# Mysql block info for zabbix by sultan

**Mysql\_block\_info.py**

#!/usr/bin/python

# encoding:utf-8

# 功能:捕获指定数据库服务器上的Mysql 阻塞信息 并将阻塞信息发送到指定邮箱.如果未捕获到阻塞信息 将不发送邮件

import getopt

import sys

import datetime

import traceback

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

import logging

import mysql.connector

import yaml

def \_\_getlogfilename\_\_():

name = str(sys.argv[0]).split('.')

name.pop()

name.append('log')

return '.'.join(name)

def logInfo(\_\_message, \_\_Level="INFO"):

Level = \_\_Level.upper()

if Level == "INFO":

logging.info(\_\_message)

def logWarn(\_\_message, \_\_Level="WARNING"):

Level = \_\_Level.upper()

if Level == "WARNING":

logging.warning(\_\_message)

def logErr(\_\_message, \_\_Level="ERROR"):

Level = \_\_Level.upper()

if Level == "ERROR":

logging.error(\_\_message)

def \_\_usage\_\_():

print(

"Usage:%s [-H|-h|-u|-p||-P|-c|-t] [--help|--host=[127.0.0.1]|--user=[root]|--password=[]|--db=[information\_schema]|--port=[3306]|--charset=[utf8,gbk]|] <--to=> args...." % (

sys.argv[0]))

sys.exit()

def \_\_getopt\_\_():

try:

options, args = getopt.getopt(sys.argv[1:], "Hh:u:p:P:c:t:",

["help", "host=", "user=", "password=", "db=", "port=", "charset=", "to=",

"itemname="])

opts = {"host": "127.0.0.1", \

"user": "cactiuser", \

"password": "cactiuser", \

"db": "information\_schema", \

"port": "3306", \

"charset": "utf8", \

"to": "dbagroup@yooli.com"}

for name, value in options:

if name in ("-H", "--help"):

\_\_usage\_\_()

elif name in ("-h", "--host"):

opts["host"] = value

logInfo("host Is %s" % (opts["host"]))

elif name in ("-u", "--user"):

opts["user"] = value

logInfo("user Is %s" % (opts["user"]))

elif name in ("-p", "--password"):

opts["password"] = value

logInfo("password Is %s" % (opts["password"]))

elif name in ("--db",):

opts["db"] = value

logInfo("database Is %s" % (opts["db"]))

elif name in ("-P", "--port"):

opts["port"] = value

logInfo("Port Is %s" % (opts["port"]))

elif name in ("-c", "--charset"):

opts["charset"] = value

logInfo("charset Is %s" % (opts["charset"]))

elif name in ("-t", "--to"):

opts["to"] = value

logInfo("mailto Is %s" % (opts["to"]))

elif name in ("--itemname"):

opts["port"] = str(filter(lambda x: x.isdigit(), value)) # 迭代函数 提取value中的数字

logInfo("itemname Is %s,port is %s" % (value, opts["port"]))

if "to" not in opts.keys():

raise KeyError

return opts

except KeyError:

print "必要参数--to未定义。"

sys.exit()

except Exception, ex:

sys.exit()

def \_\_send\_mail\_\_(body, to, subjet='Mysql Block Info'):

# print body

sender = 'dba@yooli.com'

smtp\_server = 'smtp.exmail.qq.com'

user\_name = 'dba@yooli.com'

password = '#EDC5tgb!QAZ'

msg\_root = MIMEMultipart('alternative')

msg\_root['Subject'] = subjet

msg\_root["From"] = sender

msg\_root["to"] = to

msg\_text = MIMEText(body, "plain", 'utf-8')

msg\_root.attach(msg\_text)

smtp = smtplib.SMTP()

smtp.connect(smtp\_server)

smtp.login(user\_name, password)

smtp.sendmail(msg\_root["From"], msg\_root["to"], msg\_root.as\_string())

class mysqlBlockInfo():

