PHP 2017·北京 全球开发者大会

高可用的 PHP

PHP与APM： 技术内幕和最佳实践

高驰涛 2017-06-11

ABOUT ME

高驰涛 neeke@php.net

云智慧研发总监，
PECL开发组成员。
SeasLog & JsonNet & PHP-Druid & GoCrab等多项开源软件作者。

提纲

What is APM

What does APM mean for PHP

Make PHPAgent working

WHAT IS APM

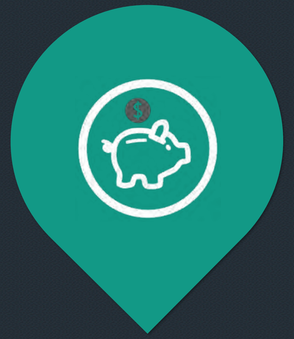
Application Performance Management

一个抽象的复杂应用架构

APM的5个要求

End User Experience

Runtime Application Architecture



Business Transactions

Deep Dive Component Monitoring

Analytics Reporting

应用APM的优势

1

关注真实用户体验

终端用户的真实体验，才是衡量应用性能是否良好的最终标准

2

自动发现和主动探测

帮助架构师和管理师，充分了解应用的运行时构成和潜在问题点

弥补脑力不足，从应用内部进行监测

3

从业务角度看性能

每一种错误或异常、缓慢，对多少用户的什么业务造成影响

每个错误或异常、缓慢，影响的具体是谁

实现APM的难点

1 用户无感知

不影响任何终端用户的任何体验

2 工程无感知

不影响原工程的结构与代码

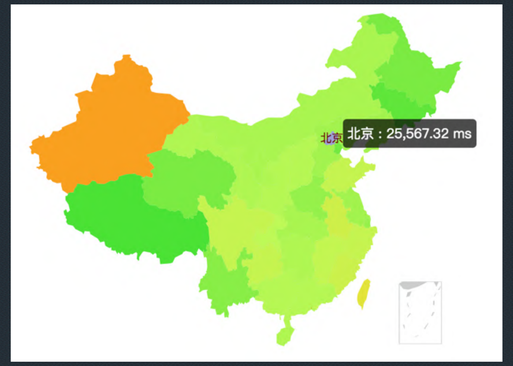
3 业务无感知

不影响原应用的任何业务

4 实时

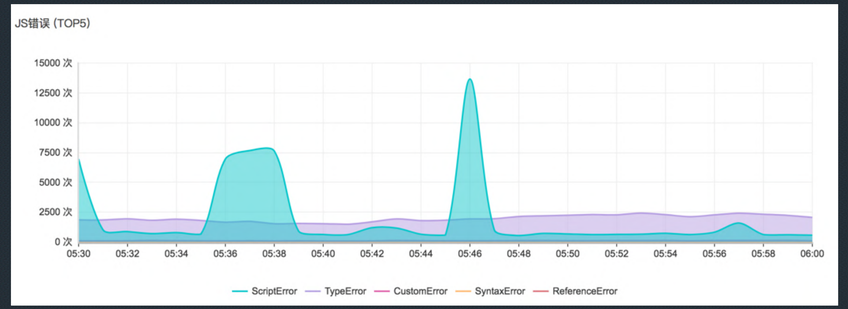
WHAT DOES APM MEAN FOR PHP

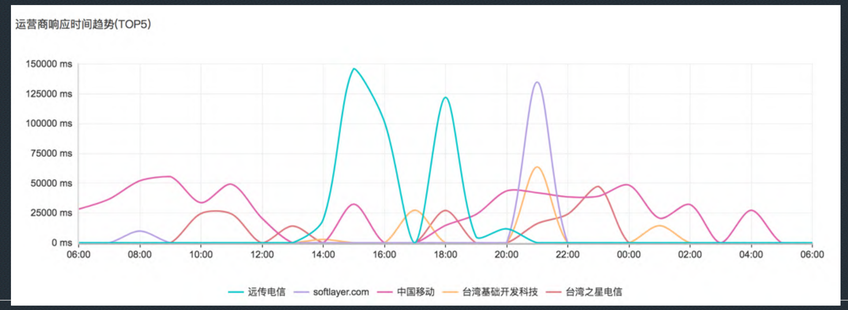
准确感知真实用户体验



web事务	http://star.a.com/zihao?from=topbarfeed				
错误类型	IndexSizeError				
错误信息	IndexSizeError (DOM Exception 1): The index is not in the allowed range.				
发生时间	2017-06-05 23:06:33	IP	111.23.151.62	地域	湖南
浏览器	qq		版本号	5.7	
UA数据	Mozilla/5.0 (iPad; CPU OS 10_3_1 like Mac OS X) AppleWebKit/603.1.30 (KHTML, like Gecko) Version/8.0 MQQBrowser/5.7.3 Mobile/14E304 Safari/600.1.4				
堆栈	end@[native code] getVideoDetail@http://star.a.com/zihao?from=topbarfeed:52:49 global code@http://star.a.com/zihao?from=topbarfeed:1:42				

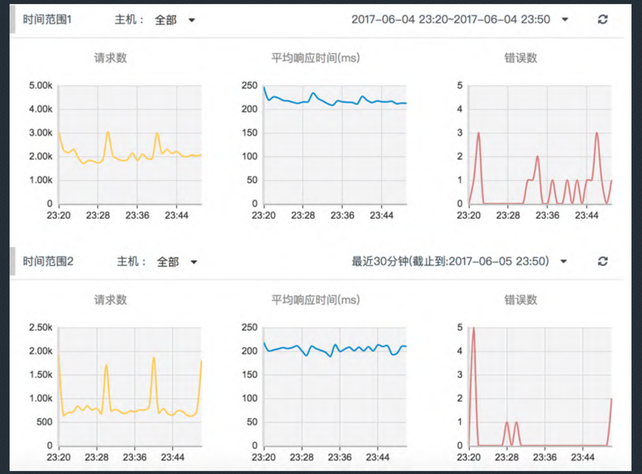
准确感知真实用户体验

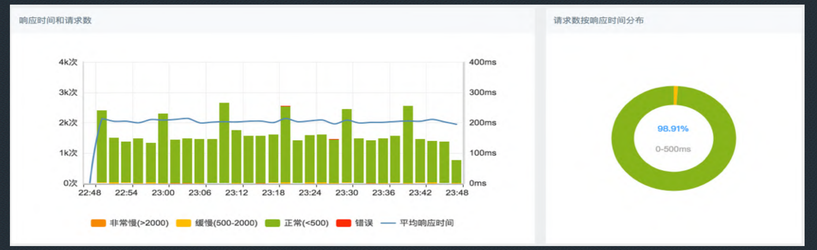

JS错误 (TOP5)
15000 次
12500 次
10000 次
7500 次
5000 次
2500 次
0 次
05:30 05:32 05:34 05:36 05:38 05:40 05:42 05:44 05:46 05:48 05:50 05:52 05:54 05:56 05:58 06:00
ScriptError TypeError CustomError SyntaxError ReferenceError

