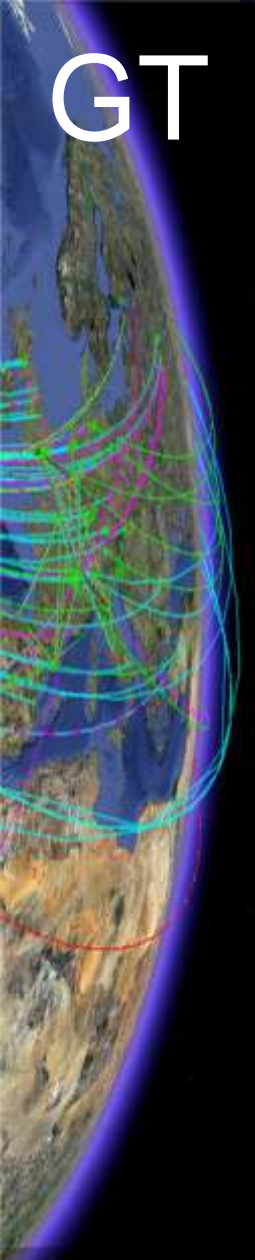
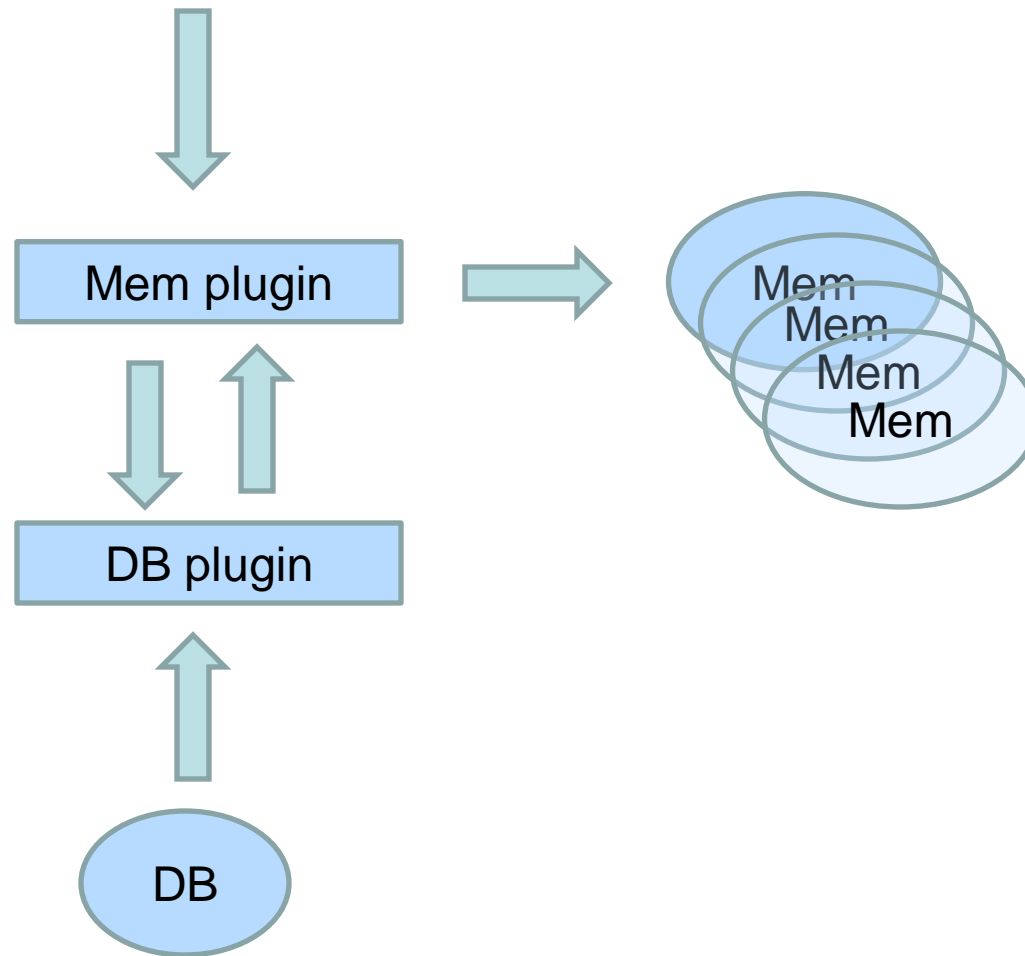


