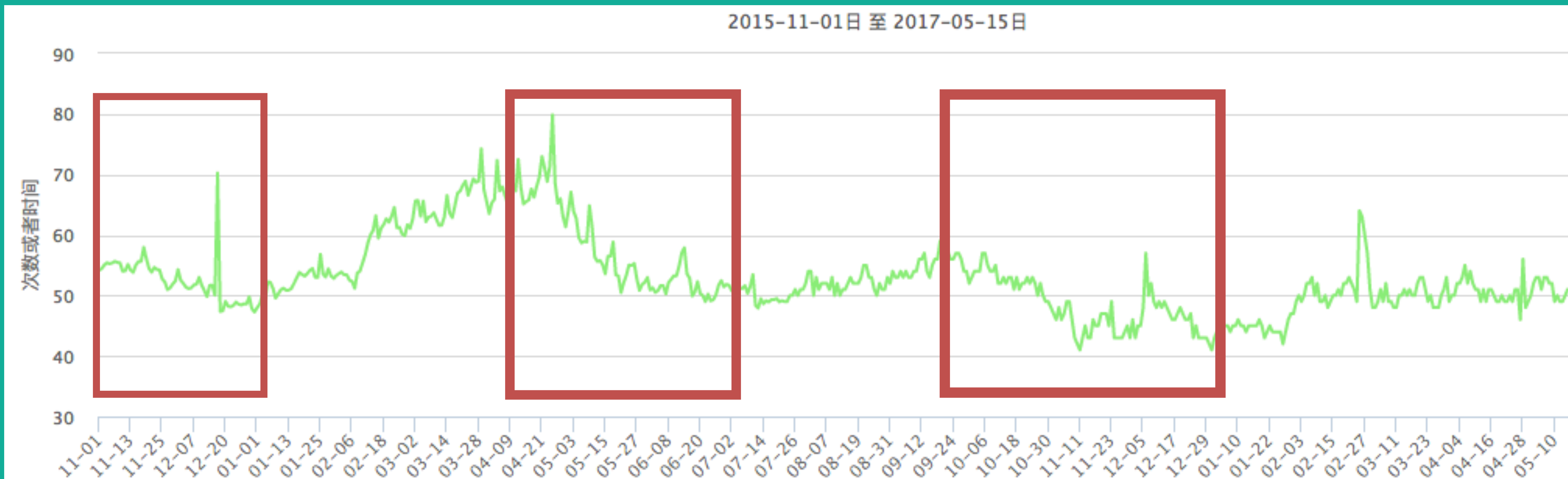


# 复杂PHP系统性能瓶颈排查及优化

MOMO陌陌 高永芝  
Glowdan

# MOMO API 耗时走势



# PHP 系统性能优化方案



## 代码分析

- 性能更好的语法
- SQL 优化
- ...

## 性能更高的工具或者系统版本

- php5 替换为 php7
- 开启Opcache
- 使用扩展替换php代码
- ...

