MySQL Connectors
PHP, C/C++, C/OOo

Ulf Wendel
Senior Software Engineer
Sun Microsystems

Das MySQL Connectors-Team

1 Teamleiter, 9 ½ Stellen für Entwickler, 10 Entwickler, 3 Länder (Deutschland, Ukraine, USA), 5 Zeitzonen, 6 MySQL Produkte, Aktive Betreuung von Community-Treibern, häufige Releases

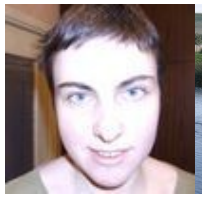
















Die MySQL Connectors

Name	Version	Status
Connector/ODBC	٣.٥١/٥.١.٥	GA – Duale Lizenz
Connector/J	٥.١.٦	GA – Duale Lizenz
Connector/NET	٥.١.٦	GA – Duale Lizenz
Connector/MXJ	١.x/٥.٠.٩	GA/Beta – Duale Lizenz
Connector/C++	١.٠.٠	Preview
Connector/OpenOffice.org	١.٠.٠	Preview
PHP: ext/mysql, ext/mysqli, PDO_MYSQL		Community – PHP Lizenz

MySQL Connector/ODBC

- 3.51 GA – ehemals MyODBC
- 5.1 GA – der partielle Neuanfang
  - MySQL >= 4.1
  - ODBC-Core und Teile von Level 1 und Level 2
  - Unicode im Treiber, SQL_WCHAR Datentyp
  - Verbesserung der Windows 64bit-Unterstützung
  - Neuer Windows Installer
  - SQL_NUMERIC_STRUCT für genaue Infos zu numerischen Werten
- Jim Winstead, Jess Balint

MySQL Connector/J

- 5.0 GA
- 5.1 GA
  - MySQL >= 4.1
  - Type 4 pure Java JDBC Driver
  - JDBC 3.0, JDBC 4.0
  - JDBC 4.0 XML
  - Verteilte Transaktionen: XA-Support
- Mark Matthews*, Eric Herman*

MySQL Connector/MXJ

- 1.x GA
- 5.0 Beta
  - MySQL Server als Java-Komponente
  - Verwendbar als “plain old Java object” (POJO), Connector/J Socket-Fabrik, JMX MBean
  - 5.0.9 – MySQL 5.0.51a Community/MySQL 5.0.54 Ent.
  - Linux (x86), Mac OS X (x86, PPC), Microsoft Windows (x86) Sun Solaris (SPARC, x86)
- Eric Herman*, Mark Matthews*

MySQL Connector/NET

- 5.0 GA
- 5.1 GA
  - MySQL >= 4.0
  - Fully compatible ADO.NET driver interface
  - ADO.NET 2.0 Interfaces and Subclasses
  - Compact Framework 2.0
  - Visual Studio Plugin
- Reggie Burnett, Eric MaLossi

MySQL Connector/C++

- 1.0 Preview
  - C++ - API anstelle von MySQL C-API
  - Unterstützung der objekt-orientierten Programmierung
  - Orientierung am JDBC 3.0 Standard
  - 335 Methoden (75%) implementiert aus den Klassen: Connection, DatabaseMetaData, Driver, PreparedStatement, ResultSet, ResultSetMetaData, Savepoint, Statement
  - Kommerzielle Lizenz und Support angedacht

- Andrey Hristov, Lawrenty Novitsky

MySQL Connector/OpenOffice.org

- 1.0 Preview
  - > OpenOffice.org 2.4 Extension
  - > keine Notwendigkeit Connector/ODBC oder Connector/J zu installieren und zu konfigurieren
  - > Zukunft: OpenOffice.org 3.0
- Andrey Hristov, Georg Richter

ext/mysql, ext/mysqli, PDO_MYSQL

- GA – alle
  - eine der ersten DB-Schnittstellen für PHP x.y
  - erste hybride (prozedural und OO) Schnittstelle
  - eine der ersten Unicode = PHP 6 Adaptierungen
  - erste Implementierung von “Read-Only” Variablen
  - gleich mehr...
- Andrey Hristov, Georg Richter, Johannes Schlueter

WFT – Windows Task Force

- Hilfe bei der Portierung von MySQL Produkten auf Windows – allen vorran: MySQL Server

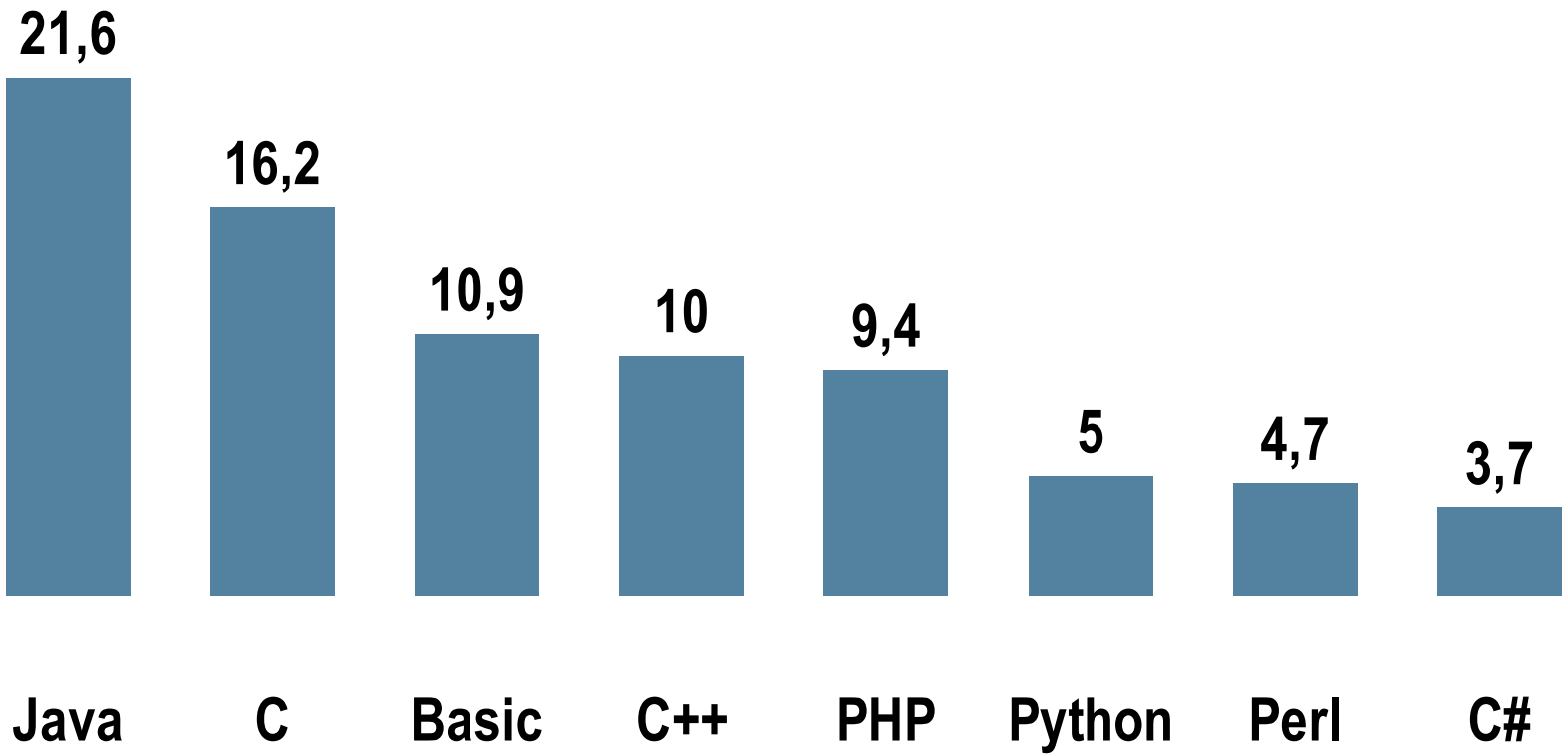
- Reggie Burnett, Ignacio "Iggy" Galarza, Georg Richter

Was wir (noch) nicht machen...

