Getting Started with PHP-FFI

Thomas Bley, June 2021 @ ipc

About me

- Senior PHP Developer
- Linux, PHP, MySQL since 2001
- studied at TU München
- working for Bringmeister in Berlin

What is PHP-FFI?

- PHP Foreign Function Interface
- PHP extension
- allows
  - loading of shared libraries (.DLL or .so)
  - calling C functions and accessing C data structures
  - in pure PHP during runtime
- requires converting PHP types to C types

source:
php.net/manual/en/intro.ffi.php
phpconference.com/blog/php-ffi-and-what-it-can-do-for-you/

PHP-FFI use cases

- call C/C++ as shared libraries
  - allows direct hardware access (e.g. webcams, GPU)
  - gives more performance (e.g. image or video rendering)
  - integrate databases that have no PHP integration (e.g. DuckDB, RocksDB)
- call Go / Rust compiled as shared libraries
- easier to use than writing PHP extensions

source: github.com/gabrielrcouto/awesome-php-ffi

What is Go?

- programming language
- designed by Google
- statically typed
- memory safety, fast garbage collection
- more performance and concurrency
- can be compiled as a shared library

➔ combining PHP and Go gives a lot of new opportunities

source: en.wikipedia.org/wiki/Go_(programming_language)

PHP-Example: Ackermann function

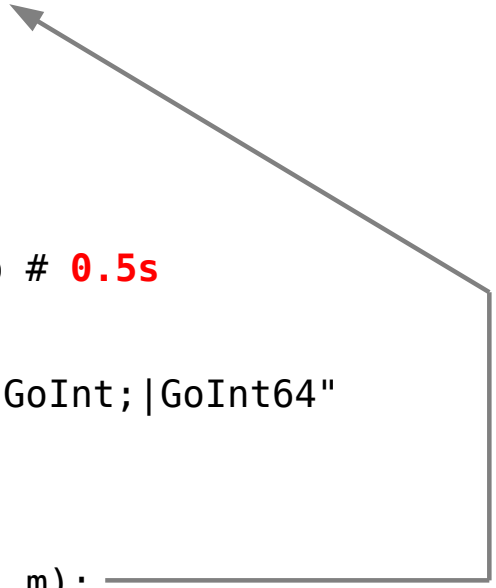
```php
<?php
error_reporting(E_ALL);
function ackermann(int $n, int $m): int {
    if ($n == 0) {
        return $m + 1;
    } else if ($m == 0) {
        return ackermann($n - 1, 1);
    } else {
        return ackermann($n - 1, ackermann($n, $m - 1));
    }
}
echo ackermann(3, 11);

// time php -dopcache.enable_cli=0 ackermann.php # 12.8s
// time php -dopcache.enable_cli=1 ackermann.php # 4.8s
// time php -dopcache.enable_cli=1 -dopcache.jit_buffer_size=32M ackermann.php # 3.2s
source: https://en.wikipedia.org/wiki/Ackermann_function
```

Go-Example: Ackermann function

```go
package main
import "C"

func main() {
    println(ackermann(3, 11));
}

//export ackermann
func ackermann(n int, m int) int {
    if n == 0 {
        return m + 1
    } else if m == 0 {
        return ackermann(n-1, 1)
    }
    return ackermann(n-1, ackermann(n, m-1))
}

// time go run ackermann.go # 0.7s
// go build ackermann.go && time ./ackermann # 0.5s (C/C++/Rust: 0.2s)
```

Calling Go from PHP

```
// go build -o ackermann.so -buildmode=c-shared ackermann.go

<?php
error_reporting(E_ALL);

$ffi = FFI::cdef('long ackermann(long n, long m);', __DIR__ . '/ackermann.so');

echo $ffi->ackermann(3, 11);
unset($ffi);

// time php -dffi.enable=1 ackermann.php # 0.5s

// cat ackermann.h | grep -E "ackermann|GoInt;|GoInt64"
// typedef long long GoInt64;
// typedef GoInt64 GoInt;
// extern GoInt ackermann(GoInt n, GoInt m);
```



Using complex data types

```php
// php
$ffi = FFI::cdef('long ackermann(char* input);', __DIR__ . '/ackermann.so');
$result = $ffi->ackermann_json(json_encode([3, 11]));
echo json_decode(FFI::string($result));
FFI::free($result); // avoid memory leak
unset($ffi);
```

```go
// go
import "encoding/json"
//export ackermann_json
func ackermann_json(input *C.char) *C.char {
    var params [2]int
    err := json.Unmarshal([]byte(C.GoString(input)), &params)
    if err != nil { return nil }

    data, err := json.Marshal(ackermann(params[0], params[1]))
    if err != nil { return nil }

    return C.CString(string(data))
}
```

Checking memory leaks

```
// php
$ffi = FFI::cdef('long ackermann(char* input);', __DIR__ . '/ackermann.so');

for ($i=0; $i < 10000000) {
    $result = $ffi->ackermann_json(json_encode([0, 1]));
    FFI::free($result);
}
unset($ffi);

php -dffi.enable=1 ackermann.php &

while :; do pmap -x `pgrep php` | grep total; sleep 1; done
total kB         1489596   44040   16884
total kB         1489596   43864   16708
total kB         1489596   44568   17412
total kB         1489596   44048   16892
total kB         1489596   43896   16740
...
```

DEMO

Thanks for listening!

Questions?

slides and sources:

github.com/thomasbley/php-go-integration

github.com/thomasbley/php-duckdb-integration