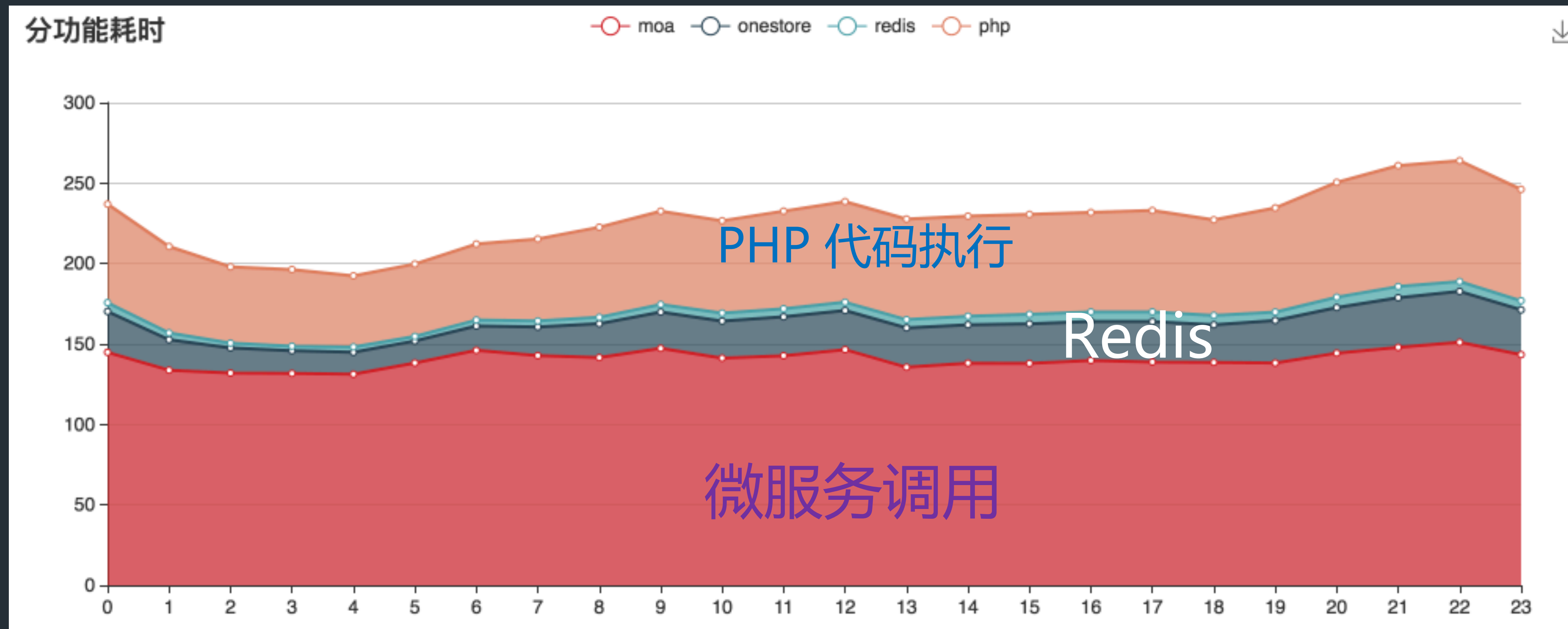
## 压力测试

- 负载高导致有问题的节点
- ...

## 架构优化

- MySQL转换为Redis
- 微服务
- ...

# “附近的人” 耗时分布





# 优化方向的选择



优化重点接口



优化有问题的代码

优化语言层面



优化外部调用

重构代码



重构服务



通过对**Xhprof**数据进行**筛选、统计**  
**分析**出系统瓶颈，进而进行性能调优

## 性能检测工具Xhprof

**Xhprof** 是轻量级的分层性能测量分析器。在数据收集阶段，它跟踪调用次数与测量数据。

能够检测每一个函数，方法调用的执行时间，调用次数，内存占用，**CPU**消耗等信息。

提供了完整的运行时调用链。可以跟踪每一个函数，方法的调用关系。

## Xhprof在陌陌的使用

部署于所有服务器上。

按照不同需要，进行不同比例的采样。通用比例为万分之一。

生成的数据，使用**Redis**的pub功能存储到**Mysql**中。

每天定时对数据进行统计处理。



# 数据筛选及统计



# 按照url进行数据统计



api性能分析2017-06-05			
类型	昨天	今天	变化率
请求总量(次)			0.71 %
总耗时(ms)			2.48 %
平均耗时(ms)	49.0000	49	0.83 %

总耗时排名前50的为						
uri	次数(昨)	次数(今)	变化率	平均耗时(ms)(昨)	平均耗时(ms)(今)	变化率
			-1.86 %	231	239.64	3.74 %
			1.48 %	179	179.21	0.12 %
			-2.73 %	173	174.32	0.76 %
			-1.46 %	72	74.78	3.86 %
			-1.47 %	124	125.79	1.44 %
			-2.49 %	53	54.32	2.49 %
			4.87 %	53	54.73	3.26 %
			-2.18 %	301	316.95	5.30 %
			3.66 %	219	222.60	1.64 %
			-2.67 %	146	152.10	4.18 %
			-0.42 %	81	82.89	2.33 %

平均耗时排名前50的为						
uri	次数(昨)	次数(今)	变化率	平均耗时(ms)(昨)	平均耗时(ms)(今)	变化率
			7.29 %	493	510.45	3.54 %
			-15.38 %	487	428.60	-11.99 %
			23.44 %	376	385.02	2.40 %
			1.41 %	351	370.63	5.59 %
			-22.86 %	326	349.92	7.34 %
			15.57 %	361	347.23	-3.81 %
			2.56 %	325	338.62	4.19 %
			4.92 %	320	332.64	3.95 %
			5.22 %	325	320.95	-1.25 %
			-2.18 %	301	316.95	5.30 %



# 将函数或者方法从列表拆分，筛选出调用次数和耗时多的函数



stack	ct	wt	cpu	mu	pmu	uri wt(ms)	uri ct(1/10000)
V1_Nearby_Controller::index==>UserNearbyManager::searchNearbyMoa	14892	144.20	28.94	144203.84	144203.84	233.45	14916
Mpf_App::run==>V1_Nearby_Controller::index	14916	143.97	52.64	143971.82	143971.82	233.45	14916
main()=>Mpf_App::run	14916	143.97	57.29	143971.82	143971.82	233.45	14916
main()	14916	143.97	57.49	143971.82	143971.82	233.45	14916
UserNearbyManager::searchNearbyMoa==>UserNearbyManager::parallelMoa	14890	95.87	7.88	144223.21	144223.21	233.45	14916
UserNearbyManager::parallelMoa==>Base_Parallel_Ctx::execute	14890	88.75	1.31	144223.21	144223.21	233.45	14916
Base_Parallel_Ctx::execute==>Base_Moa_Client::execute	14890	88.50	1.14	144223.21	144223.21	233.45	14916
Base_Moa_Client::execute==>Base_Moa_Client::__call	14890	88.49	1.14	144223.21	144223.21	233.45	14916
Base_Shout_Manager_Publish::nearbyInterestRecommend==>Base_Shout_Manager_Publish::searchNearbyMomentUsers	290	85.84	1.31	22164.00	0.00	233.45	14916
Base_Moa_Client::searchNearbyVideoUser==>Base_Moa_Client::__call	290	81.96	0.61	5193.74	0.00	233.45	14916

