mysqlnd
A new æra

PHP Vikinger
Skien, Norway

June, 9th 2007

Andrey Hristov <andrey@mysql.com>

Who is talking to you?

Andrey Hristov

- works for MySQL as a Software Developer
- MySQL Connectors departement
- author of MySQL Event Scheduler (magister exam)
- hacks PHP since ages
- has developed most of mysqlnd
- takes care of ext/mysqli together with Georg Richter

- ... tries to answer your questions – just ask!
- ... replies to (some) mails send to andrey@mysql.com

What does „mysqlnd“ mean?

MySQL Server

- **The world's most popular open source database**
- '**LAMP**' (Linux, Apache, **MySQL**, **PHP** / Perl / Python)
- http://dev.mysql.com/

mysqlnd - MySQL native driver for PHP

- **Native driver to connect from PHP to the MySQL Server**
- Tightly integrated into PHP, able to support PHP goodies
- **Not a new programming API**
- http://dev.mysql.com/downloads/connector/php-mysqlnd/

What is mysqlnd?

Library and not a new extension

- not exposed to the userland
- implemented in C

Drop-in replacement for libmysql

- works with existing software
- almost: some experimental functions are no longer available

Compile-time switch for ext/mysqli

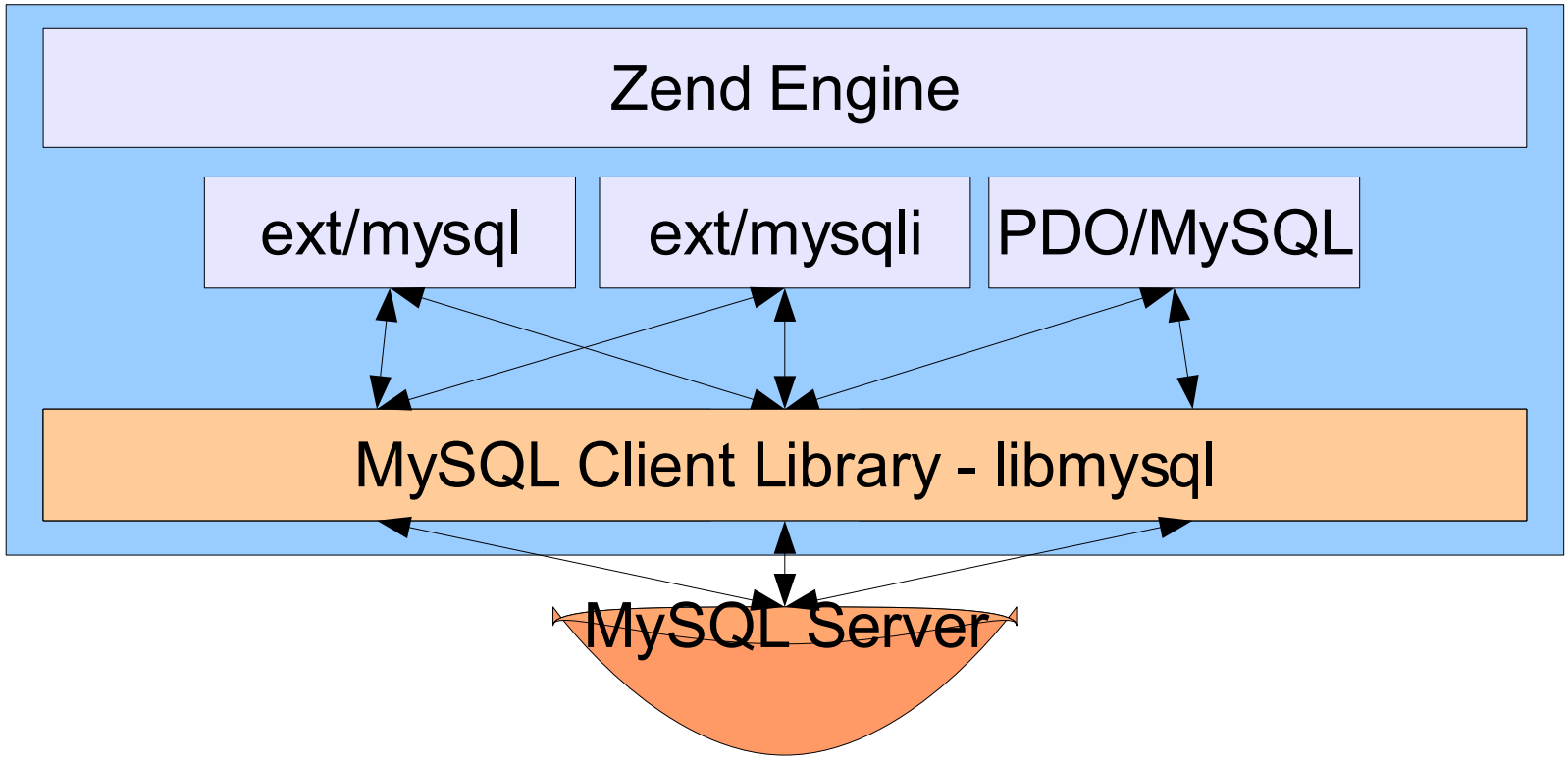
- ext/mysql support in the development tree
- PDO/MySQL support planned

PHP version independent: PHP 5, PHP 6

- theoretically even PHP 4, you might be able to port it!

How PHP connects to MySQL

- MySQL Client Library option remains, libmysql is not deprecated
- You can „mix“ mysqlnd and libmysql in one PHP binary
- PDO/MySQL support is planned



Why mysqlnd? The pros (1)

GPL/Dual-License

http://www.gnu.org/copyleft/gpl.html

http://www.mysql.com/company/legal/licensing/

„Open Source and free. If you want closed source and earn money, share your win and pay for a commercial license“

BSD-Style

http://www.php.net/license

„Open Source. Use for whatever you need it for as long as you keep the license unchanged.“

FOSS License Exception

http://www.mysql.com/company/legal/licensing/foss-exception.html

- MySQL Server
- MySQL Client Libraries
  e.g. libmysql

♥ PHP

FOSS License Exception not needed!

♥ mysqlnd

Why mysqlnd? The pros (2)

Libmysql comes with the MySQL Server

- updating libmysql means updating the MySQL Server
- bound to the server release cycle

mysqlnd is a library (stored in the ext/mysqli directory)

- upgrading, e.g. after a bug fix is simple
- has its own release and development cycle
- no longer linking/compiling issues
- no need to install libmysql on the build server
- leaner code

Tightly coupled with PHP and Zend Engine

- can do magic to get more juice out of your CPU(s)
- better integration into PHP (see below)

What are the cons?