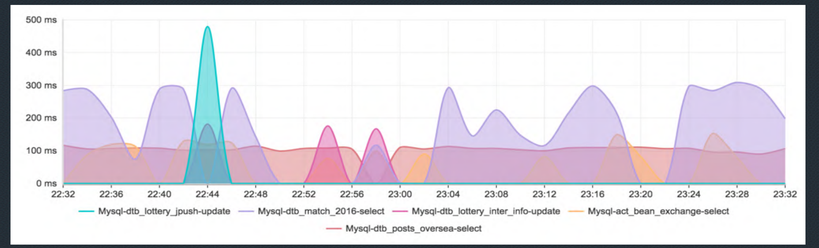

运营商响应时间趋势(TOP5)
150000 ms
125000 ms
100000 ms
75000 ms
50000 ms
25000 ms
0 ms
06:00 08:00 10:00 12:00 14:00 16:00 18:00 20:00 22:00 00:00 02:00 04:00 06:00
远传电信 softlayer.com 中国移动 台湾基础开发科技 台湾之星电信


性能评分
98
分
APP得分
响应时间 89分
崩溃 99分
HTTP错误 98分
网络错误 100分
平均响应时间
585.52ms
次数：4.07m
崩溃率
0.42%
影响用户：66
HTTP错误率
1.61%
影响用户：13.84k
网络失败率
0.61%
影响用户：4.57k
响应时间分布
50th percentile(300ms)
90th percentile(800ms)
95th percentile(1400ms)
1.00m次
800k次
600k次
400k次
200k次
0次
100 ms 400 ms 700 ms 1000 ms 2200 ms 3400 ms 4600 ms 10000 ms 17500 ms 25000 ms 30000+ ms

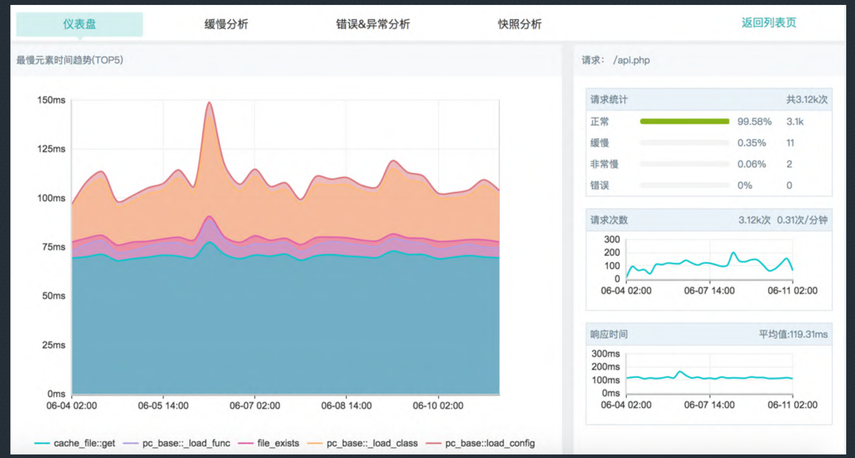
运行时监控

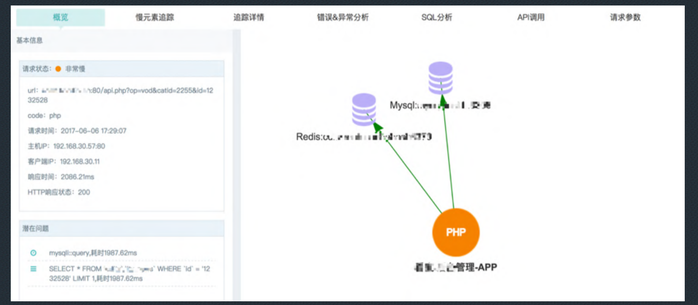

时间范围1
主机： 全部
2017-06-04 23:20~2017-06-04 23:50
请求数
平均响应时间(ms)
错误数
时间范围2
主机： 全部
最近30分钟(截止到:2017-06-05 23:50)
请求数
平均响应时间(ms)
错误数


响应时间和请求数
请求数按响应时间分布
98.91%
0-500ms
非常慢(>2000)
缓慢(500-2000)
正常(<500)
错误
平均响应时间


Mysql-dtb_lottery_jpush-update
Mysql-dtb_match_2016-select
Mysql-dtb_lottery_inter_info-update
Mysql-act_bean_exchange-select
Mysql-dtb_posts_oversea-select

运行时监控


仪表盘
缓慢分析
错误&异常分析
快照分析
返回列表页
最慢元素时间趋势(TOP5)
150ms
125ms
100ms
75ms
50ms
25ms
0ms
06-04 02:00
06-05 14:00
06-07 02:00
06-08 14:00
06-10 02:00
cache_file::get
pc_base::_load_func
file_exists
pc_base::_load_class
pc_base::load_config
请求： /api.php
请求统计
共3.12k次
正常
99.58%
3.1k
缓慢
0.35%
11
非常慢
0.06%
2
错误
0%
0
请求次数
3.12k次 0.31次/分钟
300
200
100
0
06-04 02:00
06-07 14:00
06-11 02:00
响应时间
平均值:119.31ms
300ms
200ms
100ms
0ms
06-04 02:00
06-07 14:00
06-11 02:00


概览
慢元素追踪
追踪详情
错误&异常分析
SQL分析
API调用
请求参数
基本信息
潜在问题
PHP


概览
慢元素追踪
追踪详情
错误&异常分析
SQL分析
API调用
请求参数
慢元素
0 %
20 %
40 %
60 %
80 %
100 %


概览
慢元素追踪
追踪详情
错误&异常分析
SQL分析
API调用
请求参数


概览
慢元素追踪
追踪详情
错误&异常分析
SQL分析
API调用
请求参数
SELECT * FROM
WHERE `id` = '1232528' LIMIT 1

