《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，6,7,8 直接用yum 或 apt 安装即可，例如：

sudo yum install java-1.6.0-openjdk java-1.6.0-openjdk-devel

sudo yum install java-1.7.0-openjdk java-1.7.0-openjdk-devel

sudo yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel

更高版本请参考如下URL：

OpenJDK：http://openjdk.java.net/install/

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

export JRE\_HOME=$JAVA\_HOME/jre

export PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin

CLASSPATH=:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

#最好把上述命令写入 /etc/profile

1. 安装Maven：

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

1. 下载Tomcat容器：

请参考：<https://tomcat.apache.org/download-90.cgi>

wget <http://mirrors.shu.edu.cn/apache/tomcat/tomcat-9/v9.0.12/bin/apache-tomcat-9.0.12.tar.gz>

tar -xvf apache-tomcat-9.0.12.tar.gz

1. 安装Pinpoint
2. 克隆代码并编译

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

tar -xvf pinpoint

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>

tar -xvf hbase-1.0.3-bin.tar.gz

初始化并启动：

sudo cp -r hbase-1.0.3 /usr/local/hbase

sudo vim /usr/local/hbase/conf/ hbase-env.sh

#写入内容

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

export HBASE\_CLASSPATH=/usr/local/hbase/conf

export HBASE\_OPTS="-XX:+UseConcMarkSweepGC"

export HBASE\_MANAGES\_ZK=true

sudo mkdir -p /usr/local/hbase/data/root

sudo mkdir -p /usr/local/hbase/data/zookeeper

sudo vim /usr/local/hbase/conf/hbase-site.xml

#写入如下内容

<configuration>

<property>

<name>hbase.rootdir</name>

<value>/usr/local/hbase/data/root</value>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/usr/local/hbase/data/zookeeper</value>

</property>

</configuration>

sudo vim /etc/profile

#写入如下内容

export HBASE\_HOME=/usr/local/hbase

export PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin:$HBASE\_HOME/bin

source /etc/profile

iptables -A INPUT -p tcp --dport 2181 -j ACCEPT

iptables -A OUTPUT -p tcp --sport 2181 -j ACCEPT

sudo /usr/local/hbase/bin/start-hbase.sh

sudo /usr/local/hbase/bin/hbase shell ./pinpoint/hbase/scripts/hbase-create.hbase

#注意：这样安装hbase是单机模式，性能一般，也不方便扩展，如果想分布式安装请参考：http://abloz.com/hbase/book.html

1. 安装收集器

sudo mkdir -p /usr/local/tomcat/pinpoint

sudo cp -r ~/apache-tomcat-9.0.12 /usr/local/tomcat/pinpoint/collector

sudo rm -rf /usr/local/tomcat/pinpoint/collector/webapps/ROOT

sudo unzip pinpoint-collector-1.8.1-SNAPSHOT.war -d /usr/local/tomcat/pinpoint/collector/webapps/ROOT

export CATALINA\_HOME2=/usr/local/tomcat/pinpoint/collector

export CATALINA\_BASE2=/usr/local/tomcat/pinpoint/collector

export TOMCAT\_HOME2=/usr/local/tomcat/pinpoint/collector

sudo vim /usr/local/tomcat/pinpoint/collector/conf/server.xml

#修改如下行

<Server port="28006" shutdown="SHUTDOWN">

<Connector port="28081" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="28443" />

<Connector port="28010" protocol="AJP/1.3" redirectPort="28443" />

sudo vim /usr/local/tomcat/pinpoint/collector/webapps/ROOT/WEB-INF/classes/hbase.properties

#修改如下行

hbase.client.host=localhost

hbase.client.port=2181

sudo vim /usr/local/tomcat/pinpoint/collector/webapps/ROOT/WEB-INF/classes/pinpoint-collector.properties

#修改如下行

collector.receiver.base.ip=0.0.0.0

collector.receiver.base.port=9994

collector.receiver.stat.udp.ip=0.0.0.0

collector.receiver.stat.udp.port=9995

collector.receiver.span.udp.ip=0.0.0.0

collector.receiver.span.udp.port=9996

sudo vim /usr/local/tomcat/pinpoint/collector/bin/catalina.sh

#在开头加入两行

export CATALINA\_BASE=$CATALINA\_BASE2

export CATALINA\_HOME=$CATALINA\_HOME2

#启动tomcat

sudo /usr/local/tomcat/pinpoint/collector/bin/startup.sh

1. 安装Pinpoint的Web UI界面

sudo cp -r ~/apache-tomcat-9.0.12 /usr/local/tomcat/pinpoint/web

sudo rm -rf /usr/local/tomcat/pinpoint/web/webapps/ROOT

sudo unzip ~/pinpoint/web/target/pinpoint-web-1.8.1-SNAPSHOT.war -d /usr/local/tomcat/pinpoint/web/webapps/ROOT

export CATALINA\_HOME=/usr/local/tomcat/pinpoint/web

export CATALINA\_BASE=/usr/local/tomcat/pinpoint/web

export TOMCAT\_HOME=/usr/local/tomcat/pinpoint/web

sudo vim /usr/local/tomcat/pinpoint/web/conf/server.xml

#修改如下行

<Server port="28005" shutdown="SHUTDOWN">

<Connector port="28080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="28443" />