Memcached for dmlite





Memcached in the stack





- **extendedStat | readLink**
 - create, unlink, makeDir, removeDir, symlink, chmod, chown, setACL, utime
- **get | getReplicas**
 - addReplica, deleteReplica, replicaSet*
- **openDir | readDir | closeDir**
- **getComment**
 - setComment



- `extendedStat(path)` ← unchanged
- `extendedStat(inode)`
`extendedStat(parentInode, name):`
 - `memcKey = keyFromAny(STAT, inode)`
`memcVal = memcached_get(memcKey)`
`if (memcVal)`
 - `stat = deserialize(memcVal)``else`
 - `DELEGATE_ASSIGN(stat, extendedStat, inode)`
 - `memcVal = serialize(stat)`
 - `memcached_set(memcKey, memcVal)`
`return stat`



- Memcached Representation

stat

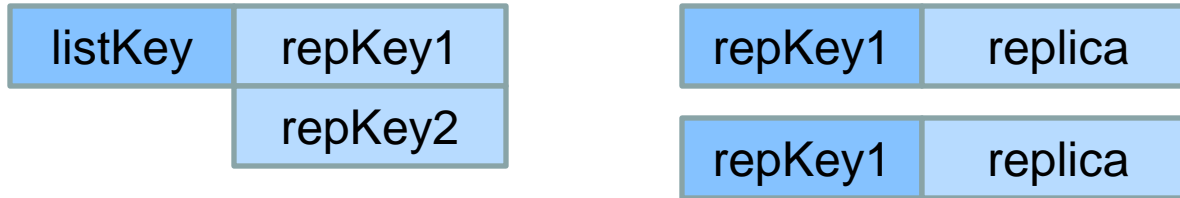
STAT:inode	stat
------------	------

STAT:parentInode:name	stat
-----------------------	------



- Side effects:
 - create, unlink, ... → st_nlink changes of parent
- Solution:
 - memcached_del(inode)
 memcached_del(parentOf(inode))
- In practice:
 - inode = extendedStat(path)
 key = keyFromAny(STAT, inode)
 memcached_del(key)
 pInode = getParent(inode)
 key = keyFromAny(STAT, pInode)
 memcached_del(key)

- Memcached List Implementation:



Get the list:

```
listKey = keyFromAny(DIR, inode)
list = memcached_get(listKey)
if (list)
    replicas = memcached_multiget(list)
```

```
memcached_multiget(keyList):
memcached_mget(keyList)
while (val = memcached_fetch())
    list.append(val)
```

Make the list:

```
listKey = keyFromAny(DIR, inode)
memcached_set(listKey, NULL)
```

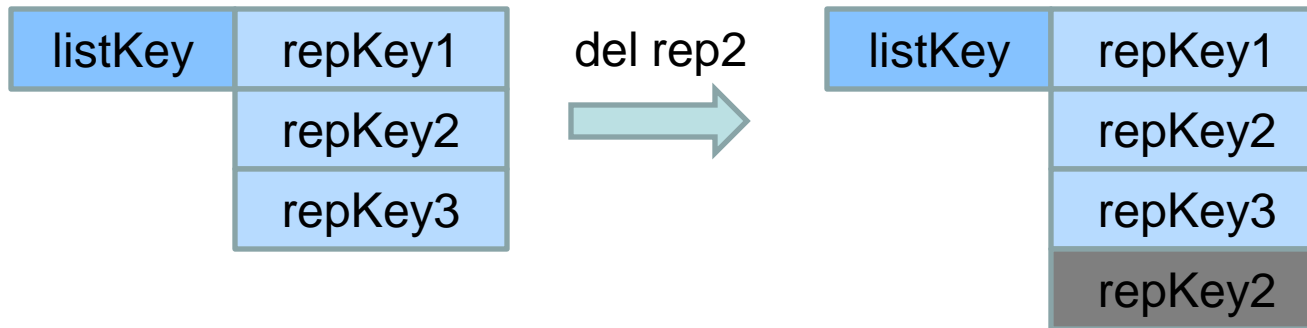
(...)

```
repKey = keyFromAny(REPLICA, uri)
memcached_set(repKey, replica)
memcached_append(listKey, repKey)
```