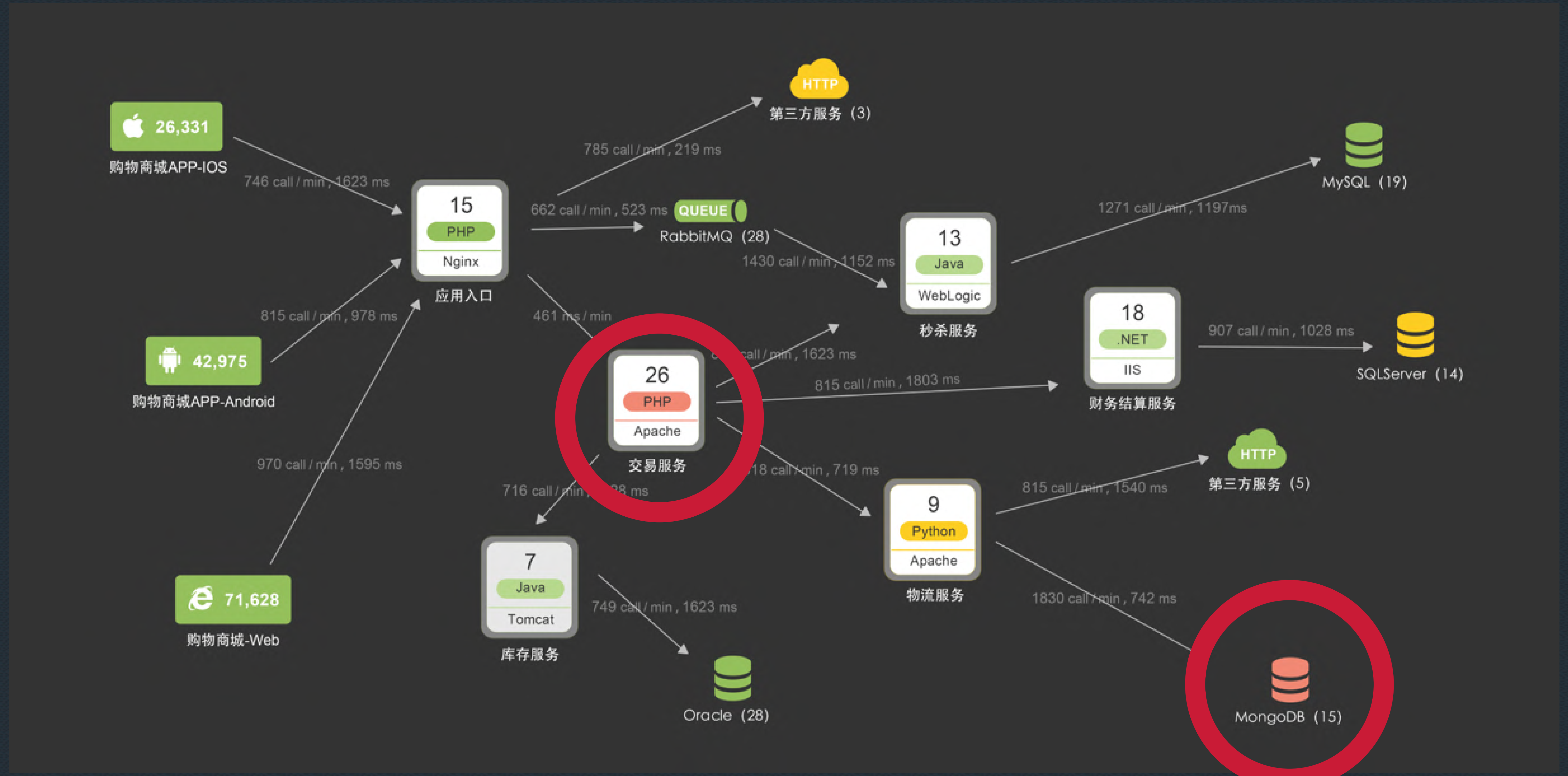
洞察业务故障


业务B(/app/score/score)
正常
82.23
每分钟执行次数
292.49 ms
平均响应时间
0 %
错误率
0
错误请求数
4934
执行次数
业务A(/app/score/match_base_detail)
正常
112.98
每分钟执行次数
244.29 ms
平均响应时间
0 %
错误率
0
错误请求数
6779
执行次数
业务C(/App/Topic/postTopic)
正常
0.08
每分钟执行次数
198.01 ms
平均响应时间
0 %
错误率
0
错误请求数
5
执行次数


app.a.com:80 > 事务 > 业务C
最近 1小时 (截止到:2017-06-05 23:41)
概要分析
错误分析
SQL分析
事务快照
错误列表
异常列表
第一次发生时间
最后一次发生时间
外部事务
消息
发生次数
2017-06-05 23:03:40
2017-06-05 23:21:07
app.a.com:80/App/Topic/postTopic
/var/disk/web/a/Application/Common/Service/PayService.class.php文件在 344 行,发生异常：mb not enough
2
2017-06-05 23:03:40
2017-06-05 23:21:07
app.a.com:80/App/Topic/postTopic
/var/disk/web/a/Application/App/Model/ZqmfV2Model.class.php文件在 2014 行,发生异常：支付失败!
2


URL
第一次发生时间
最后一次发生时间
发生次数
app.a.com:80/App/Topic/postTopic
2017-06-05 23:03:40
2017-06-05 23:21:07
2
第1/2个同类错误
发生时间 2017-06-05 23:21
时间:
2017-06-05 23:21
请求参数:
uri:/App/Topic/postTopic
cta: {"user_id":"8888888","topic_id":"6666","dealer_id":"333333","coupon_ids":[],"isUsewallet":true,"platform":"mofang","versionNum":"3.20","form":"android","iostype":"0","idfa":"abcdrfg","mf_token":"hijklmn"}
mnn: opqrst
错误信息:
/var/disk/web/a/Application/App/Model/ZqmfV2Model.class.php文件在 2014 行,发生异常： 支付失败!

预测架构瓶颈


26,331
购物商城APP-IOS
42,975
购物商城APP-Android
71,628
购物商城-Web
15
PHP
Nginx
应用入口
第三方服务 (3)
QUEUE
RabbitMQ (28)
13
Java
WebLogic
秒杀服务
MySQL (19)
18
.NET
IIS
财务结算服务
SQLServer (14)
26
PHP
Apache
交易服务
第三方服务 (5)
9
Python
Apache
物流服务
7
Java
Tomcat
库存服务
Oracle (28)
MongoDB (15)
746 call / min , 1623 ms
785 call / min , 219 ms
662 call / min , 523 ms
1430 call / min , 1152 ms
1271 call / min , 1197ms
815 call / min , 978 ms
907 call / min , 1028 ms
815 call / min , 1803 ms
970 call / min , 1595 ms
815 call / min , 1540 ms
749 call / min , 1623 ms
1830 call / min , 742 ms


APP ABCD
概览 | 应用(1) | 事务(TOP10) | 数据库(8) | 调用者(0) | 主机(1)
业务健康度
正常 1
平均响应时间 235.02ms
请求数 :36.23k次 603.8次/min
错误数 :9次 0.15次/min
异常数 :14次 0.23次/min

