

The Apache DBD Framework

Nick Kew

WebThing Limited <http://www.webthing.com/>

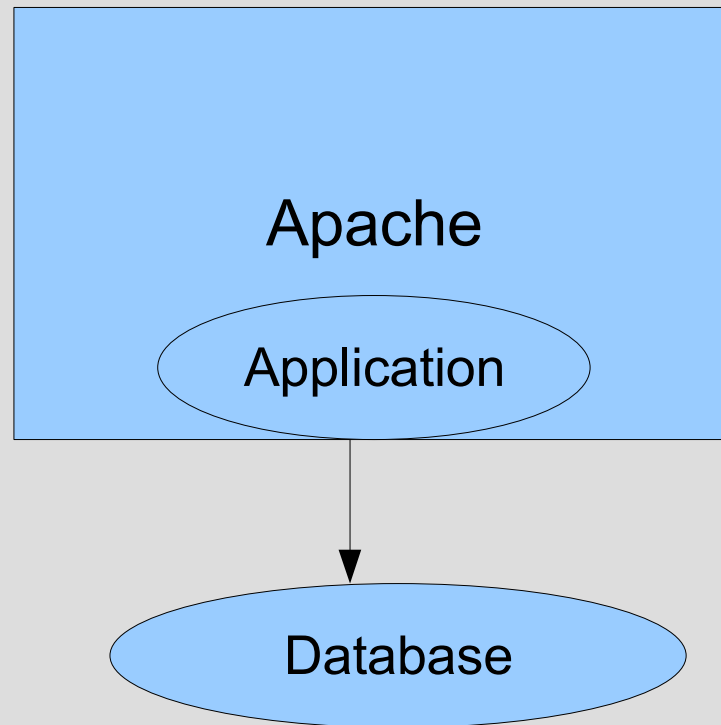
and

Apache Software Foundation <http://www.apache.org/>

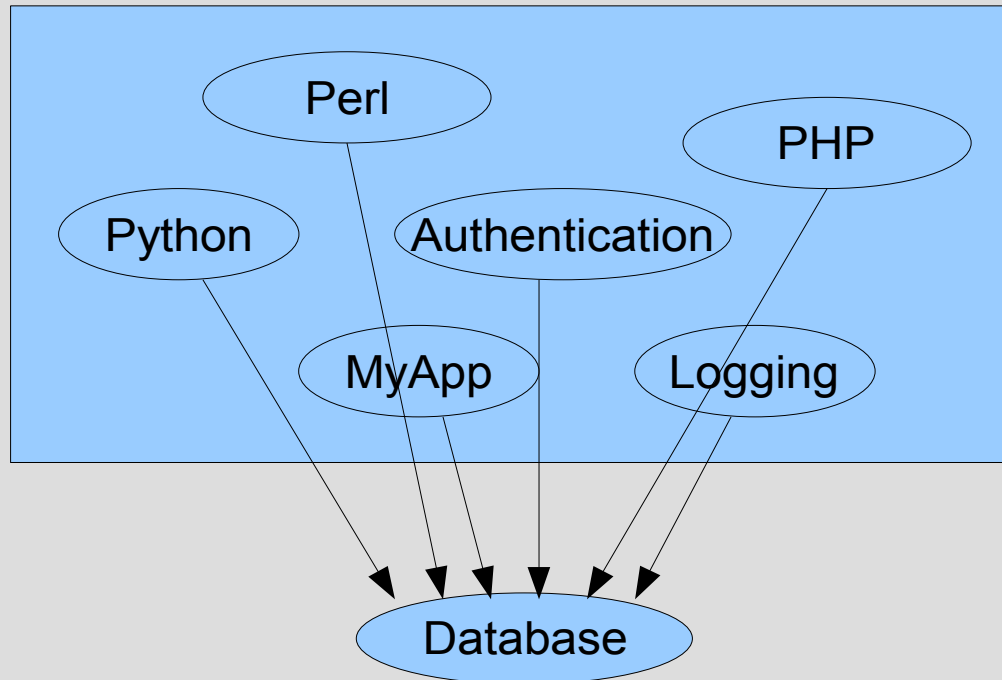
The Apache DBD Framework

- The Need for DBD
- Historical Context
- DBD Architecture
- `apr_dbd`
- DBD Drivers
- `mod_dbd`
- DBD for Server Administrators
- DBD for Module Developers
- DBD for Scripting
- DBD for Standalone Applications