INNODB\_LOCKS\_RESULT = []

INNODB\_LOCK\_WAITS\_RESULT = []

INNODB\_TRX\_RESULT = []

PROCESSLIST\_RESULT = []

def \_\_init\_\_(self, user, passwd, host, port=3307, db="information\_schema", charset="utf8"): # 初始化数据

connection = mysql.connector.connect(user=user, password=passwd, host=host, database=db,

port=int(port), charset=charset)

cursor = connection.cursor()

cursor.execute(

'select lock\_id, lock\_trx\_id, lock\_mode, lock\_type, lock\_table, lock\_index, lock\_space, lock\_page, lock\_rec, lock\_data from INNODB\_LOCKS;')

self.INNODB\_LOCKS\_RESULT = self.\_\_excute2object\_\_(cursor.column\_names, cursor.fetchall())

cursor.execute(

'select requesting\_trx\_id, requested\_lock\_id, blocking\_trx\_id, blocking\_lock\_id from INNODB\_LOCK\_WAITS;')

self.INNODB\_LOCK\_WAITS\_RESULT = self.\_\_excute2object\_\_(cursor.column\_names, cursor.fetchall())

cursor.execute(

'select trx\_id, trx\_state, trx\_started, trx\_requested\_lock\_id, trx\_wait\_started, trx\_weight, trx\_mysql\_thread\_id, trx\_query, trx\_operation\_state, trx\_tables\_in\_use, trx\_tables\_locked, trx\_lock\_structs, trx\_lock\_memory\_bytes, trx\_rows\_locked, trx\_rows\_modified, trx\_concurrency\_tickets, trx\_isolation\_level, trx\_unique\_checks, trx\_foreign\_key\_checks, trx\_last\_foreign\_key\_error, trx\_adaptive\_hash\_latched, trx\_adaptive\_hash\_timeout, trx\_is\_read\_only, trx\_autocommit\_non\_locking from INNODB\_TRX;')

self.INNODB\_TRX\_RESULT = self.\_\_excute2object\_\_(cursor.column\_names, cursor.fetchall())

cursor.execute(

'select ID, USER, HOST, DB, COMMAND, TIME, STATE, INFO, TIME\_MS, ROWS\_SENT, ROWS\_EXAMINED from PROCESSLIST;')

self.PROCESSLIST\_RESULT = self.\_\_excute2object\_\_(cursor.column\_names, cursor.fetchall())

cursor.close

connection.close

def \_\_initialize\_\_(self): # 创建测试数据

self.INNODB\_LOCKS\_RESULT = [

{'lock\_id': u'222697825:588:194:10', 'lock\_mode': u'X', 'lock\_type': u'RECORD', 'lock\_rec': 10L,

'lock\_space': 588L,

'lock\_trx\_id': u'222697825', 'lock\_index': u'PRIMARY', 'lock\_page': 194L, 'lock\_data': u'5436',

'lock\_table': u'`loanorg\_management`.`t\_loan`'},

{'lock\_id': u'222697820:588:194:10', 'lock\_mode': u'X', 'lock\_type': u'RECORD', 'lock\_rec': 10L,

'lock\_space': 588L,

'lock\_trx\_id': u'222697820', 'lock\_index': u'PRIMARY', 'lock\_page': 194L, 'lock\_data': u'5436',

'lock\_table': u'`loanorg\_management`.`t\_loan`'},

{'lock\_id': u'222697824:588:194:4', 'lock\_mode': u'X', 'lock\_type': u'RECORD', 'lock\_rec': 4L,

'lock\_space': 588L,

'lock\_trx\_id': u'222697824', 'lock\_index': u'PRIMARY', 'lock\_page': 194L, 'lock\_data': u'5430',

'lock\_table': u'`loanorg\_management`.`t\_loan`'},

{'lock\_id': u'222697820:588:194:4', 'lock\_mode': u'X', 'lock\_type': u'RECORD', 'lock\_rec': 4L,

