Интеграция WCS в Zabbix

Zabbix - система мониторинга с открытым исходным кодом под лицензией GPL v2.Для передачи параметров работы WCS в Zabbix используется набор скриптов, запускаемых Zabbix агентом на наблюдаемом сервере

В данном разделе мы не будем останавливаться на получении информации о системе, ограничиваясь только метриками WCS.

Общие сведения о настройке Zabbix агента

Файлы конфигурации параметров для передачи на Zabbix сервер мониторинга располагаются в каталоге /etc/zabbix/zabbix_agent.d

Файлы скриптов для сбора метрик располагаются в каталоге/etc/zabbix/scripts.d

Основные параметры работы WCS

Конфигурация

Основные параметры работы WCS перечисляются в файле webcallserver.conf

```
UserParameter=wcs.all,/etc/zabbix/scripts.d/get_stat.py
UserParameter=tcp.all,/etc/zabbix/scripts.d/tcp_status.sh
UserParameter=gc.type_gc,/etc/zabbix/scripts.d/gc.sh type_gc
UserParameter=gc.heap_size,/etc/zabbix/scripts.d/gc.sh heap_size
UserParameter=gc.used_mark_start,/etc/zabbix/scripts.d/gc.sh used_mark_start
UserParameter=gc.used_relocate_end,/etc/zabbix/scripts.d/gc.sh used_relocate_end
UserParameter=gc.pause_sum,/etc/zabbix/scripts.d/gc.sh pause_sum
UserParameter=websocket.check,/etc/zabbix/scripts.d/websocket_check.sh
UserParameter=wcs.check,/etc/zabbix/scripts.d/wcs_check.sh
```

Здесь

- wcs.all - параметры статистики WCS
- tcp.all - статистика TCP соединений
- gc.type_gc - тип сборщика мусора (GC)
- gc.heap_size - текущий размер Java heap
- gc.used_mark_start - используемый объем Java heapна момент начала работы GC
- gc.used_relocate_end -используемый объем Java heapна момент окончания работы GC
- gc.pause_sum - длительность работы GC (в это время приостанавливается работа JVM)
- websocket.check - проверка доступности WCS по Websocket
- wcs.check - проверка наличия процесса WCS в списке процессов узла

Скрипты

Получение параметров статистики

Параметры статистики собирает скриптget_stat.py

get_stat.py

```
#!/usr/bin/env python

import json,requests,socket
from pyzabbix import ZabbixSender, ZabbixMetric

sender_host = socket.gethostname()

r = requests.get("http://localhost:8081/?action=stat")
res = r.text.split("\n")

sender = ZabbixSender('192.168.1.179',use_config=True,chunk_size=250)

metrics = []

for item in res:
    if not item.startswith("---") and item != "":
        key = item.split("=")[0]
        value = item.split("=")[1]
        if key.startswith("native_resources"):
                pass
        else:
                m = ZabbixMetric(sender_host, 'wcs[' + key + ']', value)

            metrics.append(m)

sender.send(metrics)

print(1)
```

Скрипт разбирает страницу статистикиhttp://localhost:8081/?action=stat,и отправляет параметры на Zabbix сервер по указанному в скрипте IP адресу, кроме раздела native_resources

На стороне Zabbix к параметрам необходимо обращаться следующим образом:

```
wcs[connections]
wcs[wcs_version]
```

Получение статистики TCP соединений

Статистику соединений собирает скриптtcp_status.sh

tcp_status.sh

```
#!/bin/bash

HOST=`/bin/hostname`

/usr/sbin/ss -ant | awk "{if (NR>1) {state[\$1]++}} END {host = \"${HOST}\"; \
   for (i in state) {s=i; \
   sub (/ESTAB/, \"establ\", s); sub (/LISTEN/, \"listen\", s); sub (/SYN-SENT/, \"synsent\", s);  \
   sub (/SYN-RECV/, \"synrecv\", s); sub (/FIN-WAIT-1/, \"finw1\", s); sub (/FIN-WAIT-2/, \"finw2\", s); \
   sub (/CLOSE-WAIT/, \"closew\", s); sub (/TIME-WAIT/, \"timew\", s); print host, \"tcp.\"s, state[i]}}" \
   | /usr/bin/zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -s ${HOST} --port '10051' -i - >/dev/null 2>&1
echo "1"
exit 0
```

Скрипт собирает текущее состояние TCP соединений и отправляет на Zabbix сервер

Получение статистики сборки мусора (GC)

