《Pinpoint & Pinpoint C Agent（支持PHP） 简易安装文档》

1. Pinpoint基础概念及技术细节

请参考：<http://www.tangrui.net/2016/pinpoint-plugin-development.html>

1. 环境依赖
2. 安装JDK，OpenJDK 1.6 至 1.9，JAVA SE 6 至 9，都需要安装，6 和 7 直接用yum 或 apt 安装即可，更高版本可以到如下地址下载：

OpenJDK：<http://jdk.java.net/java-se-ri/8>

JAVA SE：<https://www.oracle.com/technetwork/java/javase/downloads/index.html>

1. 设置JAVA\_HOME，例如：

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0

export JAVA\_6\_HOME=/usr/lib/jvm/java-1.6.0

export JAVA\_7\_HOME=/usr/lib/jvm/java-1.7.0

export JAVA\_8\_HOME=/usr/lib/jvm/java-1.8.0

export JAVA\_9\_HOME=/usr/lib/jvm/java-1.9.0

1. 安装Maven：

请参考：<http://maven.apache.org/install.html>

1. 安装Tomcat：

请参考：<http://tomcat.apache.org/tomcat-9.0-doc/setup.html>

1. 安装Pinpoint
2. 克隆代码并编译

git clone <https://github.com/naver/pinpoint.git>

cd pinpoint

./mvnw install -DskipTests=true

1. 安装hbase-1.0.3

下载并解压：<http://archive.apache.org/dist/hbase/hbase-1.0.3/hbase-1.0.3-bin.tar.gz>

初始化并启动：

/path\_to\_hbase/bin/ hbase shell /path\_to\_pinpoint/ hbase/scripts/ hbase-create.hbase

/path\_to\_hbase/bin/start-hbase.sh

配置hbase：

vim /path\_to\_hbase/conf/hbase-site.xml

其实默认配置不需要修改

1. 安装收集器

Pinpoint编译完成后，收集器的war包就放在 /path\_to\_pinpoint/collector/ 中，配置文件放在/path\_to\_pinpoint/collector/src/main/resources中，通过pinpoint-collector.properties配置文件可以修改收集器相关配置，比如端口号：

profiler.collector.tcp.port=9994

profiler.collector.stat.port=9995

profiler.collector.span.port=9996

通过 hbase.properties 配置文件可以修改与hbase通信相关的配置，默认配置即可。

把收集器的war 包放入java容器即可启动收集器。

1. 安装Pinpoint的Web UI界面

Pinpoint编译完成还有，WebUI的war包就放在/path\_to\_pinpoint/web/中，配置文件放在/home/manhong/pinpoint/web/src/main/resources中，通过pinpoint-web.properties配置文件可以修改WebUI相关配置，比如hbase端口号：

hbase.client.host=localhost

hbase.client.port=2181

把WebUI的war包放入java容器即可启动。

1. 快速开始

如果是PHPer嫌配置Tomcat太麻烦，或者只是想运行一个可以演示的DEMO。可以使用Pinpont自带quickstart包。pinpoint编译完成后，quickstart包放在 /path\_to\_pinpoint/quickstart/ 中。

1. 初始化Hbase：

下载HBase-1.0.3-bin.tar.gz并解压到/path\_to\_pinpoint/quickstart/hbase/hbase

/path\_to\_pinpoint/quickstart/bin/start-hbase.sh

/path\_to\_pinpoint/quickstart/bin/init-hbase.sh

1. 启动&关闭Hbase：

启动：/path\_to\_pinpoint/quickstart/bin/start-hbase.sh

关闭：/path\_to\_pinpoint/quickstart/bin/stop-hbase.sh

1. 启动&关闭收集器：

启动：/path\_to\_pinpoint/quickstart/bin/start-collector.sh

关闭：/path\_to\_pinpoint/quickstart/bin/stop-collector.sh

1. 启动&关闭Web界面：

启动：/path\_to\_pinpoint/quickstart/bin/start-web.sh

关闭：/path\_to\_pinpoint/quickstart/bin/stop-web.sh

注意：默认Web界面的端口号是28080，访问：http://localhost:28080

1. 启动&关闭Java测试项目：

启动：/path\_to\_pinpoint/quickstart/bin/start-testapp.sh

关闭：/path\_to\_pinpoint/quickstart/bin/stop-testapp.sh

注意：JAVA测试项目的默认端口号是28081，访问：http://localhost:28081；测试项目启动时，通过字节码的方式挂载了Pinpoint-JAVA-Agent。