单次请求调用函数耗时排行

stack	ct	wt	cpu	mu	pmu	uri wt(ms)	uri ct(1/10000)
main()=>load::code_base/init.php	14916	0.04	0.00	12426.74	9483.06	233.45	14916
run_init::code_base/init.php==>load::mpf/autoload.php	14916	0.00	0.00	1096.00	1096.00	233.45	14916
run_init::code_base/init.php==>run_init::mpf/autoload.php	14916	0.00	0.00	568.00	496.00	233.45	14916
run_init::code_base/init.php==>Mpf_Autoloader::registerAutoload	14916	0.01	0.00	1048.00	808.00	233.45	14916
run_init::code_base/init.php==>load::core/classloader.php	14916	0.00	0.00	1480.00	1432.00	233.45	14916
run_init::core/classloader.php==>load::helper/cache_helper.php	14916	0.01	0.00	4056.00	4000.00	233.45	14916
run_init::helper/cache_helper.php==>CacheHelper_Yac::__construct	14916	0.00	0.00	648.00	648.00	233.45	14916
run_init::helper/cache_helper.php==>CacheHelper::setAdapter	14916	0.00	0.00	568.00	560.00	233.45	14916
run_init::core/classloader.php==>run_init::helper/cache_helper.php	14916	0.02	0.00	2320.00	2320.00	233.45	14916
run_init::code_base/init.php==>run_init::core/classloader.php	14916	0.03	0.01	6848.00	6760.00	233.45	14916
CacheHelper_Yac::formatKey==>strlen	4604884	0.00	0.00	1.74	1.74	233.45	14916
CacheHelper_Yac::get==>CacheHelper_Yac::formatKey	2502290	0.00	0.00	6.87	6.87	233.45	14916

单次请求调用次数排行

# 筛选数据库和微服务调用次数及耗时进行统计

uri	moa	moa wt	moa ct	uri wt	uri ct	weight
xxxxxx/index	getOnestore	23.81	267	173.60	15240	635.83
xxxxxx/index	getService	23.32	627	173.60	15240	1462.41
xxxxxx/index	queryMarkingEntityForSingle	12.25	15233	173.60	15240	18666.63
xxxxxx/index	queryFeeds	11.79	15146	173.60	15240	17862.58
xxxxxx/index	getFeedsCount	10.36	13019	173.60	15240	13491.36
xxxxxx/index	getMicroVideoCount	7.87	15146	173.60	15240	11925.94
xxxxxx/index	write	7.84	14969	173.60	15240	11731.78
xxxxxx/index	exists	6.04	14969	173.60	15240	9040.40

微服务耗时排行

uri	moa	moa wt	moa ct	uri wt	uri ct	weight
xxxxxx/index	mgetResult	1.82	54003	173.60	15240	9810.28
xxxxxx/index	getUserKickType	0.96	30253	173.60	15240	2916.72
xxxxxx/index	queryUserLocations	1.86	30092	173.60	15240	5603.10
xxxxxx/index	queryFeedListByIds	2.73	27036	173.60	15240	7371.05
xxxxxx/index	queryCommunitySiteById	1.55	25890	173.60	15240	4012.97
xxxxxx/index	queryFeedByFeedId	0.93	15482	173.60	15240	1436.31
xxxxxx/index	apiAuth	1.01	15240	173.60	15240	1539.43
xxxxxx/index	getUserLevelPrivilegeBatch	4.95	15234	173.60	15240	7539.00

微服务调用次数排行



# 微服务耗时排行榜（调用次数，总耗时）

uri	moa	moa wt	moa ct	uri wt	uri ct	weight
...e/nearby	queryFoursquareSite	2504.95	1	256.32	61	252.06
...rch/map	queryFoursquareSite	1608.14	2	97.03	107	322.67
...p/share/send->	sendGroupActionMessage	292.79	1	285.19	66	31.16
...ts/phone/upload->index	addOrMergeContactsNReturnStatus	291.59	149	326.00	240	4352.57
...ts/phone/contacts->index	getContactsStatus	277.59	10	446.05	10	278.04
...by/index	searchNearbyUserForGuestWithAge	250.46	61	349.17	63	1529.99
...t/weixinpay/sign	addWeiXinPaySdkOrder	240.97	8	238.10	15	193.14
...comment/publish->	sendGroupActionMessage	233.79	1	257.85	1	23.40

全部微服务平均耗时排行

uri	moa	moa wt	moa ct	uri wt	uri ct	weight
...index	mgetJsonResult	1.75	55282	245.35	14661	10061.20
...file/index	mgetJsonResult	1.82	54003	173.60	15240	10074.85
...mon/upload	apiAuth	1.89	52014	10.33	52014	9896.68
...cdn/lists	apiAuth	1.10	41392	9.33	41392	4576.60
...index	queryUserLocations	1.42	41238	245.35	14661	6211.23
...d/look	write	1.96	40447	54.04	18756	8011.88
.../base	mgetJsonResult	1.89	39706	177.30	14982	7779.56
...file/index	mgetJsonResult	1.80	37302	125.91	8892	6822.53