Traditional Application

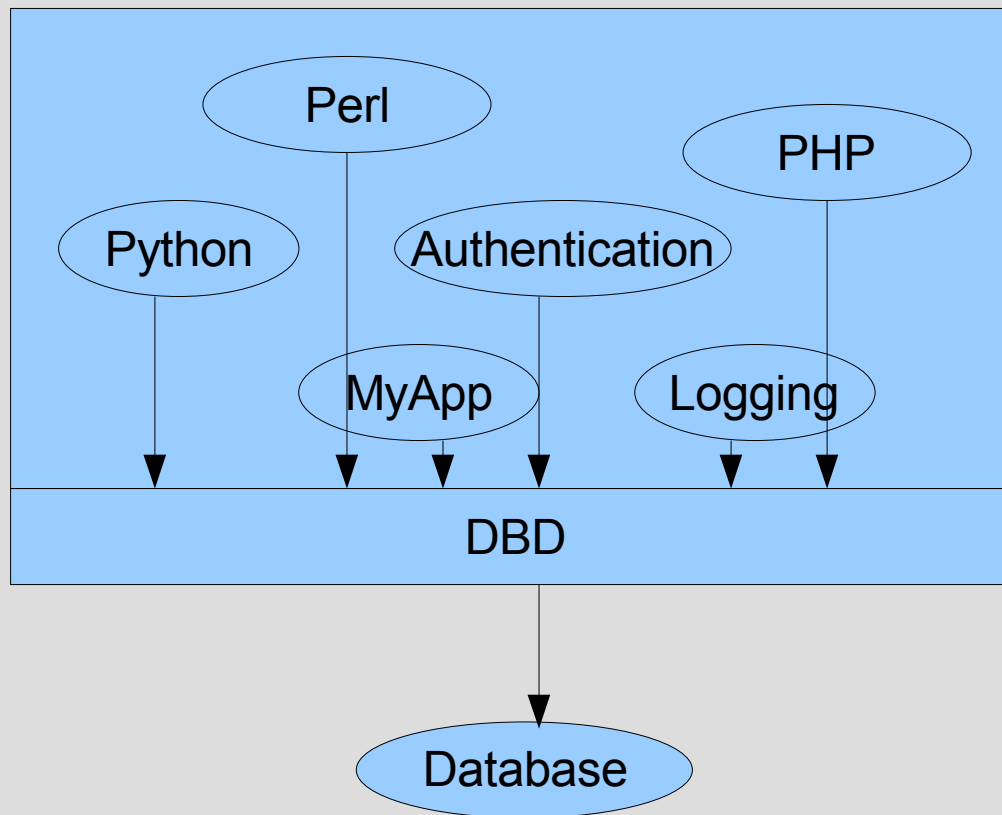


Traditional Applications



N Applications = N expensive connections

DBD Applications



N Applications = 1 expensive connection

Traditional Connections

- Trad. CGI: one connection per request

Problem: opening and closing connections is expensive, so this grows very inefficient as hit rates rise.

- LAMP: one (or N) connection per worker

Problem: expensive connections held even when not in use, and number increases linearly with number of workers.

DBD Connections

- Shared Pool of database connections
- Persistent – no per-request overhead
- N:M – no per-worker overhead
- Dynamically resizes to meet traffic levels