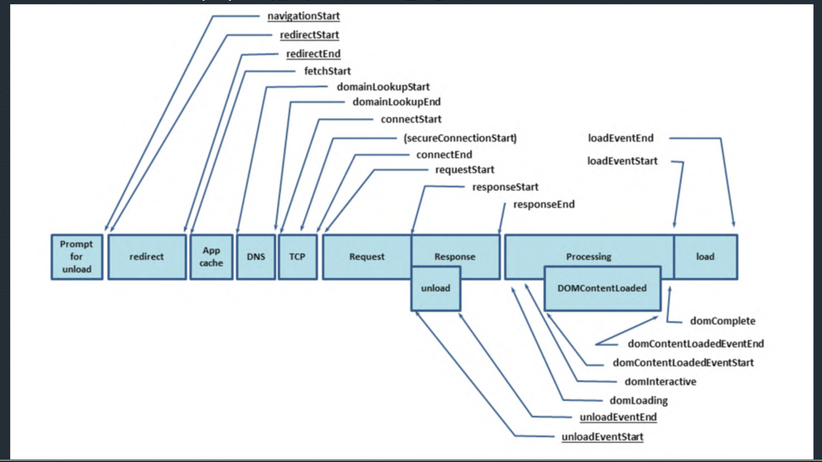

APP ABCD
概览 | 应用(1) | 事务(TOP10) | 数据库(8) | 主机(1)
请求应用 | 数据库类型 | 数据库实例 | 平均操作耗时 | 吞吐率
app.a.com:80 | mysql | dbslavedb2.mys | 500.42 ms | 2.23k rpm
app.a.com:80 | mysql | slavedb4.mysql.r. | 300.34 ms | 1.58k rpm
app.a.com:80 | mysql | slavedb2.mysql.r... | 95.22k | 1.59k rpm
app.a.com:80 | mysql | slavedb3.mysql.r... | 93.25k | 1.55k rpm
app.a.com:80 | mysql | dbslavedb1.mysq... | 2.82 ms | 123.16k | 2.05k rpm

HOW TO MAKE PHPAGENT WORKING

怎样感知终端用户体验

1 window.performance

IE9+ Chrome11+ Firefox7+
IE6,7,8 需采用补足方案



2 window.onerror

停止无休止的try catch

```
/**
 * @param {String}  errMessage   错误信息
 * @param {String}  scriptURL    错误文件URL
 * @param {Long}    lineNumber   错误代码行号
 * @param {Long}    columnNumber 错误代码列号
 * @param {Object}  errObj       错误信息对象
 */
window.onerror = function(errMessage, scriptURL, lineNumber,columnNumber,errObj) {
    var s = {};
    s.msg = errMessage;
    s.url = scriptURL;
    s.line = lineNumber;
    s.column = columnNumber && columnNumber ? columnNumber : 0;
    s.detail = errObj && errObj.stack ? errObj.stack : "";
    EndUserAgent.error_list.push(s);
}
```

怎样感知终端用户体验

3 window.XMLHttpRequest

无侵入Hook所有Ajax请求

```
// readyState --- 0: 请求未初始化, 1: 服务器连接已建立,2: 请求已接收, 3: 请求处理中,4: 请求已完成, 且响应已就绪
if (window.XMLHttpRequest) {
    var original_XMLHttpRequest = XMLHttpRequest;
    XMLHttpRequest = function () {
        var xhr = new original_XMLHttpRequest(arguments);
        util.emit('new-xhr', [xhr]);
        xhr.onreadystatechange = function () {
        };
        if (original_XMLHttpRequest.prototype.addEventListener) {
            xhr.addEventListener('readystatechange', function () {
                util.eventLisenerWrapper(this, fns, "-xhr-");
            });
            util.eventLisenerWrapper(xhr, ["addEventListener", "removeEventListener"], '-xhr-');
            xhr.addEventListener('loadstart', function () {
            }, false);
        } else {
            xhr.onreadystatechange = function () {
            };
        }
        return xhr;
    };
    XMLHttpRequest.my_original = original_XMLHttpRequest;
    XMLHttpRequest.prototype = original_XMLHttpRequest.prototype;

    /* 执行open方法开始之前 */
    util.on("open-xhr-start", function (xhr, args) {
        util.eventLisenerWrapper(xhr, fns, "-xhr-");
    });

    /* send方法开始 */
    util.on("send-xhr-start", function (xhr, args) {
        util.eventLisenerWrapper(xhr, fns, "-xhr-");
    });
```

4 Data Send

Ajax & Image

```
if (window.XMLHttpRequest) {
    // 重写XMLHttpRequest方法内，创建xhr时触发
    util.on('new-xhr', function (xhr, args) {
        xhr.my_metrics = {
            eve_type: 'ajax'
        };
    });

    /* 执行open方法开始之前 */
    util.on("open-xhr-start", function (xhr, args) {
        xhr.my_metrics.req_url = urlFilter(args[1]);
        xhr.my_metrics.req_metnod = args[0].toLocaleLowerCase();
        xhr.my_metrics.is_asyn = args[2];
    });

    /* send开始 */
    util.on("send-xhr-start", function (xhr, args) {
        xhr.my_metrics.req_time = (new Date()).getTime();
        xhr.my_metrics.req_size = countSize(args[0]);
    });

    /* onreadystatechange方法开始 */
    util.on("onreadystatechange-xhr-start", function (xhr, args) {
        readystatechangeStart(xhr);
    });
    /* onreadystatechange方法结束 */
    util.on("onreadystatechange-xhr-end", function (xhr, args) {
        readystatechangeEnd(xhr);
    });

    util.on('onload-xhr-start', function (xhr, args) {
        readystatechangeStart(xhr);
    });

    util.on('onload-xhr-end', function (xhr, args) {
        readystatechangeEnd(xhr);
    });
```

```
    // 请求响应结束
    function readystatechangeStart(xhr) {
        // 接收数据的首字节时间
        if (xhr.readyState == 3) {
            xhr.my_metrics.firstbyte_time = (new Date()).getTime();
        }

        if (xhr.readyState == 4) {

            xhr.my_metrics.res_time = xhr.my_metrics.cb_start_time = xhr.my_metrics.lastbyte_time = (new Date()).getTime

            xhr.my_metrics.rep_code = xhr.status;
            xhr.my_metrics.code_text = xhr.statusText;
            xhr.my_metrics.rep_size = responseSize(xhr);
            xhr.my_metrics.timeout = xhr.timeout;
            xhr.my_metrics.is_err = xhr.status < 400 ? 0 : 1; // 0—无错误，1—有错误

        }
    }

    // user callback执行完
    function readystatechangeEnd(xhr) {
        if (xhr.readyState == 4) {
            xhr.my_metrics.cb_end_time = (new Date()).getTime();
            window.my_rum_events.events.push(xhr.my_metrics);
        }
    }
```