- ... keine Entwicklung der MySQL C-API (MySQL Client Library, AKA “libmysql”), obwohl das Server-Team die meisten Bugs uns zur Bearbeitung zuweist.
- ... keine Verantwortung für die MySQL Client-Tools angefangen vom MySQL-Prompt “mysql” bis hin zum Werkzeug “mysql_config”. Beides betreut das Server-Team.
- ... Sun/Connector (grafisches Installationstool)

Darum Connector/C++ !

TIOBE Programming Community Index for August 2008
http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html



Status Quo: geht und fehlt

Achtung: PREVIEW

- Ausführen von Anfragen, Prepared Statements, grundlegende Metadaten
- keine Anpassung an C++ : keine Überladung der IOStream-Operatoren, keine STL-Unterstützung
- kein Connection Pooling, ...

Roadmap

Achtung: PREVIEW

- Fortentwicklung zur GA
- Erweiterung der API um weitere JDBC 3.0 Kernfunktionen
- ... je nach Rückmeldung der Anwender

Aufbau von Connector/C++

Driver	DriverManager
Connection	DatabaseMetaData
Statement	PreparedStatement
ResultSet	ResultSetMetaData
DataType	Exception, RowID, ...

MySQL Client Library

Installation

- http://forge.mysql.com/wiki/Connector_++
  - CMake (Cross-Platform Make), http://cmake.org
  - MySQL Server (MySQL Client Library), GLib

```
> cmake .
> make

> # Troubleshooting
> make clean; rm CmakeCache.txt
> cmake -L
> cmake -D CMAKE_BUILD_TYPE:STRING=Debug \
    -D MYSQL_DIR:PATH=/path/to/my/mysql/server .
```

Beispiel: einfache Anfrage

```
driver = new sql::mysql::MySQL_Driver;
con = driver->connect(EXAMPLE_HOST, EXAMPLE_PORT,
  EXAMPLE_USER, EXAMPLE_PASS);


stmt = con->createStatement();
res = stmt->executeQuery("SELECT id, label FROM
  test”);
while (res->next()) {
  cout << "id = " << res->getInt(0);
  cout << ", label = '";
  cout << res->getString("label");
  cout << "'" << endl;
}
```

Beispiel: Prepared Statement

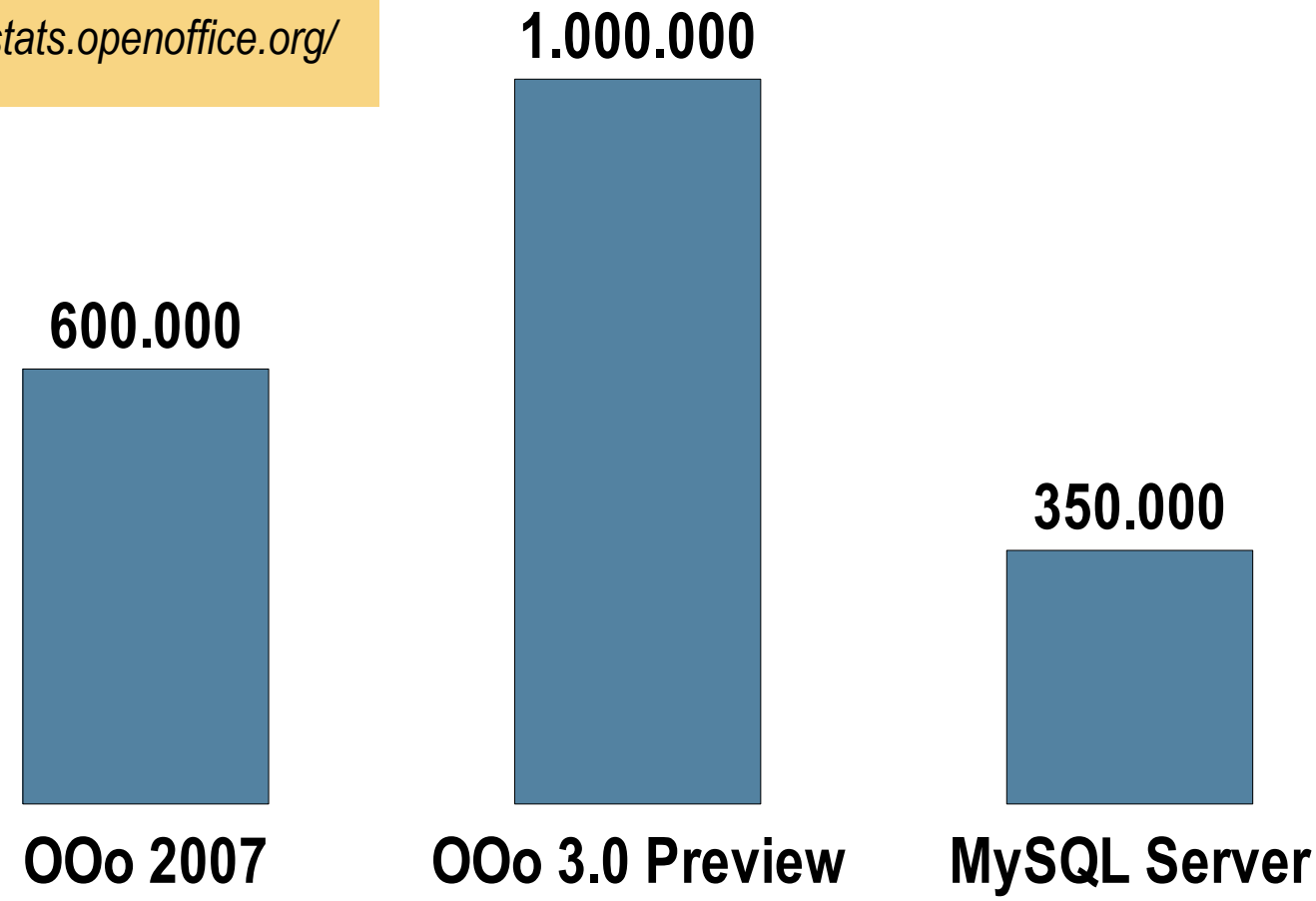
```
prep_stmt = con->prepareStatement("INSERT INTO
  test(id, label) VALUES (?, ?)");

prep_stmt->setInt(1, 1);
prep_stmt->setString(2, "a");
prep_stmt->execute();

prep_stmt->setInt(1, 2);
prep_stmt->setString(2, "b");
prep_stmt->execute();
```

Connector/C++ und MySQL++

- Beide: Mailingliste unter lists.mysql.com
- MySQL++ liegt als Produktionsversion vor
- C/C++ liegt als Preview vor (Pre-Alpha!)
- MySQL++ verwendet properitäre API
- C/C++ orientiert sich an JDBC 3.0
- MySQL++ unterstützt STL, C/C++ nicht
- MySQL++ verwendet LGPL
- C/C++ verwendet Duale Lizenz (GPL + kommerz.)

