GSoC 2024 Ideas list

This page is auto-generated! Please do NOT edit it, all changes will be lost on next update

Contents

- Airavata
  - Local user interface for Airavata MFT
- Apache Dubbo
  - Unified IDL control for multiple protocols
  - Python integration & AI Traffic Management
  - Traffic Management for Dubbo-go, feature enhancement and demonstration.
  - Universal control plane and console for Apache Dubbo
- Apache NuttX
  - NuttX NAND Flash Subsystem
  - Rust integration on NuttX
  - Device Tree support for NuttX
  - Micro-ROS integration on NuttX
  - Add X11 graphic support on NuttX using NanoX
  - TinyGL support on NuttX
- Kvrocks
  - [GSoC] Kvrocks Support time series data structure and commands like Redis
  - [GSoC] Kvrocks Support embedded storage for Kvrocks cluster controller
- ShenYu
  - Apache ShenYu KitexPlugin
- Doris
  - [GSoC][Doris] Support UPDATE for Doris Duplicate Key Table
  - [GSoC][Doris] Dictionary encoding optimization
- OpenDAL
  - Apache OpenDAL ofs, OpenDAL File System via FUSE
  - Apache OpenDAL OVFS Project Proposal
  - Apache OpenDAL oltp, OpenDAL FTP Server
  - Apache OpenDAL Ofs via CloudFilter Project
- EventMesh
  - Apache EventMesh Enhance the serverless ability for EventMesh
- ShardingSphere
  - [GSO] ShardingSphere Support for PostgreSQL Search Path Feature
  - [GSO] ShardingSphere Scan and create issues for classes that have not implemented unit tests
  - [GSO] ShardingSphere Supports executing DistSQL via JDBC
- RocketMQ
  - Optimizing Lock Mechanisms in Apache RocketMQ
  - RocketMQ Dashboard Supports RocketMQ 5.0 Architecture and Enhances Usability
- Comdev GSOC
  - [GSoC][HugeGraph] Support Memory Management Module
  - [GSoC][HugeGraph] Support Graph Traversal Expansion API
- SkyWalking
  - [GSO] SkyWalking Add Overview page in BanyanDB UI
  - [GSO] SkyWalking Self-Observability of the query subsystem in BanyanDB
- Beam
  - [GSO][Beam] Build out Beam Use Cases
  - [GSO][Beam] Add connectors to Beam ManagedIO
  - [GSO][Beam] Build out Beam Yaml features
- James Server
  - Implement RFC-8617 The Authenticated Received Chain (ARC) Protocol
- Nutch
  - Overhaul the legacy Nutch plugin framework and replace it with PF4J
- Openmeetings
  - Add blur background filter options on video sharing - AI-ML
- UIMA
  - Support Aggregate Engines in Apache UIMACPP

Airavata

Local user interface for Airavata MFT

Note: This is an issue in github - https://github.com/apache/airavata-mft/issues/114 cross posting in Jira for GSoC purposes.

Currently, Airavata MFT can be accessed through its command line interface and the gRPC API. However, it is really easy if a Docker desktop-like user interface is provided for a locally running Airavata MFT. The functionalities of such an interface can be summarized as follows:

1. Start / Stop MFT Instance
2. Register/ List/ Remove Storage endpoints
3. Access data (list, download, delete, upload) in configured storage endpoints
4. Move data between storage endpoints
5. Search data across multiple storage endpoints
6. Analytics - Performance numbers (data transfer rates in each agent)

We can use ElectronJS to develop this cross-platform user interface. The node.js backend of ElectronJS can use gRPC to connect to Airavata MFT to perform management operations.

**Difficulty:** Major  
**Project size:** ~350 hour (large)  
**Potential mentors:**  
Suresh Marru, mail: smarru (at) apache.org  
Project Devs, mail: dev (at) airavata.apache.org

Apache Dubbo

**Unified IDL control for multiple protocols**

Unified IDL control for multiple protocols

Client and Server layer APIs can support both IDL and Non-IDL modes. For IDL mode (Triple + Protobuf), defining proto file and making use of protoc-gen-triple to generate related code are straightforward. Generated code (XXX.triple.go) would contain statements that invoking APIs provided by Client and Server layers. For Non-IDL mode, it needs users to write invoking code by themselves and is not convenient. Take Dubbo+Hessian2 as example:

**Client Side**

```go
cli, err := client.NewClient()
cli, err := client.NewClient{
    client.WithClientProtocolDubbo(), }
withClientProtocolDubbo(), } if err != nil {
    panic(err)
}
conn, err := cli.Dial("GreetProvider",
    clientWithURL("127.0.0.1:20000"), )
    if err != nil {
        panic(err)
    }
    var resp string
    if err := conn.CallUnary(context.Background(), []interface{}{"hello", "new", "dubbo"}, &resp,
        "Greet"); err != nil {
        logger.Errorf("GreetProvider.Greet err: %s", err)
        errorf("GreetProvider.Greet err: %s", err)
    }
```

**Server Side**
Proposal

Even in Non-IDL mode, code is generated using protobuf IDL. In this way, whether you need schema (Protobuf) or not (Hessian2, Msgpack), it's all uniformly used: Protobuf IDL + generated code.

Details:

1. Generate Dubbo + Hessian2 related code with the help of Protobuf IDL. Compared to XXX.pb.go, XXX.hessian2.go would have much less content (due to Hessian2 schema-free), only structure definitions and the corresponding registration function (hessian2.Register(POJO)).
2. Non-IDL (Hessian2) may not map perfectly to Protobuf IDL, and we need to define our own dialect in a way that is compatible with the official semantics of Protobuf IDL.
3. XXX.dubbo.go content is basically similar to XXX.triple.go, generating code that uses the APIs of Client layer and Server layer.

Prerequisite:

1. Provide tools for Dubbo side users to automatically convert Dubbo interface definitions into Protobuf IDL.
2. Protobuf IDL can support extensions (add Hessian2-specific tag extensions, generate Hessian2-specific content)

Results:

Not only Dubbo + Hessian2, but also Triple + Hessian2, Triple + Json and other Non-IDLs can use the interface in a unified way.

Mentor

- Mentor: Albumen Kevin, Apache Dubbo PMC, albumenj@apache.org
- Mailing List: dev@dubbo.apache.org

Python integration & AI Traffic Management

Background
Dubbo is an easy-to-use, high-performance remote procedure call framework. Most of the AI frameworks are running on Python and suffering unbalanced load for GPUs.

Objectives

1. Enhance Dubbo on Python[1] and support the brand new Triple protocol in Dubbo-Java
2. Introduce a new load balance algorithm for AI, which can gather the metrics from GPUs and select the most idle one to invoke


Recommended Skills

1. Familiar with Python
2. Have a basic understanding of RPC
3. Have a basic understanding of traffic management

Mentor

- Mentor: Albumen Kevin, Apache Dubbo PMC, albumenj@apache.org
- Mailing List: dev@dubbo.apache.org

Traffic Management for Dubbo-go, feature enhancement and demonstration.

Background and Goal

Dubbo is an easy-to-use, high-performance microservice framework that provides both RPC and rich enterprise-level traffic management features.

The community has been working on the improvement of Dubbo's traffic management abilities, to make it support rich features like traffic splitting, canary release, a/b testing, circuit breaker, mocking, etc. Although we have made big progress in the past months, there's still a long way to go before we can announce the first official release.

Now, we want volunteers from the open-source community to join us to work on this task together, including traffic management feature development (xds adaption, etc.), demos that can demonstrate the usage and ability of the traffic management feature, and documentation writing.

Relevant Skills

1. Familiar with Golang
2. Familiar with Service Mesh and Microservice architectures

Potential Mentors

- Jun Liu, Apache Dubbo PMC Chair, junliu@apache.org
- dev@dubbo.apache.org

Universal control plane and console for Apache Dubbo

Background and Goal

Dubbo is an easy-to-use, high-performance microservice framework that provides RPC and rich enterprise-level traffic management features.
The community is working on building a Universal control plane and console, the source code can be found in the incubating repo here. As you can see from the repo, we already had a working prototype that can provide most of the core features like a visualizable UI console, adapting with kubernetes, etc. But it's still far from the complete version we expected, we hope that developers can join us to help improve the kubernetes adaptation, support cross cluster interoperability, and improve compatibility with old Nacos registration centers and other scenarios.

**Relevant Skills**

1. Familiar with Golang
2. Familiar with Service Mesh and Microservice architectures
3. Familiar with Kubernetes

**Potential Mentors**

1. Jun Liu, Apache Dubbo PMC Chair, junliu@apache.org
2. dev@dubbo.apache.org

---

**Apache NuttX**

**NuttX NAND Flash Subsystem**

Currently NuttX has support only for NOR Flash and eMMC as solid state storage.

Although for low-end embedded systems NOR Flash still much used, for some devices that need bigger storage, NAND Flash is a better option, because its price per MB is very low.