怎样感知终端用户体验

5

JS文件注入

php_output_start_internal

Nginx / Apache output filter

```
//输出钩子 添加rum.js
static void output_rum_hook(TSRMLS_D)
{
//AgentDebug("output_rum_hook -> start");
    if (!SMARTAGENT_G(trace_rum)) {
        return;
    }

    if (SG(sapi_headers).http_response_code) {
        char *tsb_output_rum_handler = "my_output_rum_handler";

#if PHP_API_VERSION >= 20100412
        php_output_start_internal(ZEND_STRL(tsb_output_rum_handler), output_rum_handler, 0, PHP_OUTPUT_HANDLER_STDFLAGS ^
TSRMLS_CC);
#else
        php_ob_set_internal_handler(output_rum_handler, 0, tsb_output_rum_handler, PHP_OUTPUT_HANDLER_END TSRMLS_CC);
#endif

    }

//AgentDebug("output_rum_hook -> end");
}
```

```
        if (have_head || have_body_end) {
            if (have_head) {
                have_head_len = strlen(have_head);
                have_script = strstr(output_raw, tsb_REPLACE_TARGET_TAG_SCRIPT);
                have_head_end = strstr(output_raw, tsb_REPLACE_TARGET_TAG_HEAD_END);
            }

            if (have_script && have_head_end) {
                uint have_script_len = strlen(have_script);
                uint have_head_end_len = strlen(have_head_end);
                uint have_script_pos = have_head_len - have_script_len;
                uint have_head_end_pos = have_head_len - have_head_end_len;

                if (have_script_pos < have_head_end_pos) {
                    insert_pos = have_script;
                } else {
                    insert_pos = have_head_end;
                }
            } else if (have_head_end) {
                insert_pos = have_head_end;
            } else {
                need_preload = 0;
                insert_pos = have_body_end;
            }

            uint insert_len = strlen(insert_pos);
            char *request_id_tmp;
            uint request_id_tmp_len;

            if (need_preload) {
                request_id_tmp_len = spprintf(&request_id_tmp,0,"<script>var my_request_id = \"%s\"</script><script src=\"//%s\"></script>",SMARTAGENT_G(my_request_id),2
SMARTAGENT_G(rum_js_preload_path));
            } else {
                request_id_tmp_len = spprintf(&request_id_tmp,0,"<script>var my_request_id =\"%s\"</script><script src=\"//%s\"></script>",SMARTAGENT_G(my_request_id),2
SMARTAGENT_G(rum_js_path));
            }

            char pre_body[output_len - insert_len];
            strncpy(pre_body,output,output_len - insert_len);
            pre_body[output_len - insert_len] = '\0';

            char *out;
            uint out_len = spprintf(&out, 0, "%s%s%s", pre_body, request_id_tmp, insert_pos);

            *handled_output = estrdup(out);
            *handled_output_len = out_len;
```

怎样获取生产错误和异常

```c
//初始化钩子
inline void initErrorHooks(TSRMLS_D)
{
    if (SMARTAGENT_G(trace_error)) {
        old_error_cb = zend_error_cb;
        zend_error_cb = my_error_cb;
    }

    if (SMARTAGENT_G(trace_exception)) {
        if (zend_throw_exception_hook) {
            old_throw_exception_hook = zend_throw_exception_hook;
        }

        zend_throw_exception_hook = my_throw_exception_hook;
    }
}

//恢复钩子
inline void recoveryErrorHooks(TSRMLS_D)
{
    if (SMARTAGENT_G(trace_error)) {
        if (old_error_cb) {
            zend_error_cb = old_error_cb;
        }
    }

    if (SMARTAGENT_G(trace_exception)) {
        if (old_throw_exception_hook) {
            zend_throw_exception_hook = old_throw_exception_hook;
        }
    }
}
```

```c
//error call back
void my_error_cb(int type, const char *error_filename, const uint error_lineno, const char *format, va_list args)
{
    if (type == E_ERROR || type == E_PARSE || type == E_CORE_ERROR || type == E_COMPILE_ERROR || type == E_USER_ERROR || type == E_RECOVERABLE_ERROR) {
        char *msg;
        va_list args_copy;
        TSRMLS_FETCH();

        va_copy(args_copy, args);
        vspprintf(&msg, 0, format, args_copy);
        va_end(args_copy);

        MAKE_STD_ZVAL(SMARTAGENT_G(error_detail));
        array_init(SMARTAGENT_G(error_detail));
        add_assoc_long_ex(SMARTAGENT_G(error_detail),"type",5,type);
        add_assoc_string_ex(SMARTAGENT_G(error_detail),"file",5,(char *) error_filename,1);
        add_assoc_long_ex(SMARTAGENT_G(error_detail),"line",5,error_lineno);
        add_assoc_string_ex(SMARTAGENT_G(error_detail),"msg",4,msg,1);

    }

    old_error_cb(type, error_filename, error_lineno, format, args);
}
```

```c
//exception hook
void my_throw_exception_hook(zval *exception TSRMLS_DC)
{
    zval *message, *file, *line, *code;
#if PHP_VERSION_ID >= 70000
    zval rv;
#endif
    zend_class_entry *default_ce;

    if (!exception) {
        return;
    }

    default_ce = zend_exception_get_default(TSRMLS_C);

#if PHP_VERSION_ID >= 70000
    message = zend_read_property(default_ce, exception, "message", sizeof("message")-1, 0, &rv);
    file = zend_read_property(default_ce, exception, "file", sizeof("file")-1, 0, &rv);
    line = zend_read_property(default_ce, exception, "line", sizeof("line")-1, 0, &rv);
    code = zend_read_property(default_ce, exception, "code", sizeof("code")-1, 0, &rv);
#else
    message = zend_read_property(default_ce, exception, "message", sizeof("message")-1, 0 TSRMLS_CC);
    file = zend_read_property(default_ce, exception, "file", sizeof("file")-1, 0 TSRMLS_CC);
    line = zend_read_property(default_ce, exception, "line", sizeof("line")-1, 0 TSRMLS_CC);
    code = zend_read_property(default_ce, exception, "code", sizeof("code")-1, 0 TSRMLS_CC);
#endif

    char *error_file = estrdup(Z_STRVAL_P(file));
    ulong line_no = Z_LVAL_P(line);
    char *error_msg = estrdup(Z_STRVAL_P(message));
    ulong exception_type = Z_LVAL_P(code);
    if (!exception_type) exception_type = E_EXCEPTION;

    char *exception_key = "";
    int exception_key_len = 0;
    exception_key_len = spprintf(&exception_key, 0, "%s:%ld", error_file, line_no);

    HashTable *maps_ht,*maps_keys_ht,*_to_ht;
    zval      **_maps_data_;

    if(!SMARTAGENT_G(exception_maps) || !SMARTAGENT_G(exception_maps_keys)) {
        if (old_throw_exception_hook) {
            old_throw_exception_hook(exception TSRMLS_CC);
        }
        return;
    }
```

