PostgreSQL GSoC Contributor Proposal

Summary
self link:
https://github.com/xRay2016

This project aims to integrate the PostgreSQL Write-Ahead Log (WAL) data structures into pgmoneta.

Owner: xray20161@gmail.com
Contributors: Donglin Xie
Created: 2022.04.15

Project Abstract

In this project, we would replace the usage of pg_receivewal with a native solution of pgmoneta. Currently, the pgmoneta use the system(cmd) to call pg_receivewal, so it is hard to track the reason of failure. This project create native solution and other infrastructure for WAL interaction.

Problem Background

*In the pgmoneta, the main event loop registers periodic event of **wal_streaming**. The **wal_streaming_cb** callback function is triggered periodically to start backing up the server's WAL. https://github.com/pgmoneta/pgmoneta/blob/5357694ffb223f0b22487b1dac544a5d4077c843/src/main.c#L604*

*In the callback function, the **pgmoneta_wal** function is called to fetch i-th server's WAL. In the function **pgmoneta_wa,** the shell command is spliced to call executable program pg_receivewal. The pg_receivewal stream the write-ahead log from a running PostgreSQL to the specific directory in real time. The write-ahead log is streamed over a regular PostgreSQL connection and uses the replication protocol. The pg_reveivewal will run until terminated by the SIGINT signal. The source code of pg_receivewal is in https://doxygen.postgresql.org/pg__receivewal_8c.html#a3c04138a5bfe5d72780bb7e82a18e627.*

Design Ideas

In my opinion, the native solution could be divided into three parts.

1. the options analysis and storage
2. create the replication connection with remote server, create replication slot and determine the segment size
3. keep cycling to get the stream WAL data.

According to the design idea, the most important part is the **streamCtl** struct. It is used to storage the streaming configurations and the specific WAL writing method.

The **StreamLog** method use **streamCtl** as the configuration while reading. Before reading, the StreamLog initialize the connection and get the segment size from the remote server.

During reading, use the deterministic finite automaton to parse the replication protocol. Until the end of the stream, the content in the reading buffer is flushed to the disk with the specific WAL format. The WAL format is defined by the function pointer in **streamCtl.**