Less tested than ext/mysqli - libmysql combo

- we have more than doubled the number of tests and pass them all
- we have even found libmysql bugs with the new tests
- we have done mainly CLI tests no web tests
- we have no/few user feedback

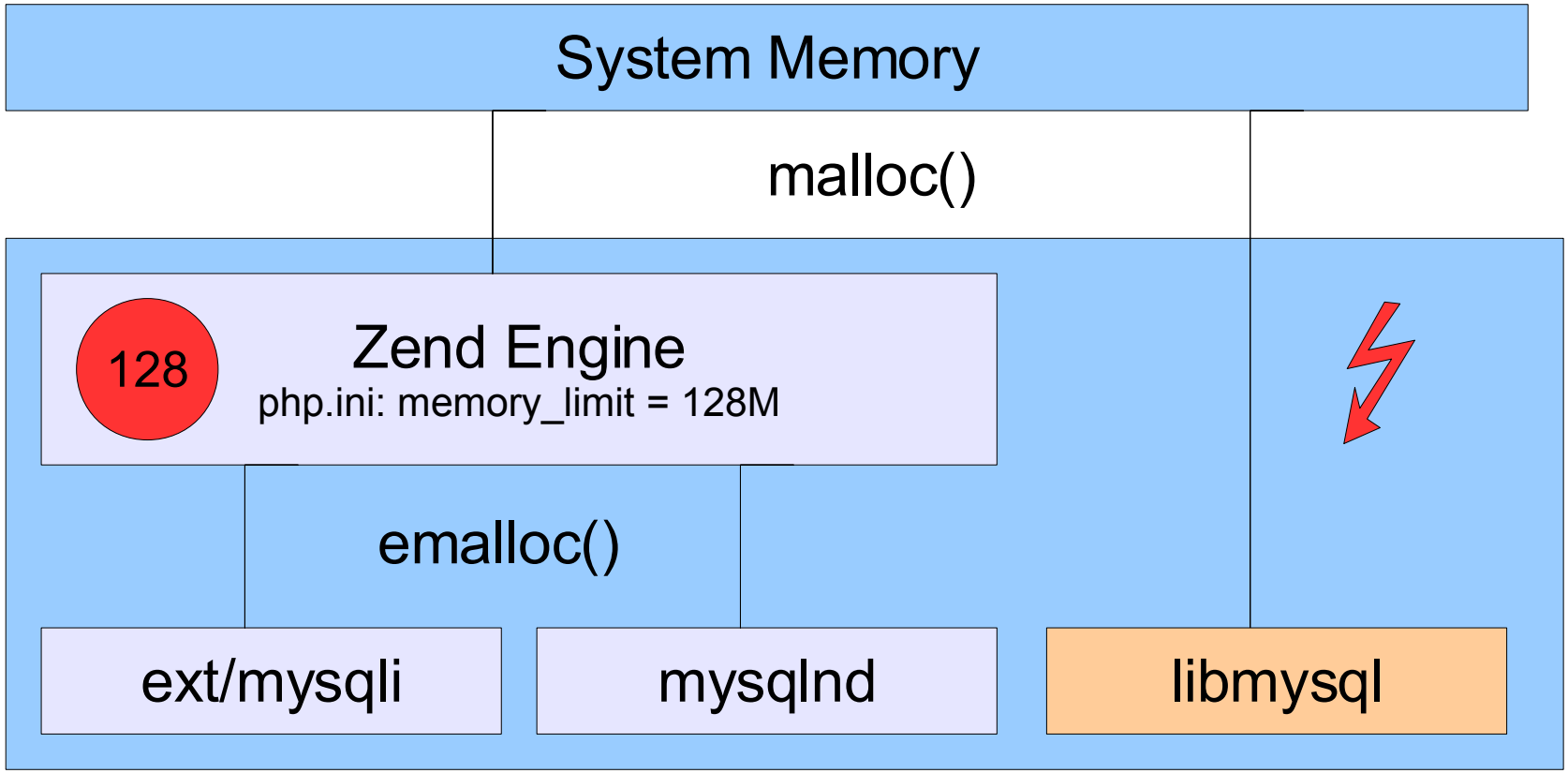
Some (experimental) functions are no longer available

- `mysqli_embedded_*`
- Replication related, e.g. `mysqli_enable_rpl_parse`

Good old times are gone...

Abracadabra: memory limit

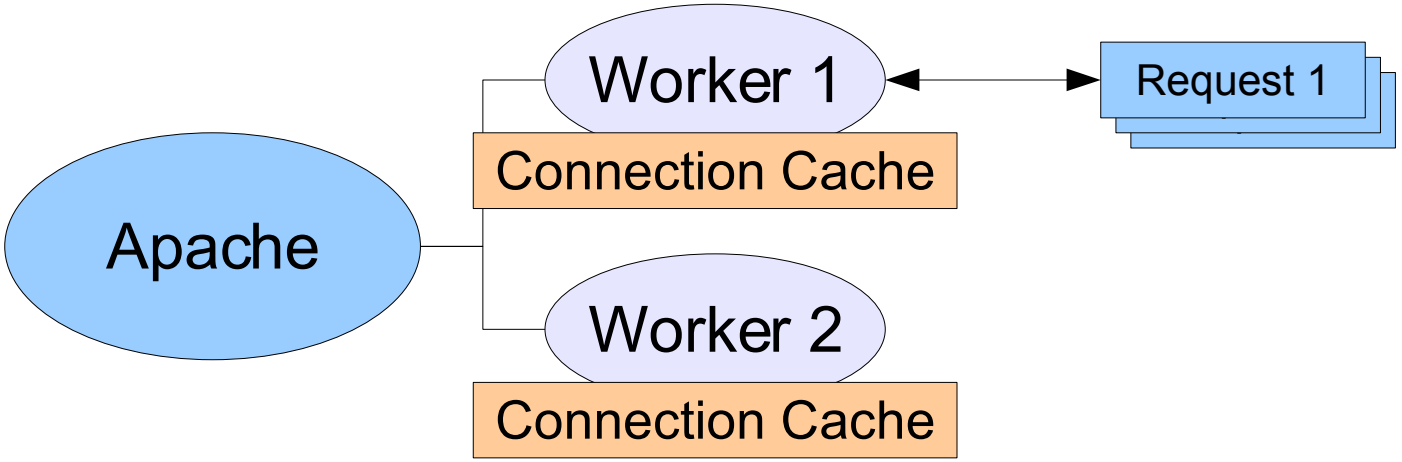
Memory limit now supported!



Abrecadabra: persistent connections

Persistent connections like ext/mysql and libmysql

- saves connection overhead: +5% for Dell DVD Store
- works with Apache prefork/threads and Fast-CGI
- Per process resp. per thread cache
- Future: optional flag to reset connections properly
- `mysqli_connect('p:localhost', ...)`



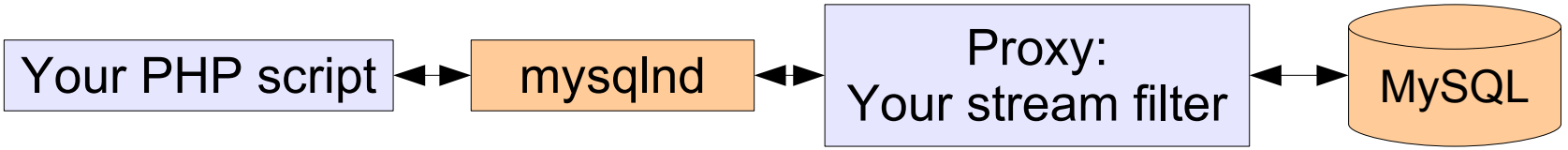
Abracadabra: PHP streams

mysqlnd uses PHP streams

- portable and robust
- **not implemented:** expose stream to users
- **theoretically**: you could implement a proxy

```
<?php
[...]
mysqli_query($link, 'SELECT ...');
[...]
```

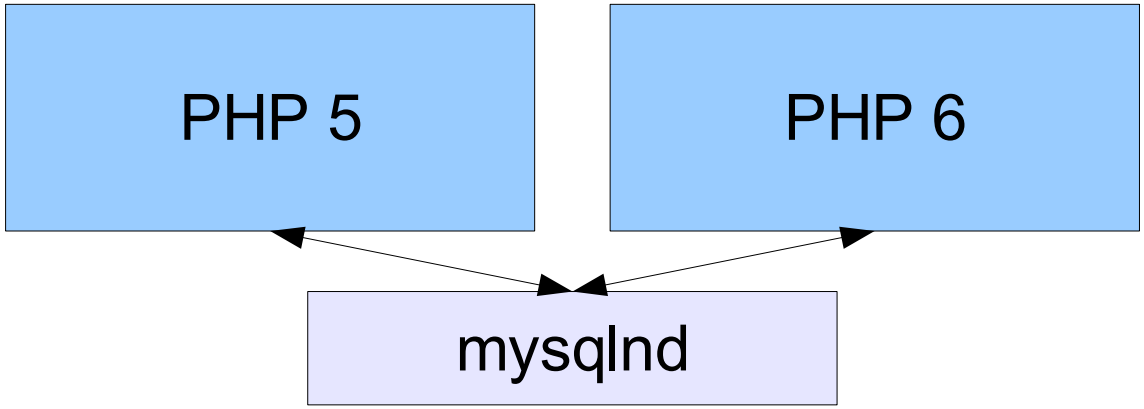
```
<?php
class auto_explain extends
  php_user_filter { [...] }

class load_balance extends
  php_user_filter { [...] }
```