怎样获取运行时代码栈

Rewrite
- zend_execute_ex
- zend_execute_internal

Ginit
- Build hashmap white functions

Minit & Rinit
- Backup zend_execute_ex & zend_execute_internal
- Rewrite zend_execute_ex & zend_execute_internal

Runtime
- Run my_zend_execute_ex before zend_execute_ex
  - Get class name & function name
  - Match function name & function params & get start time & get start memory
  - Get end time & get end memory
- Run zend_execute_ex
- Build functions map & stack tree

Rshutdown & Mshutdown
- Give back zend_execute_ex & zend_execute_internal

Rewrite
- native function

Ginit
- Build proxy white functions
- Rewrite native function by proxy function

Runtime
- Run proxy function before native function
  - Match function params & get start time & get start memory
  - Get end time & get end memory
- Run native function
- Build functions map & stack tree

REWRITE ZEND_DLEXPORT

Rewrite
- zend_execute_ex
- zend_execute_internal

```
typedef struct _func_entry_t
{
    int     is_null;        //is null
    char    *mn;            //方法名
    int     is_white;       //是否白名单
    int     mn_l;           //方法名长度
} func_entry_t;

typedef struct _method_entry_t
{
    long int        level;          //节点级别
    char            *from;          //来源方法
    int             from_l;         //来源方法名长度
    long int        from_node;      //来源方法node_id
    func_entry_t    *func_entry;
    long int        node_id;        //当前node_id
    long            code_line;      //方法所在的行数
    zval            *params;        //参数

    struct rusage   start_ru;
    struct rusage   end_ru;

    long int        start_mu;
    long int        start_pmu;
    uint64          start_tsc;

    long int        end_mu;
    long int        end_pmu;
    uint64          end_tsc;

    char            *tree[TREE_LEVEL_MAX];
    int             has_tree;
    struct          method_entry_t *prev_entry;
} method_entry_t;

typedef struct _zptr_entry_t
{
    zval *ptr;
    method_entry_t *prev_entry;
} zptr_entry_t;
```

```
    //初始化类的白名单
    zend_hash_init(&smart_agent_globals->classMethodWhite, 0, NULL, NULL, 1);


    //PDO
    HashTable pdoMethod;
    zend_hash_init(&pdoMethod, 0, NULL, NULL, 1);
    zend_hash_add_empty_element(&pdoMethod,TS_S_L("__construct"));
    zend_hash_add_empty_element(&pdoMethod,TS_S_L("query"));
    zend_hash_add(&smart_agent_globals->classMethodWhite, "PDO",strlen("PDO")+1, (void *)&pdoMethod, 2
%sizeof(HashTable), NULL);

    //PDOStatement
    HashTable pdoStatementMethod;
    zend_hash_init(&pdoStatementMethod, 0, NULL, NULL, 1);
    zend_hash_add_empty_element(&pdoStatementMethod,TS_S_L("execute"));
    zend_hash_add(&smart_agent_globals->classMethodWhite, "PDOStatement", strlen("PDOStatement")+1, 2
%(void *)&pdoStatementMethod, sizeof(HashTable), NULL);

    //mysqli
    HashTable mysqliMethod;
    zend_hash_init(&mysqliMethod, 0, NULL, NULL, 1);
    zend_hash_add_empty_element(&mysqliMethod,TS_S_L("mysqli"));
    zend_hash_add_empty_element(&mysqliMethod,TS_S_L("query"));
    zend_hash_add(&smart_agent_globals->classMethodWhite, "mysqli",strlen("mysqli")+1, (void 2
%*)&mysqliMethod, sizeof(HashTable), NULL);

    //mysqli_stmt
    HashTable mysqliStmtMethod;
    zend_hash_init(&mysqliStmtMethod, 0, NULL, NULL, 1);
    zend_hash_add_empty_element(&mysqliStmtMethod,TS_S_L("prepare"));
    zend_hash_add_empty_element(&mysqliStmtMethod,TS_S_L("execute"));
    zend_hash_add(&smart_agent_globals->classMethodWhite, "mysqli_stmt",strlen("mysqli_stmt")+1, 2
%(void *)&mysqliStmtMethod, sizeof(HashTable), NULL);
```

```
#if PHP_VERSION_ID < 50500
    _zend_execute = zend_execute;
    zend_execute  = my_execute;
#else
    _zend_execute_ex = zend_execute_ex;
    zend_execute_ex  = my_execute_ex;
#endif

    _zend_execute_internal = zend_execute_internal;
    zend_execute_internal = my_execute_internal;
```

```
#if PHP_VERSION_ID < 50500
ZEND_DLEXPORT void my_execute (zend_op_array *ops TSRMLS_DC)
{
#else
ZEND_DLEXPORT void my_execute_ex (zend_execute_data *execute_data TSRMLS_DC)
{
    zend_op_array *ops = execute_data->op_array;
#endif

    func_entry_t  *func;
    int  code_line = 0;
    zend_execute_data *data = EG(current_execute_data);

    func = my_get_function_name(data TSRMLS_CC);

    if (func->mn) {
        method_entry_t *method_entry = ecalloc(sizeof(method_entry_t),1);
        METHOD_START(&SMARTAGENT_G(entries),method_entry,data,func);

#if PHP_VERSION_ID < 50500
        _zend_execute(ops TSRMLS_CC);
#else
        _zend_execute_ex(execute_data TSRMLS_CC);
#endif

        METHOD_END(&SMARTAGENT_G(entries),method_entry,data,func);
        efree(func->mn);
        efree(method_entry);
    } else {

#if PHP_VERSION_ID < 50500
        _zend_execute(ops TSRMLS_CC);
#else
        _zend_execute_ex(execute_data TSRMLS_CC);
#endif
    }
    efree(func);
}
```

