Alibaba Summer of Code Proposal

Part 1 Biographical Information

Name: -Deepanshu Udhwani

Email Id: -deepanshu1422@gmail.com

Contact Number: -+91-7018765080

University: Thapar Institute of Engineering and Technology

Qualification: - (Bachelors in Computer Science, currently perusing Masters)

Time zone: IST (GMT + 0530)

Country of Residence: India

GitHub ID:- deepanshu1422

Project :-https://github.com/alibaba/cloud-kernel/issues/9

Mentors:- Joseph Qi,Wen Yang,@wangxiaoguang

Alibaba Cloud Linux

Personal Experience:

Being a computer science student with a specialization in Electronics and Communication I am interested in kernels from my freshman year, I think this is a very good opportunity for me to get hands-on development experience in this field. I am prepared to learn more about it in the summers and explore this interesting field. I think given my interests, I would like to add more functionalities to my kernel patches after ASOC as and when time permits me.

I have a certification in kernel development along with a fairly good experience https://www.youracclaim.com/badges/82d60cbd-8330-4eb2-b053-9ec2547dca2b

Recently,I did a virtual internship with the giant KPMG in the field of data analytics where PostgreSQL was the major database in use
https://github.com/deepanshu1422/MyCertificates/blob/master/KPMG.pdf

I love to help community thats why I joined
https://people.communitybridge.org/project/71777df4-4bad-46e9-8813-693043a57ae4
as mentee and doing my part

I also interned at the Computation Acceleration team at Thapar Institute Of Engineering And Technology wherein I developed a synthesizable arbitrary precision fixed and floating-point library for their High-Level Synthesis tool which also works with Vivado and Calypto. I thus feel that I have experience with the concepts and skills needed to complete this project.

Coding Preferences

I have been interested in Open Source programming since my freshman year at college. I was a part of the Autonomous Vehicle team as a Software engineer where we developed the entire stack for our (which reached the semi-finals at RoboSub in 2016, 2017 and 2018). I have also done a bunch of other parallel computing, computer vision and image processing like a chess-playing robot and written a lot of utility software including Django servers and mobile kernels in accordance with TWRP.

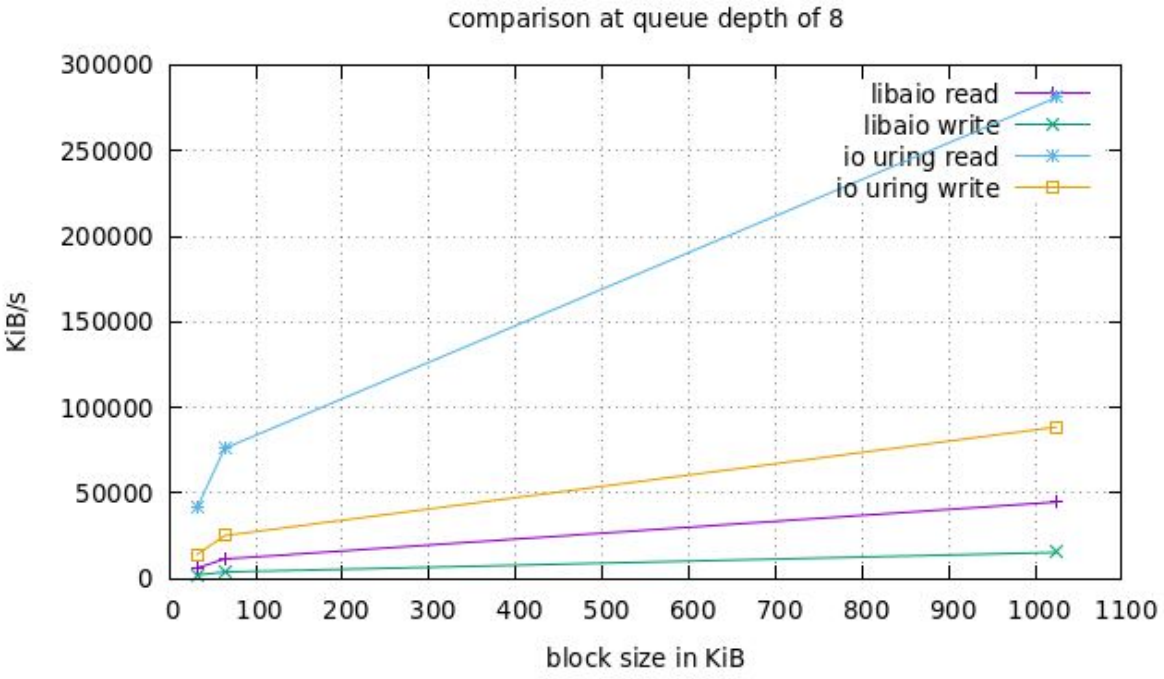
I code on my Ubuntu Desktop using vim as I think the availability of a large number of plugins for vim makes it customizable to suit my needs. I am comfortable with a lot of programming languages but I specifically like C++ because of its intuitive interface and a very broad spectrum of usage from super speed. One specific feature I like about C is playing close to the kernel :).