1. 安装Pinpoint-C-Agent

1，安装先决条件

Centos：sudo yum install automake libtool flex bison pkgconfig gcc-c++

Ubuntu：sudo apt-get install automake bison flex g++ git libtool make pkg-config

安装PHP，并确保phpize在搜索路径当中

export PATH=/path\_to\_phpize/:$PATH

1. 快速编译

git clone <https://github.com/naver/pinpoint-c-agent>

cd pinpoint-c-agent/pinpoint\_php

./Build.sh

export LD\_LIBRARY\_PATH=$PWD/../thirdlibray/var/:$LD\_LIBRARY\_PATH

sudo make install

1. 源码编译

从快速编译后 pinpoint-c-agent/thirdlibray 目录中的内容来看，pinpoint-c-agent需要一些第三方库，比如：boost和thrift；快速编译时，脚本将boost和thrift编译后的库文件放在了pinpoint-c-agent/thirdlibray中，并修改了环境变量：LD\_LIBRARY\_PATH；但这样并不利于第三方库的管理，所以我们可能需要单独安装第三方库。

1. 安装Boost

wget <https://jaist.dl.sourceforge.net/project/boost/boost/1.63.0/boost_1_63_0.tar.gz>

tar -zxvf boost\_1\_63\_0.tar.gz && cd boost\_1\_63\_0

./bootstrap.sh

sudo ./b2 install --prefix=/usr/local/

1. 安装Thrift

wget <http://apache.fayea.com/thrift/0.11.0/thrift-0.11.0.tar.gz>

tar zxvf thrift-0.11.0.tar.gz

cd thrift-0.11.0

./configure CXXFLAGS="-DFORCE\_BOOST\_SMART\_PTR" --with-cpp --with-php=no --with-python=no --with-ruby=no --with-nodejs=no --with-qt4=no --with-java=no -with-boost=/usr/local/

sudo make && make install

export WITH\_BOOST\_PATH=/usr/local/

export WITH\_THRIFT\_PATH=/usr/local/

#如果/usr/local不在LD\_LIBRARY\_PATH中的话

export LD\_LIBRARY\_PATH=/usr/local/:$LD\_LIBRARY\_PATH

1. 编译PHP扩展

git clone <https://github.com/naver/pinpoint-c-agent>

cd pinpoint-c-agent/pinpoint\_php

phpize

sudo make && make install

1. 配置PHP Agent

sudo mkdir -p /var/pinpoint/php/config

sudo mkdir -p /var/pinpoint/php/log

sudo vim /var/pinpoint/php/config/pinpoint\_agent.conf

##在配置文件pinpoint\_agent.conf中插入如下行

[common]

AgentID=PHP\_Agent

ApplicationName=PHP\_App

CollectorSpanIp=127.0.0.1

CollectorSpanPort=9996

CollectorStatIp=127.0.0.1

CollectorStatPort=9995

CollectorTcpIp=127.0.0.1

CollectorTcpPort=9994

TraceLimit=-1

SkipTraceTime=-1

PPLogLevel=DEBUG

LogFileRootPath=/var/pinpoint/php/log

AgentType=NGINX

##字段含义

AgentID：确保每个agent 的ID是唯一的。

ApplicationName：项目名称，用项目域名就好了

CollectorXXXIp，CollectorXXXPort：收集器相关IP和端口号，请参考之前收集器的配置。

TraceLimit：单个Trace中Span的最大个数，如果不限制则传-1

SkipTraceTime：不发送时间过短的Trace到收集器，如果不限制则传-1

PPLogLevel：日志级别DEBUG | INFO | WARNING | ERROR | STDERR

LogFileRootPath：日志文件路径

AgentType：Agent类型NGINX | APACHE | PHP | TEST 但是感觉用处不大

1. 配置php.ini

vim /usr/local/php/php.ini

#在php.ini文件最后插入如下行

[pinpoint]

extension=pinpoint.so

pinpoint\_agent.config\_full\_name=/var/pinpoint/php/config /pinpoint\_agent.conf

1. 编写拦截器插件

sudo mkdir -p /var/pinpoint/php/plugins

vim test\_plugin.php

//插件文件test\_plugin.php中写入如下内容

//定义插件类

class TestPlugin extends \Pinpoint\Plugin

{

public function \_\_construct()

{

parent::\_\_construct();

//定义简单的函数拦截器，监控代码中的sleep，getenv函数的调用

$this->addSimpleInterceptor("sleep", -1);

$this->addSimpleInterceptor("getenv", -1);

//注册自定义的拦截器

$i = new CustomInterceptor();

$this->addInterceptor($i, "testNameSpace\\TestClass::getVarAdd", " test\_plugin.php");

}

}

//自定义拦截器

class CustomInterceptor extends \Pinpoint\Interceptor

{

public $apiId = -1;

public function \_\_construct()

{

$this->apiId = pinpoint\_add\_api("testNameSpace\\TestClass::getVarAdd ", -1);

}

//函数调用开始前调用

public function onBefore($callId, $args)

{

$trace = pinpoint\_get\_current\_trace(); //取得trace

if ($trace)

{

$event = $trace->traceBlockBegin($callId);

$event->markBeforeTime(); //标记调用开始时间

$event->setApiId($this->apiId);