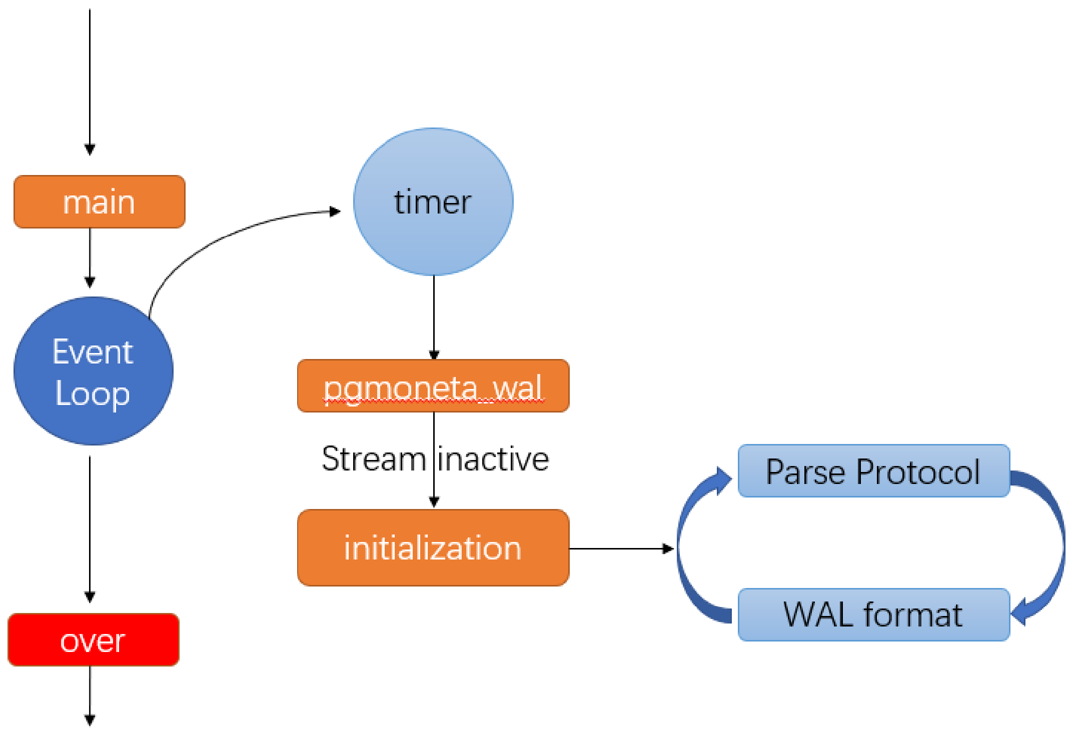

Figure 1 Workflow of WAL backup

Deliverables

1. *A native solution to replacement the usage of pg_receivewal*
2. *A foundation for a pgmoneta library that can be used for WAL interaction*

3. *Detailed code documentation*

Schedule of Deliverables (timeline)

May 20 - June 1

- Study the PostgreSQL format of WAL, and the replication protocol over a regular PostgreSQL connection.

June 1 - July 27 (Phase I)

- Code for native solution of WAL and unit test.
- (1 week) design the streamCtl struct and the server connection initialization before stream reading
- (2 weeks) develop the reading and parsing part for the initialized connection
- (2 weeks) develop the part WAL disk flush part
- (2 week) make unit test and fix bugs. reserve one week to prevent accidents

July 25 - September 12 (Phase II)

- Other foundation for a pgmoneta library and write the code document

About Me

My name is Donglin Xie, a student pursuing a master degree in Computer Science at Zhejiang University, China. I am passionate about programing and open source. I hope I can get the opportunity to contribute to the PostgreSQL community.
I have abundant experience in C/C++, linux and network programing. Here are some of my experience.

1. Tencent Rhino Bird Open Source Training Program-Tars(C++)
(1) Realizes support for the subset function in Tarscpp. It can distribute traffic to different Subsets based on the configured Subset flow rules, achieving more effective development and testing (grayscale testing).
(2) Realize the transparent transmission of TARS_ROUTE_KEY in the Tars structure.
(3) Realize proportional (consistent hashing), select according to request parameters (regular matching) and subset of random rules
github: https://github.com/TarsCloud/TarsCpp

2. High-performance Web server based on C++(C++)
(1) A single Reactor multi-threaded model is realized by using multiple IO multiplexing technology Epoll and thread pool
(2) Use regular matching and state machines to parse HTTP request messages to achieve static resource processing requests
(3) Based on the timer implemented by the heap, close the timeout inactive connection
(4) Realizes the database connection pool of the RAII mechanism, reducing the overhead of establishing and closing database connections
(5) A blocking queue is used to implement an asynchronous log system to record the running status of the server
github: https://github.com/xRay2016/CppWebServer

3. ByteDance Back-end Training Camp: Red Packet Rain Project(Go)
(1) Use Redis to cache Mysql data, realize efficient data reading, and introduce Kafka to write to Mysql asynchronously
(2) The Bloom filter is introduced in the local cache, to reduce the redis network request as much as possible and improve the throughput
(3) The token bucket algorithm is used to realize the current limit of the interface
(4) Use Hystrix-go to deal with the surge in traffic
github: https://github.com/MySuperSoul/teccamp-envelop-rain

Availability Scheduler and Others

I can commit more than 40 hours every week to achieve the goals and deliverables. In weekdays, the working hours are from 6pm to 11pm. The working hours in weekend are from 10am to 9pm. Every weeks, I will make a progress report to the mentor.

The contribution to the open-source project is not limited in GSoC. I would like to continue working on the open-source project. It is a challenging but interesting thing. After the GSoC, do more thing for the PostreSQL community.