REWRITE ZEND_DLEXPORT

Rewrite
- zend_execute_ex
- zend_execute_internal

```
#define METHOD_START(entries,method_entry,data,func)                    \
do{                                                                     \
    method_entry->start_mu = zend_memory_usage(0 TSRMLS_CC);            \
    method_entry->start_pmu = zend_memory_peak_usage(0 TSRMLS_CC);      \
    method_entry->start_tsc = cycle_timer(TSRMLS_C);                    \
    getrusage(RUSAGE_SELF, &method_entry->start_ru);                    \
    method_entry->func_entry = (func);                                  \
    method_entry->code_line = get_method_code_line(data TSRMLS_CC);     \
    method_entry->prev_entry = (*(entries));                            \
    method_entry->level = (*(entries))->level + 1;                      \
    method_entry->from_node = (*(entries))->node_id;                    \
    int node_id = process_node_id(func->mn,func->mn_l TSRMLS_CC);       \
    method_entry->node_id = node_id;                                    \
    assemble_trace_tree((method_entry) TSRMLS_CC);                      \
    method_entry->params = get_method_params((func),data TSRMLS_CC);    \
    (*(entries)) = (method_entry);                                      \
}while(0)

#define METHOD_END(entries,method_entry,data,func)                      \
do{                                                                     \
    method_entry->end_mu = zend_memory_usage(0 TSRMLS_CC);              \
    method_entry->end_pmu = zend_memory_peak_usage(0 TSRMLS_CC);        \
    method_entry->end_tsc = cycle_timer(TSRMLS_C);                      \
    getrusage(RUSAGE_SELF, &method_entry->end_ru);                      \
    assemble_maps_data((method_entry), data TSRMLS_CC);                 \
    (*(entries)) = (method_entry_t *)(*(entries))->prev_entry;          \
    if (method_entry->params) {                                         \
        EX_ARRAY_DESTROY(method_entry->params);                         \
    }                                                                   \
} while(0)
```

```
//计算start节点entry开始值
inline int processStartEntryBegin(TSRMLS_D)
{
    method_entry_t *method_entry_start = ecalloc(sizeof(method_entry_t),1);
    method_entry_start->start_mu = zend_memory_usage(0 TSRMLS_CC);
    method_entry_start->start_pmu = zend_memory_peak_usage(0 TSRMLS_CC);
    method_entry_start->start_tsc = cycle_timer(TSRMLS_C);
    getrusage(RUSAGE_SELF, &method_entry_start->start_ru);
    SMARTAGENT_G(start_method_entry) = method_entry_start;

    return SUCCESS;
}

//计算start节点entry结束值
inline int processStartEntryEnd(TSRMLS_D)
{
    SMARTAGENT_G(start_method_entry)->end_mu = zend_memory_usage(0 TSRMLS_CC);
    SMARTAGENT_G(start_method_entry)->end_pmu = zend_memory_peak_usage(0 TSRMLS_CC);
    SMARTAGENT_G(start_method_entry)->end_tsc = cycle_timer(TSRMLS_C);
    getrusage(RUSAGE_SELF, &SMARTAGENT_G(start_method_entry)->end_ru);

    return SUCCESS;
}
```

```
//根据maps的数据
static void assemble_maps_data(method_entry_t *method_entry, zend_execute_data *execute_data TSRMLS_DC) {
    zval *data;
    void **data;

    zval *tmpCopy;
    MAKE_STD_ZVAL(tmpCopy);
    array_init(tmpCopy);

    long int wt = method_entry->end_tsc - method_entry->start_tsc;

    long int cpu = get_us_interval(&(method_entry->start_ru.ru_utime), &(method_entry->end_ru.ru_utime)) +
                   get_us_interval(&(method_entry->start_ru.ru_stime), &(method_entry->end_ru.ru_stime));

    long int mu = method_entry->end_mu - method_entry->start_mu;
    long int pmu = method_entry->end_pmu - method_entry->start_pmu;

    func_entry_t *func_entry = method_entry->func_entry;

    zval *vPsData = NULL;

    if(zend_hash_index_find(HASH_OF(SMARTAGENT_G(trace_maps)),method_entry->node_id,&data) == SUCCESS) {
        _data = *(zval **) data;
        HashTable *_ht;
        _ht = HASH_OF(_data);

        void *vCtData, *vWtData, *vCpuData, *vMuData, *vPmuData;

        ulong ctHashVal = zend_get_hash_value("ct",3);
        ulong wtHashVal = zend_get_hash_value("wt",3);
        ulong cpuHashVal = zend_get_hash_value("cpu",4);
        ulong muHashVal = zend_get_hash_value("mu",3);
        ulong pmuHashVal = zend_get_hash_value("pmu",4);
        ulong psHashVal = zend_get_hash_value("ps",3);

        //update 调用次数
        zend_hash_quick_find(_ht,"ct",3,ctHashVal,&vCtData);
        Z_LVAL_PP((zval **)vCtData) += 1; //调用次数+1

        //update 执行时间
        zend_hash_quick_find(_ht,"wt",3,wtHashVal,&vWtData);
        Z_LVAL_PP((zval **)vWtData) += wt;

        //update cpu时间
        zend_hash_quick_find(_ht,"cpu",4,cpuHashVal,&vCpuData);
        Z_LVAL_PP((zval **)vCpuData) += cpu;

        //update mu的内存使用
        zend_hash_quick_find(_ht,"mu",3,muHashVal,&vMuData);
        Z_LVAL_PP((zval **)vMuData) += mu;

        //update mu的内存使用
        zend_hash_quick_find(_ht,"pmu",4,pmuHashVal,&vPmuData);
        Z_LVAL_PP((zval **)vPmuData) += pmu;
    } else {
        if (wt > SMARTAGENT_G(user_function_time_limit_bar) && method_entry->func_entry->is_white == FAILURE) {
            while(DelTree(SMARTAGENT_G(trace_tree),method_entry TSRMLS_CC);
            return;
        }

        add_assoc_long_ex(tmpCopy, "pid", 4, method_entry->node_id);
        add_assoc_string_ex(tmpCopy,"mn",3,func_entry->mn,1);
        add_assoc_long_ex(tmpCopy, "cl", 3, method_entry->code_line);
        add_assoc_long_ex(tmpCopy, "rt", 3, (long) method_entry->start_tsc / 1000);
        add_assoc_long_ex(tmpCopy, "ct", 3, 1);
        add_assoc_long_ex(tmpCopy, "wt", 3, wt);
        add_assoc_long_ex(tmpCopy, "cpu", 4, cpu);
        add_assoc_long_ex(tmpCopy, "mu", 3, mu);
        add_assoc_long_ex(tmpCopy, "pmu", 4, pmu);


        add_index_zval(SMARTAGENT_G(trace_maps),method_entry->node_id,tmpCopy);
    }

    setMethodCommonProperty(func_entry->mn,method_entry->params,tmpCopy,vPsData,execute_data TSRMLS_CC);
#if ZEND_DEBUG
    zptr_entry_t *zptr_entry = my_alloc_zptr_entry(TSRMLS_C);
    zptr_entry->ptr = tmpCopy;
#endif
}
```

