Fourth Annual

MySQL®

Users Conference 2006

Wikipedia: cheap & explosive scaling with LAMP

Domas Mituzas
MySQL AB,
Wikipedia

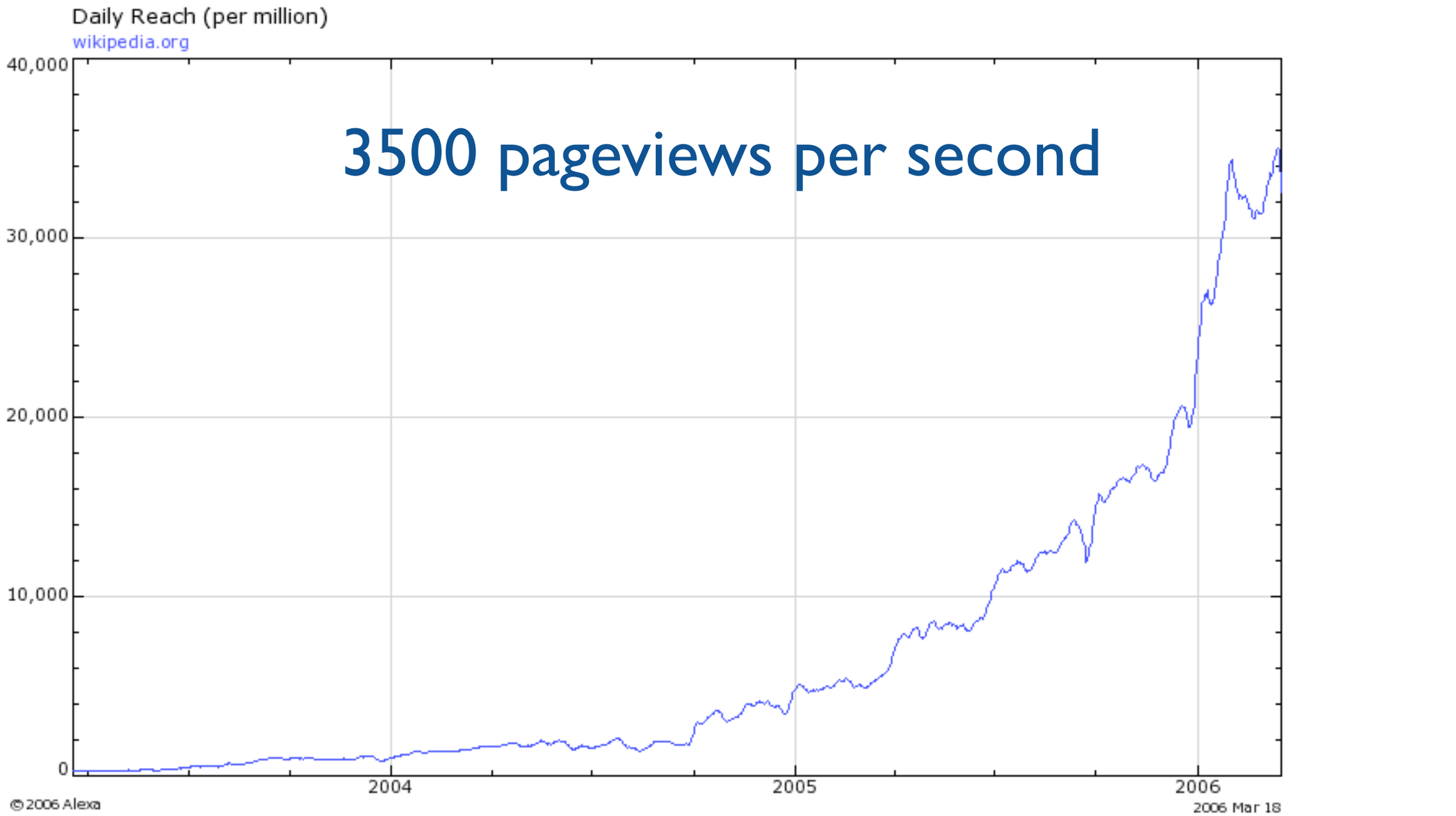
domas@mysql.com

Brion Vibber
Wikipedia

brion@wikimedia

Presented by MySQL O'REILLY

Growth



3500 pageviews per second

Experience

- It is:
  - Open
  - Entertaining
  - Educating
  - Free

Key components

- Mediawiki on top of:
  - Linux
  - Apache
  - MySQL
  - PHP

Key components cont.