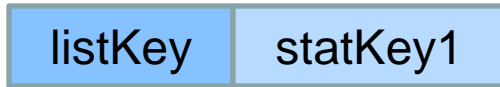
No List Download

- Side effects:
 - addReplica, delReplica
- Solution: blacklists



- In Practice:
 - lazy add: only create list item

- cached as list:

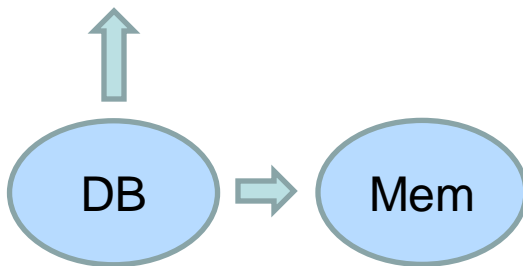


- List extension: isComplete bit

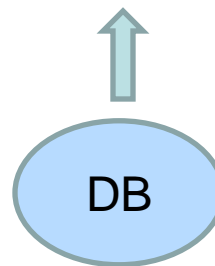


- 3 cases:

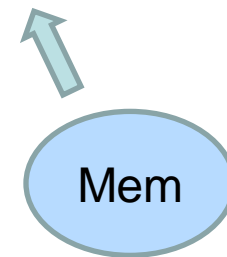
Not Cached



List not complete



List complete



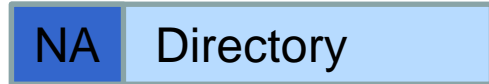


- Make a List:
- openDir():
 - memcached_set(listKey, NULL, complete = False)
- readdir():
 - memcached_set(stat)
 - memcached_append(listKey, stat)
- readdir() == 0x00:
 - mark_list_complete(listKey, complete = True)

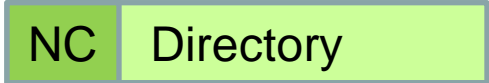


- Concurrency:

openDir() =



openDir() =



readDir() =

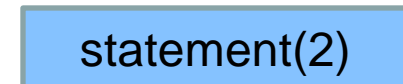
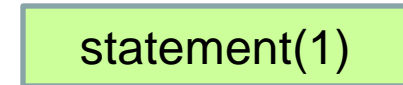
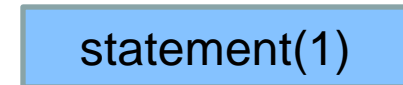
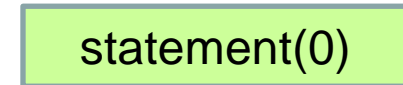
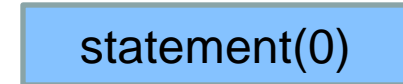
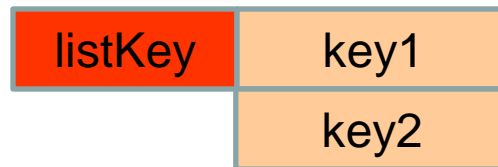
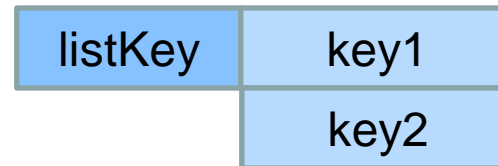
readDir() =

readDir() =

readDir() = 0x00

openDir() =

time





- memcached 1M limit:
 - Num Replicas per File ~7M
 - Num Files per Dir ~40M
- MemcacheDir hack:
 - Num inodes ~1 Billion
- readdir issue:
 - No lazy file-fetching
 - Memory limited



