Inside:SWOOLE
非同期処理はどのようにして動くのか

小山哲志
@koyhoge

PHPerKaigi 2020 / 2020-02-09

自己紹介

小山哲志 (こやま てつじ)

合同会社ほげ技研

ユーザ会いくつか

- 日本UNIXユーザ会
- 日本PostgreSQLユーザ会
- 日本PHPユーザ会

rakumo株式会社 エンジニア

映画を観る人 ← **NEW**

@koyhoge　koyhoge

PHPerKaigi 2020 / 2020-02-09

Swooleで実装する
PHP非同期処理の世界

小山哲志
@koyhoge

PHPカンファレンス北海道2019 2019-09-21

PHPカンファレンス北海道2019

「SwooleによるPHP非同期処理」
WEB+DB PRESS vol.113

みんなのPHP 現場で役立つ最新ノウハウ!

PHPerKaigi 2020 / 2020-02-09

Swooleとは

- PHPで非同期処理を実現する機能拡張
- 2012年から開発
- 元は英国Transfonによって
- 現在は中国圏の開発者が猛烈に

非同期処理

- 並行/並列
- コルーチン
- n月間ラムダノート
  vol.1,No.1に詳しい

Swooleの実現法を探る

- 対象は本日(2020-02-09)時点の
  github master
- v 4.4.15 (+α)

スタートはここから

```
go(function() {
      // something
})
```

PHPerKaigi 2020 / 2020-02-09

go()

swoole.cc

```
473         if (SWOOLE_G(use_shortname))
474         {
475             SW_FUNCTION_ALIAS(CG(function_table),
"swoole_coroutine_create", CG(function_table), "go");
476             SW_FUNCTION_ALIAS(CG(function_table),
"swoole_coroutine_defer", CG(function_table), "defer");
477         }
```

swoole_coroutine_create()

swoole_coroutine.cc

```
931 PHP_FUNCTION(swoole_coroutine_create)
932 {
933     zend_fcall_info fci;
934     zend_fcall_info_cache fci_cache;
935
936     ZEND_PARSE_PARAMETERS_START(1, -1)
937         Z_PARAM_FUNC(fci, fci_cache)
938         Z_PARAM_VARIADIC('*', fci.params, fci.param_count)
939     ZEND_PARSE_PARAMETERS_END_EX(RETURN_FALSE);
940
941     if (sw_unlikely(SWOOLE_G(req_status) == PHP_SWOOLE_CALL_USER_SHUTDOWNFUNC_BEGIN))
```

swoole_coroutine.cc

```
  951       long cid = PHPCoroutine::create(&fci_cache,
fci.param_count, fci.params);
```

PHPCoroutine::create()

swoole_coroutine.cc

```
  814 long PHPCoroutine::create(zend_fcall_info_cache *fci_cache, uint32_t
argc, zval *argv)
  815 {
    :
  838     php_coro_args php_coro_args;
  839     php_coro_args.fci_cache = fci_cache;
  840     php_coro_args.argv = argv;
  841     php_coro_args.argc = argc;
  842     save_task(get_task());
  843
  844     return Coroutine::create(main_func, (void*) &php_coro_args);
  845 }
```

Coroutine::create()

include/coroutine.h

```
121     static inline long create(coroutine_func_t fn, void* args = nullptr)
122     {
123         return (new Coroutine(fn, args))->run();
124     }
```

Coroutine::Coroutine()

include/coroutine.h

```
205     Coroutine(coroutine_func_t fn, void *private_data) :
206             ctx(stack_size, fn, private_data)
207     {
208         cid = ++last_cid;
209         coroutines[cid] = this;
210         if (sw_unlikely(count() > peak_num))
211         {
212             peak_num = count();
213         }
214     }
```

Coroutine::run()

include/coroutine.h

```
216     inline long run()
217     {
218         long cid = this->cid;
219         origin = current;
220         current = this;
221         ctx.swap_in();
222         check_end();
223         return cid;
224     }
```

Coroutine::ctx

include/coroutine.h

```
199     sw_coro_state state = SW_CORO_INIT;
200     long cid;
201     void *task = nullptr;
202     Context ctx;
203     Coroutine *origin;
```

Context::swap_in()

src/coroutine/context.cc

```
110 bool Context::swap_in()
111 {
112     jump_fcontext(&swap_ctx_, ctx_, (intptr_t) this, true);
113     return true;
114 }
```

jump_fcontext()