Part 2 Your Proposal

[SoC2020] PostgreSQL: add io_uring support

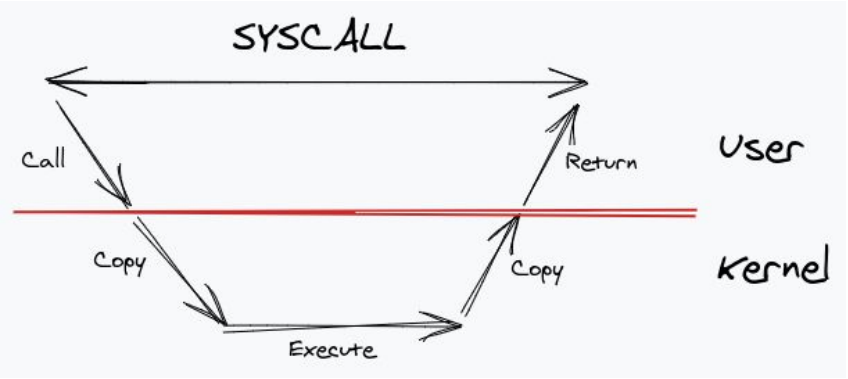
The native Linux AIO framework suffers from various limitations, which io_uring aims to overcome:

- It does not support buffered I/O, only direct I/O is supported.
- It has non-deterministic behaviour which may block under various circumstances.
- It has a sub-optimal API, which requires at least two system calls per I/O, one to submit a request, and one to wait for its completion.
- Each submission needs to copy 64 + 8 bytes of data, and each completion needs to copy 32 bytes.