- Example:

.proto file:

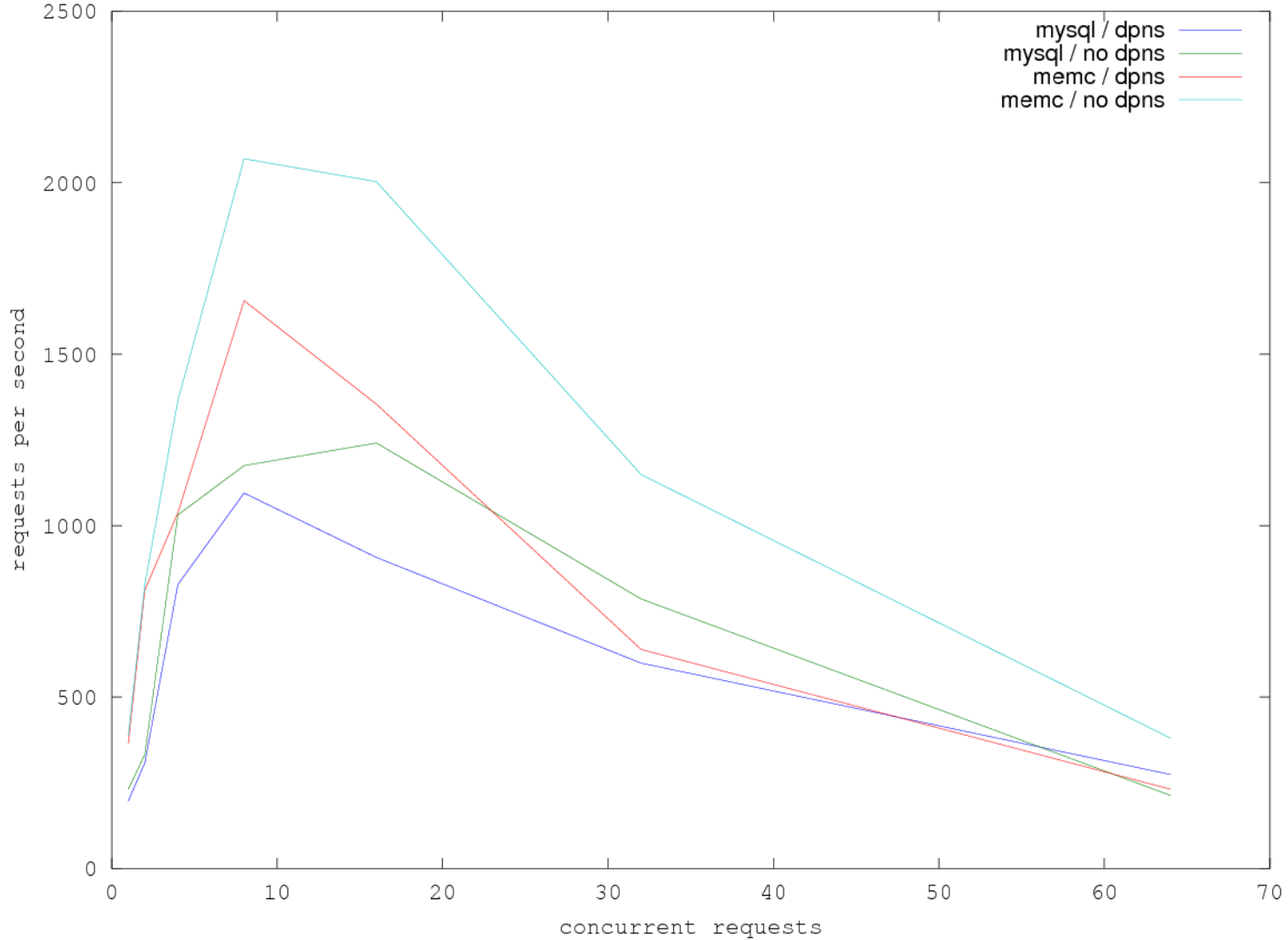
```
message SerialKeyList {  
  required bool iscomplete = 1;  
  repeated SerialKey key = 2;  
}  
  
message SerialKey {  
  required string key = 1;  
  required bool white = 2 [default = true];  
}
```

```
def serialize(list = [k1, k2, k3 ,k4])  
  
  SerialKeyList skList;  
  string serializedList  
  
  for key in list  
    skList.add_key(key)  
  
  serializedList =  
    skList.SerializeAsString()  
  
  return serializedList
```