- As well as with:
  - Squid
  - Memcached
  - Lucene (on Mono)
  - lighttpd

Server ‘optimization’

- 90% of memory in InnoDB buffer pool
- Unsafe configurations:
  - Rebuilding server doesn’t take too much time
  - Servers do not crash often (they should not at all!!!)
  - Didn’t see any mysqld crashes for years
  - RAID0 -> RAIS (Inexpensive? Instable? :)
  - flush logs=0

Numbers (English Wiki, 30%)

Table	Rows	Data	Index
revision	25m	8g	11g
page	4m	0.5g	0.5g
pagelinks	45m	4.5g	3g
text	25m	0.3t	-
watchlist	8m	1g	9.5g
user	1m	1.2g	60m
recent	5m	1.3g	1g

Other data...

- multiple links tables
- media
- logging
- job queues
- cached reports
- profiling

Data is...

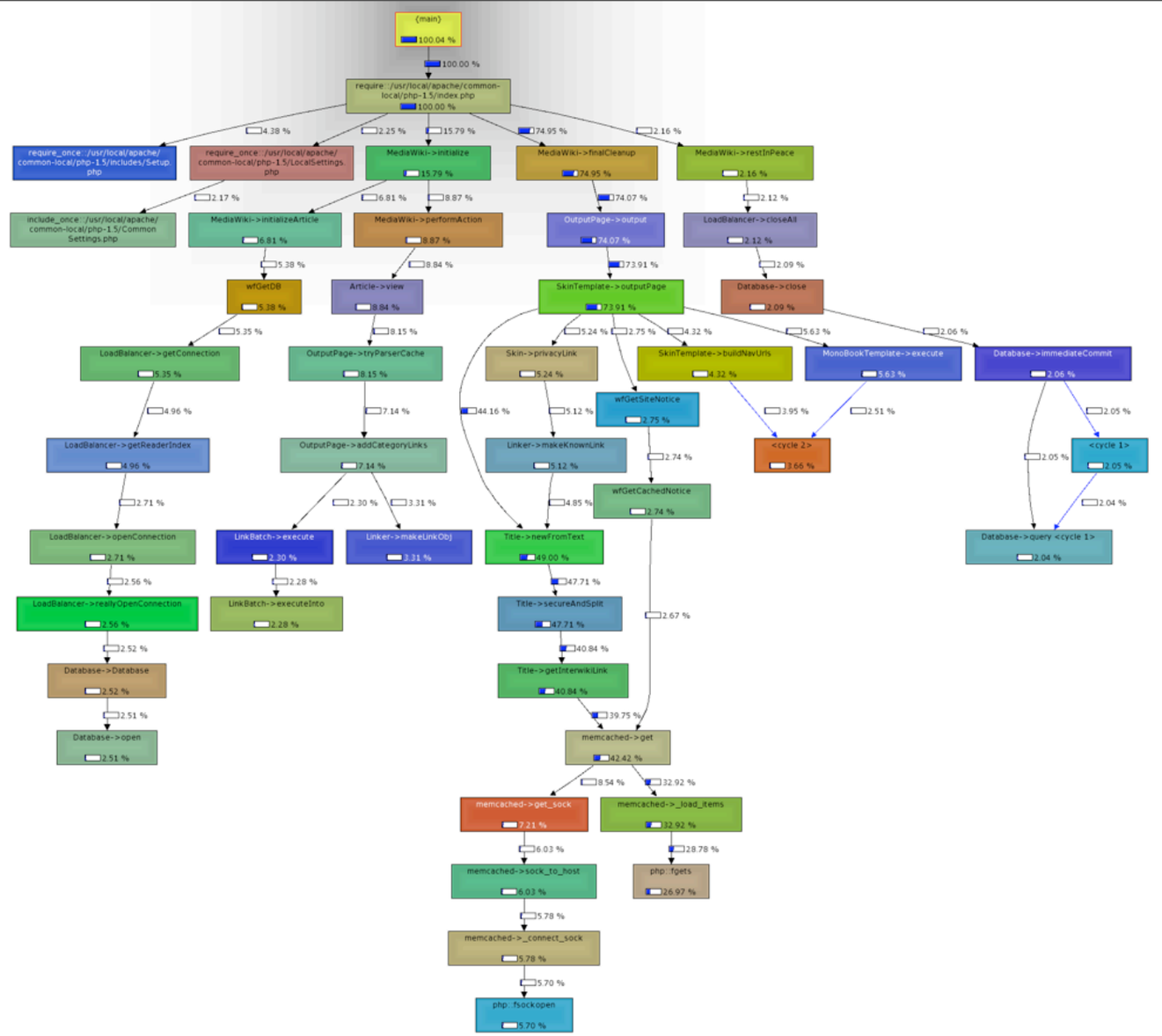
- Aggregated
- Compressed
- Duplicated
- Replicated
- Distributed
- **Used**