全部微服务调用次数排行



# 计算耗时同比数据



#	uri	moa	wt	wt(昨)	wt rate	ct	ct(昨)	ct rate
172		execute			4.18%			3.16%
128		queryActivateList			2.81%			1.83%
57		findMatchLikeUsersForCardPageV2			7.67%			6.54%
532		queryNearbyFeeds			12.94%			9.43%
171		searchNearByUsersWithAge			0.87%			2.84%
30		exists			11.68%			7.84%
231		queryCommentListByCommentIdList			17.88%			5.87%
104		queryFeeds			1.48%			3.00%
109		queryMarkingEntityForSingle			5.15%			3.16%
138		queryFeeds			3.62%			1.76%

# 整合所有请求数据到一个请求中，看所有的请求的平均水平



CacheHelper_Yac::formatKey==>strlen	5534403	0.00	0.00	1.52	1.52	174.82	15713
CacheHelper_Yac::get==>CacheHelper_Yac::formatKey	3078370	0.00	0.00	5.88	5.88	174.82	15713
CacheHelper::get==>CacheHelper_Yac::get	2780727	0.01	0.00	10.44	10.44	174.82	15713
ClassLoader::_fetchClassPath==>CacheHelper::get	2454045	0.01	0.00	15.78	15.78	174.82	15713
ClassLoader::import==>ClassLoader::_fetchClassPath	1990207	0.01	0.00	24.32	24.32	174.82	15713
CacheHelper_Yac::set==>CacheHelper_Yac::formatKey	2456033	0.00	0.00	3.53	3.53	174.82	15713
CacheHelper::set==>CacheHelper_Yac::set	2454178	0.01	0.00	11.47	12.91	174.82	15713
ClassLoader::_storeClassPath==>CacheHelper::set	2454045	0.01	0.00	15.42	15.42	174.82	15713
ClassLoader::import==>ClassLoader::_storeClassPath	1990207	0.01	0.00	23.87	23.87	174.82	15713
ClassLoader::import==>load::config/momo_setting.php	15713	0.00	0.00	776.00	768.00	174.82	15713
ClassLoader::import==>run_init::config/momo_setting.php	15713	0.04	0.01	5000.00	5000.00	174.82	15713
run_init::code_base/init.php==>ClassLoader::import	62852	0.05	0.01	3266.00	3266.00	174.82	15713
ClassLoader::import==>load::config/momo_setting_dev.php	15713	0.00	0.00	780.07	628.07	174.82	15713
ClassLoader::import==>run_init::config/momo_setting_dev.php	15713	0.02	0.01	17816.00	17224.00	174.82	15713
ClassLoader::import==>load::config/momo_group_config.php	15713	0.00	0.00	792.00	512.00	174.82	15713
ClassLoader::import==>run_init::config/momo_group_config.php	15713	0.00	0.00	568.00	568.00	174.82	15713

Overall Summary  
Total Incl. Wall Time 155,531 (microsec): microsecs  
Total Incl. CPU (microsecs): 37,994 microsecs  
Total Incl. MemUse (bytes): 4,019,680 bytes  
Total Incl. PeakMemUse (bytes): 4,386,736 bytes  
Number of Function Calls: 6,847

[\[View Full Callgraph\]](#)

Displaying top 100 functions: Sorted by Calls [\[ display all \]](#)

Function Name	<a href="#">Calls</a>	<a href="#">Calls%</a>	<a href="#">Incl. Wall Time (microsec)</a>	<a href="#">IWall%</a>	<a href="#">Excl. Wall Time (microsec)</a>	<a href="#">EWall%</a>	<a href="#">Incl. CPU (microsecs)</a>	<a href="#">ICpu%</a>	<a href="#">Excl. CPU (microsec)</a>	<a href="#">ECPU%</a>	<a href="#">Incl. MemUse (bytes)</a>	<a href="#">IM</a>
<a href="#">Mpf_Ctx::_get</a>	575	8.4%	7,136	4.6%	1,056	0.7%	1,000	2.6%	-1,999	-5.3%	644,976	
<a href="#">strlen</a>	402	5.9%	15	0.0%	15	0.0%	0	0.0%	0	0.0%	4,920	
<a href="#">CacheHelper_Yac::formatKey</a>	333	4.9%	720	0.5%	707	0.5%	0	0.0%	0	0.0%	1,704	
<a href="#">BaseContextFactory::_get</a>	307	4.5%	5,779	3.7%	611	0.4%	2,000	5.3%	1,000	2.6%	741,160	
<a href="#">CacheHelper_Yac::get</a>	185	2.7%	2,029	1.3%	1,568	1.0%	0	0.0%	0	0.0%	49,320	
<a href="#">CacheHelper::get</a>	167	2.4%	1,633	1.0%	359	0.2%	0	0.0%	0	0.0%	4,216	
<a href="#">ClassLoader::_storeClassPath</a>	148	2.2%	1,516	1.0%	286	0.2%	0	0.0%	0	0.0%	4,128	
<a href="#">CacheHelper::set</a>	148	2.2%	1,230	0.8%	324	0.2%	0	0.0%	0	0.0%	2,408	
<a href="#">CacheHelper_Yac::set</a>	148	2.2%	906	0.6%	647	0.4%	0	0.0%	0	0.0%	1,792	
<a href="#">ClassLoader::_fetchClassPath</a>	148	2.2%	1,781	1.1%	310	0.2%	0	0.0%	0	0.0%	4,408	