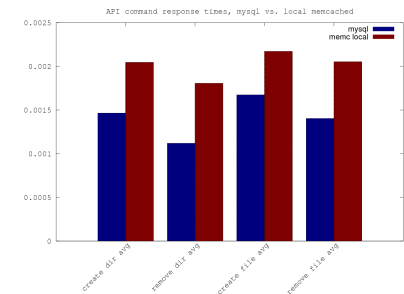
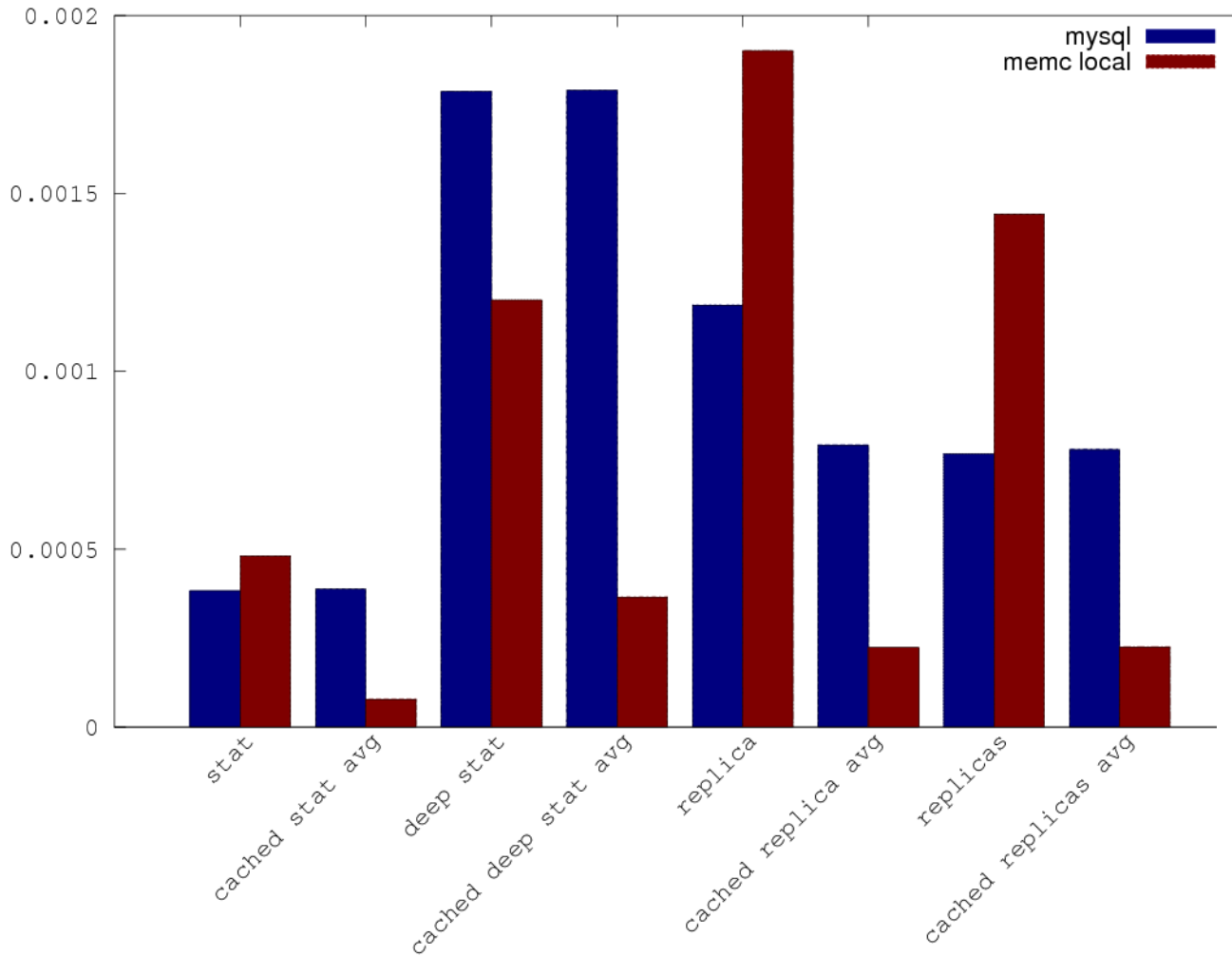
webDAV Performance /1