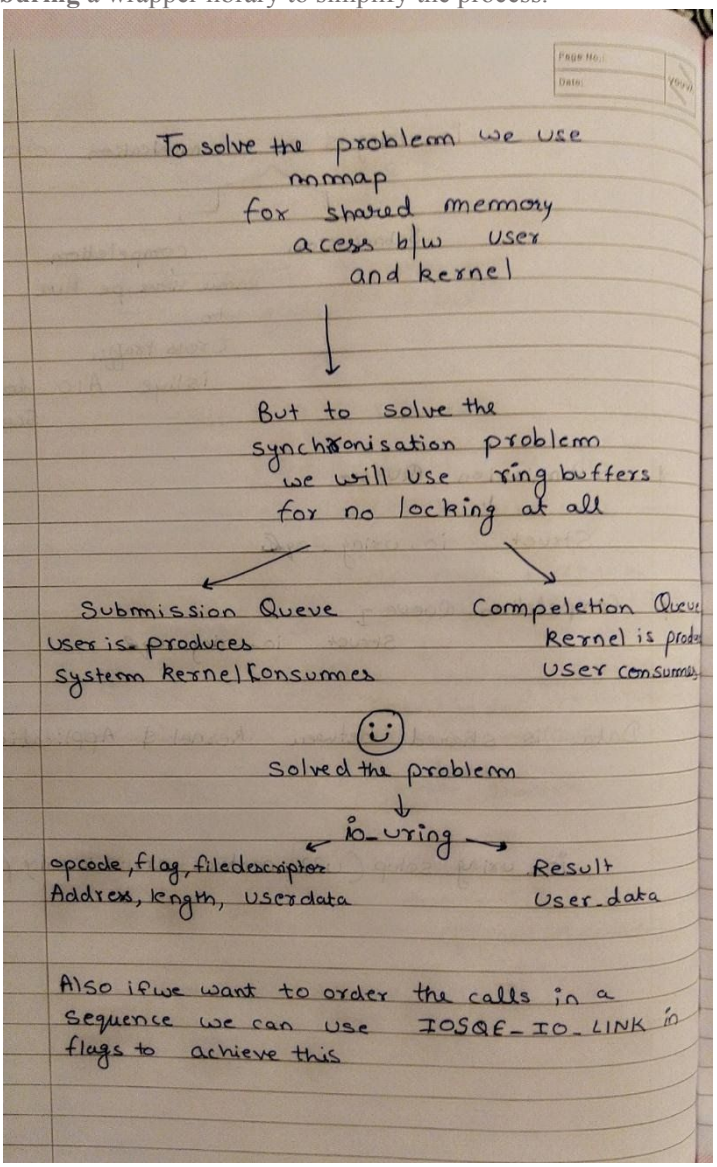
I believe that io_uring is the biggest thing to happen to the Linux kernel in a very long time. It will change everything for PostgreSQL from disk load speeds to processing speeds. It started as a way to do real async disk IO without needing to use O_DIRECT, but its scope has expanded and it will continue to support more and more kernel functionality over time due to its ability to batch large numbers of different syscalls.

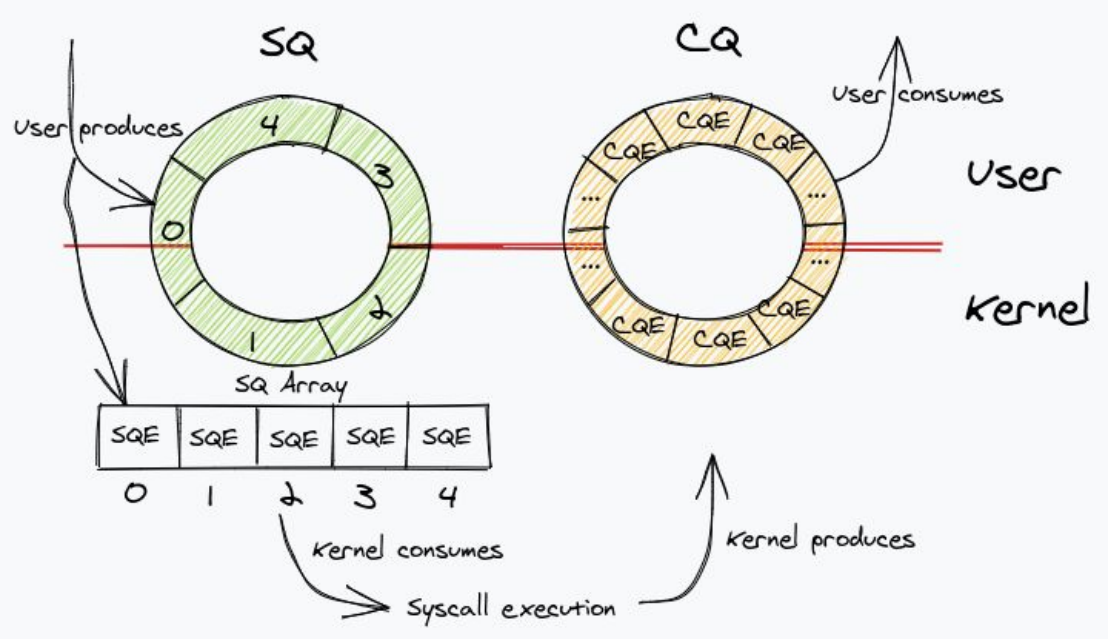
Implementation plan



Here is evident, there are multiple copy operations in AIO and we could minimize this by using submission queue and completion queue.