Abracadabra: one code base

PHP 5 and PHP 6 use the same code base

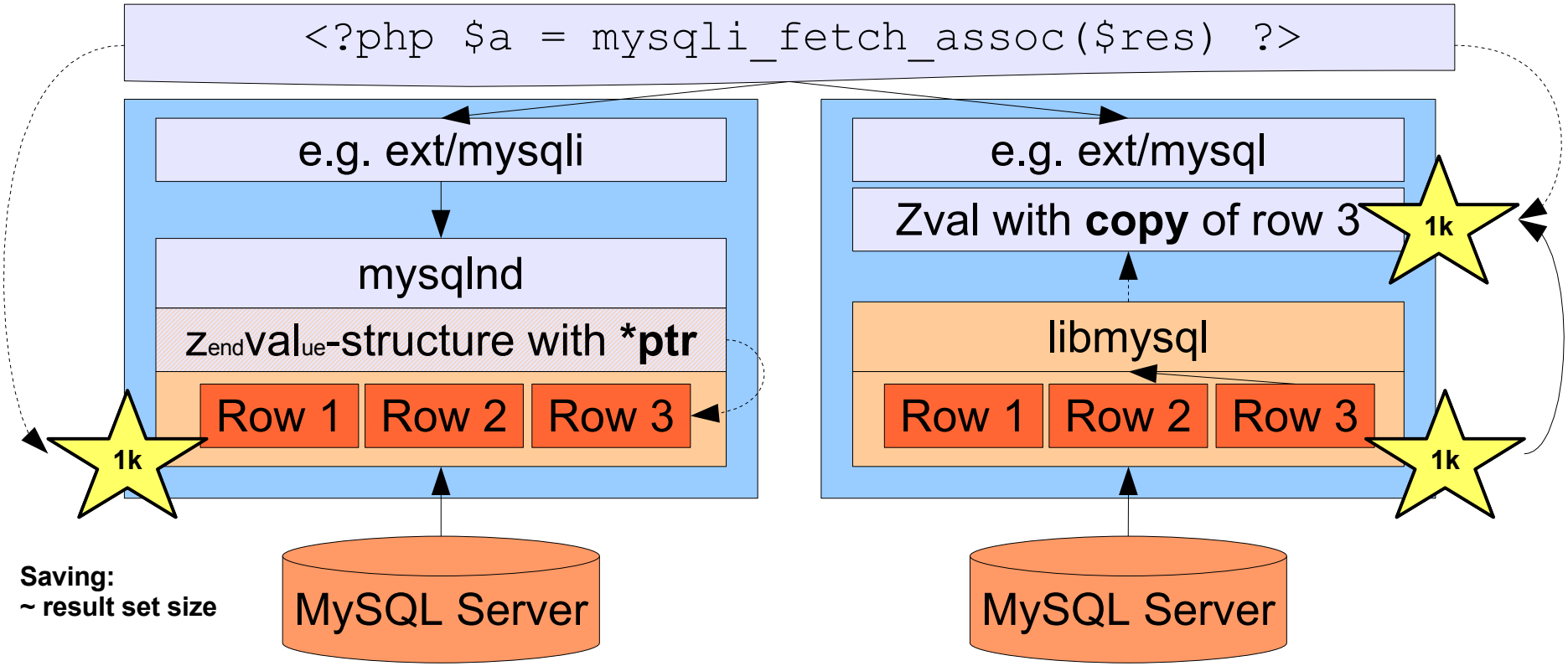
- all extensions use one code base of mysqlnd
- one fix, fixes it all (one bug, breaks it all)
- Unicode is compiled out for PHP 5
- nevertheless the code has only few #ifdef and is readable
- If php.net wants mysqlnd they can go for one or two branches



Abracadabra: memory savings

„Read-only“ variables before PHP offers them

- tries to hold results only once in memory: up to 50% saving
- often less operations to perform



Abracadabra: read-only variables (2)

What ext/mysqli (or ext/mysql, ...) used with libmysql does:

- mysqli sends a query
- (1) result set get fetched into libmysql buffers
- (2) mysqli allocates zvals, then new buffers
- (3) mysqli copies data from libmysql to its own buffers
- calls `mysql_free_result()` and deallocates libmysql buffers

What ext/mysqli used with mysqlnd does:

- mysqli sends a query
- (1) result set is fetched row by row, every row is different buffer
- (2) mysqlnd creates a result set of zvals, pointing to the buffers
- calls `mysqlnd_free_result()` which could be lightweight

Abracadabra: read-only variables (3)

What ext/mysqli (or ext/mysql, ...) with libmysql does:

- one extra allocation for mysqli buffers
- one extra data copy
- one extra zval allocation,
  which can be saved with the zval cache

Tricks and goodies of mysqlnd:

- mysqlnd does thread-safe zval caching -
  Zend Engine does this too, but we bring it to an extreme!
- mysqlnd records cache statistics (e.g. `phpinfo()` output)
- mysqlnd collects query statistics (more later)
- statistics are per process, even in a threaded environment

copy_on_write_saved	2002
copy_on_write_performed	194
command_buffer_too_small	0

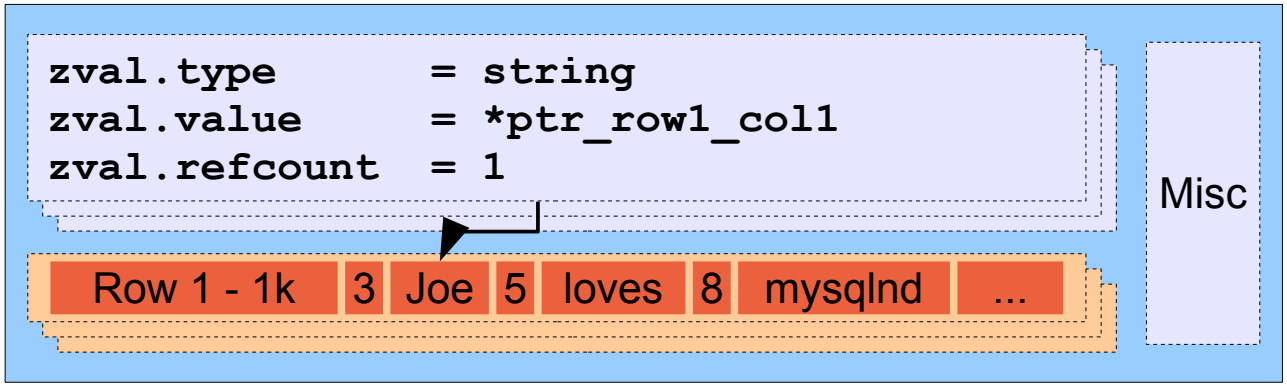
Read-only variable implementation

Result set consists of:

