs to be serviced by database in RDBMS layer.
- On failure, storing all data once again is costly

Limitations Due To Scale Out Approach

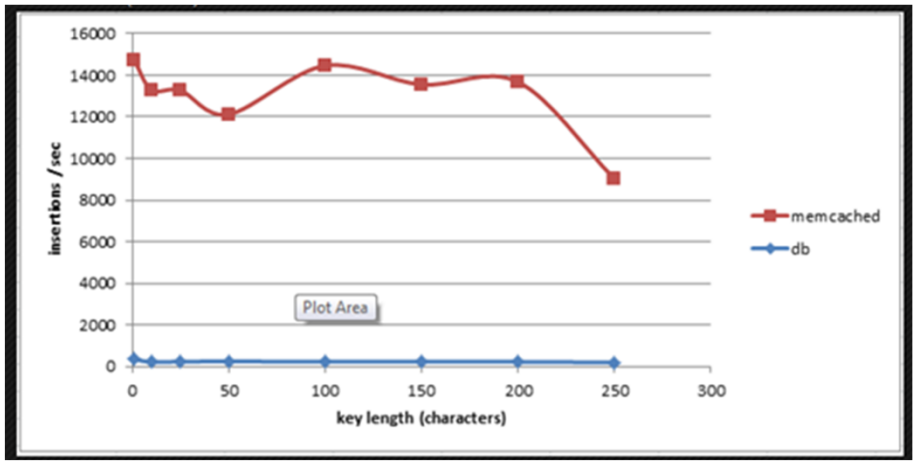
- Scale out approach used instead of scale up when more memory needed.
- Adding new a new node to existing N nodes, around 1/(N+1) keys needs to be remapped to different nodes.
- Client needs to be updated remapping of keys to avoid data loss or incorrect data delivery.
- Application forced to send query to RDBMS or return incorrect data to user.

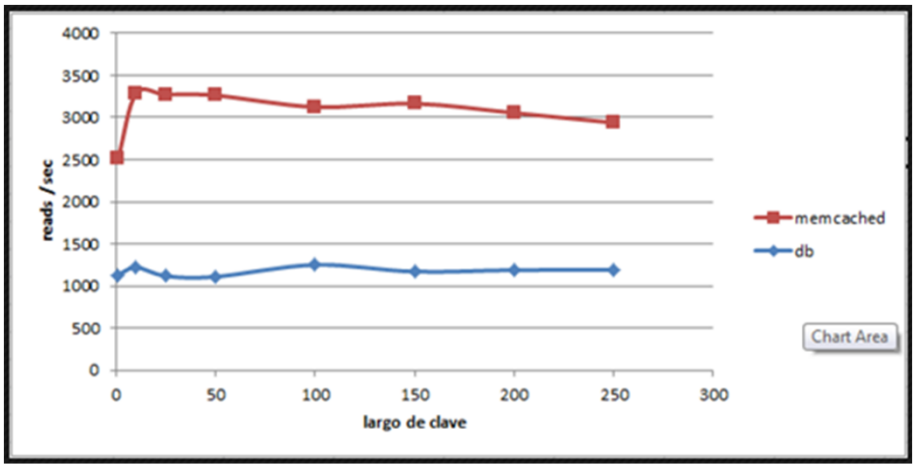
Stale Data Access

- No strict state for where a given key lives in memcached hashing
- No up-to-date key server mapping info, client might read or write from wrong memcached server – stale or inconsistent data
- Clients without updated key-server remapping info will read stale data.