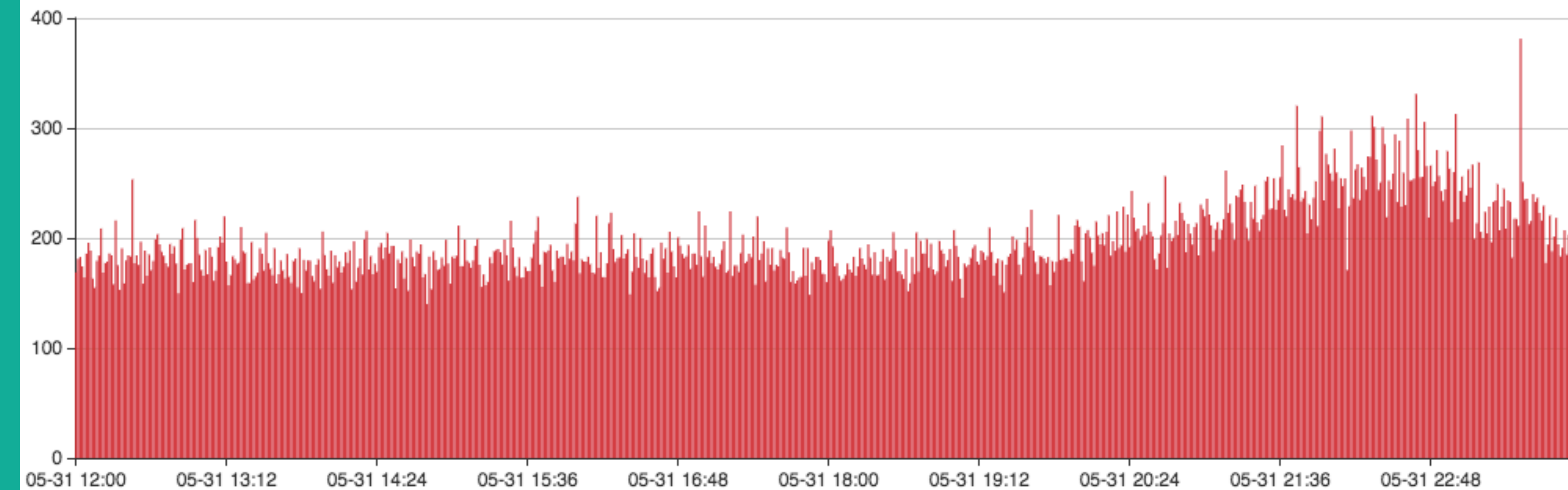
# 可视化性能指标



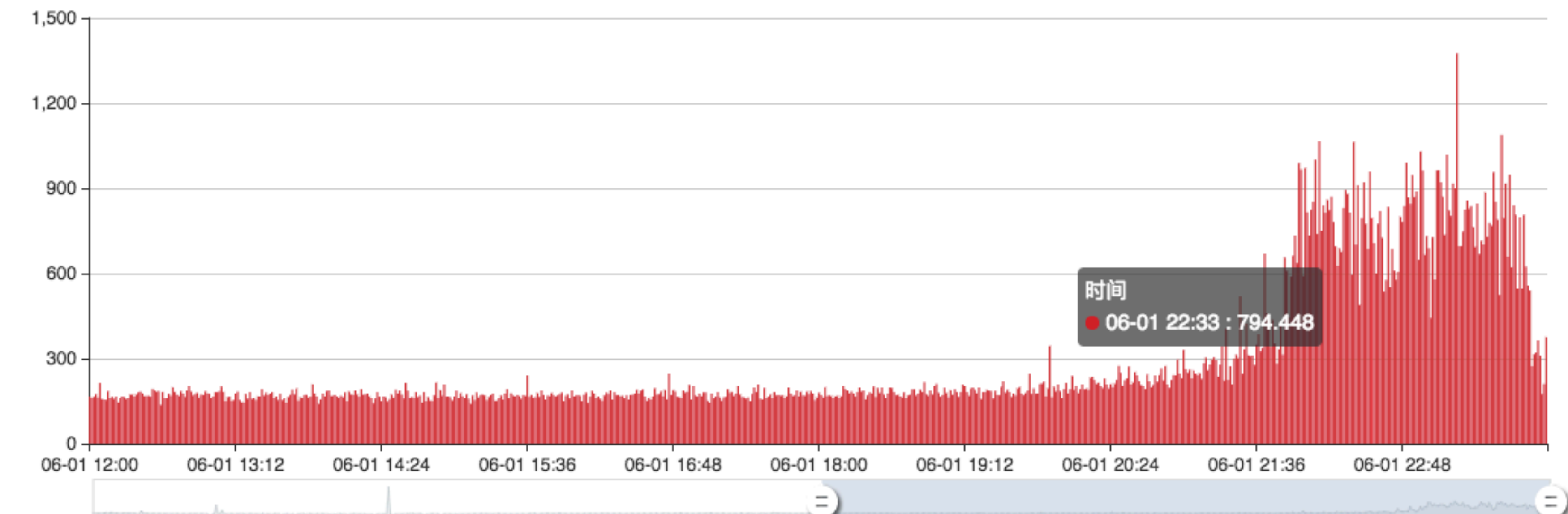
# 耗时分布-按照自然时间展示