- 2D zval buffer
- result butter
- miscellaneous

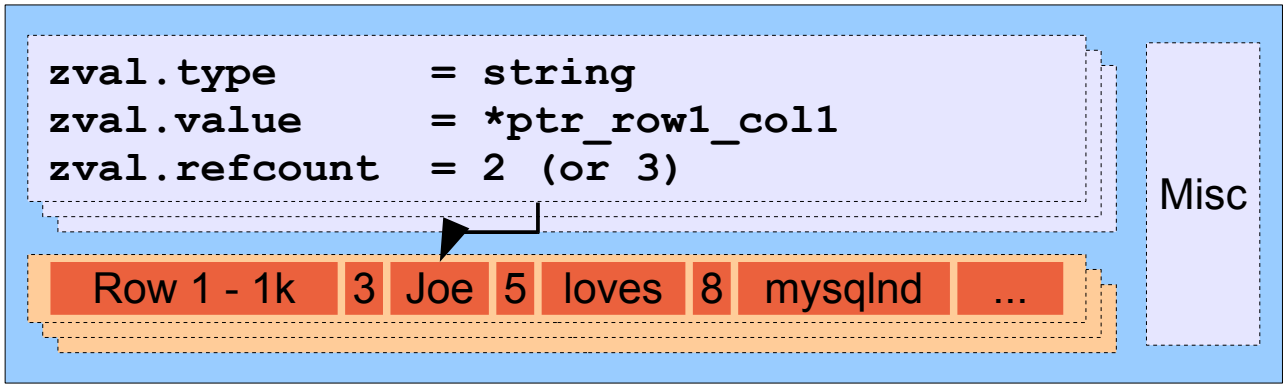
zval owned by result set:

- refcount = 1



Fetching data:

- *fast:* refcount++
- unlike libmysql always correct hash size and no rehashing
- `unset($row)`: refcount--



mysqli_free_result():

- Separate (zval_copy_ctor()) all zvals for which refcount > 1, for example:
  `$joe = $row['firstname']`

```
zval.type      = string
zval.value     = 'Joe'
zval.refcount  = 1
```

Abracada*sure?*: read-only variables

Memory allocators are synchronized

- synchronisation is slow
- happens even in single-threaded applications

mysqlnd allocates bigger chunks

- are we wasting memory?
- minimizes heap fragmentation, less work for the allocator

mysqlnd tries to reuse the zvals

- saves CPU cycles
- less memory fragmentation
- But: will you notice it?

Expertadabra: zval cache

- On module start-up the zval cache is initialized. Specific number of zvals, ini setting, are pre-allocated on a contiguous block of memory.
- When mysqlnd needs a zval, it asks the cache, if the cache has a free entry, it returns a pointer to one. Otherwise, allocates a zval.
- When a result set is freed, the zvals are destructed through the cache. If a zval is not from the cache normal ZE is called. Of course we need to separate values, if refcount > 1.
- If the zval is from the cache, efree() is not called, which means no critical sections in the allocator. The zval can be used again.
- If a cache zval had refcount > 1, it won't be returned for re-usage till the end of the script. Limitation, which can be lifted in further versions by using garbage collection on next free result.

Expertadabra: zval cache (2)

The global (per process) cache consists of:

- a pre-allocated zval block
- a free list stack
- a reference counter
- a mutex
- other variables for operation and statistics
- created at MINIT

The local (per thread) cache consists of:

- a garbage collection stack
- reference counter
- pointer to the global cache
- created at RINIT

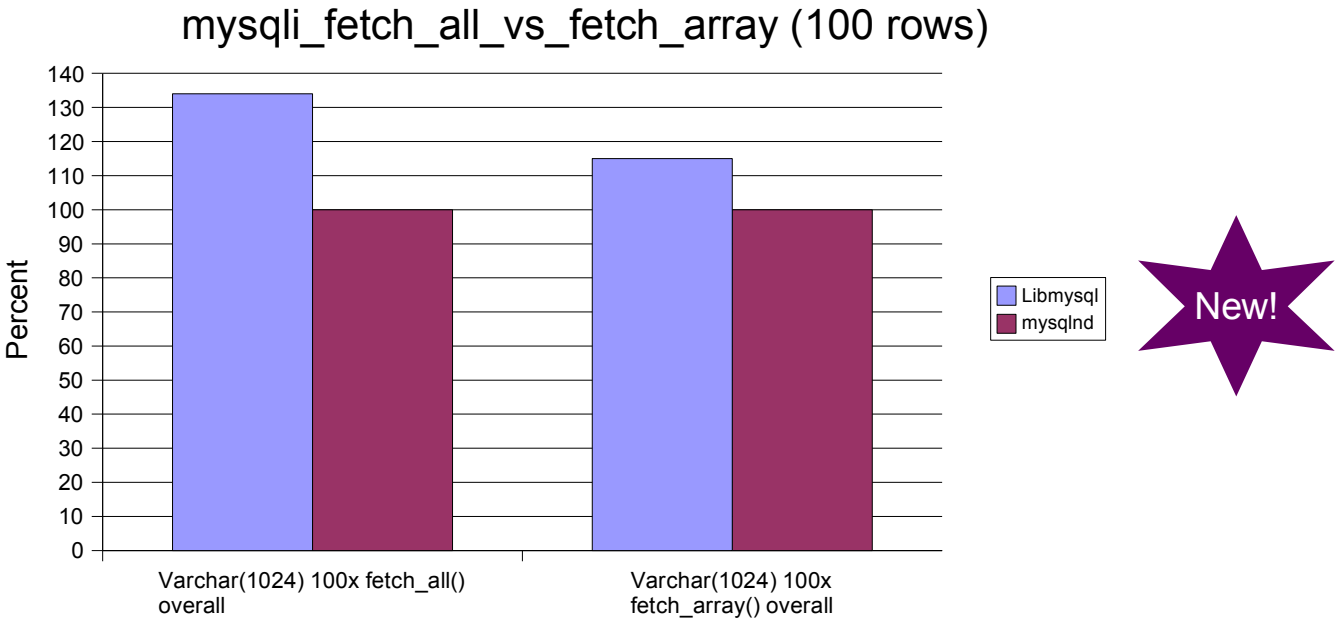
Expertadabra: zval cache (3)

- Because complexity of sequential scan of the block for finding free zval is O(n), the access time would grow with the cache size. Thus, an optimisation called free-list-stack was introduced.
- Free-list-stack holds pointers to free zvals, thus GET operation has O(1) complexity.
- PUT is also is also leight-weight because the zval has to be added to the free-list.
- The stack is organised backwards. New elements get lower memory addresses. Thus if there is a batch of GET operations the CPU will cache the line and the first GET will hit the DATA L1/L2 cache.
- Garbage collection at end of request is very cheap as those „stucked“ zvals are kept on a garbage collection list and only they are freed. This is the actual reason for splitting the cache into two parts – global and local (per thread).