Darum Connector/OpenOffice.org !

Downloads / Woche
http://stats.openoffice.org/



Status Quo: geht und fehlt

Achtung: PREVIEW

- Ausführen von Anfragen, grundlegende Metadaten
- GUI zur Connection-Parameter Konfiguration
- keine Unterstützung von OpenOffice.org 3.0

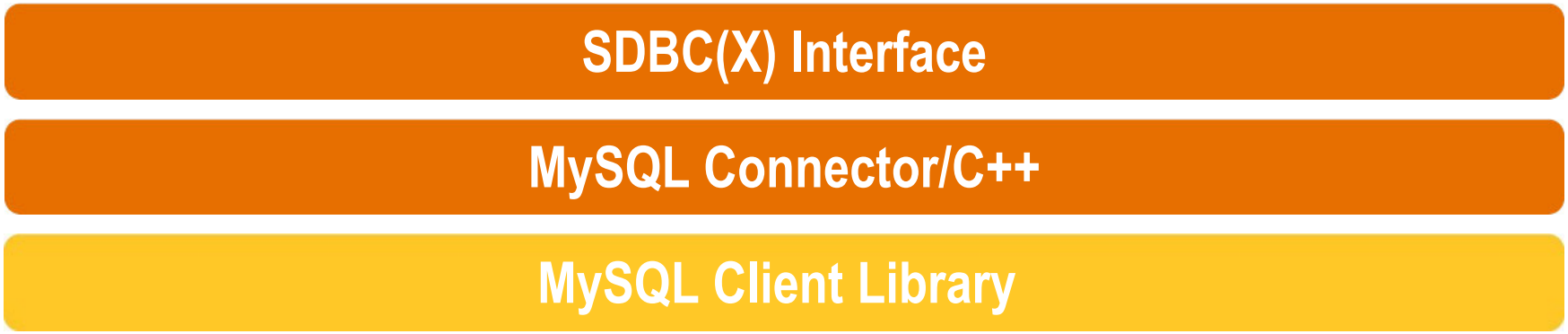
Roadmap

Achtung: PREVIEW

- Fortentwicklung zur GA
- Umstellung von MySQL Client API auf C/C++
- Fortentwicklung zur OOo 3.x Kernkomponente*
- ... more to come