ser/my/base 2017/05/31 每分钟耗时

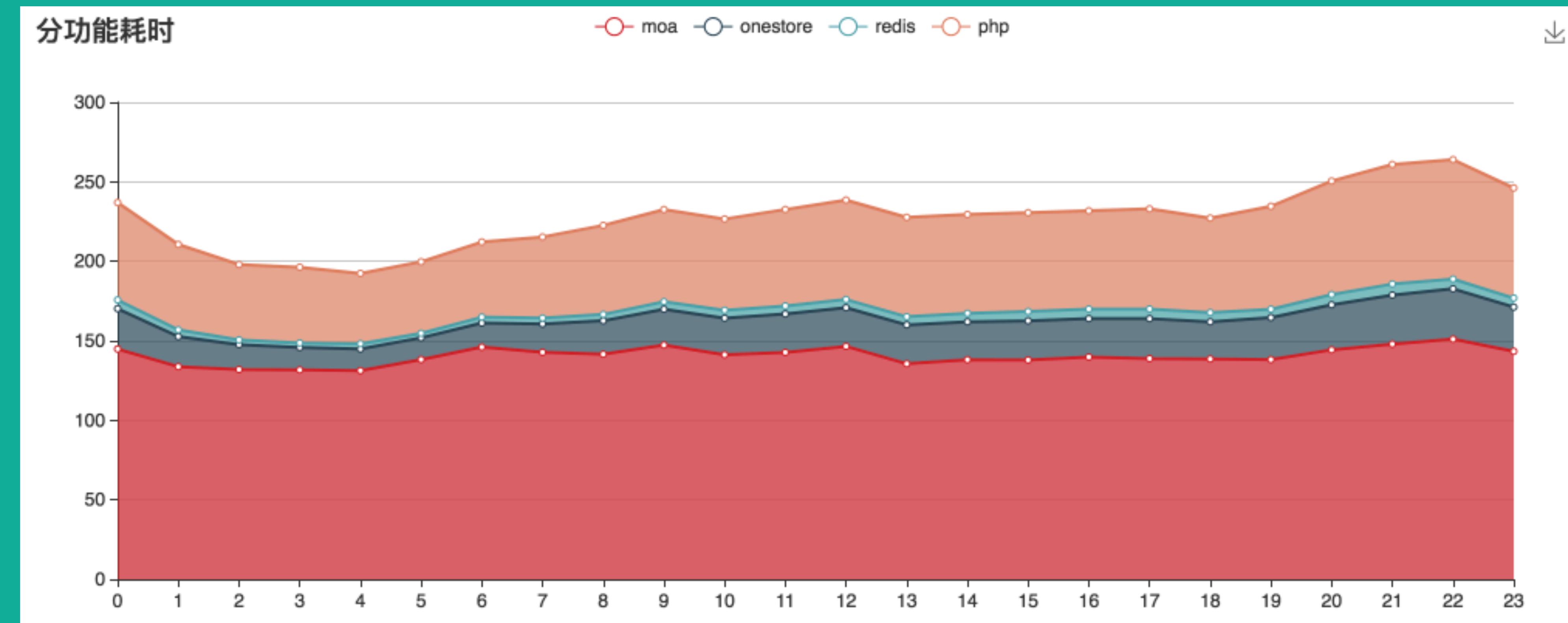
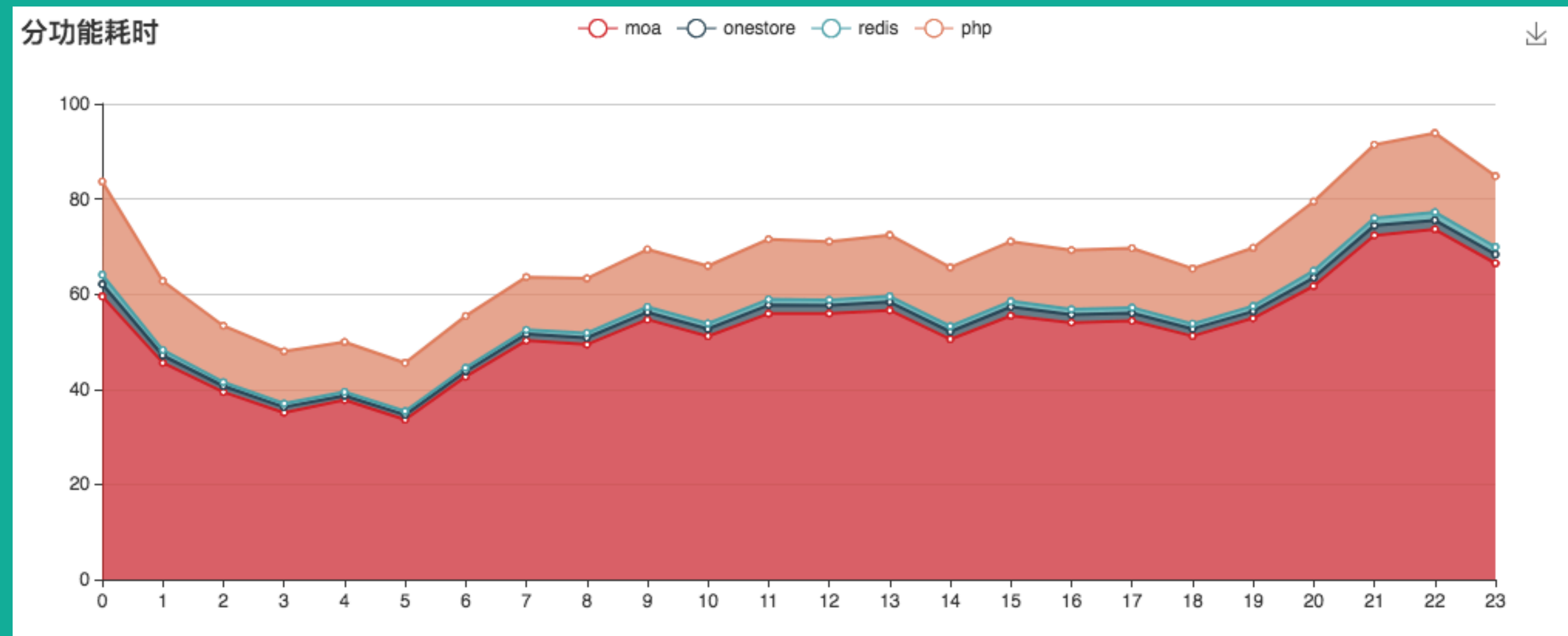