- swoole-src 内にはない
- boost.context の関数

boost.context

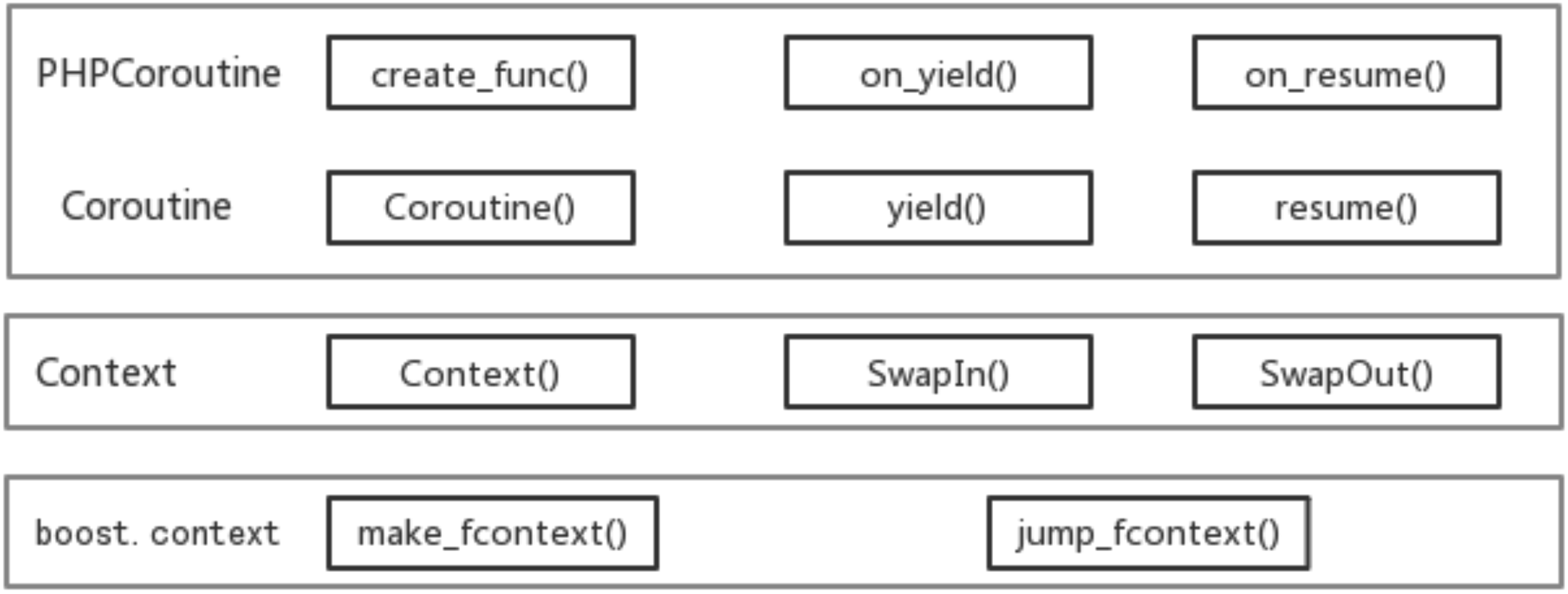
- https://github.com/boostorg/context
- C++でシングルスレッドマルチタスクを実現するライブラリ
- setjmp(), longjmp() の進化版

タスク切換えはasmで

```
briareoth:context koyama$ ls src/asm
./                                      make_ppc32_ppc64_sysv_macho_gas.S
../                                     make_ppc32_sysv_elf_gas.S
jump_arm64_aapcs_elf_gas.S              make_ppc32_sysv_macho_gas.S
jump_arm64_aapcs_macho_gas.S            make_ppc32_sysv_xcoff_gas.S
jump_arm_aapcs_elf_gas.S                make_ppc64_sysv_elf_gas.S
jump_arm_aapcs_macho_gas.S              make_ppc64_sysv_macho_gas.S
jump_arm_aapcs_pe_armasm.asm            make_ppc64_sysv_xcoff_gas.S
jump_combined_sysv_macho_gas.S          make_riscv64_sysv_elf_gas.S
jump_i386_ms_pe_gas.asm                 make_s390x_sysv_elf_gas.S
jump_i386_ms_pe_masm.asm                make_x86_64_ms_pe_gas.asm
jump_i386_sysv_elf_gas.S                make_x86_64_ms_pe_masm.asm
jump_i386_sysv_macho_gas.S              make_x86_64_sysv_elf_gas.S
jump_i386_x86_64_sysv_macho_gas.S       make_x86_64_sysv_macho_gas.S
jump_mips32_o32_elf_gas.S               ontop_arm64_aapcs_elf_gas.S
jump_mips64_n64_elf_gas.S               ontop_arm64_aapcs_macho_gas.S
jump_ppc32_ppc64_sysv_macho_gas.S       ontop_arm_aapcs_elf_gas.S
jump_ppc32_sysv_elf_gas.S               ontop_arm_aapcs_macho_gas.S
jump_ppc32_sysv_macho_gas.S             ontop_arm_aapcs_pe_armasm.asm
jump_ppc32_sysv_xcoff_gas.S             ontop_combined_sysv_macho_gas.S
jump_ppc64_sysv_elf_gas.S               ontop_i386_ms_pe_gas.asm
jump_ppc64_sysv_macho_gas.S             ontop_i386_ms_pe_masm.asm
jump_ppc64_sysv_xcoff_gas.S             ontop_i386_sysv_elf_gas.S
jump_riscv64_sysv_elf_gas.S             ontop_i386_sysv_macho_gas.S
jump_s390x_sysv_elf_gas.S               ontop_i386_x86_64_sysv_macho_gas.S
jump_x86_64_ms_pe_gas.asm               ontop_mips32_o32_elf_gas.S
jump_x86_64_ms_pe_masm.asm              ontop_mips64_n64_elf_gas.S
jump_x86_64_sysv_elf_gas.S              ontop_ppc32_ppc64_sysv_macho_gas.S
jump_x86_64_sysv_macho_gas.S            ontop_ppc32_sysv_elf_gas.S
make_arm64_aapcs_elf_gas.S              ontop_ppc32_sysv_macho_gas.S
```

参考図

- **【Swoole源码研究】深入理解Swoole协程实现**
  - https://segmentfault.com/a/1190000019089997



タスクはどう保持されている?

include/coroutine.h

```
114     static std::unordered_map<long, Coroutine*> coroutines;
```

PHPerKaigi 2020 / 2020-02-09

まとめ

- Swooleはboost.contextのタスク切換を利用している
- 一見魔法のように見えても、ひとつずつ辿っていくと理解できる
- C++コワくないよw