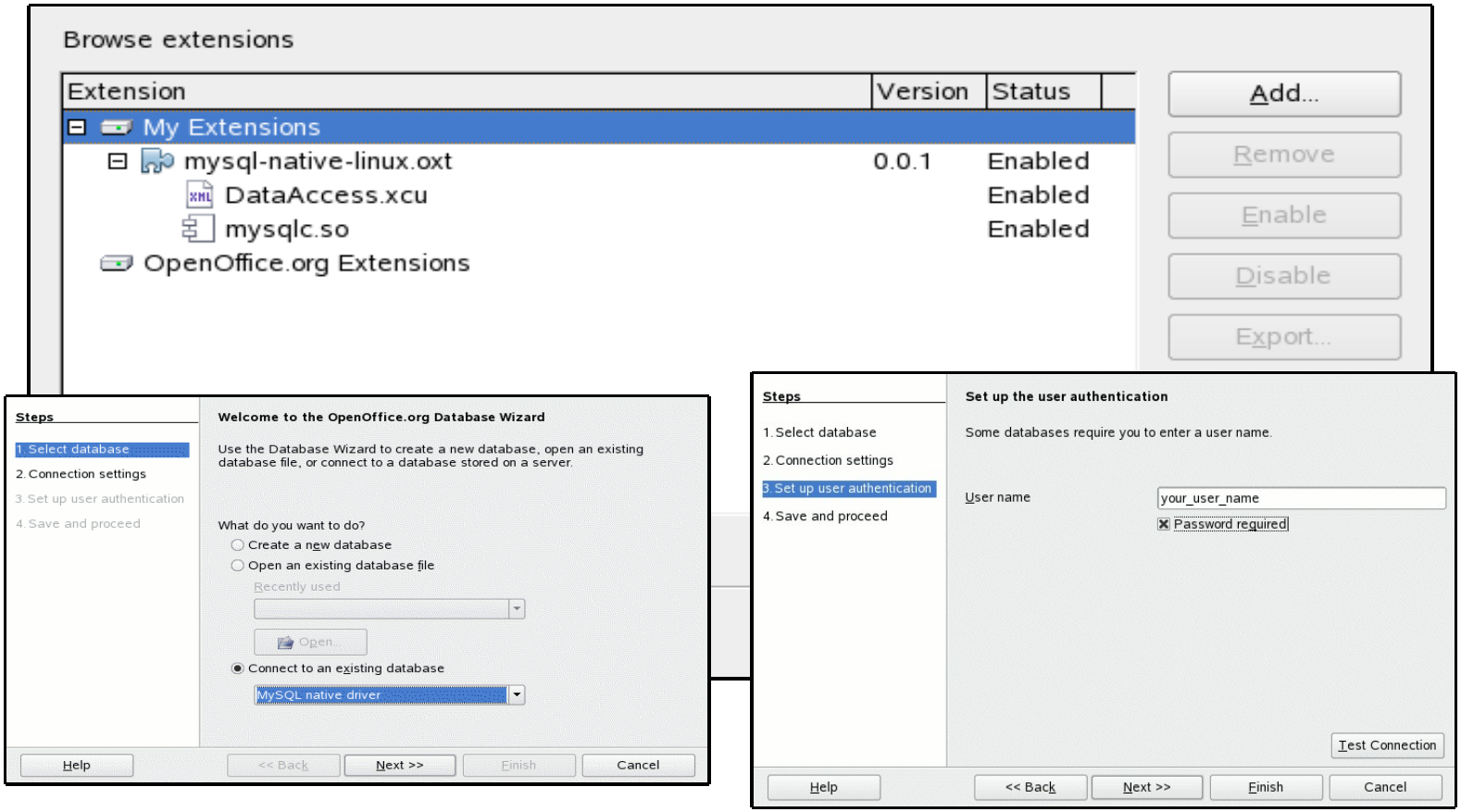
Aufbau von Connector/OO.org

- SDBC(X) ist von JDBC abgeleitet
- Connector/C++ bietet JDBC 3.0



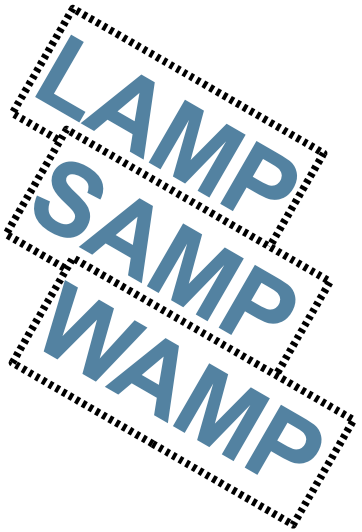
Installation

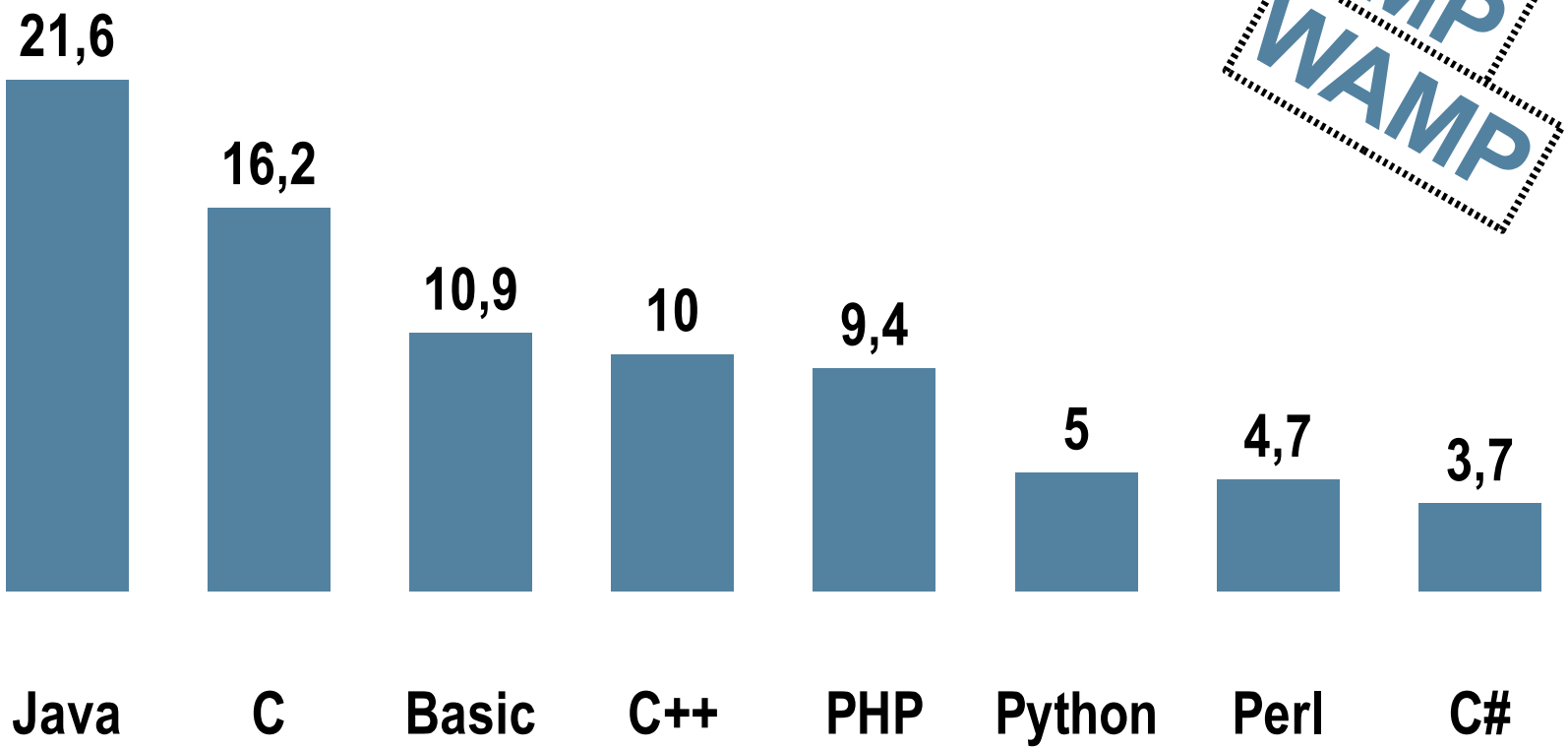
- http://forge.mysql.com/wiki/Connector_OpenOffice



Darum PHP !

TIOBE Programming Community Index for August 2008
http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