Expertadabra: zval cache problems

- Recall that on mysqli_free_result() the result set has to be scanned and zvals with refcount > 1 have to be separated. However, as there is (are) user variable(s) pointing to a zval cache entry, we cannot perform a PUT operation. Thus, we are stuck with this till the end of the script.
- This means that actually with mysqlnd it's better if the user doesn't call very quickly `mysqli_free_result()`.
- This limitation can be lifted by trying checking the garbage collection list of the local cache for zvals with refcount = 1

Brand new: mysqli_fetch_all()




```
<?php
...
$res = mysqli_query('SELECT ...');
$users =
   mysqli_fetch_all($res);
...
?>
```

```
<?php
...
$res = mysqli_query('SELECT ...');
while ($users[] =
   mysqli_fetch_row($res)) ;
...
?>
```

Brand new: statistics

mysqli

Mysqli Support	enabled
Client API library version	mysqlnd 5.0.1-beta - 070402 - $Revision: 341 $
Client statistics	
bytes_sent	44495
bytes_received	69654
packets_sent	583
packets_received	1932
protocol_overhead_in	7728
protocol_overhead_out	2332
result_set_queries	145
non_result_set_queries	135
no_index_used	0
bad_index_used	0
buffered_sets	145
unbuffered_sets	0
ps_buffered_sets	0
ps_unbuffered_sets	0
flushed_normal_sets	0
flushed_ps_sets	0
rows_fetched_from_server	503
rows_fetched_from_client	503
rows_skipped	0
copy_on_write_saved	2002
copy_on_write_performed	194
command_buffer_too_small	0
connect_success	101
connect_failure	0
connection_reused	0
explicit_close	44
implicit_close	57
disconnect_close	0

More information:

- 40+ new figures
- simplifies bottleneck analysis
- easy monitoring

Simple to identify a script that:

- selects more rows than it consumes
- opens more connection than it needs, ...

Accessible through:

- `phpinfo()`
- `mysqli_get_client_stats()`
- `mysqli_get_connection_stats()`
- `mysqli_get_cache_stats()`

Brand new: statistics (2)

Network traffic

- Bytes send and received
- Packets send and received
- Protocol overhead incoming and outgoing

SQL related

- Result set queries (SELECT)
- Non result set queries (all other)
- Number of queries not using indexes
- Number of queries using bad indexes

Result set related

- Number of buffered and unbuffered sets
- Number of flushed sets
- Rows fetched from Server
- Rows fetched from Client
- Rows skipped
- Copy on write saved, write performed

Result set related (continued)

- Explicit free result (e.g. mysqli_free_result())
- Implicit free result (e.g. end of script)

Connection related

- Connect success and failure
- Connection reused (persistent connections)
- Explicit close (e.g. mysqli_close())
- Implicit close (end of script)
- Disconnect and close (server disconnect)
- Disconnect during command exec. (network)

Prepared Statements related

- Number of buffered and unbuffered result sets
- Number of flushed result sets
- Explicit close (e.g. mysqli_stmt_close())
- Implicit close (end of script)

... and more: command buffer, qcache...

Brand new: the statistics API calls

Statistics from the previous slide are available:

- Per script:
  `mysqli_get_client_stats()`
- Per connection:
  `mysqli_get_connections_stats(link mysqli)`

zval cache Experticadabra can be monitored with:

- `mysqli_get_cache_stats()`
- Put hits and misses
- Get hits and missed
- Size
- Number of free (unused) items
- Number of references

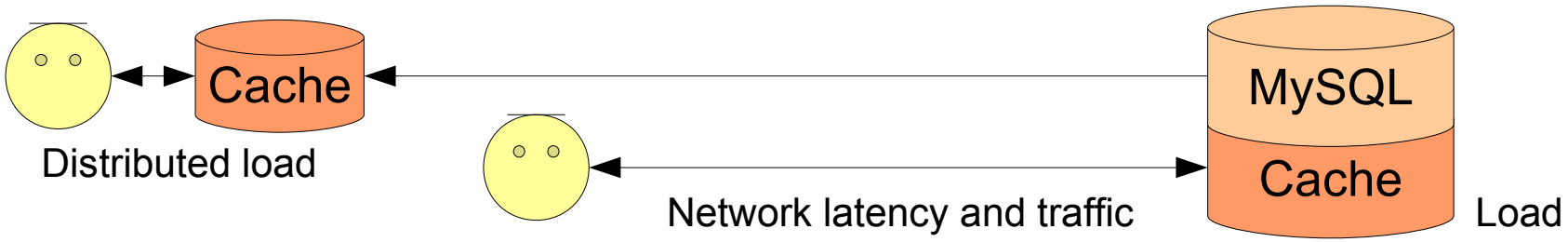
Brand new: client-side query cache

Did Rasmus ever ask for it, did he say „br...d..th“?

- High load? You start counting system calls
- High load? You die without remote/server caches
- High load? You die without network caches
- High load? You die without client-side cache

Client- vs. Remote/Server cache

- Remote/Server cache is often central: one for all clients
- Remote/Server cache means: load on the (one?) server
- Remote/Server cache still causes: network traffic and latency
- Client-side cache: entries can be tricky to invalidate/synch



Experimental: TTL-based

Experimental implementation in the SVN

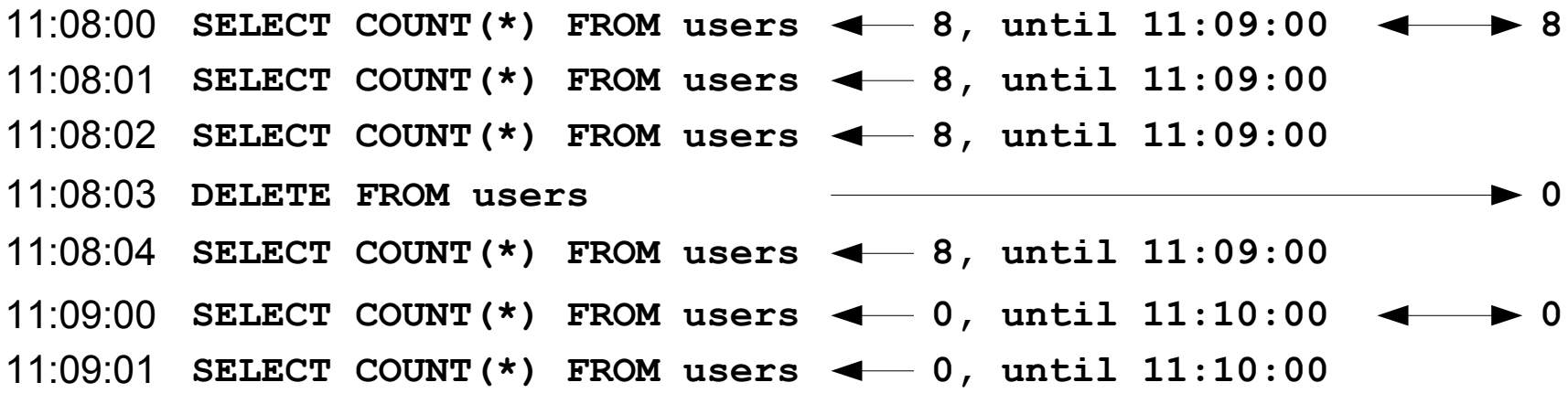
- not part of the released version, experimental, will change!
- Time-to-live (TTL) invalidation: stale data can be delivered
- On the API-level: simple to use, no extra tools needed
- Customizeable storage (memcached, file, ...) planned
- `mysqli_query(string query, boolean cache)`
- `php.ini: mysqli.qc_hash_buffer_size = <kB>`
- `php.ini: mysqli.qc_ttl = <seconds>`

Fast: 'SELECT 1' takes 6x less!