1000 stat requests on 1000 files with n threads on lxfsra04a04, zoomed.

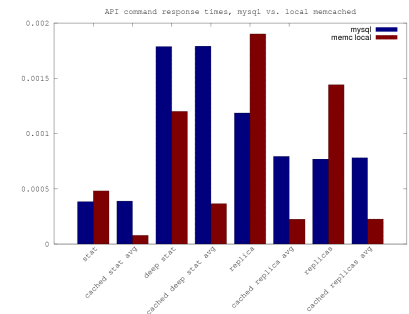
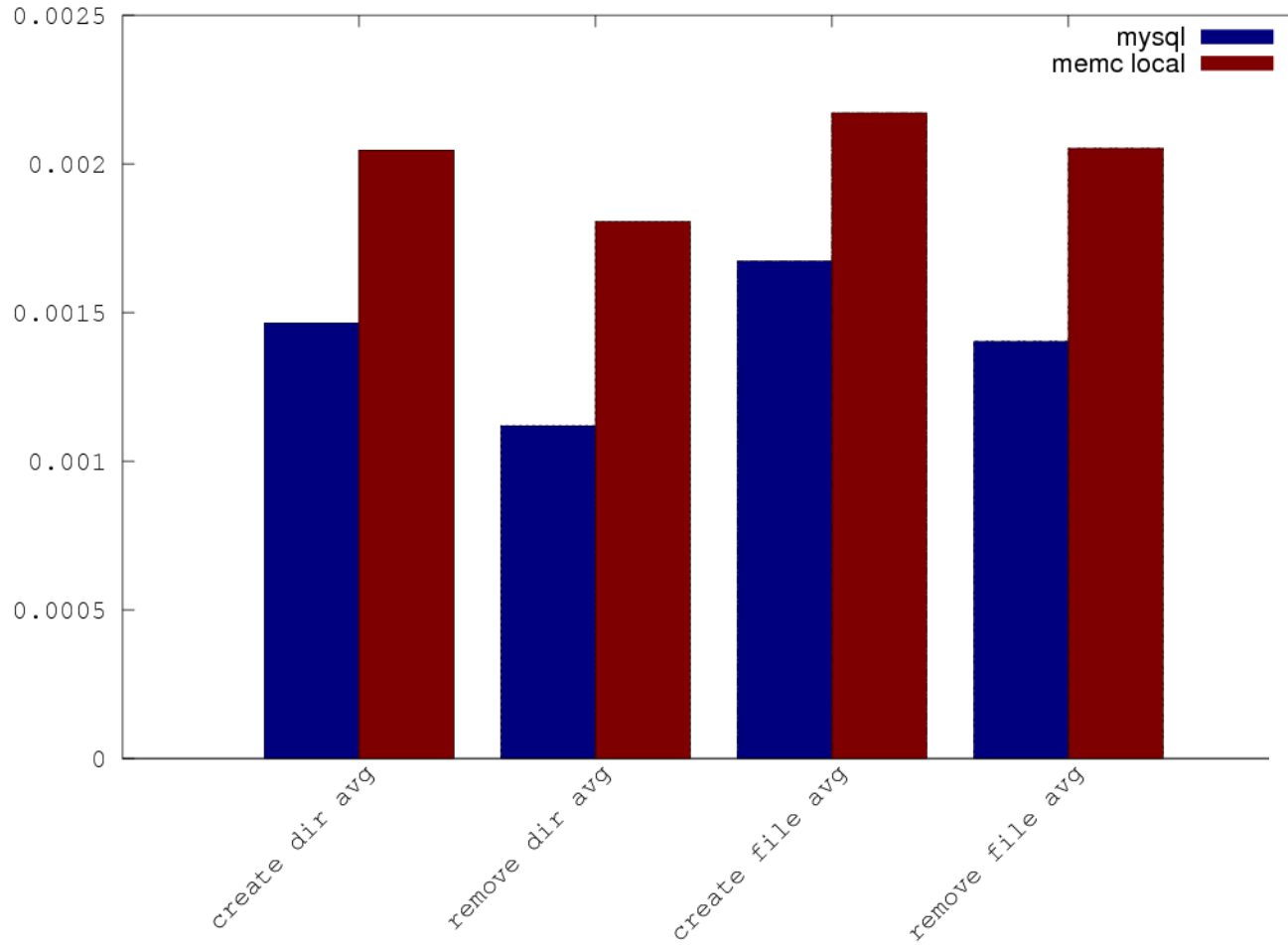