Статистику работы сборщика мусора (GC) JVM собирает скриптgc.sh

gc.sh

```
#!/bin/bash

TYPE=$1

WCS_HOME="/usr/local/FlashphonerWebCallServer"
LAST_LOG=$(ls -t ${WCS_HOME}/logs/ | grep gc-core | head -1)
LOG="${WCS_HOME}/logs/${LAST_LOG}"
JAVA_VER=$(java -version 2>&1 | head -n 1 | awk -F '"' '{print $2}')

TYPE_GC="$(grep -Pv '^(#|$)' ${WCS_HOME}/conf/wcs-core.properties | grep -oE 'ConcMarkSweepGC|ZGC')"

if [[ -n "TYPE_GC" ]]; then
    if [[ $TYPE == "type_gc" ]]; then
        echo $TYPE_GC
    fi
else
    echo "Garbage ollector configuration not found in WCS core.properties"
    exit 1
fi

# Used OpenJDK 1.x or Java x; GC - ConcMarkSweepGC (only)
if [[ $JAVA_VER != "1"[0-9]* ]]; then

# 2019-08-29T03:48:51.481+0700: 153426.640: [GC (Allocation Failure) 2019-08-29T03:48:51.481+0700: 153426.640:
[ParNew: 34153K->384K(38080K), 0.0045083 secs] 52756K->18988K(122752K), 0.0047446 secs] [Times: user=0.01 sys=0.
00, real=0.01 secs]

    if [[ $TYPE == "heap_size" ]]; then
        grep 'Allocation Failure' $LOG | tail -1 | awk -F'->' '{print $3}' | sed -rn 's/([0-9]+)K\(([0-9]+)K\),
([0-9]+.[0-9]+).*/\2/p' | awk '{printf "%i\n", $1 * 1024}'

    elif [[ $TYPE == "used_mark_start" ]]; then
        grep 'Allocation Failure' $LOG | tail -1 | awk -F'->' '{print $2}' | sed -rn 's/.* ([0-9]+)K$/\1/p' |
awk '{printf "%i\n", $1 * 1024}'

    elif [[ $TYPE == "used_relocate_end" ]]; then
        grep 'Allocation Failure' $LOG | tail -1 | awk -F'->' '{print $3}' | sed -rn 's/([0-9]+)K\(([0-9]+)K\),
([0-9]+.[0-9]+).*/\1/p' | awk '{printf "%i\n", $1 * 1024}'

    elif [[ $TYPE == "pause_sum" ]]; then
        grep 'Allocation Failure' $LOG | tail -1 | awk -F'->' '{print $3}' | sed -rn 's/([0-9]+)K\(([0-9]+)K\),
([0-9]+.[0-9]+).*/\3/p' | tr , . | awk '{printf "%f\n", $1 * 1000 }'
    fi

# Used OpenJDK 1x or Java 1x; GC - ConcMarkSweepGC, ZGC
elif [[ $JAVA_VER == "1"[0-9]* ]]; then

    if [[ $TYPE_GC == "ConcMarkSweepGC" ]]; then

# [26002.922s][info][gc] GC(133) Pause Young (Allocation Failure) 101M->21M(1014M) 4.239ms
        if [[ $TYPE == "heap_size" ]]; then
            grep 'Allocation Failure' $LOG | tail -1 | awk '{print $7}' | awk -F'->' '{print $2}' | sed -rn 's/
[0-9]+M\(([0-9]+)M\).*/\1/p' | awk '{printf "%i\n", $1 * 1024 * 1024}'

        elif [[ $TYPE == "used_mark_start" ]]; then
            grep 'Allocation Failure' $LOG | tail -1 | awk '{print $7}' | sed -rn 's/([0-9]+).*/\1/p' | awk
'{printf "%i\n", $1 * 1024 * 1024}'

        elif [[ $TYPE == "used_relocate_end" ]]; then
            grep 'Allocation Failure' $LOG | tail -1 | awk '{print $7}' | awk -F'->' '{print $2}' | sed -rn 's/
([0-9]+).*/\1/p' | awk '{printf "%i\n", $1 * 1024 * 1024}'

        elif [[ $TYPE == "pause_sum" ]]; then
            grep 'Allocation Failure' $LOG | tail -1 | awk '{print $8}' | sed 's/ms$//'
        fi
    fi

    if [[ $TYPE_GC == "ZGC" ]]; then
```

```
# [6.960s][info][gc,heap     ] GC(1)  Capacity:     4096M (100%)       4096M (100%)       4096M (100%)
4096M (100%)       4096M (100%)       4096M (100%)
        if [[ $TYPE == "heap_size" ]]; then
            grep 'GC(.*)*Capacity' $LOG | tail -1 | awk -F 'Capacity:' '{print $2}' | sed -rn 's/^[ ]*([0-9]+).*
/\1/p' | awk '{printf "%i\n", $1 * 1024 * 1024}'

# 6.960s][info][gc          ] GC(1) Garbage Collection (Metadata GC Threshold) 186M(5%)->70M(2%)
        elif [[ $TYPE == "used_mark_start" ]]; then
            grep 'Garbage Collection (.*) ' $LOG | tail -1 | awk -F 'Allocation Rate)|Proactive)|Warmup)
|Threshold)' '{print $2}' | sed 's/^ *\([0-9]*\).*/\1/' | awk '{printf "%i\n", $1 * 1024 * 1024}'

        elif [[ $TYPE == "used_relocate_end" ]]; then
            grep 'Garbage Collection (.*) ' $LOG | tail -1 | awk -F 'Allocation Rate)|Proactive)|Warmup)
|Threshold)' '{print $2}' | awk -F'->' '{print $2}' | sed -rn 's/([0-9]+).*/\1/p' | awk '{printf "%i\n", $1 *
1024 * 1024}'

        elif [[ $TYPE == "pause_mark_start" ]]; then
            grep '.*GC.*Pause Mark Start' $LOG | tail -1 | awk -F 'Pause Mark Start ' '{print $2}' | sed 's/ms$
//'

        elif [[ $TYPE == "pause_mark_end" ]]; then
            grep '.*GC.*Pause Mark End' $LOG | tail -1 | awk -F 'Pause Mark End ' '{print $2}' | sed 's/ms$//'

        elif [[ $TYPE == "pause_relocate_start" ]]; then
            grep '.*GC.*Pause Relocate Start' $LOG | tail -1 | awk -F 'Pause Relocate Start ' '{print $2}' |
sed 's/ms$//'

        elif [[ $TYPE == "pause_sum" ]]; then
            grep '.*GC.*Pause' $LOG | awk -F 'Pause Mark Start|End|Relocate Start' '{print $2}' | tail -3 | sed
's/ms$//' | awk '{a=$1; getline;b=$1;getline;c=$1;getline;t=a+b+c;print t}'

        fi
    fi
fi
```