Memcached vs DB Cache

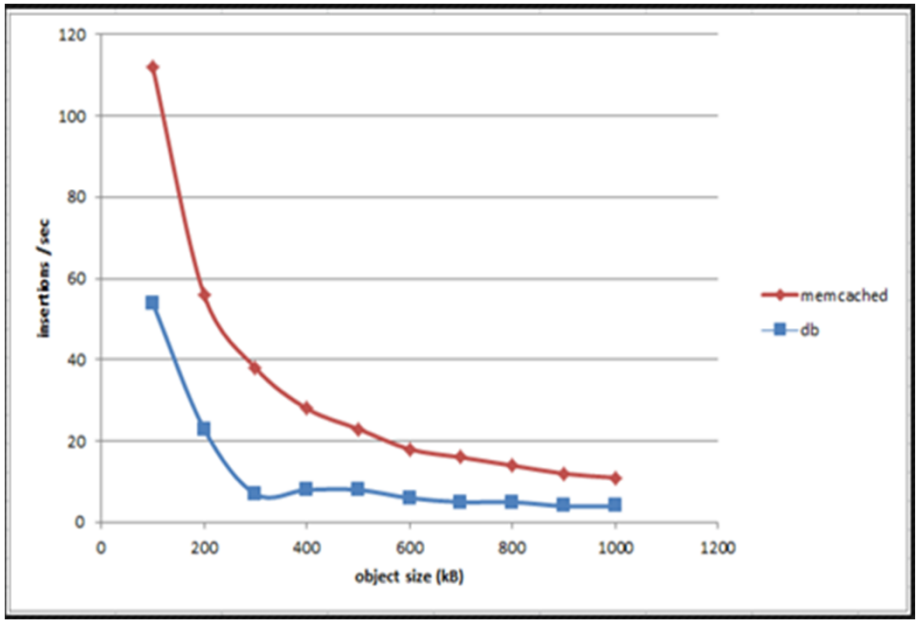
Variable length of key. Item Size = 6KB

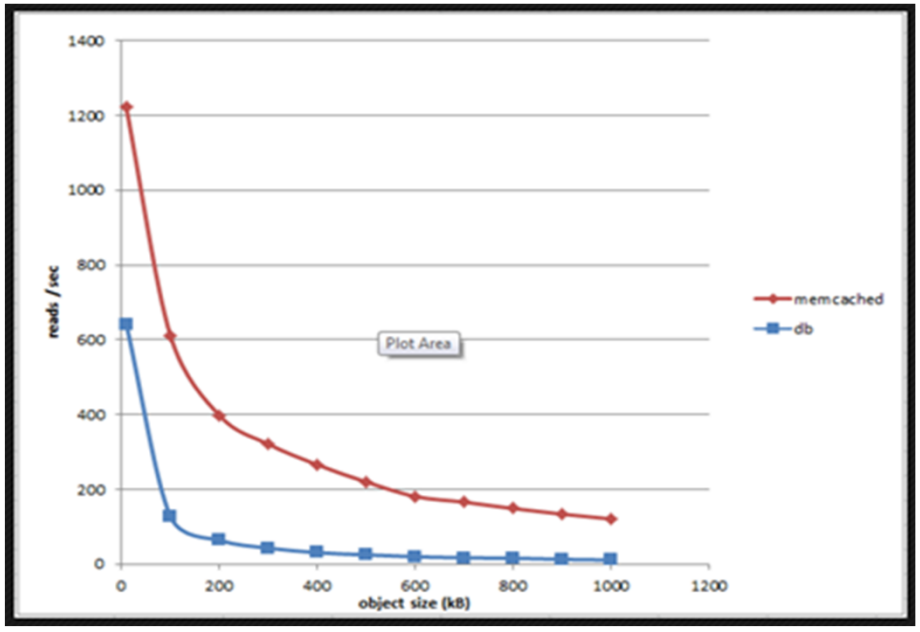




Memcached vs DB Cache

Variable size object. Key Size = 100 bytes



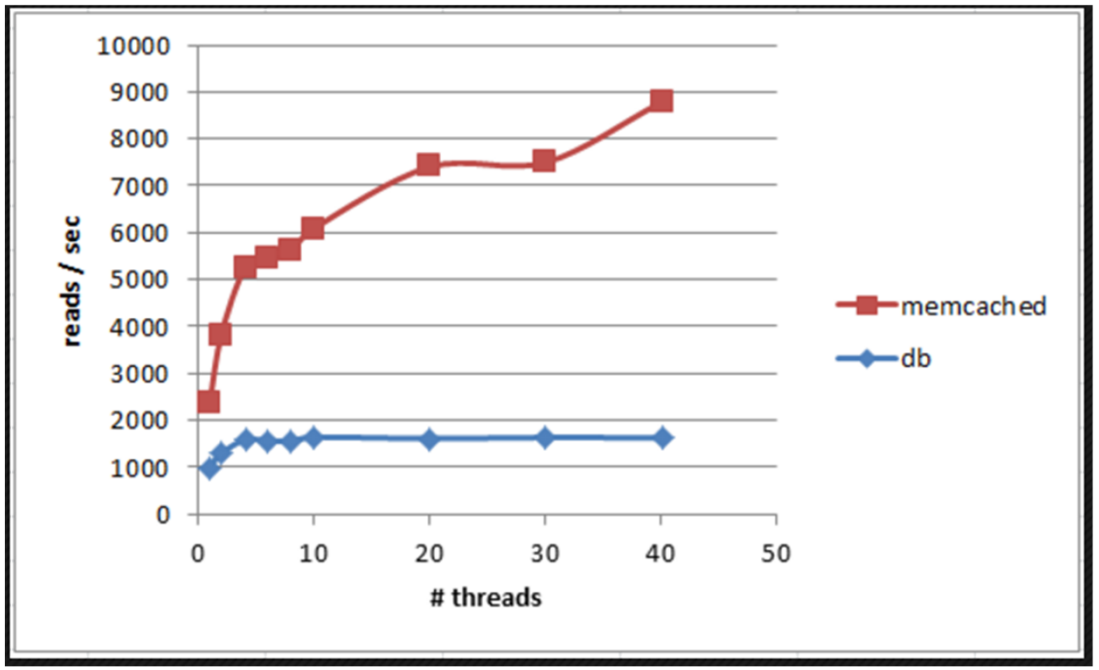


Memcache vs DB Cache

Variable #of threads.

Length of key = 100 Bytes

Item Size = 6kB



Memcache vs DB Cache

Conclusion :

- Memcache always performs better no matter what the variable.
- Even with large number of connections, the performance is robust.
- Memcache performance degrades with with big objects, but performs very well
- with average size objects.

Thank You.