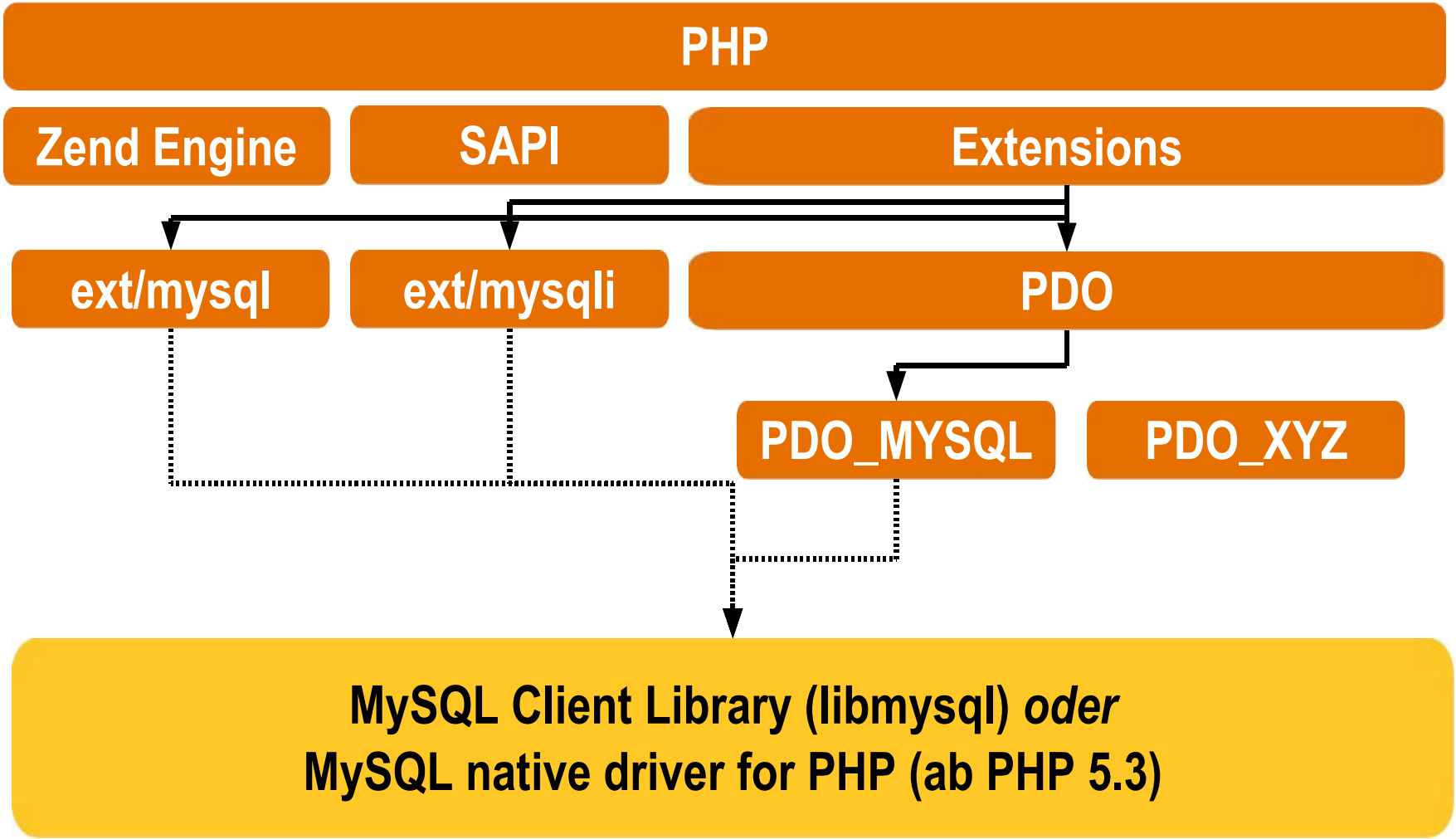




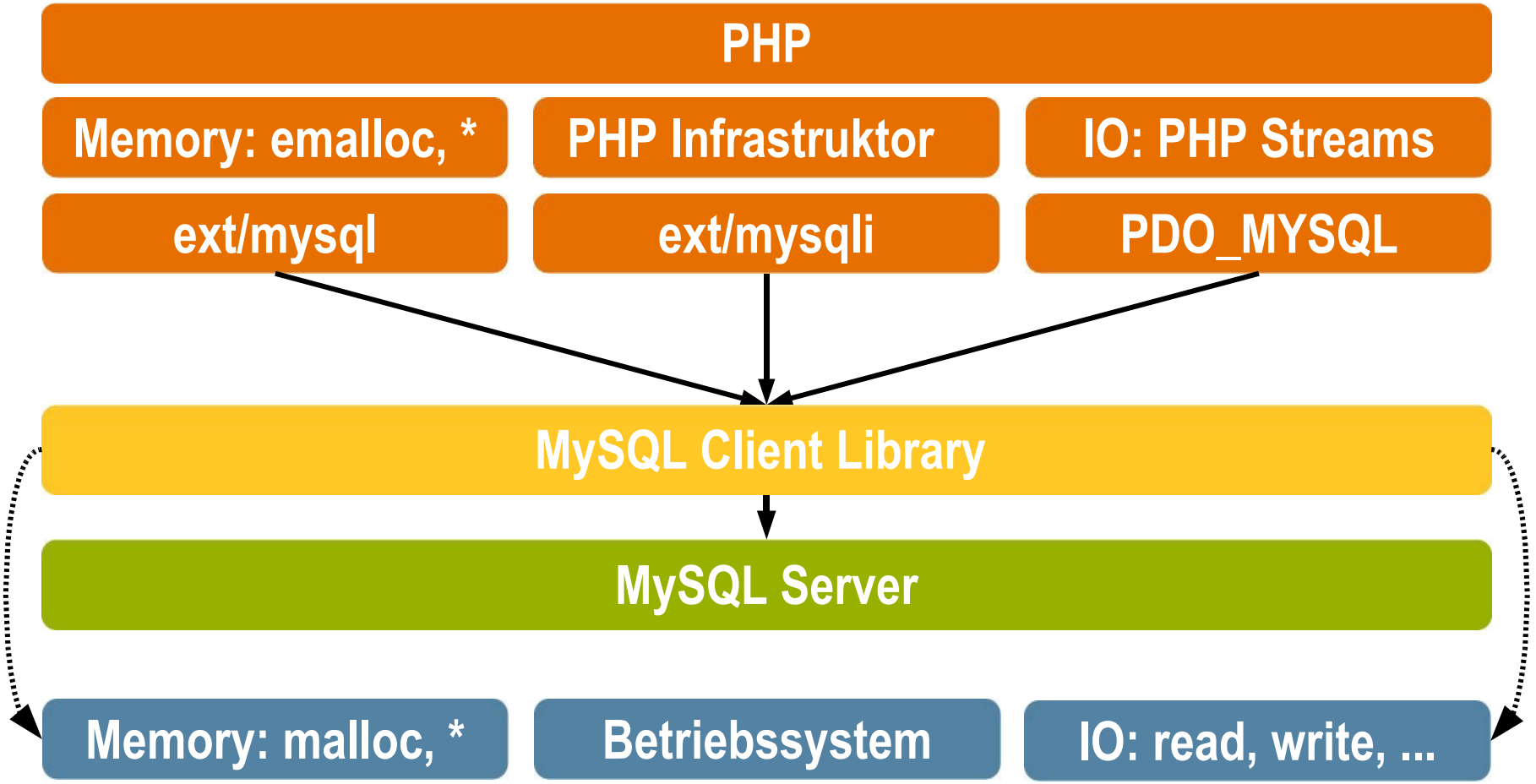
Welches ... hätten S' denn gerne?

	ext/mysql	ext/mysqli	PDO_MYSQL
Wartung durch MySQL	ja	ja	ja
Fortentwicklung durch MySQL	nein	ja	ja
Bestandteil von PHP 4	ja	nein	nein
Bestandteil von PHP 5	ja	ja	ja
Bestandteil von PHP 6	ja	ja	ja
Unterstützung von MySQL < 4.1	ja	nein	ja
Unterstützung von MySQL >= 4.1	unvollst.	ja	unvollst.
MySQL Client Library	ja	ja	ja
MySQL native driver for PHP	ja	ja	ja