ser/my/base 2017/06/01 每分钟耗时

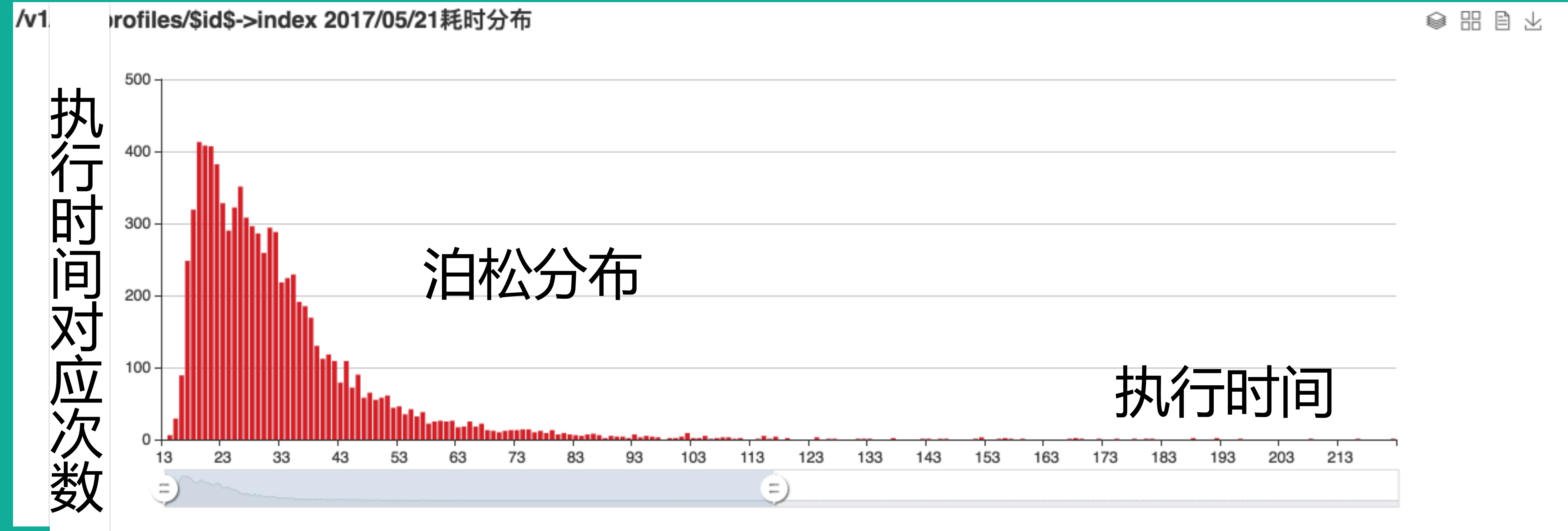




# 分离PHP执行时间和数据库，微服务时间视图



# 耗时分布-按照耗时展示





# 高效优化方案

## 多级缓存

对读取到的数据进行缓存，避免二次调用

缓存在PHP代码中有很多层次。

1. （外部服务）缓存服务，比如Redis, memcache
2. （本机）文件缓存，Yac共享内存
3. （本次调用）程序内缓存



使用全局变量、类中的变量等方法将远程数据进行缓存。  
第二次或更多次调用直接读取。



# 高效优化方案



## 单个请求，替换为批量请求

替换多次调用为批量。

Redis中get换为mget。