- Saves communication overhead
- Saves memory allocation overhead
- On `mysqli_free_result()` the result set gets optimized for quick cache access

Time-to-live (TTL) properties

- TTL is simple to implement and to use
- Returning stale data is possible



- MySQL (Server) Query Cache never serves stale data
- MySQL Query Cache invalidates on a per table basis
- We dream of – **no promise** we ever make it – per table and per row invalidation

Abracadabra: Performance

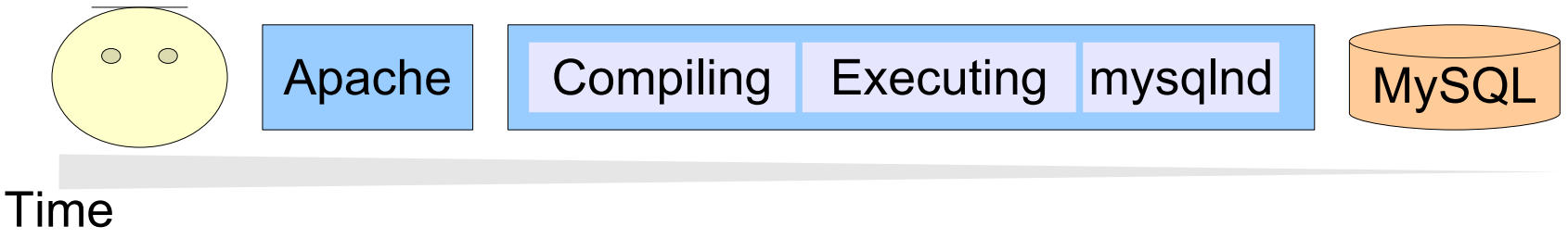
Benchmark your particular application yourself!

The API impact in a web setup can be very small

- it makes no sense to optimize something down from 2% to 1%

Excellent micro benchmark results can melt down

- Let overall time be 10s and the time spend in libmysql 0.1s
- Let mysqlnd be twice as fast (200%) as libmysql: 0.05s
- You get down from 10s to 9.95s, that's only 5% faster!



Abracadabra: Dell DVD Store

OLTP application with small result sets

- Online shop simulation, uses transactions, reminds of TCP-C
- Project page: http://linux.dell.com/projects.shtml#dvdstore
- Used for a contest in which **LAMP** had by far outperformed others
- C't contest: http://firebird.sourceforge.net/connect/ct-dbContest.html
- Contest submission: http://www.heise.de/ct/dbcontest/teilnehmer.shtml
- Not perfect for mysqlnd, because of small result sets

We found for the contest submission:

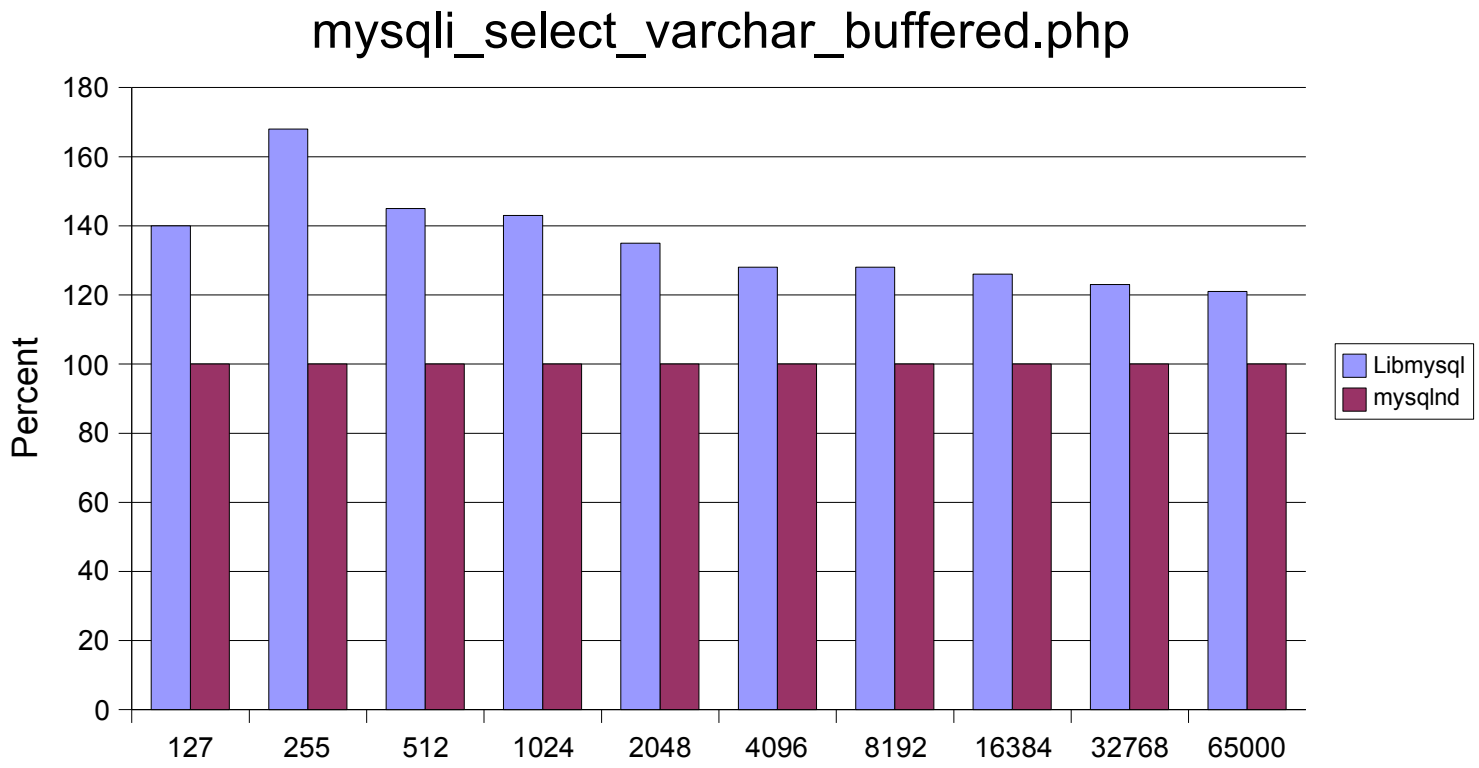
- ext/mysqli with mysqlnd: fastest
- ext/mysqli with libmysql: 6% slower
- ext/mysql with libmysql: 3% slower

mysqlnd is about as fast as libmysql, sometimes faster

Sometimes faster?

Lower bars are faster, libmysql is blue, mysqlnd is purple

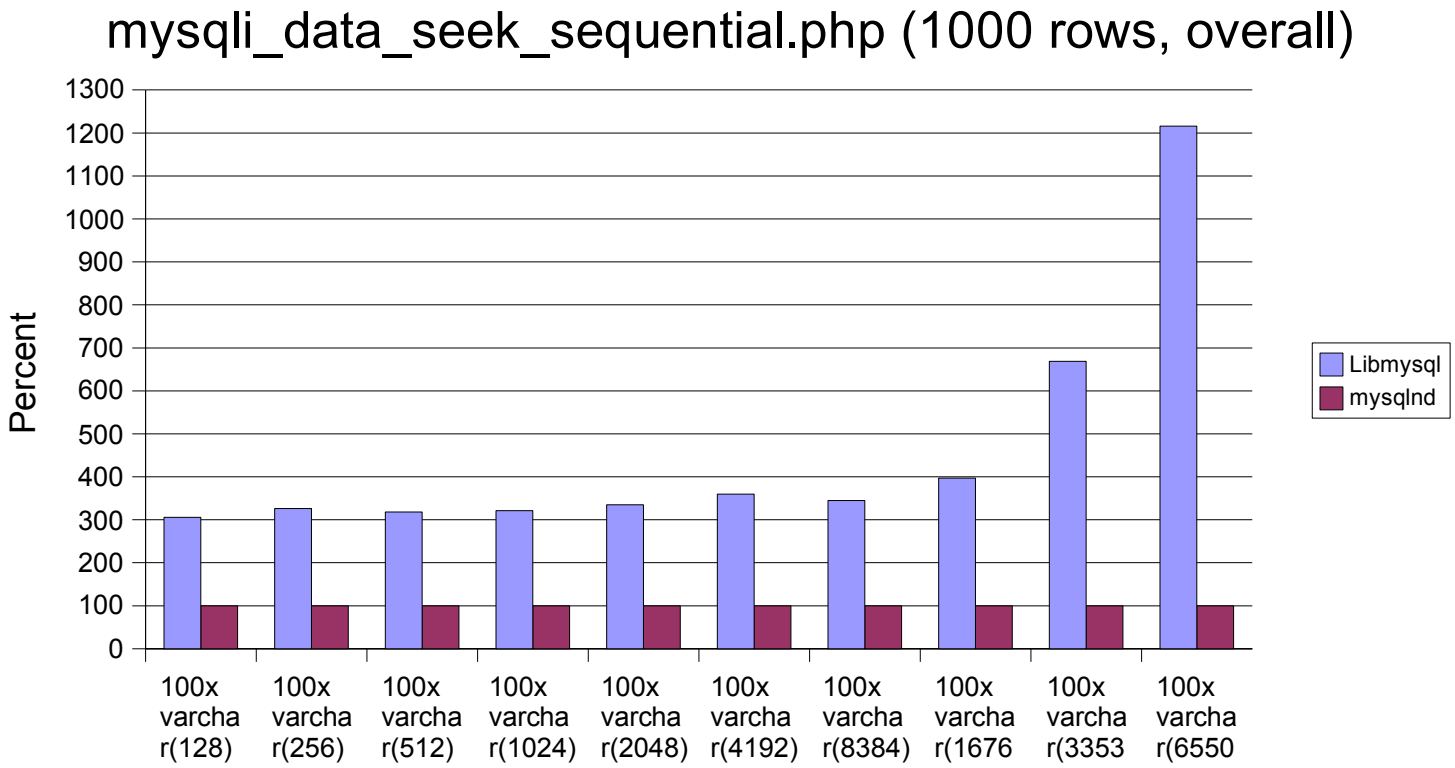
Fetching 100 rows with one varchar column of a variable size. Data shown for100 rows, varchar(n), 100x overall.