'lock\_space': 588L,

'lock\_trx\_id': u'222697820', 'lock\_index': u'PRIMARY', 'lock\_page': 194L, 'lock\_data': u'5430',

'lock\_table': u'`loanorg\_management`.`t\_loan`'}]

self.INNODB\_LOCK\_WAITS\_RESULT = [

{'blocking\_lock\_id': u'222697820:588:194:10', 'requesting\_trx\_id': u'222697825',

'blocking\_trx\_id': u'222697820',

'requested\_lock\_id': u'222697825:588:194:10'},

{'blocking\_lock\_id': u'222697820:588:194:4', 'requesting\_trx\_id': u'222697824',

'blocking\_trx\_id': u'222697820',

'requested\_lock\_id': u'222697824:588:194:4'}]

self.INNODB\_TRX\_RESULT = [{'trx\_foreign\_key\_checks': 1L,

'trx\_query': u'SELECT\r\n\t\*\r\nFROM\r\n\tt\_loan\r\nWHERE\r\n\tloanid BETWEEN 5436\r\nAND 5440 FOR UPDATE',

'trx\_isolation\_level': u'REPEATABLE READ', 'trx\_lock\_structs': 2L,

'trx\_lock\_memory\_bytes': 360L,

'trx\_mysql\_thread\_id': 3806484L, 'trx\_last\_foreign\_key\_error': None,

'trx\_tables\_locked': 1L,

'trx\_adaptive\_hash\_timeout': 10000L, 'trx\_rows\_modified': 0L,

'trx\_started': datetime.datetime(2015, 6, 23, 14, 54, 42), 'trx\_is\_read\_only': 0L,

'trx\_concurrency\_tickets': 0L, 'trx\_tables\_in\_use': 1L, 'trx\_rows\_locked': 1L,

'trx\_wait\_started': datetime.datetime(2015, 6, 23, 14, 54, 42),

'trx\_state': u'LOCK WAIT',

'trx\_adaptive\_hash\_latched': 0L, 'trx\_requested\_lock\_id': u'222697825:588:194:10',

'trx\_operation\_state': u'starting index read', 'trx\_autocommit\_non\_locking': 0L,

'trx\_id': u'222697825', 'trx\_unique\_checks': 1L, 'trx\_weight': 2L},

{'trx\_foreign\_key\_checks': 1L,

'trx\_query': u'SELECT\r\n\t\*\r\nFROM\r\n\tt\_loan\r\nWHERE\r\n\tloanid BETWEEN 5420\r\nAND 5435 FOR UPDATE',

'trx\_isolation\_level': u'REPEATABLE READ',

'trx\_lock\_structs': 5L,

'trx\_lock\_memory\_bytes': 1184L,

'trx\_mysql\_thread\_id': 3806489L,

'trx\_last\_foreign\_key\_error': None,

'trx\_tables\_locked': 1L,

'trx\_adaptive\_hash\_timeout': 10000L,

'trx\_rows\_modified': 0L,

'trx\_started': datetime.datetime(

2015, 6, 23, 14, 54, 41),

'trx\_is\_read\_only': 0L,

'trx\_concurrency\_tickets': 0L,

'trx\_tables\_in\_use': 1L,

'trx\_rows\_locked': 12L,

'trx\_wait\_started': datetime.datetime(

2015, 6, 23, 14, 54, 41),

'trx\_state': u'LOCK WAIT',

'trx\_adaptive\_hash\_latched': 0L,

'trx\_requested\_lock\_id': u'222697824:588:194:4',

'trx\_operation\_state': u'fetching rows',

'trx\_autocommit\_non\_locking': 0L,

'trx\_id': u'222697824',

'trx\_unique\_checks': 1L,

'trx\_weight': 5L},

{'trx\_foreign\_key\_checks': 1L, 'trx\_query': None,

'trx\_isolation\_level': u'REPEATABLE READ',

'trx\_lock\_structs': 3L, 'trx\_lock\_memory\_bytes': 360L,

'trx\_mysql\_thread\_id': 3806407L,

'trx\_last\_foreign\_key\_error': None, 'trx\_tables\_locked': 0L,

'trx\_adaptive\_hash\_timeout': 10000L,

'trx\_rows\_modified': 0L, 'trx\_started': datetime.datetime(2015, 6, 23, 14, 51, 56),