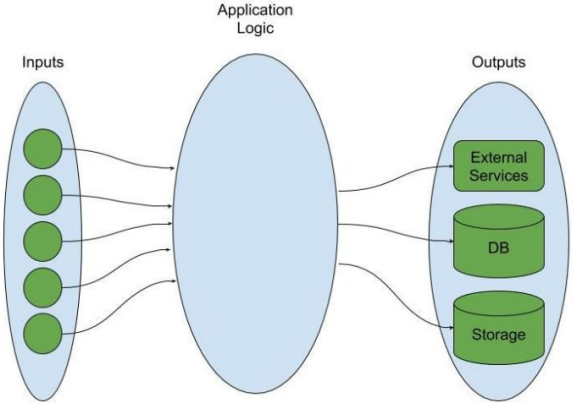
I'll prefer to use **liburing** a wrapper library to simplify the process.

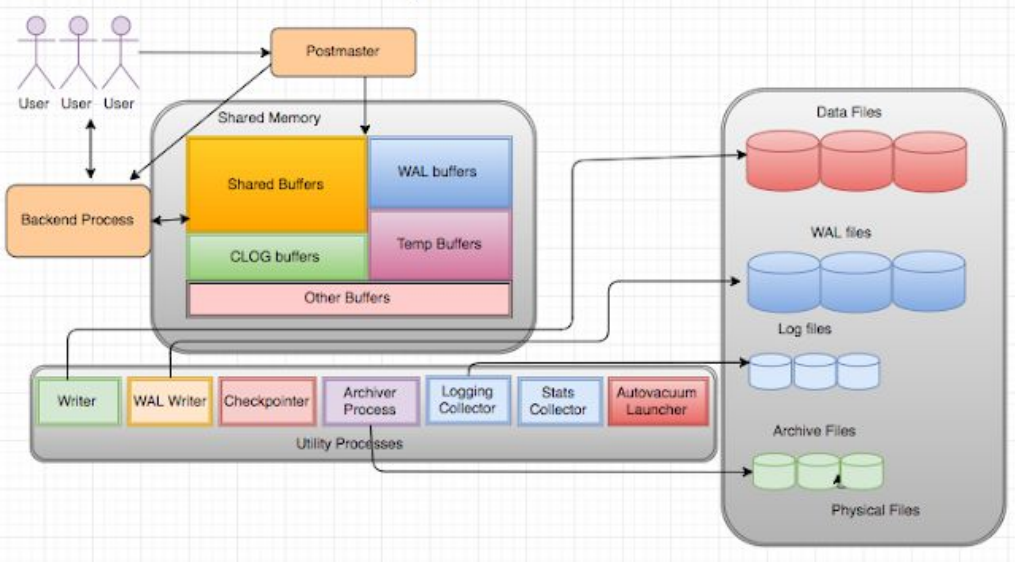




With liburing, I would roughly/have to do mainly these set of steps to make it work with PostgreSQL

- io_uring_queue_(init|exit) to set up and tear down the ring.
- io_uring_get_sqe to get an SQE.
- io_uring_prep_(readv|writev|other) to mark which syscall to use.
- io_uring_sqe_set_data to mark the user data field.
- io_uring_(wait|peek)_cqe to either wait for CQE or peek for it without waiting.
- io_uring_cqe_get_data to get back the user data field.
- io_uring_cqe_seen to mark the CQE as done.
- My initial step in this direction will be checkout the pre existing code base and the previous implementation of I/O calls
- The tuning of PostgreSQL will be based on mainly these factors i.e. Indexing, Partitioning, Checkpoints and I/O problems