In the other NAND Flash brings many challenges: you need to map and track all the bad-blocks, you need to have a good filesystem for wear leveling. Currently the SmartFS and LittleFS offer some kind wear leveling for NOR Flash. It needs to be adapted to NAND Flash.

- **Difficulty**: Major
- **Project size**: ~350 hour (large)

**Potential mentors**:

- Alan Carvalho de Assis, mail: acassis (at) apache.org
- Project Devs, mail: dev (at) nuttx.apache.org

---

**Rust integration on NuttX**

The Rust language is gain some momentum as an alternative to C and C++ for embedded system (https://www.rust-lang.org/what/embedded) and it should be very useful to be able to develop NuttX applications using Rust language.

Sometime Yoshiro Sugino already ported the Rust standard libraries, but it was not a complete port and wasn't integrated on NuttX. Anyway this initial port could be used as starting point for some student willing to add official support on NuttX.

Also it needs to pave the way to support developing NuttX driver in Rust and a complement to C drivers.

- **Difficulty**: Normal
- **Project size**: ~350 hour (large)

**Potential mentors**:

- Alan Carvalho de Assis, mail: acassis (at) apache.org
- Project Devs, mail: dev (at) nuttx.apache.org

---

**Device Tree support for NuttX**

Device Tree will simplify the way as boards are configured to support NuttX. Currently for each board the developer/user need to manually create an initialization file for each feature or device (expect when the device is already in the common board folder).

Matias Nitsche (aka v0id) create a very descriptive and information explanation here: https://github.com/apache/incubator-nuttx/issues/1020

The goal for this project is to add Device Tree support for NuttX and let it to be configurable (low end board should be able to avoid using Device Tree for instance).
Micro-ROS integration on NuttX

Micro-ROS ([https://micro.ros.org](https://micro.ros.org)) is a ROS2 support to Microcontrollers. Initially the project was developed over NuttX by Bosch and other EU organizations. Later on they added support to FreeRTOS and Zephyr. After that NuttX support started ageing and we didn’t get anyone working to fix it (with few exceptions like Roberto Bucher work to test it with pysimCoder).

Add X11 graphic support on NuttX using NanoX

NanoX/Microwindows is a small graphic library which allow Unix/Linux X11 application to run on embedded systems that cannot support X-Server because it is too big. Add it to NuttX will allow many applications to be ported to NuttX. More importantly: it will allow FLTK 1.3 run on NuttX and that could big Dillo web browser.

TinyGL support on NuttX

TinyGL is a small 3D graphical library created by Fabrice Bellard (same creator of QEMU) designed for embedded systems. Currently NuttX RTOS doesn’t have a 3D library and this could enable people to add more 3D programs on NuttX.

Kvrocks

[GSOC] Kvrocks Support time series data structure and commands like Redis

RedisTimeSeries is a redis module used to operate and query time series data, giving redis basic time series database capabilities.

As Apache Kvrocks is characterized by being compatible with the Redis protocol and commands, we also hope to provide temporal data processing capabilities that are compatible with RedisTimeSeries.

This task is to implement the time series data structure and its commands on Kvrocks. Note: Since Kvrocks is an on-disk database based on RocksDB, the implementation will be quite different from Redis.

Recommended Skills

Modern C++, Database Internals (especially for time series databases), Software Engineering and Testing

References

[https://redis.io/docs/data-types/timeseries/](https://redis.io/docs/data-types/timeseries/)

[https://kvrocks.apache.org/community/data-structure-on-rocksdb](https://kvrocks.apache.org/community/data-structure-on-rocksdb)
Mentor

Mentor: Mingyang Liu, Apache Kvrocks PMC Member, twice@apache.org
Mailing List: dev@kvrocks.apache.org
Website: https://kvrocks.apache.org

Difficulty: Major
Project size: ~350 hour (large)
Potential mentors:
  Mingyang Liu, mail: twice (at) apache.org
  Project Devs, mail: dev (at) kvrocks.apache.org

[GSoc] [Kvrocks] Support embedded storage for Kvrocks cluster controller

Currently, the Kvrocks controller supports using multiple external storages like Apache Zookeeer / ETCD and also plans to support more common databases in the future. However, using external components will bring extra operation complexity for users. So it would be great if we could support embedded storage inside the controller, making it easier to maintain the controller service.

We would like participants to help design and implement the solution.

Recommended Skills

Familiar with the Go programming language and Know how the Raft algorithm works.