'trx\_is\_read\_only': 0L, 'trx\_concurrency\_tickets': 0L, 'trx\_tables\_in\_use': 0L,

'trx\_rows\_locked': 12L, 'trx\_wait\_started': None, 'trx\_state': u'RUNNING',

'trx\_adaptive\_hash\_latched': 0L, 'trx\_requested\_lock\_id': None,

'trx\_operation\_state': None,

'trx\_autocommit\_non\_locking': 0L, 'trx\_id': u'222697820', 'trx\_unique\_checks': 1L,

'trx\_weight': 3L}]

self.PROCESSLIST\_RESULT = [

{'INFO': None, 'ROWS\_EXAMINED': 5L, 'ROWS\_SENT': 5L, 'TIME\_MS': 2818907L, 'DB': None, 'STATE': u'',

'HOST': u'192.168.1.141:51427', 'COMMAND': u'Sleep', 'USER': u'dan.su', 'TIME': 2819L, 'ID': 3803415L},

{'INFO': None, 'ROWS\_EXAMINED': 51L, 'ROWS\_SENT': 51L, 'TIME\_MS': 282763L, 'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'192.168.1.141:51466', 'COMMAND': u'Sleep', 'USER': u'baojin.qi', 'TIME': 283L,

'ID': 3806319L},

{'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 249591L, 'DB': u'test', 'STATE': u'',

'HOST': u'192.168.1.141:51471', 'COMMAND': u'Sleep', 'USER': u'dan.su', 'TIME': 249L, 'ID': 3806355L},

{'INFO': None, 'ROWS\_EXAMINED': 100L, 'ROWS\_SENT': 100L, 'TIME\_MS': 2029973L, 'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'106.39.52.26:33913', 'COMMAND': u'Sleep', 'USER': u'qiang.li', 'TIME': 2030L,

'ID': 3804219L},

{

'INFO': u'select ID, USER, HOST, DB, COMMAND, TIME, STATE, INFO, TIME\_MS, ROWS\_SENT, ROWS\_EXAMINED from PROCESSLIST',

'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 0L, 'DB': u'information\_schema', 'STATE': u'executing',

'HOST': u'106.39.52.26:46233', 'COMMAND': u'Query', 'USER': u'dan.su', 'TIME': 0L, 'ID': 3806632L},

{'INFO': u'SELECT\r\n\t\*\r\nFROM\r\n\tt\_loan\r\nWHERE\r\n\tloanid BETWEEN 5436\r\nAND 5440 FOR UPDATE',

'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 8142L, 'DB': u'loanorg\_management',

'STATE': u'Sending data',

'HOST': u'192.168.1.141:51476', 'COMMAND': u'Query', 'USER': u'dan.su', 'TIME': 8L, 'ID': 3806484L},

{'INFO': None, 'ROWS\_EXAMINED': 400L, 'ROWS\_SENT': 10L, 'TIME\_MS': 83417L, 'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'192.168.1.141:51468', 'COMMAND': u'Sleep', 'USER': u'baojin.qi', 'TIME': 83L,

'ID': 3806326L},

{'INFO': None, 'ROWS\_EXAMINED': 63L, 'ROWS\_SENT': 63L, 'TIME\_MS': 218170L, 'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'192.168.1.141:51470', 'COMMAND': u'Sleep', 'USER': u'dan.su', 'TIME': 218L,

