Centralized server data logging to ClickHouse DB

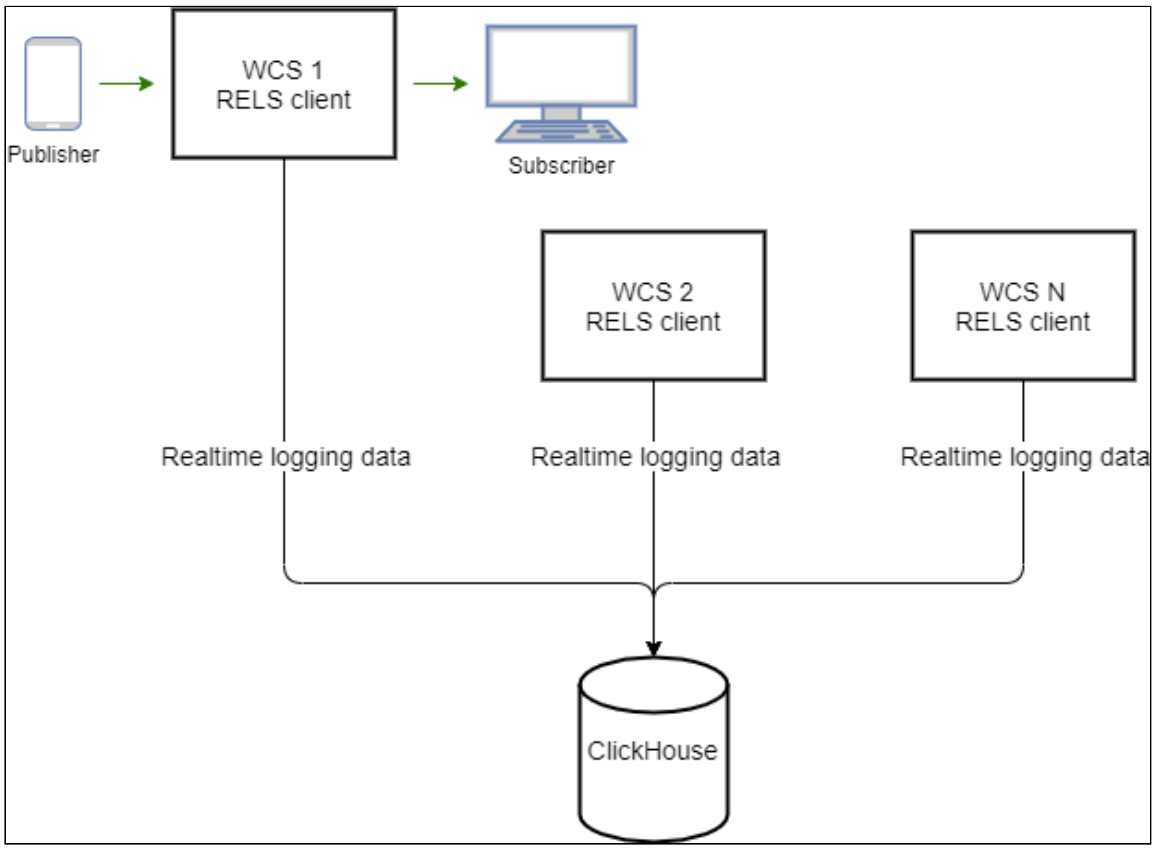
Overview

It may be necessary to collect stream, client connections and CDN events data while managing a big number of WCS servers, to debug a streaming problems. In fact, the information that logged on every server, should be collected at one point. Note that logging itself is minimized in production use, to prevent server disk excessive load.

To collect such big amount of data, time series databases are good choice. Since build 5.2.774, the Remote Event Logging System (RELS) based on open source time series DB ClickHouse can be used to collect server logs.

Architecture

Every WCS server sends logging data to ClickHouse DB independently using JDBC-driver and HTTP connection. To optimize ClickHouse server load, the data are buffered and sent by time interval batch



Data table description

The logging data are collected to ClickHouse tables listed below. To speed up, an integer event identifiers are written to the tables. There is a textual string dictionary decribing all the events for each table to display human readable selection results.

