

A decorative header at the top of the slide features four overlapping spheres. From left to right, they are light green, light blue, light red, and light yellow. The spheres are partially cut off by the top edge of the slide.

Part 4: Memcache API + Thoughts on Scaling



Recap

So far, we've created a wiki that:

- Allows users to create and edit wiki pages
- Stores the revision of each page
- Has user profiles
- Sends email to admins

Memcache API

Every time we serve a page we make at least 1 query to the datastore. This costs us

- Time
- Money

Memcache provides a distributed in-memory data cache that our application can use to cache data



The API

Memcache lets you store a string key, value, and an expiration time. It provides the following functions:

- `add()` [only if the value doesn't exist]
- `set(key, value, time)`, `set_multi(...)` [even if value exists]
- `get(key)`, `get_multi(...)`
- `delete(key)`, `delete_multi(...)`
- `replace(...)`
- `incr(key, delta)`
- `decr(key, delta)`
- `flush_all()`
- `get_stats()`

Using Memcache

```
def get_data():  
    data = memcache.get("key")  
    if data is not None:  
        return data  
    else:  
        data = self.query_for_data()  
        memcache.add("key", data, 60)  
        return data
```



Brazenly Stolen Scalability Slides

Watch the original!

- <http://sites.google.com/site/io/>

Loading Python modules on every request can be slow

- **Reuse** `main()` to addresses this:

```
def main():  
    wsgiref.handlers.CGIHandler().run(my_app)  
if __name__ == "__main__":  
    main()
```

- **Lazy-load** big modules to reduce the "warm-up" cost

```
def my_expensive_operation():  
    import big_module  
    big_module.do_work()
```



More Tips

Avoid large result sets

- In-memory sorting and filtering can be slow
- Make the Datastore work for you

Avoid repeated queries

- Landing pages that use the same query for everyone
- Incoherent caching
- Use memcache for a consistent view

Use Keys!

- Key corresponds to the Bigtable row for an Entity
- Bigtable accessible as a distributed hashtable
- Get() by Key: Very fast! No scanning, just copying data
- Limitations:
 - Only one ID or key_name per Entity
 - Cannot change ID or key_name later
 - 500 bytes

Last Hack Session

Add Memcache to your wiki!