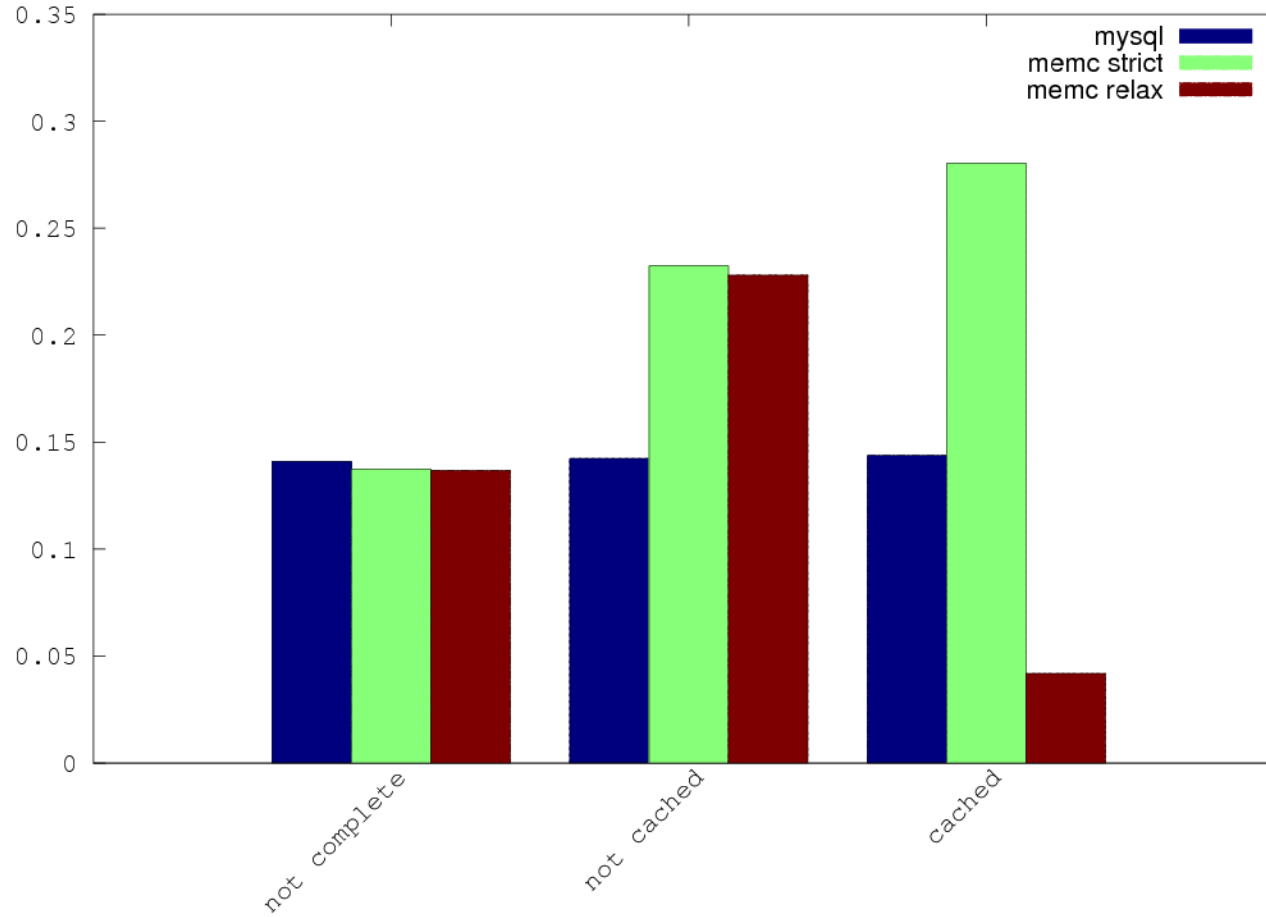
API command response times, mysql vs. local memcached