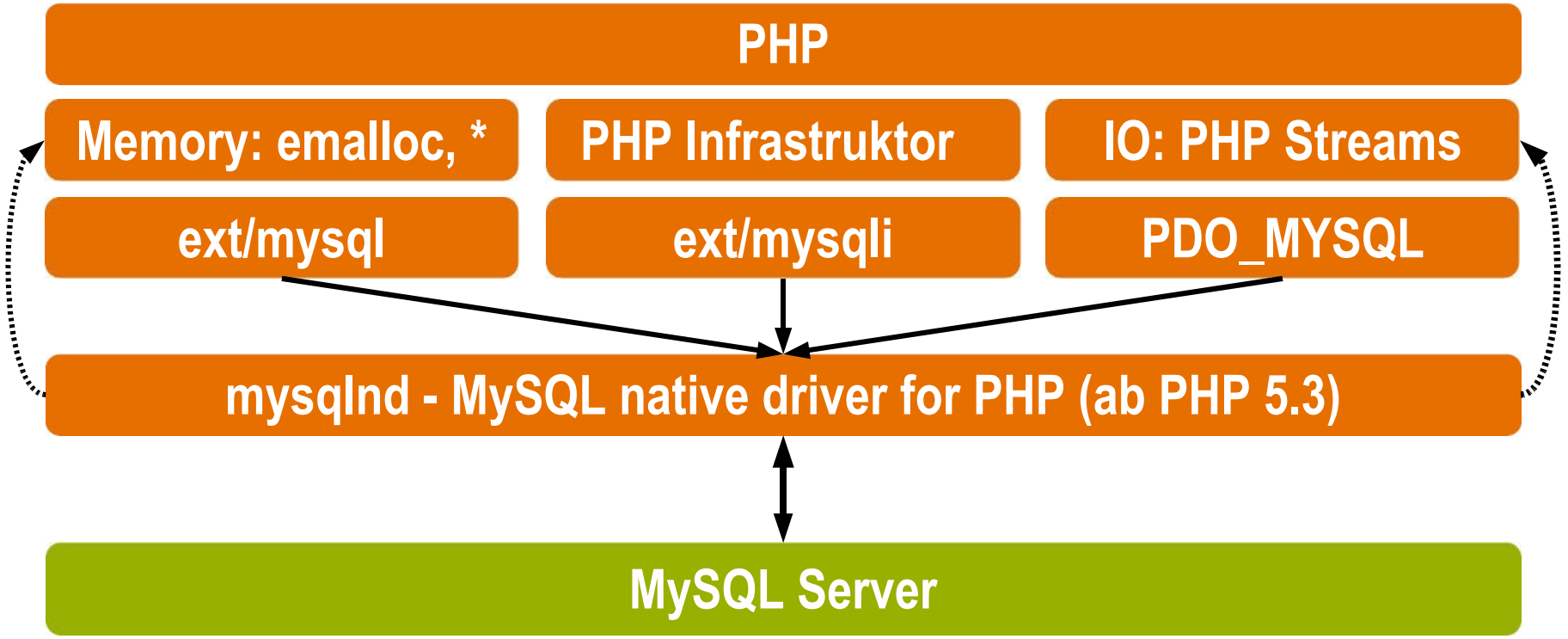
PHP – MySQL Architektur



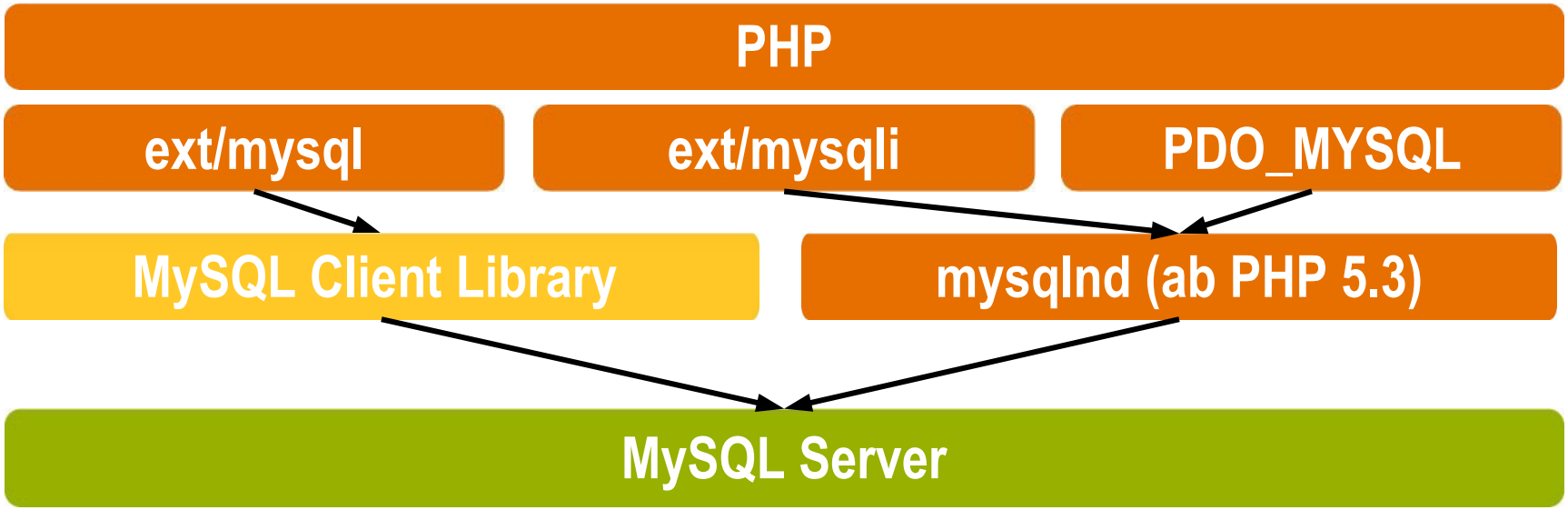
PHP mit MySQL Client Library



MySQL native driver for PHP



Gemischter Salat

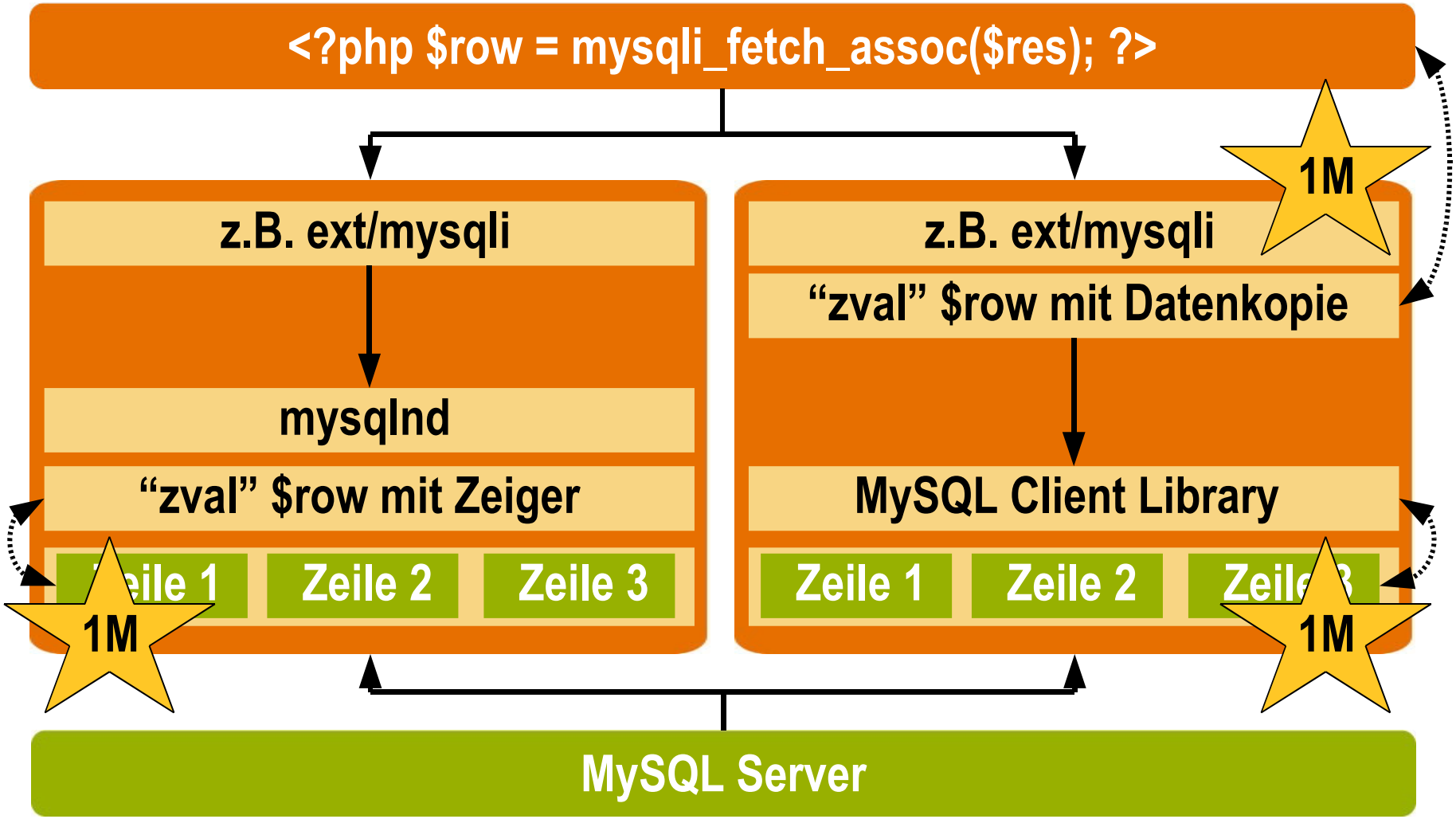



```
./configure --with-mysql=/path/to/mysql_config \
  --with-mysqli=mysqlnd \
  --with-pdo-mysql=mysqlnd
```