'ID': 3806334L},

{'INFO': None, 'ROWS\_EXAMINED': 5L, 'ROWS\_SENT': 5L, 'TIME\_MS': 785144L, 'DB': None, 'STATE': u'',

'HOST': u'192.168.1.141:51458', 'COMMAND': u'Sleep', 'USER': u'baojin.qi', 'TIME': 785L,

'ID': 3805737L},

{'INFO': None, 'ROWS\_EXAMINED': 70L, 'ROWS\_SENT': 70L, 'TIME\_MS': 290921L, 'DB': u'information\_schema',

'STATE': u'', 'HOST': u'192.168.1.141:51459', 'COMMAND': u'Sleep', 'USER': u'baojin.qi', 'TIME': 291L,

'ID': 3805740L},

{'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 550356L, 'DB': u'information\_schema',

'STATE': u'',

'HOST': u'192.168.1.141:51460', 'COMMAND': u'Sleep', 'USER': u'baojin.qi', 'TIME': 550L, 'ID': 3805744L},

{'INFO': None, 'ROWS\_EXAMINED': 63L, 'ROWS\_SENT': 63L, 'TIME\_MS': 173512L, 'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'192.168.1.141:51472', 'COMMAND': u'Sleep', 'USER': u'dan.su', 'TIME': 173L,

'ID': 3806407L},

{'INFO': None, 'ROWS\_EXAMINED': 1814979L, 'ROWS\_SENT': 52068L, 'TIME\_MS': 401413L,

'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'localhost', 'COMMAND': u'Sleep', 'USER': u'root', 'TIME': 401L, 'ID': 3805343L},

{'INFO': None, 'ROWS\_EXAMINED': 63L, 'ROWS\_SENT': 63L, 'TIME\_MS': 140173L, 'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'192.168.1.141:51475', 'COMMAND': u'Sleep', 'USER': u'dan.su', 'TIME': 140L,

'ID': 3806475L},

{'INFO': None, 'ROWS\_EXAMINED': 378L, 'ROWS\_SENT': 16L, 'TIME\_MS': 830400L, 'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'192.168.1.141:51441', 'COMMAND': u'Sleep', 'USER': u'dan.su', 'TIME': 830L,

'ID': 3804185L},

{'INFO': None, 'ROWS\_EXAMINED': 63L, 'ROWS\_SENT': 63L, 'TIME\_MS': 156158L, 'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'192.168.1.141:51473', 'COMMAND': u'Sleep', 'USER': u'dan.su', 'TIME': 156L,

'ID': 3806451L},

{'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 2101349L, 'DB': u'loanorg\_management',

'STATE': u'',

'HOST': u'192.168.1.141:51429', 'COMMAND': u'Sleep', 'USER': u'dan.su', 'TIME': 2101L, 'ID': 3803424L},

{'INFO': u'SELECT\r\n\t\*\r\nFROM\r\n\tt\_loan\r\nWHERE\r\n\tloanid BETWEEN 5420\r\nAND 5435 FOR UPDATE',

'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 10L, 'TIME\_MS': 9400L, 'DB': u'loanorg\_management',

'STATE': u'Sending data',

'HOST': u'192.168.1.141:51477', 'COMMAND': u'Query', 'USER': u'dan.su', 'TIME': 9L, 'ID': 3806489L},

{'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 53837L, 'DB': u'loanorg\_management',

'STATE': u'',

'HOST': u'192.168.1.141:51478', 'COMMAND': u'Sleep', 'USER': u'baojin.qi', 'TIME': 54L, 'ID': 3806579L},

{'INFO': None, 'ROWS\_EXAMINED': 29L, 'ROWS\_SENT': 29L, 'TIME\_MS': 853261L, 'DB': u'loanorg\_management',

'STATE': u'', 'HOST': u'192.168.1.141:51430', 'COMMAND': u'Sleep', 'USER': u'dan.su', 'TIME': 853L,

'ID': 3803427L}, {'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 3365841980L, 'DB': None,

'STATE': u'Waiting for an event from Coordinator', 'HOST': u'', 'COMMAND': u'Connect',