<Connector port="28009" protocol="AJP/1.3" redirectPort="28443" />

sudo vim /usr/local/tomcat/pinpoint/web/webapps/ROOT/WEB-INF/classes/hbase.properties

#修改如下行

hbase.client.host=localhost

hbase.client.port=2181

#启动tomcat

sudo /usr/local/tomcat/pinpoint/web/bin/startup.sh

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

sudo yum install python-devel

wget ftp://xmlsoft.org/libxml2/libxml2-git-snapshot.tar.gz

tar -xvf libxml2-git-snapshot.tar.gz

cd libxml2-2.9.7/

./configure

make

sudo make install

wget https://www.openssl.org/source/openssl-1.0.2p.tar.gz

tar -xvf openssl-1.0.2p.tar.gz

cd openssl-1.0.2p

./config -fPIC enable-shared

make

sudo make install

sudo ln -s /usr/local/ssl/lib/libssl.so.1.0.0 /usr/local/lib64/libssl.so.1.0.0

sudo ln -s /usr/local/ssl/lib/libssl.so.1.0.0 /usr/local/lib/libssl.so.1.0.0

sudo ln -s /usr/local/ssl/lib/libcrypto.so.1.0.0 /usr/local/lib64/libcrypto.so.1.0.0

sudo ln -s /usr/local/ssl/lib/libcrypto.so.1.0.0 /usr/local/lib/libcrypto.so.1.0.0

wget https://nih.at/libzip/libzip-1.2.0.tar.gz

tar -zxvf libzip-1.2.0.tar.gz

cd libzip-1.2.0

./configure

make

sudo make install

wget https://curl.haxx.se/download/curl-7.61.1.tar.gz

tar -xvf curl-7.61.1.tar.gz

cd curl-7.61.1

./configure --with-ssl=/usr/local/ssl

make & make install

adduser --disabled-login --gecos php-fpm

sudo ./configure \

--prefix=/usr/local/php \

--with-config-file-path=/usr/local/php \

--enable-fpm \

--enable-shmop \

--enable-bcmath \

--enable-soap \

--enable-sockets \

--enable-wddx \

--enable-zip \

--enable-calendar \

--enable-mbstring \

--enable-exif \

--enable-ftp \

--with-fpm-user=php-fpm \

--with-fpm-group=php-fpm \

--with-mysqli=mysqlnd \

--with-pdo-mysql=mysqlnd \

--with-libxml-dir=/usr/local/include/libxml2 \

--with-openssl=/usr/local/ssl \

--with-libzip=/home/centos/libzip-1.2.0 \

--with-curl=/usr/local/include/curl

sudo cp ~/php-7.2.11/php.ini-development ./php.ini

export PHP\_HOME=/usr/local/php

export PATH=$PATH:$HBASE\_HOME/bin:$JAVA\_HOME/bin:$JRE\_HOME/bin:$PHP\_HOME/bin

#最好放入/etc/profile

#启动PHP-FPM

sudo mv /usr/local/php/etc/php-fpm.conf.default /usr/local/php/etc/php-fpm.conf

sudo mv /usr/local/php/etc/php-fpm.d/www.conf.default /usr/local/php/etc/php-fpm.d/www.conf

sudo /usr/local/php/sbin/php-fpm

1. 快速编译

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

或者 git clone https://github.com/yuslf/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 -xvf 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/

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

sudo make && make install

#从GIT安装thrift

git clone <https://github.com/apache/thrift.git>

cd thrift

./bootstrap.sh

sudo ./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=/data0/local/ --with-boost-libdir=/data0/local/lib/ --with-c\_glib=/data0/local/lib/ --prefix=/data0/local

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>

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

cd ~/pinpoint-c-agent/pinpoint\_php

./Build.sh --with-boost=/data0/local --with-thrift=/data0/local

sudo make install

1. 配置PHP Agent

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

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

sudo chmod 777 /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

sudo vim /var/pinpoint/php/plugins/quickstart\_plugin.php

//插件文件quickstart\_plugin.php中写入如下内容

<?php

//定义插件类

class QuickStartPlugin 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", "quickstart\_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);

}

}

}

}

#增加插件入库文件plugins\_create.php

sudo vim /var/pinpoint/php/plugins/plugins\_create.php

//并在plugins\_create.php中写入如下内容

<?php

$path=dirname(\_\_FILE\_\_);

foreach (glob($path ."/\*plugin.php") as $value) {

include\_once($value);

}

//include\_once($path."/quickstart\_plugin.php");

//注册插件

$p = new QuickStartPlugin();

pinpoint\_add\_plugin($p, "quickstart\_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=ExcludePlugin;

#引用的插件，多个用分号分隔

PluginInclude= PhpRequestPlugin;QuickStartPlugin;

[PHP]

#插件类所在文件

PluginEntryFile=plugins\_create.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

include(’class.php’);

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

sleep(2);

$t = new testNameSpace\TestClass();

$t->num = 80;

echo $t->getVarAdd(8);

?>

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

<?php

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

参考<https://github.com/yuslf/pinpoint-c-agent>

(2),拦截器通配符

//todo