Status quo mysqlnd

- Geschwindigkeit Praxis: 0 bis 5% schneller
- Microbenchmarks: -5% bis +1200% schneller
- Speicherverbrauch: 0 bis 40% weniger
- Einfachere Installation
- Neue (Performanz-)Statistiken – phpinfo()
- ext/mysqli: Persistente Verbindungen
- PHP >= 5.3, keine SSL-Unterstützung
- MySQL >= 4.1, kein old-password

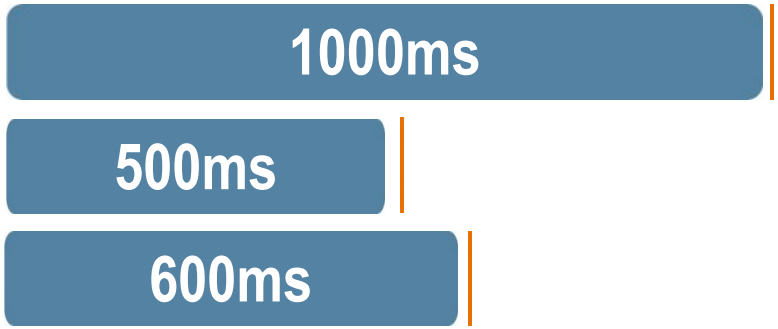
Read-Only Variablen (Copy on Write)



Synchrone vs. asynchrone Anfrage

- Zeitbedarf synchron: Summe($t_1 + t_2 + \ldots t_n$)

  Beispiel, eine Verbindung: 1000 ms + 500ms + 600ms = 2100ms



- Zeitbedarf asynchron: Maximum($t_1 + t_2 + \ldots t_n$)

  Beispiel, drei Verbindungen parallel und asynchron:
  Maximum (1000ms, 500ms, 600ms) = 1000ms

Asynchrone API

```
$queries = array(
  $con1->query("SELECT * FROM slow1",
    MYSQLI_USE_RESULT | MYSQLI_ASYNC), [...]
);
$left = count($queries);
do {
  if (mysqli_poll($queries, NULL, NULL, 1, 0)) {
    foreach ($queries as $query) {
      if ($res = $query->reap_async_query()) {
        process_result($res); $left--
      }
    }
  }
} while ($left > 0);
```

Fortsetzung Asynchrone API

- (fast) ungetestet!
- ext/mysqli/tests/mysqli_async_query.phpt

```
int mysqli_poll(
    array read   /* in:  SELECT query handles */,
    array write  /* in:  INSERT/... queries   */,
    array error  /* out: error message        */,
    long sec     /* in:  timeout seconds      */
    [, long usec /* in:  timeout microseconds */ ]
)

int mysqli_reap_async_query(object link)
```

mysqlnd Statistiken

Netzwerk-Last

- # Bytes gesendet und empfangen
- # Pakete gesendet und empfangen
- # Protokol-Overhead Senden/Empfangen

Anfragen / SQL

- # Anfragen mit Ergebnissen (SELECT)
- # Anfragen ohne Ergebnisse (!SELECT)
- # Anfragen, die Indizes verwenden
- # Anfragen, die keine Indizes verwenden

Ergebnisse

- # Gepufferte und Ungepufferte
- # teilw. gelesene gepufferte Ergebnisse
- # Zeilen gelesen vom Server
- # Zeilen gelesen vom Client
- # Zeilen übersprungen
- # Copy-on-Write: eingespart, ausgeführt

Ergebnisse (Fortsetzung)

- Explizite Freigabe (z.B. *_free_result())
- Implizite Freigabe (z.B. Skriptende)

Verbindung

- # Erfolgreich und fehlgeschlagene
- # Wiederverwendung (Persistente Verb.)
- # Explizit geschlossen (z.B. *_close())
- # Implizit geschlossen (Skriptende)
- # Abgebrochen und geschlossen (Server)
- # Kommando-Abbruch (Netzwerk)

Prepared Statements

- # Gepufferte und Ungepufferte
- # implizit freigegebene Ergebnisse
- # Explizite Freigabe (z.B. *_stmt_close())
- # Implizite Freigabe (Skriptende)
- [...]