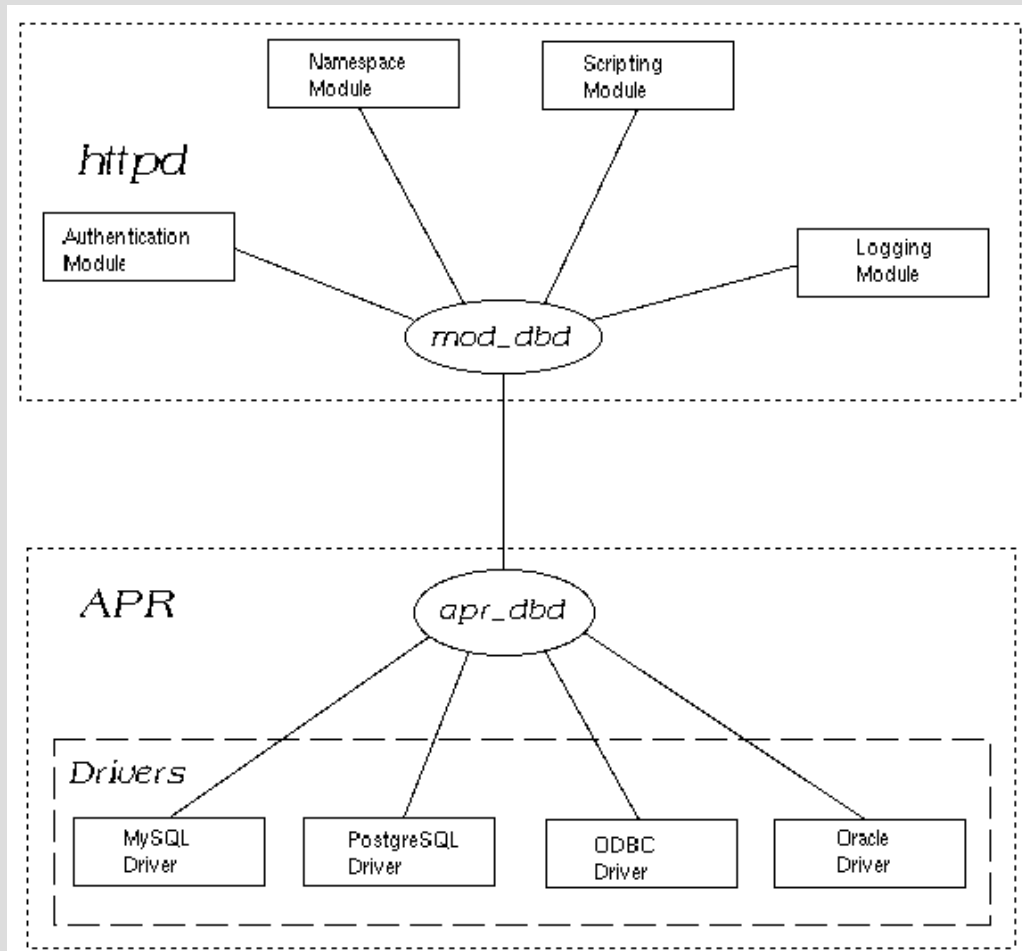
Scalability: DBD vs LAMP

- No overhead as number of database applications grows
- No overhead due to non-database traffic as number of workers (Web traffic) grows.
- Reduced overhead in database traffic
- Worst case: reduces to traditional model
- Best case: Reduces overhead from $O(\text{NumApps} * \text{NumWorkers})$ to $O(1)$

A Brief History

- Prior Art: Perl DBI/DBD; libdbi, etc
- Prerequisite: apr_reslist
- Site Valet: DBD-like layer with connection pooling
- ApacheCon 2003: BOF session
- Connection Pooling Modules for individual databases
- Connection Pooling Module for libdbi
- The current architecture:
 mod_dbd + apr_dbd

DBD Architecture



apr_dbd

```
#include "apr_dbd.h"
```

- Opaque Types
- Drivers
- Interface Layer
- Queries
- Query Results
- Transactions
- Other API functions

Opaque Types

- `apr_dbd_driver_t`
- `apr_dbd_t`
- `apr_dbd_transaction_t`
- `apr_dbd_prepared_t`
- `apr_dbd_results_t`
- `apr_dbd_row_t`

Driver

```
#include "private/apr_dbd_internal.h";

struct apr_dbd_driver_t {
    /* Declare functions implementing each API func */
};

APU_DECLARE_DATA apr_dbd_driver_t
    apr_dbd_foo_driver = {
    /* Functions implementing each API func
     * based on the underlying database's API
     */
};
```

Drivers

- Statically linked or dynamically loaded
- Separate compilation an option
- Current build doesn't support dynamic load

```
apr_dbd_get_driver() {  
    /* Check for it in loaded drivers */  
    #if APR_DSO_BUILD (and not already found)  
        /* If threaded, obtain a mutex */  
        /* Load driver */  
    #endif  
}
```

Interface Layer

```
static int apr_dbd_do_something(  
    apr_dbd_driver_t* driver,  
    apr_dbd_t* handle, ...)  
{  
    /* minimal apr_dbd code if required */  
  
    return driver->do_something(handle, ...);  
}
```