$event->setServiceType(PINPOINT\_PHP\_RPC\_TYPE);

$self = $this->getSelf();

//输出调用函数时使用的参数等信息

if ($self) {

$event->addAnnotation(PINPOINT\_ANNOTATION\_ARGS,

sprintf("[ %s ] \n this.num=%d ", htmlspecialchars(print\_r($args, true) ,ENT\_QUOTES), $self->num));

} else {

$event->addAnnotation(PINPOINT\_ANNOTATION\_ARGS, htmlspecialchars(print\_r($args, true), ENT\_QUOTES));

}

}

}

//函数调用结束后调用

public function onEnd($callId, $data)

{

$trace = pinpoint\_get\_current\_trace(); //取得trace

if ($trace) {

$args = $data["args"];

$retArgs = $data["result"]; //取得函数返回的结果

$event = $trace->getEvent($callId);

//输出函数调用结果等信息

if ($event) {

if ($retArgs) {

$event->addAnnotation(PINPOINT\_ANNOTATION\_RETURN, htmlspecialchars(print\_r($retArgs, true), ENT\_QUOTES));

}

$event->markAfterTime();

$trace->traceBlockEnd($event);

}

}

}

//函数调用出错时调用

public function onException($callId, $exceptionStr)

{

$trace = pinpoint\_get\_current\_trace();

if ($trace) {

$event = $trace->getEvent($callId);

if ($event) {

$event->markAfterTime();

$event->setExceptionInfo($exceptionStr);

}

}

}

}

//注册插件

$p = new TestPlugin();

pinpoint\_add\_plugin($p, "test\_plugin.php");

#修改pinpoint\_agent.conf配置文件

sudo vim /var/pinpoint/php/config /pinpoint\_agent.conf

#排除的插件类，多个用分号分隔

#PluginExclude=TestPlugin

#引用的插件，多个用分号分隔

PluginInclude= TestPlugin

[PHP]

#插件类所在文件

PluginEntryFile=test\_plugin.php

#插件文件所在的目录

PluginRootDir=/var/pinpoint/php/plugins/

1. 配置测试PHP项目

#创建测试项目web根，和index.php文件

sudo mkdir -p /var/web/pinpoint\_test/

vim /var/web/pinpoint\_test/index.php

<?php

var\_dump(getenv('HTTP\_HOST'));

sleep(2);

$t = new testNameSpace\TestClass();

$t->num = 80;

echo $t->getVarAdd(8);

namespace testNameSpace;

class TestClass

{

public function getVarAdd($arg1)

{

return $this->num + $arg1;

}

var $num;

}

#配置Nginx

server {

listen 2828;

server\_name 127.0.0.1;

root /var/web/pinpoint\_test/;

location / {

index index.php index.html index.htm;

}

location ~ \.php($|/) {

fastcgi\_pass unix:/usr/local/php/var/run/www\_fpm\_stock;

fastcgi\_index index.php;

fastcgi\_split\_path\_info ^(.+\.php)(.\*)$;

fastcgi\_param PATH\_INFO $fastcgi\_path\_info;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

}

if (!-e $request\_filename) {

rewrite ^/(.\*)$ /index.php/$1 last;

break;

}

location ~ /\.ht {

deny all;

}

}

#重启php-fpm和Nginx就好了，访问<http://127.0.0.1:28080>访问pinpoint的Web界面，并访问<http://127.0.0.1:2828>测试效果。

1. 修改Pinpoint-C-Agent源码

(1),单个php-fpm实例支持多个Agent

vim /path\_to\_pinpoint/pinpoint\_php/php\_common.cpp

#增加如下行

string get\_host\_name\_flag()

{

TSRMLS\_FETCH();

char\* pName = "SERVER\_NAME";

char \*pTempValueStr = sapi\_getenv(pName, strlen(pName) TSRMLS\_CC);

string::size\_type nPos = 0;

if (pTempValueStr)

{

std::string value(pTempValueStr);

efree(pTempValueStr);

while(nPos = value.find(".", nPos))

{

if (nPos = string::npos) break;

value.replace(nPos, 1, "\_");

nPos ++;

}

return value;

}

return "none";

}

vim /path\_to\_pinpoint/pinpoint\_php/php\_common.h

#增加如下行

extern std::string get\_host\_name\_flag();

vim /path\_to\_pinpoint/pinpoint\_php/pinpoint\_ext.cpp

#增加如下行

Pinpoint::Agent::PinpointAgentContextPtr& contextPtr = Pinpoint::Agent::PinpointAgentContext::getContextPtr();

contextPtr->agentId = contextPtr->applicationName + ":" + get\_host\_name\_flag();

contextPtr->applicationName = contextPtr->applicationName + ":" + get\_host\_name\_flag();

#按照之前的方法重新编译PHP扩展就可以了。

(2),拦截器通配符

//todo