'USER': u'system user', 'TIME': 3365842L, 'ID': 201L},

{'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 303521L, 'DB': None,

'STATE': u'Waiting for an event from Coordinator', 'HOST': u'', 'COMMAND': u'Connect',

'USER': u'system user',

'TIME': 303L, 'ID': 202L},

{'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 591411585L, 'DB': None,

'STATE': u'Waiting for an event from Coordinator', 'HOST': u'', 'COMMAND': u'Connect',

'USER': u'system user', 'TIME': 591411L, 'ID': 199L},

{'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 3365841981L, 'DB': None,

'STATE': u'Waiting for an event from Coordinator', 'HOST': u'', 'COMMAND': u'Connect',

'USER': u'system user',

'TIME': 3365842L, 'ID': 200L},

{'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 303531L, 'DB': None,

'STATE': u'Slave has read all relay log; waiting for the slave I/O thread t',

'HOST': u'', 'COMMAND': u'Connect', 'USER': u'system user', 'TIME': 303L,

'ID': 198L},

{'INFO': None, 'ROWS\_EXAMINED': 0L, 'ROWS\_SENT': 0L, 'TIME\_MS': 3365841983L, 'DB': None,

'STATE': u'Waiting for master to send event', 'HOST': u'', 'COMMAND': u'Connect', 'USER': u'system user',

'TIME': 3365842L, 'ID': 197L}]

# 数据库结果转对象

def \_\_excute2object\_\_(self, \_\_column\_names, \_\_Indata):

\_\_Outdata = []

for data in \_\_Indata:

\_\_Outdata.append(dict(zip(\_\_column\_names, data)))

return \_\_Outdata

# 递归获取根阻塞事务ID

def \_\_Get\_Root\_Block\_Trxid\_\_(self, \_\_INNODB\_LOCK\_WAITS, \_\_ListNum): # listNum:列表序号

Root\_Info = {"Root\_Block\_Trxid": \_\_INNODB\_LOCK\_WAITS[\_\_ListNum]["blocking\_trx\_id"],

"Root\_lock\_id": \_\_INNODB\_LOCK\_WAITS[\_\_ListNum]["blocking\_lock\_id"]}

for INNODB\_LOCK\_WAITS in \_\_INNODB\_LOCK\_WAITS:

if INNODB\_LOCK\_WAITS['requesting\_trx\_id'] == Root\_Info["Root\_Block\_Trxid"] and INNODB\_LOCK\_WAITS[

'requested\_lock\_id'] == Root\_Info["Root\_lock\_id"] and \_\_INNODB\_LOCK\_WAITS.index(

INNODB\_LOCK\_WAITS) > \_\_ListNum:

Root\_Info = self.\_\_Get\_Root\_Block\_Trxid\_\_(\_\_INNODB\_LOCK\_WAITS,

\_\_INNODB\_LOCK\_WAITS.index(INNODB\_LOCK\_WAITS))

return Root\_Info

# 排重阻塞列表操作

def \_\_Distinct\_INNODB\_LOCK\_WAITS\_\_(self, \_\_INNODB\_LOCK\_WAITS\_RESULT):

Distinct\_INNODB\_LOCK\_WAITS = []

for data in \_\_INNODB\_LOCK\_WAITS\_RESULT:

Distinct\_INNODB\_LOCK\_object = {}

Root\_Info = self.\_\_Get\_Root\_Block\_Trxid\_\_(\_\_INNODB\_LOCK\_WAITS\_RESULT,

\_\_INNODB\_LOCK\_WAITS\_RESULT.index(data))

Distinct\_INNODB\_LOCK\_object["blocking\_trx\_id"] = Root\_Info["Root\_Block\_Trxid"]

Distinct\_INNODB\_LOCK\_object["blocking\_lock\_id"] = Root\_Info["Root\_lock\_id"]

Distinct\_INNODB\_LOCK\_object["requesting\_trx\_id"] = []

if Distinct\_INNODB\_LOCK\_WAITS == []:

