Designing A Toolkit for Distributed Storage in Web Applications

Neha Narula* and Robert Morris
MIT CSAIL, *Google

Goals
• Help high-traffic web sites cache and partition data for better performance
• Offer performance comparable to a customized memcached + MySQL database

Current Practice
• LAMP stack (Linux, Apache, MySQL, PHP)
• Memcached for caching and multiple MySQL databases to partition the write workload
• Caching avoids recomputing expensive queries and spreads read load
• Partitioning allows concurrent writes

Challenges
• Caching is hard due to invalidation
  • Programmer must decide freshness policy
  • Partitioning is hard due to tradeoffs
  • Web app data doesn’t necessarily cleanly partition
  • You only get one partition plan
  • Often unclear which is best
• Change to data layout is hard but necessary
• Changes of caching and partitioning often require extensive application modifications

Approach
• Separate application logic from data plan
  • Allows changes to data plan w/o app changes
  • Application logic:
    • Can choose from a set of pre-defined queries
    • Data plan defines those queries
    • Configured by developer
    • Constructed from a toolkit of query execution modules
    • Modules for caching, partitioning, joins, etc
  • Developer composes modules to define queries

Related Work
• Existing frameworks for horizontally partitioning MySQL DBs
  • MySQL NDB, HiveDB, Hibernate, HSCALE
• Still hard to change partitions or manage caches.
  • Simply structured key-value stores
    • CouchDB, Amazon’s SimpleDB
  • Less structured model
  • Facebook uses precomputed JOINs on memcached

A Pligg Query
Recent Public Links

```
SELECT link_id FROM links
LEFT JOIN groups ON links.link_group_id = groups.group_id
WHERE link_status='published' AND link_category in (1)
AND (groups.group_privacy!='private' OR ISNULL(groups.group_privacy))
ORDER BY link_published_date DESC LIMIT 0,8
```

New Pligg Queries

Challenges
• Caching is hard due to invalidation
  • Programmer must decide freshness policy
  • Partitioning is hard due to tradeoffs
  • Web app data doesn’t necessarily cleanly partition
  • You only get one partition plan
  • Often unclear which is best
• Change to data layout is hard but necessary
• Changes of caching and partitioning often require extensive application modifications

Approach
• Separate application logic from data plan
  • Allows changes to data plan w/o app changes
  • Application logic:
    • Can choose from a set of pre-defined queries
    • Data plan defines those queries
    • Configured by developer
    • Constructed from a toolkit of query execution modules
    • Modules for caching, partitioning, joins, etc
  • Developer composes modules to define queries

Related Work
• Existing frameworks for horizontally partitioning MySQL DBs
  • MySQL NDB, HiveDB, Hibernate, HSCALE
• Still hard to change partitions or manage caches.
  • Simply structured key-value stores
    • CouchDB, Amazon’s SimpleDB
  • Less structured model
  • Facebook uses precomputed JOINs on memcached