API command response times, mysql vs. local memcached



readDir response times, mysql vs. local memcached with and w/o MemcachedStrictConsistency



Components > Dmlite > memcached

The screenshot shows a web browser window with the URL `https://svnweb.cern.ch/trac/lcgdm/wiki/Dpm/Dev/Dmlite/Plugins/Memcache`. The page content is as follows:

DPM (Disk Pool Manager)

The Disk Pool Manager (DPM) is a lightweight solution for disk storage management.

- Support: [DPM Users Forum](#), [GGUS Helpdesk](#), [Dev List](#)
- User Docs: ([Tutorial](#), [Codebook/FAQ?](#), [API Overview](#), [Command Line Tools](#))
- Admin Docs: ([Installation](#), [Configuration](#), [Maintenance](#), [Performance](#), [Monitoring](#), [Reference Card](#), [EMI Reference Card](#))
- Development: ([Savannah](#), [Certification](#), [Release Procedure](#), [Internal Testbed](#), [Components](#), [Coordination](#), [Recipes](#))

memcached Plugin

The memcached plugin caches results from dmlite API calls.

For supported API calls, the plugin creates a key using the function arguments and looks it up in memcached. If the key is not found, the call is delegated and its result cached using the previously calculated key.

The plugin uses Google Protocol Buffers to serialize the data objects into strings, which can be fed to the memcached.

Usage

The plugin should be loaded last(!) so that it is the first one in dmlite's plugin queue.

dmlite API functionality covered

- stat
- getreplicas
- getcomment

Dependencies

- libmemcached
- google.protobuf

Memcached connection

- ASCII Protocol
- binary Protocol

Item Sizes on Memcached

These sizes represent the serialized values which are sent to memcached. The serialization used is google.protobuf. The size consists of the key, the value and the memcached datastructure (48 bytes on SLS 64bit).

- ExtendedStat 8-23 (key) + 60-101 (value) + 48 = 68-124 bytes
- Replica ~31 (key) + ~93 (value) + 48 = 172 bytes

At the bottom of the browser window, there is a download bar showing files like 'SAF meeting 1...pptx', 'cessda-analy...docx', and 'dariah-analys...docx'.

<https://svnweb.cern.ch/trac/lcgdm/wiki/Dpm/Dev/Dmlite/Plugins/Memcache>



- memcached.org
- libmemcached.org
- code.google.com/p/protobuf



- **save stat by path:**
 - extendedStat can fetch the tree all at once
- **invalidate the cache asynchronously**
 - deletes and utime() to DB
- **serialized values from delegate**
 - mysql → struct → string



End

Thank you for your attention!



- openDir():
- dir = MemcacheDir()
list = memcached_get(listKey)

if not list

 DELEGATE_ASSIGN(dir, openDir, path)

 dir.isCached = notcached

elif list != complete

 dir.isCached = notcomplete

else

 dir.isCached = complete



Halde

NC	Directory
----	-----------

C	Directory
---	-----------

NA	Directory
----	-----------

NC	Directory
----	-----------

listKey	key1
	key2

listKey	repKey1
	repKey2
	repKey3
	repKey2