Скрипт возвращает один из следующих параметров:

- gc.type_gc - тип сборщика мусора (GC)
- gc.heap_size - текущий размер Java heap
- gc.used_mark_start - используемый объем Java heapна момент начала работы GC
- gc.used_relocate_end -используемый объем Java heapна момент окончания работы GC
- gc.pause_sum - длительность работы GC (в это время приостанавливается работа JVM)

Zabbix agent сам отправляет параметр на Zabbix сервер

Проверка доступности WCS по Websocket

Доступность сервера по Websocket определяет скриптwebsocket_check.sh

websocket_check.sh

```
#!/bin/bash

WCS_HOME="/usr/local/FlashphonerWebCallServer"

WSS_ADDRESS="$(grep -Pv '^(#|$)' ${WCS_HOME}/conf/flashphoner.properties | grep -E 'wss.address' | awk -F '='
'{print $2}' | sed 's/^[ \t]*//;s/[ \t]*$//')"
[ -z $WSS_ADDRESS ] && WSS_ADDRESS='0.0.0.0'

WSS_PORT="$(grep -Pv '^(#|$)' ${WCS_HOME}/conf/flashphoner.properties | grep -E 'wss.port' | awk -F '=' '{print
$2}' | sed 's/^[ \t]*//;s/[ \t]*$//')"
[ -z $WSS_PORT ] && WSS_PORT='8443'

curl --connect-timeout 30 --insecure --silent --include \
--no-buffer \
--header "Connection: Upgrade" \
--header "Upgrade: websocket" \
--header "Host: $WSS_ADDRESS:$WSS_PORT" \
--header "Origin: https://$WSS_ADDRESS:$WSS_PORT" \
--header "Sec-WebSocket-Key: SGVsbG8sIHdvcmxkIQ==" \
--header "Sec-WebSocket-Version: 13" \
https://$WSS_ADDRESS:$WSS_PORT/ | grep -q "HTTP/1.1" && [[ $? == 0 ]] && echo '1' || echo '0'
```

Скрипт возвращает одно из следующих значений:

- 1 - WCS отвечает по Websocket
- 0 - WCS не отвечает по Websocket

Zabbix agent сам отправляет параметр на Zabbix сервер

Проверка наличия WCS в списке процессов

Наличие WCS в списке процессов определяет скриптwcs_check.sh

wcs_check.sh

```
#!/bin/bash

PID="$(pgrep -f 'com.flashphoner.server.Server' | grep -v bash)"
[ -n "$PID" ] && echo "1" || echo "0"
```

Скрипт возвращает одно из следующих значений:

- 1 - WCS есть в списке процессов на сервере
- 0 - WCS нет в списке процессов на узле

Zabbix agent сам отправляет параметр на Zabbix сервер

Пример шаблона Zabbix для получения данных от агента

Пример шаблона Zabbix для получения данных от агента на WCS сервере можно скачать здесь

zbx_export_templates.xml