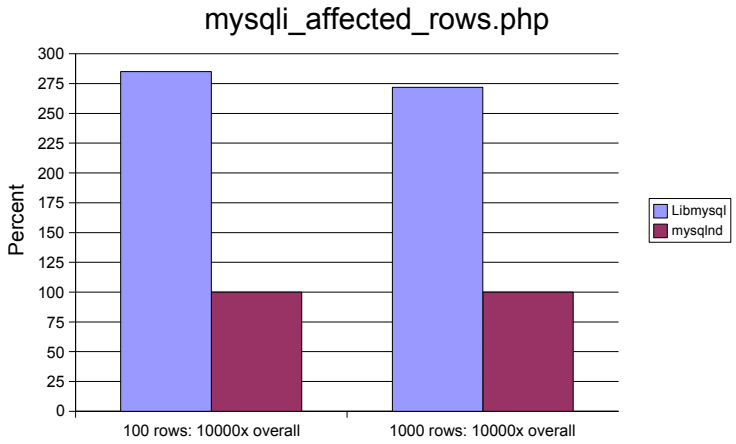
We like this micro benchmark chart...

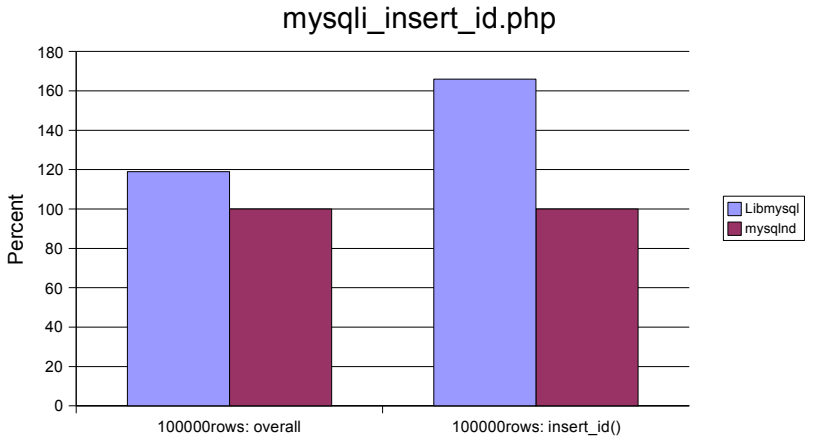
Lower bars are faster, libmysql is blue, mysqlnd is purple

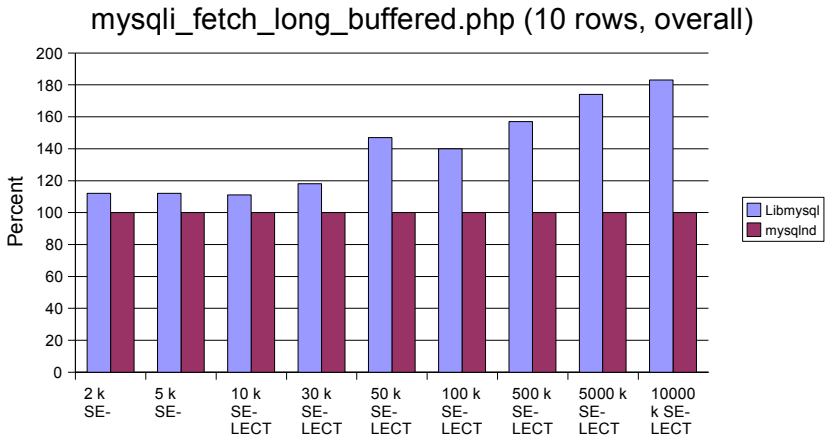
Connect, create 100-rows with varchar(m), SELECT all, 1000x times mysqli_data_seek(mt_rand()), close.

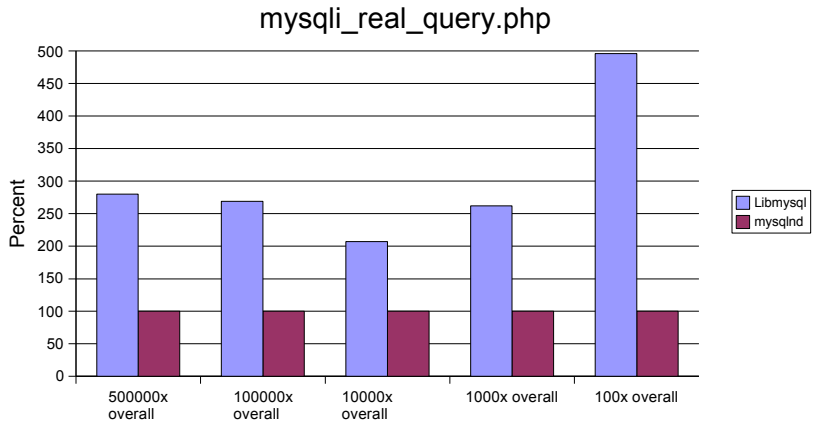


These are also acceptable...









But....

... these are micro benchmarks

... recall that the impact of the API can be very small
... libmysql and mysqlnd run thousands of op. per second
... from 0.0001s to 0.00003s often makes no difference

... we know we made it faster, th
check twice if you reall
performance improvem

Typically we found:
-5 to +15%.
But really no promise!

... if you really want more charts: we are preparing a 30 pages performance document

Roadmap

- Complete the work – be there in time for PHP 6
  - SSL support
  - Compressed protocol support
  - write documentation, educate users
- Support PDO/MySQL
  - Nobody has volunteered for the task...
  - ... looks like we have to do it
- Move development and source to php.net
  - Of course: only if the PHP fellows want mysqlnd on php.net
  - Of course: not before its stable enough not to break PHP...

The future is yours!