微服务单次调用改为同时调用多次



## 预先准备数据

预先将需要用到的数据进行整理，然后统一获取。  
在用到的时候，直接到内存中获取。

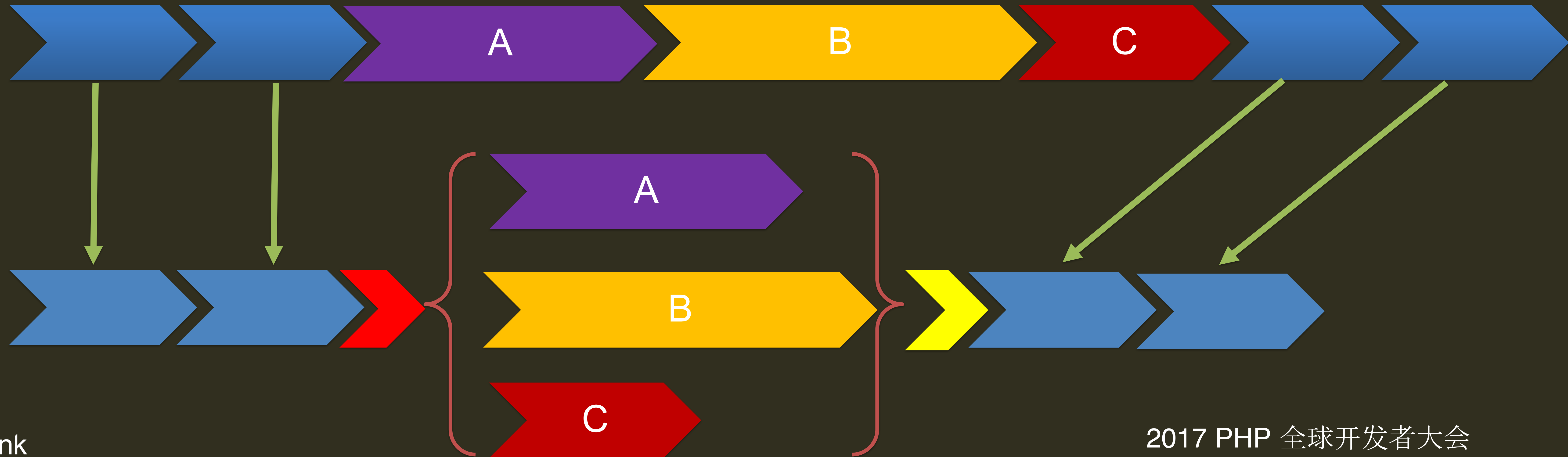
## 短路策略

在有多判断的情况下，按照发生的可能性进行从大到小排序。

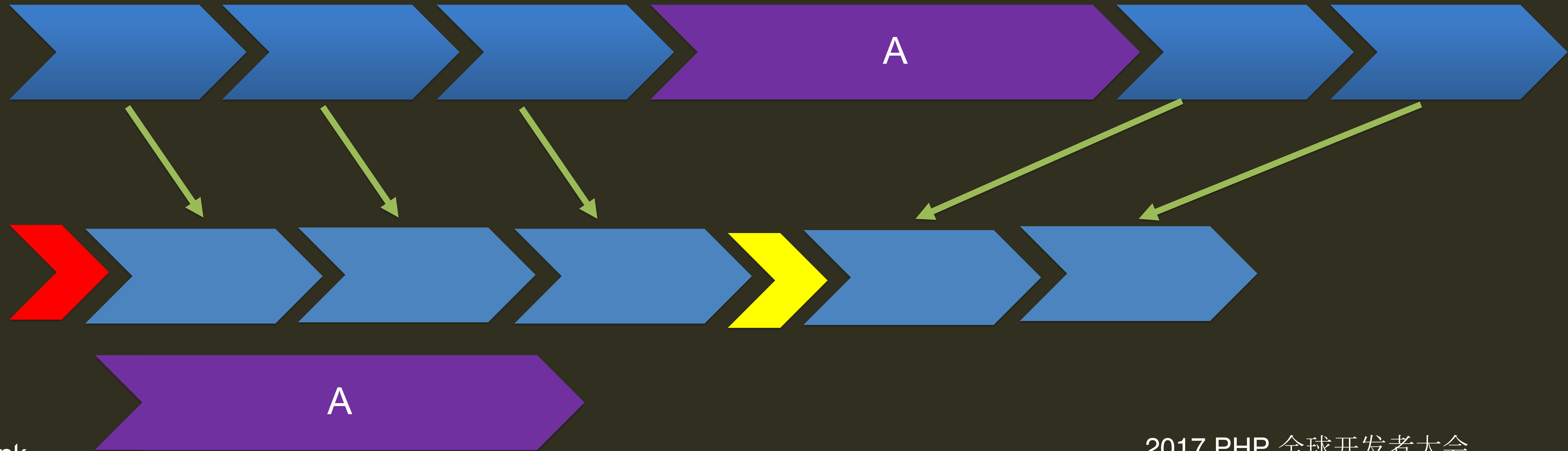


## 并行化及旁路策略

# 并行策略



# 旁路策略



# 性能专项 Tips



每日报表直达机制



使用熟悉的技术进行方案选择



放量策略



勇敢尝试新技术



2/8法则，优化80%问题



一次只改一个地方





# PHP 2017·北京

## 全球开发者大会