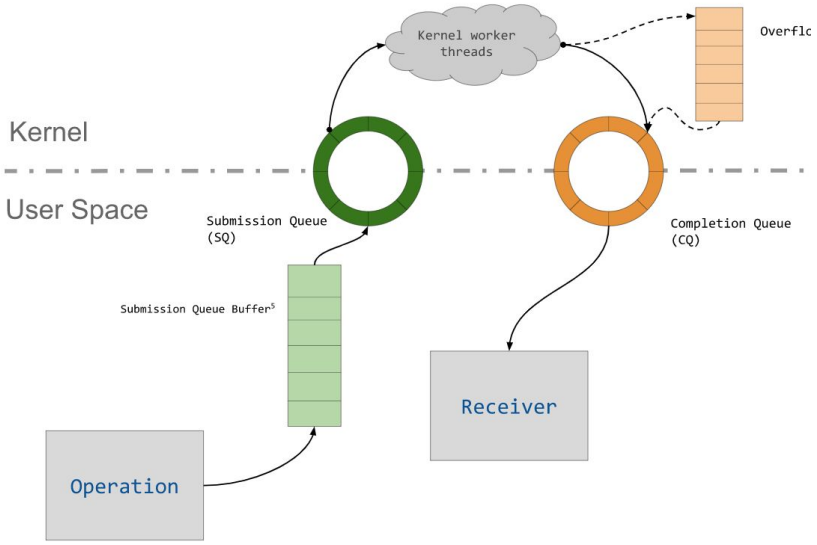




- As we know the default maximum segment size is 32 MB, which is only adequate for very small PostgreSQL installations and therefore moving it to io_uring will help in boosting its performance phenomenally

- using Pg_top to monitor database dynamically

io_uring is an interface for fully asynchronous Linux syscalls the old AIO interface forces O_DIRECT, isn't async sometimes



Mainly the project would be based upon the implementation given in :

https://kernel.dk/io_uring.pdf

https://www.phoronix.com/scan.php?page=news_item&px=Linux-5.6-IO-uring-Tests

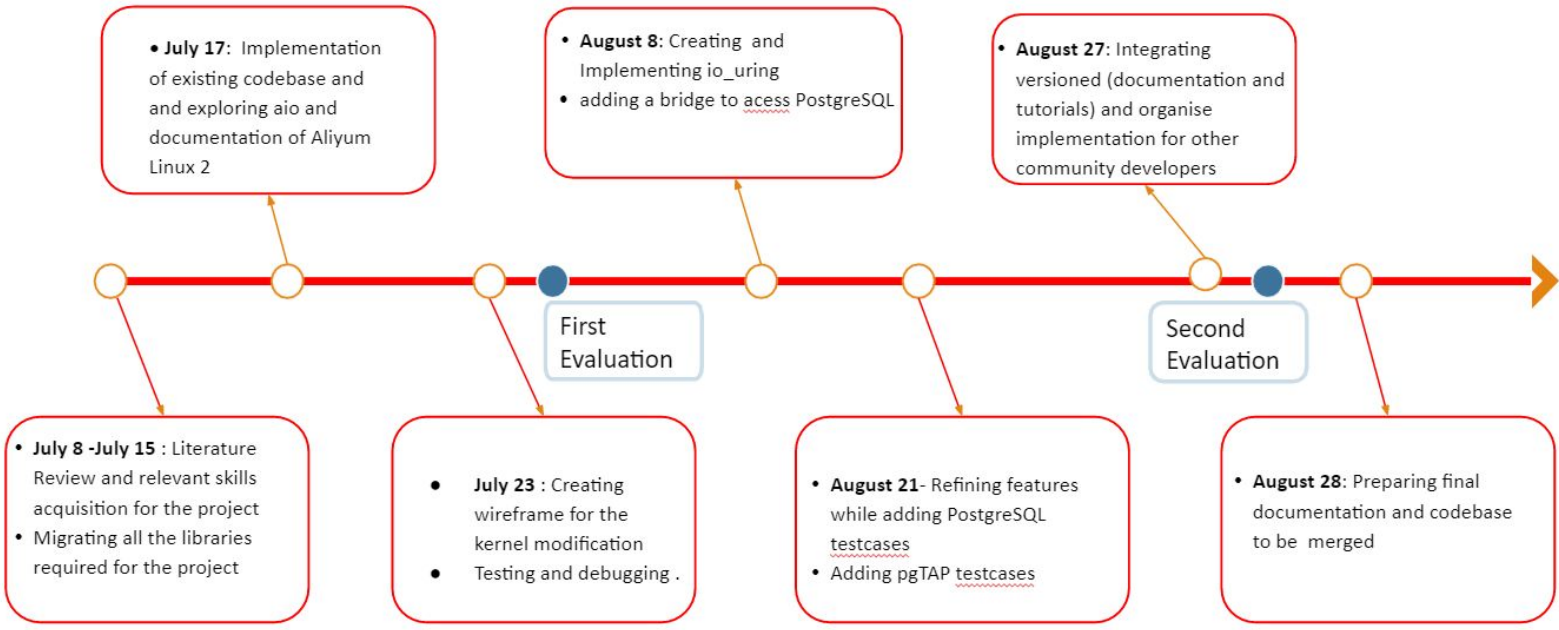
This is how the pseudo implementation of i/o thread interfacing PostgreSQL will look like

```
void io_context::run() {
   io_uring ring;
   io_uring_queue_init(URING_ENTRIES, &ring, 0);
   struct io_uring_cqe* cqe;
   while(true) {
      add_pending_operations_to_io_uring();
      io_uring_wait_cqe(&ring, &cqe); // single syscall to submit and wait
      auto* operation = operation_from_completion(cqe);
      io_uring_cqe_seen(&ring, cqe);
      execute_completion(cqe);
   }
   io_uring_queue_exit(&m_ring);
}
```