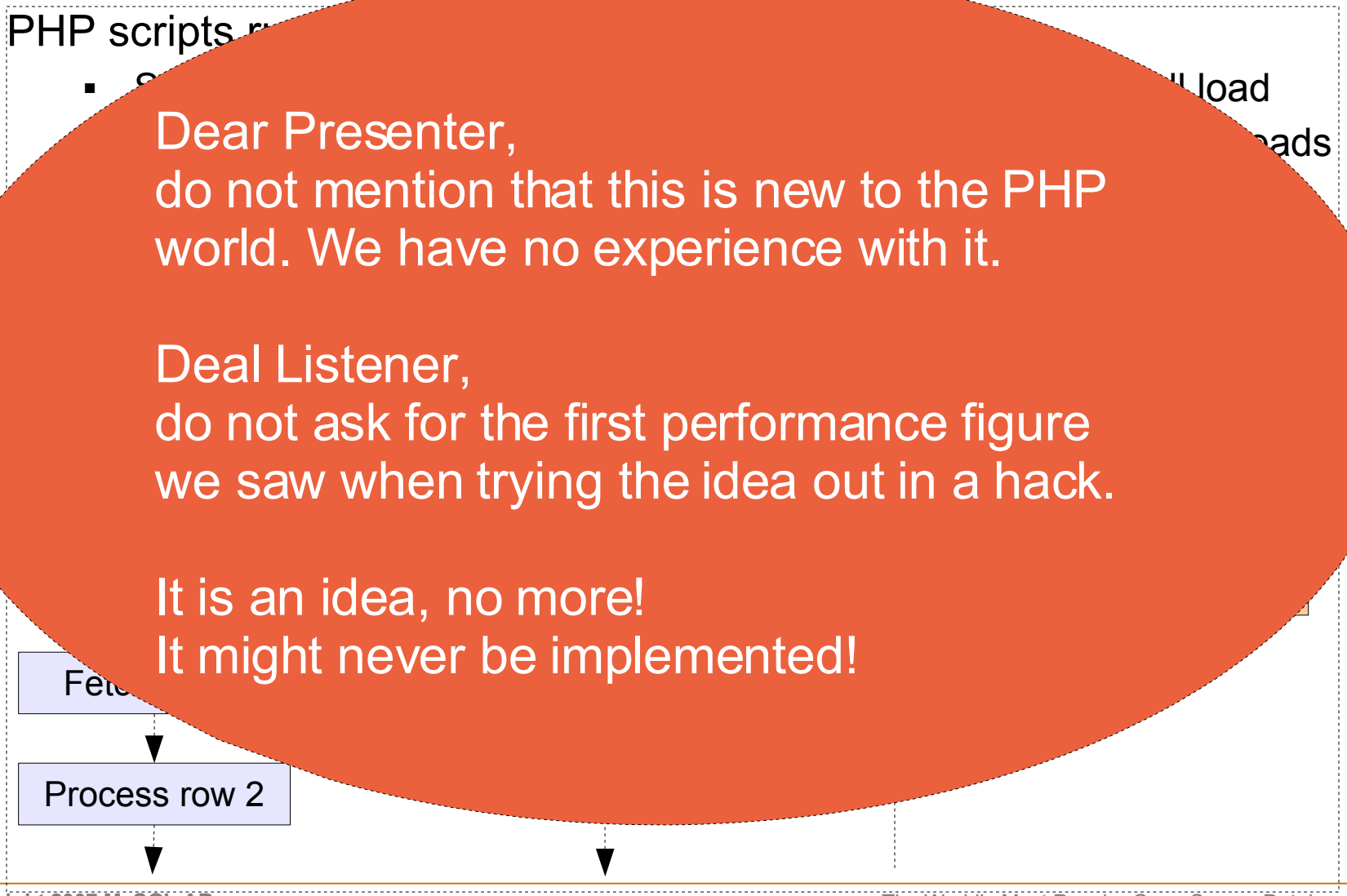
Your wishes are:

- Hands up, please!
- There is plenty of room on this slide!
- We can add more slides with your ideas and wishes!

Possible futures...

- Prepared statements
  - Cache handles: prepare is expensive, execute is fast
  - Similar to persistent connections or zval-cache
- Clever client-side query cache
  - TTL-based invalidation as a default
  - Remote and configurable storage of cache entries
  - Dream: MySQL Query Cache style invalidation
  - Dream: row-level invalidation
- Expose PHP streams to users
  - Transparent Proxy
  - Translating Proxy

Idea to try: background fetch



Dear Presenter,
do not mention that this is new to the PHP world. We have no experience with it.

Deal Listener,
do not ask for the first performance figure we saw when trying the idea out in a hack.

It is an idea, no more!
It might never be implemented!

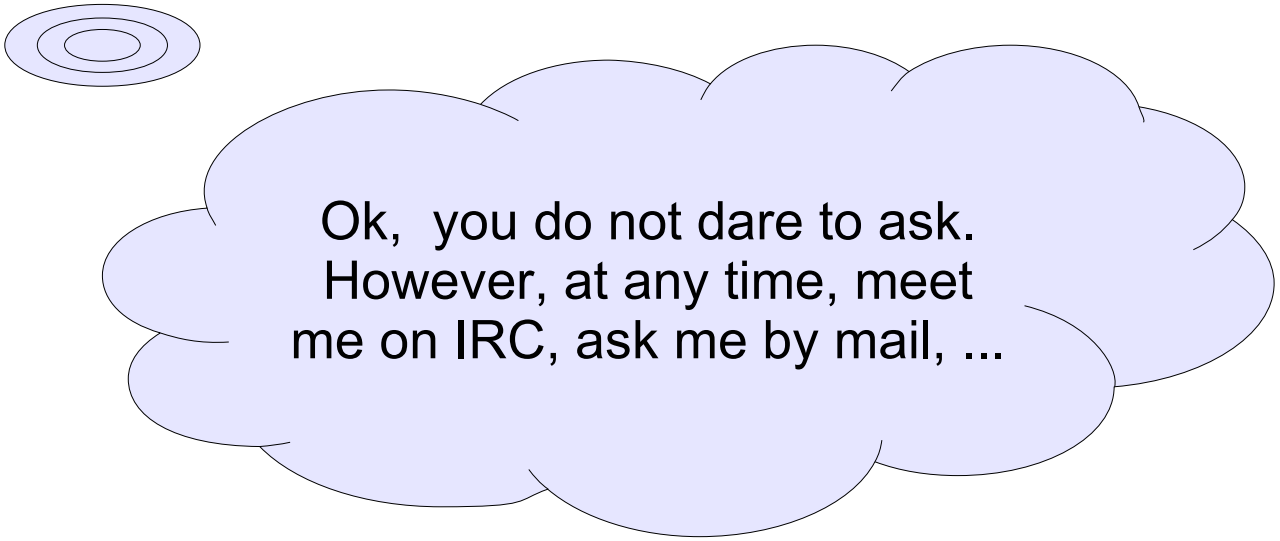
Been there, done that! Who to blame?

- Georg Richter
  - Management and development
  - If you want to deal with us, he is yours...
  - georg@mysql.com
- Andrey Hristov
  - Architecture and development
  - If you need the first-hand hacker information, I'm yours
  - andrey@mysql.com
- Ulf Wendel
  - QA, Documentation
  - The dirty work which brings sweet and tasteful fruit
  - uwendel@mysql.com

Some randomly selected Contributors

- Lukas Kahwe Smith
  - API ideas and more...
  - http://pooteeweet.org/
- Peter Zaitsev
  - Statistics, background fetch, benchmarking hints and more...
  - http://www.mysqlperformanceblog.com/about/

Questions ?



Feedback, please!

Average – Good – Excellent – Hands up!

- My overall rating for the talk is...
- This talk was too difficult to understand...
- This talk did not give enough technical information...

If I were you, I would change this... – raise your voice!

Thank you and good bye!