Queries

`apr_dbd_query` (no results)

`apr_dbd_select` (returns a result set)

`apr_dbd_prepare` (prepare a query)

`/* Using printf-like format string for query */`

`apr_dbd_pquery` (prepared query/argc-argv)

`apr_dbd_pselect`

`apr_dbd_pvquery` (prepared query/varargs)

`apr_dbd_pvselect`

Query Results

apr_dbd_num_cols
apr_dbd_num_tuples
apr_dbd_get_name
apr_dbd_get_row
apr_dbd_get_entry

Transactions

apr_dbd_transaction_start
apr_dbd_transaction_end
apr_dbd_transaction_state

Native Database

`apr_dbd_native_handle`

- get handle on the underlying database for (non-portable) operations supported by a database client library but not by `apr_dbd`

Other API

apr_dbd_init
apr_dbd_name
apr_dbd_open
apr_dbd_check_conn
apr_dbd_close
apr_dbd_select_db
apr_dbd_error
apr_dbd_escape

mod_dbd

- Threaded and Unthreaded models
- Connection pooling
- Alternatives: persistent or one-off connections
- Exports ap_dbd API

ap_dbd

```
struct ap_dbd_t {  
    /* driver, handle, statements, pool */  
};
```

ap_dbd_prepare	(startup)
ap_dbd_open	(indefinite lifetime)
ap_dbd_close	(close after an open)
ap_dbd_acquire	(request-oriented)
ap_dbd_cacquire	(connection-oriented)

Server Configuration

```
<VirtualHost foo.example.com>  
    DBDriver pgsql  
    DBDParams "host=db.example.com  
              user=apache pass=secret"  
    DBDMin 5  
    DBDKeep 20  
    DBDMax 50  
    DBDExptime 120  
    #DBDPrepareSQL, DBDPersist  
</VirtualHost>
```

DBD for Modules

- httpd.conf: prepare statements at server startup with `ap_dbd_prepare`
- Req/Conn Processing: `ap_dbd_[c]acquire` normally preferred
- **Caution**: an acquired connection is attached to the `request_rec`, and returned to every module that uses `acquire`. Always leave clean and unencumbered. Use `open/close` if your module can't share with other modules!
- `apr_dbd` API for database ops

DBD for Scripting

Goal for developers of scripting languages:
integrate apr_dbd into familiar scripting idioms

Example: perl

```
$sql = new DBD::APR driver => mysql  
                params => ... etc ...;
```

Standalone perl will use apr_dbd; mod_perl
will substitute the ap_dbd API calls for
obtaining a database connection

Embedded scripting

```
<table>
  <caption>Query Results</caption>
  [set res [apr::dbd::pselect $stmt $params]]
  <thead><tr><th>[apr::dbd::get_name $res 0]</th>
    <th>[apr::dbd::get_name $res 1]</th></tr></thead>
  <tbody>[foreach $row [apr::dbd::get_row $res]
    <tr><td>[apr::dbd::get_entry $row 0]</td>
      <td>[apr::dbd::get_entry $row 1]</td></tr>
    ] </tbody>
</table>
```

Embedded SQL

mod_sql, mod_sqil

```
<table>
```

```
<caption>Search Results</caption>
```

```
<sql:select
```

```
  query="select foo from bar where all is not lost"
```

```
  format="<tr><td>$1</td><td>$2</td>" />
```

```
</table>
```

Standalone DBD Apps

- Don't forget `apr_dbd_init` !
- `apr_dbd` API is a direct alternative to native database client libraries, and other multi-database APIs such as `libdbi`, `ODBC`.
- Selling points include resource management with APR pools, connection pooling support (as in `mod_dbd`) with `apr_reslist`, and APR optional function API.

Examples

- Authentication
- Authorization
- Embedded SQL
- Embedded Scripting
- General Scripting
- Logging to SQL
- SQL-based vhost configuration
- SQL backend for DAV

References

- Applications Development with Apache (the apache modules book) contains by far the most complete reference. ISBN 0-13-240967-4
<http://people.apache.org/~niq/book>
- http://httpd.apache.org/docs/2.2/mod/mod_dbd.html
- <http://people.apache.org/~niq/dbd.html>
- <http://apache.webthing.com/database/>
- <http://www.apachetutor.org/dev/reslist>