REWRITE FUNCTION

Rewrite
- native function

```
    override_func_t *override_curl_close = emalloc(sizeof(override_func_t));
    override_curl_close->func = PHP_FN(my_curl_close);
    php_override_func("curl_exec", override_curl_close->func, &override_curl_close->old_func TSRMLS_CC);
    zend_hash_add(&smart_agent_globals->overrideFunctions, "curl_close",
        strlen("curl_close") +1, (void*)override_curl_close, sizeof(override_func_t), NULL);
    efree(override_curl_close);

    override_func_t *override_cls_mysqli_prepare = emalloc(sizeof(override_func_t));
    override_cls_mysqli_prepare->func = PHP_MN(my_mysqli_prepare);
    php_override_cls_func("mysqli", "prepare", override_cls_mysqli_prepare->func, &
&override_cls_mysqli_prepare->old_func TSRMLS_CC);
    zend_hash_add(&smart_agent_globals->overrideFunctions, "mysqli_prepare_c",
        strlen("mysqli_prepare_c") +1, (void*)override_cls_mysqli_prepare, sizeof(override_func_t), NULL);
    efree(override_cls_mysqli_prepare);

    override_func_t *override_cls_mysqli_stmt_init = emalloc(sizeof(override_func_t));
    override_cls_mysqli_stmt_init->func = PHP_MN(my_mysqli_stmt_init);
    php_override_cls_func("mysqli", "stmt_init", override_cls_mysqli_prepare->func, &
&override_cls_mysqli_prepare->old_func TSRMLS_CC);
    zend_hash_add(&smart_agent_globals->overrideFunctions, "mysqli_stmt_init_c",
        strlen("mysqli_stmt_init_c") +1, (void*)override_cls_mysqli_prepare, sizeof(override_func_t), NULL);
    efree(override_cls_mysqli_stmt_init);

    override_func_t *override_cls_mysqli_stmt_prepare = emalloc(sizeof(override_func_t));
    override_cls_mysqli_stmt_prepare->func = PHP_MN(my_mysqli_stmt_prepare);
    php_override_cls_func("mysqli_stmt", "prepare", override_cls_mysqli_stmt_prepare->func, &
&override_cls_mysqli_stmt_prepare->old_func TSRMLS_CC);
    zend_hash_add(&smart_agent_globals->overrideFunctions, "mysqli_stmt_prepare_c",
        strlen("mysqli_stmt_prepare_c") +1, (void*)override_cls_mysqli_prepare, sizeof(override_func_t), NULL);
    efree(override_cls_mysqli_stmt_prepare);
```

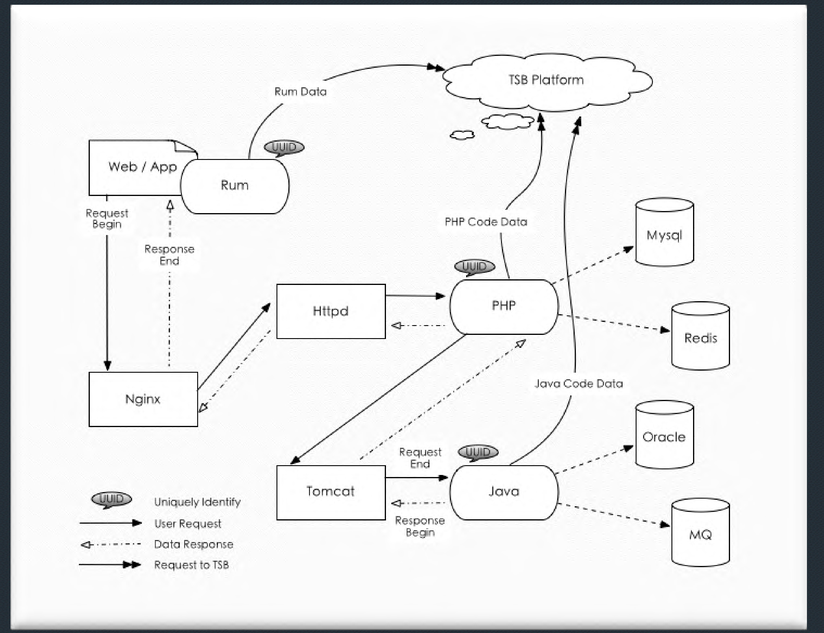
```
static void php_override_func(const char *name, php_func handler, php_func *stash TSRMLS_DC) {
    zend_function *func;
    if (zend_hash_find(CG(function_table), name, strlen(name) +1, (void **)&func) == SUCCESS) {
        if (stash) {
            *stash = func->internal_function.handler;
        }
        func->internal_function.handler = handler;
    }
}

static void php_override_cls_func(const char *cls_name, const char *name, php_func handler, php_func *stash &
TSRMLS_DC)
{
    zend_class_entry **cls_ptr;
    zend_class_entry *cls;
    zend_function *func;
    if (zend_hash_find(CG(class_table), cls_name, strlen(cls_name)+1 , (void **)&cls_ptr) == SUCCESS) {
        cls = *cls_ptr;
        if (zend_hash_find(&cls->function_table, name, strlen(name)+1, (void **)&func) == SUCCESS) {
            if (stash) {
                *stash = func->internal_function.handler;
            }
            func->internal_function.handler = handler;
        }
    }
}
```

```
PHP_METHOD(my_pdo, prepare)
{
    long stmt_lval, instance_lval;
    char *sql;
    zval **query_data;
    zval *params = get_current_method_params(TSRMLS_C);

    void *pdata;
    override_func_t *overide_func;
    if (zend_hash_find(&SMARTAGENT_G(overrideFunctions), "pdo_prepare_c",
        strlen("pdo_prepare_c") + 1, (void **)&pdata) == SUCCESS) {
        overide_func = (override_func_t *)pdata;
        php_func old_func = overide_func->old_func;
        old_func(INTERNAL_FUNCTION_PARAM_PASSTHRU);
    }

    if (zend_hash_index_find(Z_ARRVAL_P(params), 0, (void **)&query_data) == SUCCESS) {
        sql = Z_STRVAL_PP(query_data);
    }

    instance_lval = Z_OBJ_HANDLE_P(this_ptr);
    stmt_lval = Z_OBJ_HANDLE_P(return_value);

    set_db_prepare_stmt_sql(&SMARTAGENT_G(dbPrepareStmtSqlMap), stmt_lval, sql TSRMLS_CC);
    set_db_prepare_stmt_connected(&SMARTAGENT_G(dbPrepareStmtConnectedMap), stmt_lval, instance_lval TSRMLS_CC);
}
```

更快，兼容性更好
更易编码

怎样实现端到端

- 生成Trace ID
- 通过注入JS变量交给浏览器
- 通过Request Header传递
- 接收者Append当前Agent标识
- Server端习得端到端拓扑



回顾一下

- 什么是APM
  - APM的定义
  - APM的五个层次要求
  - APM的优势和难点
- APM对 PHP应用意味着什么
  - 准确感知终端用户体验
  - 运行时监控
  - 洞察业务故障
  - 预测架构瓶颈
- 动手实现一个PHPAgent
  - 怎样实现注入JS感知终端用户体验
  - 怎样获取生产的错误和异常
  - 怎样获取运行时代码栈
  - 怎样实现端到端数据收集

可供参考的项目

- RUM
  - https://www.w3.org/2013/Talks/0516-webperf
- PHPAgent
  - Xhprof
  - Xdebug
  - SeasLog
  - PHP源码

Thanks & QA

github.com/neeke

PHP 2017·北京 全球开发者大会