PDO und PDO_MYSQL

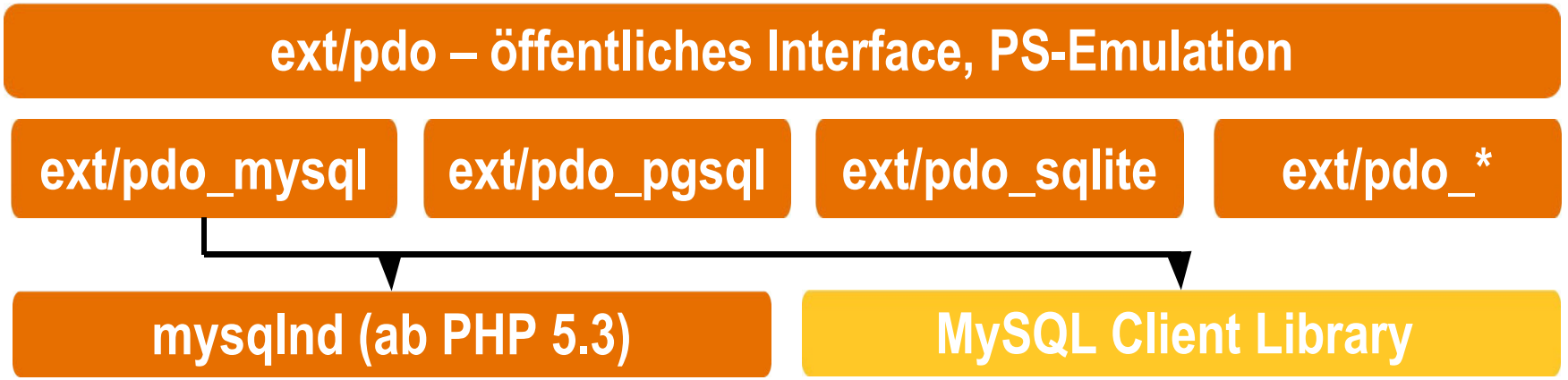
- PDO ist kein PDBC, PDO ist kein ORM
- Abstraktion der Datenzugriffs-API
  - kurze, imprägnante API
  - keine SQL-Abstraktion
  - unvollständige Unterstützung von DB-Spezifika
- Prepared Statement Emulation
- PDO_MYSQLND war ein Arbeitstitel

PDO_MYSQL kompiliert mit mysqlnd

- Native Datentypen
  - SQL INTEGER = PHP integer/double
  - BC: PDO::ATTR_STRINGIFY_FETCHED
- Fixes
  - Mehrfacher Aufruf einer SP mit PS (mysqlnd)
  - next rowset / multi-query* / SP / ~30 Bugs
- Fehlt:
  - PDO::MYSQL_ATTR_MAX_INIT_COMMAND
  - PDO::MYSQL_ATTR_MAX_BUFFER_SIZE

PDO-Architektur

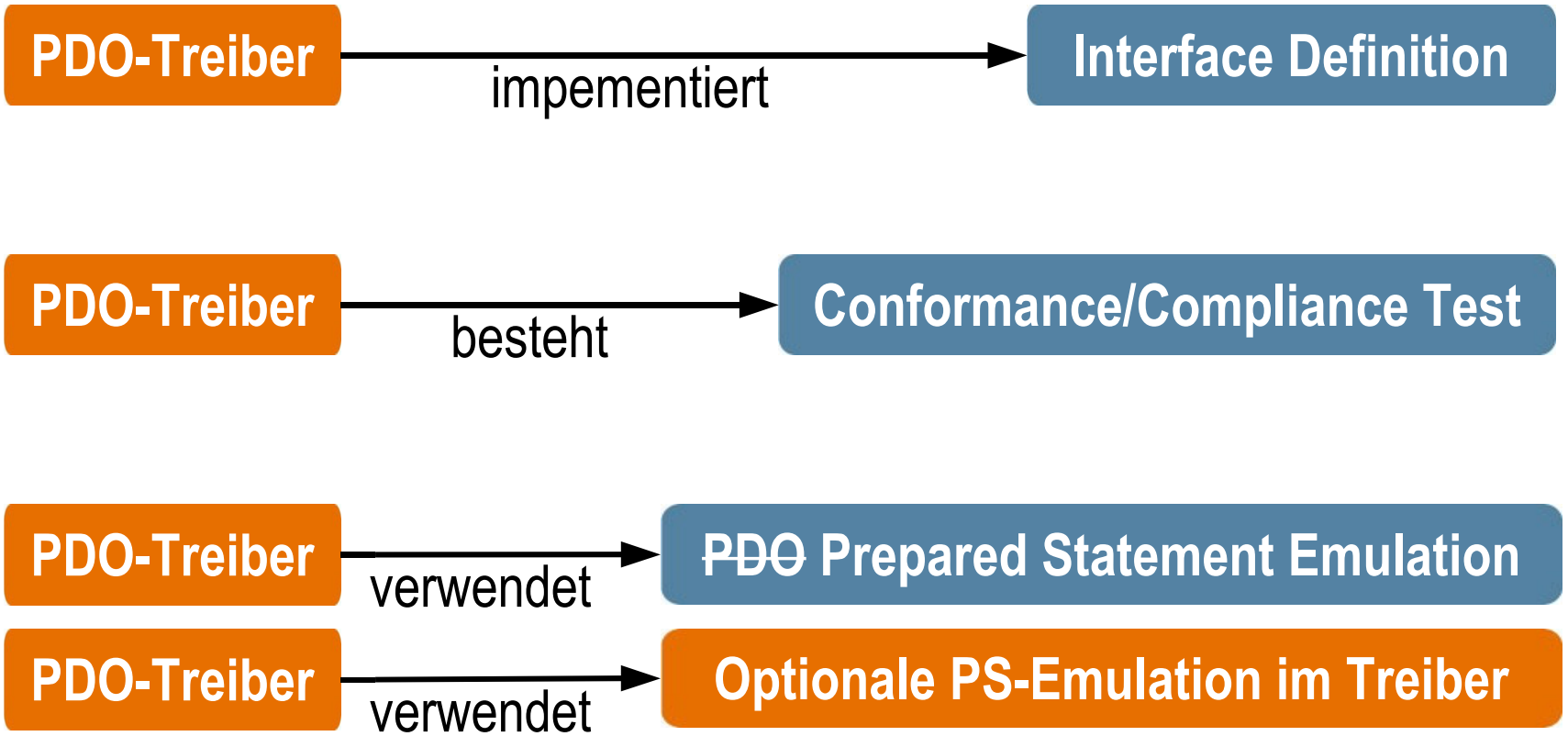
- PDO-Kern
  - öffentliches PDO-Interface
  - Prepared Statement Emulation
- PDO-Treiber
  - internes Callback-Interface
  - optionale öffentliche Interface-Erweiterungen



PDO ist broken by design

- Komplizierung der API-Übersetzung aufgrund interner Callback-Zwischenschicht
- Keine Brücksichtigung von SQL-Dialekten in der Prepared Statement Emulation
  - FAQ im Bugsystem
- Zwangsläuftig keine Berücksichtigung von Datentypen in der Prepared Statement Emulation
  - SELECT mynum FROM table WHERE mynum > “1
- Kein Maintainer? CLA-Blockade Commit
  - 93 Argumente gegen PDO unter http://bugs.php.net

~~PDO~~ PHP-Datenbankzugriff 1.0



Brainstorming Prepared Statements

Sicherheit: Trennung von Kommando und Parametern

Parse-Zeiten einsparen

bind()-basierte API

Datentransfervolumen reduzieren

Server-Resourcen cachen

Skalierung

Optimizer-Pläne wiederverwenden

MySQL PS-Protokoll: keine Konvertierung in Zeichenketten

Client-side Prepared Statements 1.1?

- **Client:** Syntaktischer Parser erstellt Parse-Baum
- **Server:** Prüfung und Anreicherung des Parse-Baum
- **Client:** Zwischenspeicherung des Parse-Baum
- **Client:** Parameterersetzungen im Parse-Baum
- **Client:** Kompilierung in Übertragungsformat
- **Server:** Bearbeitung des Kompilats

The End

Feedback: ulf.wendel@sun.com