Operation guidelines

- General availability (as opposed to High)
- Maximum efficiency (donation powered)
- Always scale (there always exists at least one performance bottleneck)
  - If it does not - more users will come

Profile!!!

- gprof
- xdebug
- KCacheGrind
- Profiler.php
- udpstats
- Premature optimization is root of [nearly] all evil



Contraints

- App servers
  - CPU
- Database servers
  - RAM
  - I/O
  - Disk space

Slow things get killed

(or be killed yourself)

- Querybane - a workaround for overloaded site
- The best way to optimize a request - kill it
- Or kill some data
  - Using shortened aggregated tables for reports - doubled overall performance
- Or kill the feature
  - Some metadata included in every page was simply too expensive to generate

Other: split

- Splits:
  - save memory
  - increase cache hits
  - distribute load
  - optimize access

Split #1: slavery

- Move read load to replicated slaves
  - write load remains same
  - read operations scale!
  - MASTER_LOG_POS() keeps bits in sync

Split #2: Babel

- Splitting by language:
  - Improved cache hits
    - Much faster servers (2x-4x)
    - Can reuse outdated database hardware
  - Less storage needed
  - Inter-language code becomes complex
    - RPC instead of cross-db queries

Split #3: guilds

- Specialize tasks:
  - Search: Lucene (real time search is too expensive for us)
    - Saves DB for other tasks
  - Texts: MySQL instances on AppServers with lots of cheap SATA storage)
    - No bulk data on our processing machines
  - Other: possibility to offload specific tasks to specific servers

Split #4: Delay

- Use job queues for large operations
- Serial execution of queue sustains performance
- Move small constant often used data to application servers (as on-disk hashes)
  - Saved up to 30% request time in some cases