Connections data (table ConnectionEvent)

Field	Type	Description
timestamp	UInt64	Time stamp

Field	Type	Description
ip	IPv4	Server address
sessionId	String	Session Id
eventType	UInt64	Event type Id
eventPayload	String	Event payload

Event types (table ConnectionEventTypes)

Field	Type	Description
id	UInt32	Event type id
type	String	Event type description

Events dictionary (table DictionaryConnectionEvents)

Field	Type	Description
id	UInt64	Event type id
type	String	Event type description

Streams data (table StreamEvent)

Field	Type	Description
timestamp	UInt64	Time stamp
ip	IPv4	Server address
sessionId	String	Session Id
mediaSessionId	String	Media session Id
streamName	String	Stream name
eventType	UInt64	Event type Id
eventPayload	String	Event payload

Event types (table StreamEventTypes)

Field	Type	Description

Field	Type	Description
id	UInt32	Event type id
type	String	Event type description

Events dictionary (table DictionaryStreamEvents)

Field	Type	Description
id	UInt64	Event type id
type	String	Event type description

CDN data (table CDNEvent)

timestamp	UInt64	Time stamp
ip	IPv4	Server address
ip	IPv4	Адрес сервера
nodeId	String	Node Id (IP address strin g)
eventType	UInt64	Event type Id
eventPayload	String	Event payload

Event types (table CDNEventTypes)

Field	Type	Description
id	UInt32	Event type id
type	String	Event type description

Events dictionary (table DictionaryCDNEvents)

Field	Type	Description
id	UInt64	Event type id
type	String	Event type description

Configuration

ClickHouse installation and setup

Server requirement

- CPU from 4 physical cores, frequency from 3 GHz, for example Intel(R) Xeon(R) CPU E3-1246 v3 @ 3.50GHz
- RAM from 32 Gb
- HDD from 2 Tb

ClickHouse installation in CentOS 7

1. Create repository file altinity_clickhouse.repo in /etc/yum.repos.d folder

```
sudo cat <<EOF > /etc/yum.repos.d/altinity_clickhouse.repo
[altinity_clickhouse]
name=altinity_clickhouse
baseurl=https://packagecloud.io/altinity/clickhouse/el/7/$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://packagecloud.io/altinity/clickhouse/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300

[altinity_clickhouse-source]
name=altinity_clickhouse-source
baseurl=https://packagecloud.io/altinity/clickhouse/el/7/SRPMS
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://packagecloud.io/altinity/clickhouse/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
EOF
```

2. Add repository

```
sudo yum -q makecache -y --enablerepo='altinity_clickhouse'
```

3. Install ClickHouse

```
sudo yum install -y clickhouse-server clickhouse-client
```

4. Launch ClickHouse

```
systemctl start clickhouse-server
```

ClickHouse configuration

1. Uncomment the following string in /etc/clickhouse-server/config.xml file to listen all the server network interfaces

```
    <listen_host>::</listen_host>
```

2. Set the following parameter for default user in /etc/clickhouse-server/users.xml file to temporary allow users managent

```
            <access_management>1</access_management>
```