Distinct\_INNODB\_LOCK\_WAITS.append(Distinct\_INNODB\_LOCK\_object)

for x in Distinct\_INNODB\_LOCK\_WAITS:

if x["blocking\_trx\_id"] <> Distinct\_INNODB\_LOCK\_object["blocking\_trx\_id"] and x["blocking\_lock\_id"] <> \

Distinct\_INNODB\_LOCK\_object["blocking\_lock\_id"]:

Distinct\_INNODB\_LOCK\_object["requesting\_trx\_id"].append(data["requesting\_trx\_id"])

Distinct\_INNODB\_LOCK\_WAITS.append(Distinct\_INNODB\_LOCK\_object)

elif data["requesting\_trx\_id"] not in \

Distinct\_INNODB\_LOCK\_WAITS[Distinct\_INNODB\_LOCK\_WAITS.index(x)]["requesting\_trx\_id"]:

Distinct\_INNODB\_LOCK\_WAITS[Distinct\_INNODB\_LOCK\_WAITS.index(x)]["requesting\_trx\_id"].append(

data["requesting\_trx\_id"])

return Distinct\_INNODB\_LOCK\_WAITS

# 获取列表内数据

def \_\_Get\_INNODB\_LOCKS\_INFO\_\_(self, \_\_INNODB\_LOCKS\_RESULT, \_\_Trx\_ID):

INNODB\_LOCKS\_INFO = {}

for data in \_\_INNODB\_LOCKS\_RESULT:

if data["lock\_trx\_id"] == \_\_Trx\_ID:

INNODB\_LOCKS\_INFO["lock\_id"] = data["lock\_id"].encode('utf8')

INNODB\_LOCKS\_INFO["lock\_mode"] = data["lock\_mode"].encode('utf8')

INNODB\_LOCKS\_INFO["lock\_type"] = data["lock\_type"].encode('utf8')

INNODB\_LOCKS\_INFO["lock\_index"] = data["lock\_index"].encode('utf8')

INNODB\_LOCKS\_INFO["lock\_table"] = data["lock\_table"].encode('utf8')

break

return INNODB\_LOCKS\_INFO

def \_\_Get\_INNODB\_Trx\_\_(self, \_\_INNODB\_TRX\_RESULT, \_\_Trx\_ID):

INNODB\_TRX\_INFO = {}

for data in \_\_INNODB\_TRX\_RESULT:

if data["trx\_id"] == \_\_Trx\_ID:

INNODB\_TRX\_INFO["trx\_id"] = data["trx\_id"].encode('utf8')

INNODB\_TRX\_INFO["trx\_state"] = data["trx\_state"].encode('utf8')

INNODB\_TRX\_INFO["trx\_started"] = data["trx\_started"].\_\_str\_\_()

INNODB\_TRX\_INFO["trx\_mysql\_thread\_id"] = data["trx\_mysql\_thread\_id"].\_\_str\_\_()

INNODB\_TRX\_INFO["trx\_query"] = data["trx\_query"].\_\_str\_\_().replace("\r\n", " ").replace("\t", " ")

INNODB\_TRX\_INFO["trx\_operation\_state"] = data["trx\_operation\_state"].\_\_str\_\_()

break

return INNODB\_TRX\_INFO

def \_\_Get\_PROCESSLIST\_\_(self, \_\_PROCESSLIST\_RESULT, \_\_thread\_id):

PROCESSLIST\_INFO = {}

for data in \_\_PROCESSLIST\_RESULT:

if str(data["ID"]) == \_\_thread\_id:

PROCESSLIST\_INFO["HOST"] = data["HOST"].encode('utf8')

PROCESSLIST\_INFO["USER"] = data["USER"].encode('utf8')

PROCESSLIST\_INFO["COMMAND"] = data["COMMAND"].encode('utf8')

PROCESSLIST\_INFO["STATE"] = data["STATE"].encode('utf8')