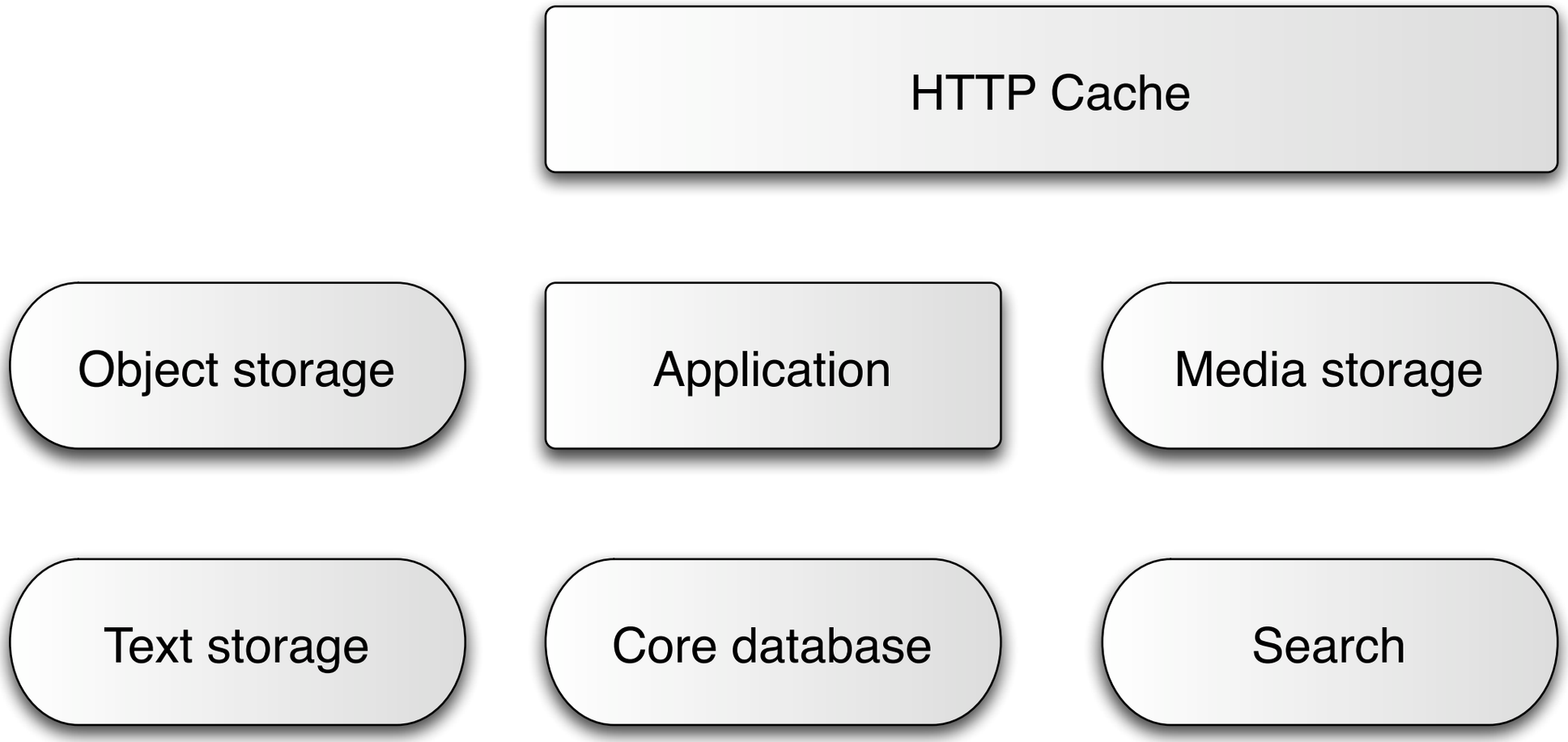
LoadBalancer.php

- Flexible configuration - PHP code
- Has per-database weights (different servers handle different languages)
- Has per-task weights (e.g. watchlists could be served by a specialized slave)
- Handles replication failures (caches replication lag information)
- Handles overloaded servers (SHOW STATUS before use)

Backups

- Dump XML of aggregated content
- Reuse old dump
- XML streams are easy to handle
- SQL dumps for internal data
- Bulk data handled outside from transaction
- READ UNCOMMITTED isolation

Bigger picture



Object cache

- Unreliably store post-processed data:
  - User objects and sessions
  - Parsed texts
  - Diffs
  - Environment state
  - Memcached, Tugela, in future possibly NDB
  - Runs on Application servers (2gb per host)

HTTP Cache

- Squid, LRU
- A fork (upstream development status unclear)
- Multicast object purges (HTCP CLR)
- Application heavily relies on it
- Distributed - Europe, Asia, US
- PowerDNS with our geo-backend used
- Serves 70% requests from cache

Media storage

- SPOF, a box with lots of disks
- lighttpd runs HTTP, unbelievably fast!
- Gearing towards distributed..
  - With coordination on MySQL..

Application

- MediaWiki
- Our in-house developed application, used by thousands of other sites
- PHP, some bits in C/C++
- Additional bits: Python, C#, Boo, JavaScript, ...

Database.php

- API abstraction
- MySQL optimizations (batched inserts, ...)
- Query builder (now seen in other frameworks)
- Injections generally avoided
- Queries tagged with /* Marks */
- Easy coding
- Do not treat SQL as text

Compression

- Concatenate, then compress
- Decompressed blobs kept in memory for cache
- Compresses up to 10%
- Reduced I/O on blob dataspaces
- Possible use of UDF

Biggest scaling problem

- People
  - System grows, people get scared
  - Volunteer vs pay issue
  - Hobby
  - US, Australia, UK, Netherlands, Lithuania, Iceland, Germany

Run, Wiki, run!

- >9000 HTTP requests per second
- >30000 SQL requests per second
- 12 db servers
- 15 app servers in ‘external storage’ role
- 20 app servers in ‘object cache’ role
- Peak #13 spot in Alexa

Questions?

- Now or..
- domas@mysql.com
- domas on Freenode, #wikimedia-tech
- http://dammit.lt/