Deliverables

1. The implementation should be complete and pass both the Unit and Integration tests.
2. To pass PostgreSQL related test cases
3. Make PostgreSQL with io_uring support run normally on Aliyun Linux 2
4. performance optimization compared to the original version
5. The implementation should be faster than the old transports
6. set of unit tests that extend the existing test suite

Timeline


• July 17: Implementation of existing codebase and and exploring aio and documentation of Aliyum Linux 2
• August 8: Creating and Implementing io_uring
• adding a bridge to acess PostgreSQL
• August 27: Integrating versioned (documentation and tutorials) and organise implementation for other community developers
First Evaluation
Second Evaluation
• July 8 -July 15 : Literature Review and relevant skills acquisition for the project
• Migrating all the libraries required for the project
• July 23 : Creating wireframe for the kernel modification
• Testing and debugging .
• August 21- Refining features while adding PostgreSQL testcases
• Adding pgTAP testcases
• August 28: Preparing final documentation and codebase to be merged

08 July – 15 July (Community Bonding Period)
Familiarize with the code and the community. Read docs, code, set up the environment, browse, write and debug C code the easy way.
Read about io_uring and its adaptations, forums etc. Check if someone did something similar already.
Discussion with mentors about workflow, upcoming challenges and limitations.

15 July – 1 August
Problem discussion and design with mentors. Write C interfaces for io_uring.
Add a new epoll-iou transport module that is at the beginning very similar to epoll.
Try to remove some example system calls in new epoll-iou transport in favour of a new API and test it.

1 August - 15 August
Continue work on removing system calls for Linux sockets operations for PostgreSQL (read, write open, close, accept).
Try to optimize the number of system calls by the correct usage of io_uring and batching.
Add tests for the work done.
Take feedback from the community and iterate on the designs and improvise on use cases. Ensure code quality by adding more test cases and working with more videos.
Work to make documents, blogs or videos to help increase the user base for this product(Subject to developer community approval).

15 August – 27 August

Documentation of the solution, optimizations, performance tests to determine how the final solution improved on Aliyun Linux 2.
If there is enough time trying to find out other ways to benefit from io_uring usage in PostgreSQL

Obligations

● I will be able to work full-time (a minimum of 40 hours a week).

● I have no obligations in July and August so I'll be able to work for a minimum of 40 hours a week.

General Notes

I believe communication is very vital to the successful completion of ASoC, as well as coordination with the Alibaba Kernel community. The efforts I will be making in this regard are:

- Participate in all bi-weekly meetings and any other relevant meetings.
- Keeping my mentor(s) informed about my progress on daily progress.
- Weekly blog posts about project progress, including:
  - Deliverables for the week.
  - Hurdles (if any) encountered while delivering for the week.
  - How the hurdles were surmounted.
- Communicate with the community on GitHub and Slack.
- Maintain a public Google Doc with daily progress.
- Create a public GitHub tracker repository with relevant information about the project progress.