PROCESSLIST\_INFO["TIME"] = data["TIME"].encode('utf8')

break

return PROCESSLIST\_INFO

def get\_mysqlBlockInfo(self): # 获取阻塞信息列表

Distinct\_INNODB\_LOCK\_WAITS = self.\_\_Distinct\_INNODB\_LOCK\_WAITS\_\_(self.INNODB\_LOCK\_WAITS\_RESULT)

self.BLOCKLIST = []

for data in Distinct\_INNODB\_LOCK\_WAITS:

Info = {}

requestdict = {}

for r in data["requesting\_trx\_id"]:

requested = {"requesting\_trx\_id": r.\_\_str\_\_()}

requested.update(

self.\_\_Get\_INNODB\_LOCKS\_INFO\_\_(self.INNODB\_LOCKS\_RESULT, requested["requesting\_trx\_id"]))

requested.update(self.\_\_Get\_INNODB\_Trx\_\_(self.INNODB\_TRX\_RESULT, requested["requesting\_trx\_id"]))

requested.update(self.\_\_Get\_PROCESSLIST\_\_(self.PROCESSLIST\_RESULT, requested["trx\_mysql\_thread\_id"]))

requestdict[data["requesting\_trx\_id"].index(r) + 1] = requested

Info["requested"] = requestdict

blocking = {"blocking\_trx\_id": data["blocking\_trx\_id"].\_\_str\_\_()}

blocking.update(self.\_\_Get\_INNODB\_LOCKS\_INFO\_\_(self.INNODB\_LOCKS\_RESULT, blocking["blocking\_trx\_id"]))

blocking.update(self.\_\_Get\_INNODB\_Trx\_\_(self.INNODB\_TRX\_RESULT, blocking["blocking\_trx\_id"]))

blocking.update(self.\_\_Get\_PROCESSLIST\_\_(self.PROCESSLIST\_RESULT, blocking["trx\_mysql\_thread\_id"]))

Info["blocking"] = blocking

self.BLOCKLIST.append(Info)

if \_\_name\_\_ == '\_\_main\_\_':

try:

logging.basicConfig(level=logging.DEBUG,

format='%(asctime)s-%(filename)s[line:%(lineno)d]-%(levelname)s-%(message)s',

datefmt='%a, %d %b %Y %H:%M:%S',

filename=\_\_getlogfilename\_\_(),

filemode='a')

console = logging.StreamHandler()

console.setLevel(logging.INFO)

formatter = logging.Formatter('%(asctime)-8s-%(levelname)-12s-%(message)s')

console.setFormatter(formatter)

logging.getLogger('').addHandler(console)

logInfo("program begin")

opts = \_\_getopt\_\_()

blockinfo = mysqlBlockInfo(host=opts["host"], user=opts["user"], passwd=opts["password"], db=opts["db"],

port=opts["port"], charset=opts["charset"])

# blockinfo.\_\_initialize\_\_()

blockinfo.get\_mysqlBlockInfo()

mailbody=yaml.dump(blockinfo.BLOCKLIST, default\_flow\_style=False)

if mailbody <> '[]\n':

logInfo("捕获到数据库阻塞信息\n%s"%(mailbody))

\_\_send\_mail\_\_(body=mailbody, to=opts["to"],

subjet="Mysql Block Info " + opts["host"] + ":" + opts["port"])

else:

logWarn("未捕获到阻塞信息,邮件发送终止")

except Exception, ex:

Error = sys.exc\_info()

for file, lineno, function, text in traceback.extract\_tb(Error[2]):

logErr("File %s, line %s,in %s\n\t\t%s" % (file, lineno, function, text))

logErr(ex)

exit()

**Execute**

python Mysql\_block\_info.py --host=119.254.115.72 -u user -p password --itemname="mysql.status[ Innodb\_row\_lock\_current\_waits,3306]" --to="dan.su@yooli.com,dba@yooli.com"