3. Restart ClickHouse

```
systemctl restart clickhouse-server
```

4. Create wcs database and tables

```
cat wcs_clickhouse.sql | clickhouse-client -mn
```

wcs_clickhouse.sql Expand source

```
CREATE DATABASE IF NOT EXISTS wcs;

DROP TABLE IF EXISTS wcs.StreamEvent;

DROP TABLE IF EXISTS wcs.ConnectionEvent;

DROP TABLE IF EXISTS wcs.CDNEvent;

DROP TABLE IF EXISTS wcs.StreamEventTypes;

DROP TABLE IF EXISTS wcs.ConnectionEventTypes;

DROP TABLE IF EXISTS wcs.CDNEventTypes;

DROP DICTIONARY IF EXISTS wcs.DictionaryStreamEvents;

DROP DICTIONARY IF EXISTS wcs.DictionaryConnectionEvents;

DROP DICTIONARY IF EXISTS wcs.DictionaryCDNEvents;

CREATE TABLE wcs.ConnectionEventTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.ConnectionEventTypes VALUES (0, 'CONNECTED'), (1,
```

```
'DISCONNECTED');

CREATE TABLE wcs.StreamEventTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.StreamEventTypes VALUES (0,'CREATED'),
(1,'LOCAL_SDP_CREATED'),(2,'REMOTE_SDP_RECEIVED'),(3,'ICE_STARTED'),
(4,'ICE_COMPLETE'),(5,'DTLS_STARTED'),(6,'DTLS_COMPLETE'),(7,'INITIALIZED'),
(8,'DISPOSING'),(9,'DISPOSED'),(10,'AUDIO_RECEIVED'),(11,'VIDEO_RECEIVED'),
(12,'VIDEO_KFRAME_RECEIVED'),(13,'AUDIO_RTCP_RECEIVED'),
(14,'VIDEO_RTCP_RECEIVED'),(15,'RESOLUTION_RECEIVED'),
(16,'VIDEO_ENCODER_CREATED'),(17,'AUDIO_ENCODER_CREATED'),
(18,'VIDEO_ENCODER_DISPOSED'),(19,'AUDIO_ENCODER_DISPOSED'),
(20,'TERMINATED'),(21,'AUDIO_SENT'),(22,'VIDEO_SENT'),
(23,'VIDEO_JITTER_BUFFER_STALL'),(24,'SENT_PLI'),(25,'RECEIVED_PLI'),
(26,'SYNC_BUFFER_FULL'),(27,'SYNC_FORCE_FAILED'),(28,'SYNC_SHIFT'),
(29,'SYNC_DEVIATION'),(30,'VIDEO_STATS'),(31,'RECORD');

CREATE TABLE wcs.CDNEventTypes
(
    `id` UInt32,
    `type` String
)
ENGINE = MergeTree()
ORDER BY id
SETTINGS index_granularity = 8192;

INSERT INTO wcs.CDNEventTypes VALUES (0, 'STATE'), (1, 'CDN_STATE'), (2,
'VERSION'), (3, 'ACL_REFRESH'), (4, 'ACL_UPDATE');

CREATE DICTIONARY wcs.DictionaryStreamEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'StreamEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE DICTIONARY wcs.DictionaryConnectionEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
```

```
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'ConnectionEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE DICTIONARY wcs.DictionaryCDNEvents (
    `id` UInt16,
    `type` String DEFAULT ''
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(
 host 'localhost'
 port 9000
 user 'default'
 password ''
 db 'wcs'
 table 'CDNEventTypes'
))
LAYOUT(FLAT())
LIFETIME(300);

CREATE TABLE wcs.StreamEvent
(
    `timestamp` UInt64,
    `ip` IPv4,
    `sessionId` String,
    `mediaSessionId` String,
    `streamName` String,
    `eventType` UInt64,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY (sessionId, mediaSessionId, streamName)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.ConnectionEvent
(
    `timestamp` UInt64,
    `ip` IPv4,
    `sessionId` String,
    `eventType` UInt64,
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY (timestamp, sessionId)
SETTINGS index_granularity = 8192;

CREATE TABLE wcs.CDNEvent
(
    `timestamp` UInt64,
    `ip` IPv4,
    `nodeId` String,
    `eventType` UInt64,
```

```
    `eventPayload` String
)
ENGINE = MergeTree()
ORDER BY (nodeId, eventType)
SETTINGS index_granularity = 8192;
```

5. Create wcs user and grant permissions to all the tables in wcs database

```
cat wcs_clickhouse_users.sql | clickhouse-client -mn
```

wcs_clickhouse_users.sql Expand source

```
CREATE USER IF NOT EXISTS wcs IDENTIFIED BY 'wcs';
GRANT ALL ON wcs.* TO wcs WITH GRANT OPTION;
```

6. Disable users management for default user by setting the following parameter in /etc/clickhouse-server/users.xml file

```
            <access_management>0</access_management>
```

7. Restart ClickHouse

```
systemctl restart clickhouse-server
```

WCS configuration

Data logging to ClickHouse is enabled by the following parameter

```
rels_enabled=true
```

ClickHouse server and database address is set by the following parameter

```
rels_database_address=jdbc:clickhouse://clickhouseserver:8123/wcs?
user=wcs&password=wcs
```

Stop data logging without WCS server restart

Data logging can be stopped without WCS restart if necessary. To do this:

1. Disable data logging in server settings

```
rels_enabled=false
```

2. Reload settings using CLI command

```
reload node-settings
```

ClickHouse server address changing without WCS server restart

ClickHouse server address can be changed without WCS restart. To do this:

1. Change address in server settings

```
rels_database_address=jdbc:clickhouse://newclickhouseserver:8123/wcs?
user=wcs&password=wcs
```

2. Disable data logging in server settings

```
rels_enabled=false
```

3. Reload settings using CLI command

```
reload node-settings
```

4. Enable data logging in server settings

```
rels_enabled=true
```

5. Reload settings using CLI command

```
reload node-settings
```

Data retrieving from DB

Logging data can be retrieved usin SQL queries in CliskHouse client

```
select
timestamp,ip,sessionId,mediaSessionId,streamName,dictGetString('wcs.DictionarySt
 eventType) as eventType from wcs.StreamEvent where streamName = 'test'
```

```
select
timestamp,ip,sessionId,dictGetString('wcs.DictionaryConnectionEvents','type',
eventType) as eventType from wcs.ConnectionEvent
```

```
select timestamp,ip,nodeId,dictGetString('wcs.DictionaryCDNEvents','type',
eventType) as eventType,eventPayload from wcs.CDNEvent
```

Attachments:

- RELS.png (image/png)