Mentor

Mentor: Hulk Lin, Apache Kvrocks PMC Member, hulk.website@gmail.com
Mailing List: dev@kvrocks.apache.org
Website: https://kvrocks.apache.org

Difficulty: Major
Project size: ~350 hour (large)
Potential mentors:
  Hulk Lin, mail: hulk (at) apache.org
  Project Devs, mail: dev (at) kvrocks.apache.org

ShenYu

Apache ShenYu KitexPlugin

Description

‘Apache ShenYu’ is a Java native API Gateway for service proxy, protocol conversion and API governance.

‘WASM’ (WebAssembly) bytecode is designed to be encoded in a size- and load-time-efficient binary format. WebAssembly aims to leverage the common hardware features available on various platforms to execute in browsers at machine code speed.

‘WASI’ (WebAssembly System Interface) allows WASM to run in non browser environments such as Linux.

This plugin should base on [WasmPlugin](https://github.com/apache/shenyu/issues/4612), which means other languages, as long as their code can be compiled into WASM bytecode (such as Rust/golang/C++), can be used to write ShenYu plugins.

[kitex](https://github.com/cloudwego/kitex) is a Go RPC framework with high-performance and strong-extensibility for building micro-services.

You can find useful information [here](https://github.com/cloudwego/kitex/issues/1237).

The usage documentation for WasmPlugin is [here](https://shenyu.apache.org/docs/next/developer/custom-plugin/).

Relevant Skills

Know the use of Apache ShenYu, especially the wasm plugin.
Familiar with Golang and Java.
Task List

- [ ] add `shenyu-client-kitex` to [shenyu-client-golang](https://github.com/apache/shenyu-client-golang);
- [ ] add `shenyu-plugin-kitex` module;
- [ ] add `shenyu-spring-boot-starter-plugin-kitex` module;
- [ ] add `shenyu-integrated-test-kitex` module;
- [ ] add doc to [shenyu-website](https://github.com/apache/shenyu-website) for `KitexPlugin`;

Links:

- website [https://shenyu.apache.org/](https://shenyu.apache.org/)
- issues: [https://github.com/apache/shenyu/issues/5425](https://github.com/apache/shenyu/issues/5425)

**Difficultly:** Major

**Project size:** ~350 hour (large)

**Potential mentors:**

- zhangzicheng@apache.org, mail: zhangzicheng@apache.org
- mahaitao@apache.org, mail: mahaitao@apache.org
- dev@shenyu.apache.org, mail: dev@shenyu.apache.org

---

Doris

**[GSoC][Doris]Support UPDATE for Doris Duplicate Key Table**

**Objectives**

Support UPDATE for Doris Duplicate Key Table

Currently, Doris supports three data models, Duplicate Key / Aggregate Key / Unique Key, of which Unique Key has perfect data update support (including UPDATE statement). With the widespread popularity of Doris, users have more demands on Doris. For example, some user needs to perform ETL processing operations inside Doris, but they use Duplicate Key table and hopes that Duplicate Key can also support UPDATE. For Duplicate Key, since there is no primary key can help we locate one specific row, UPDATE is low efficient. The usual practice is to rewrite all the data, even if the user only updates one field of a row of data, he must rewrite at least the segment file it is in. Another potentially more efficient solution is to implement Duplicate Key by combining Unique Key's Merge-on-Write, and the auto_increment column. i.e., let's change the underlying implementation of Duplicate Key to use Unique Key MoW, and add a hidden auto_increment column in the primary key, so that all the keys written by the user to the Unique Key MoW table are not duplicated, which realizes the semantics of Duplicate Key, and since each row of data has a unique primary key, we can reuse the UPDATE capability of Unique Key to support the Duplicate Key's UPDATE

We would like participants to help design and implement the solution, and perform performance testing for comparison and performance optimization.

**Recommended Skills**

- Familiar with C++ programming
- Familiar with the storage layer of Doris

**Mentor**

Mentor: Chen Zhang, Apache Doris Committer, chzhang1987@gmail.com
Mentor: Guolei Yi, Apache Doris PMC Member, yiguolei@gmail.com
Mailing List: dev@doris.apache.org
Website: [https://doris.apache.org](https://doris.apache.org)

**Difficulty:** Major

**Project size:** ~350 hour (large)

**Potential mentors:**

- ZCheng Zhang, mail: zhangzicheng (at) apache.org
- Project Devs, mail: dev (at) shenyu.apache.org

---

[https://doris.apache.org](https://doris.apache.org)
Dictionary encoding optimization

Background
Apache Doris is a modern data warehouse for real-time analytics. It delivers lightning-fast analytics on real-time data at scale.

Objectives
Dictionary encoding optimization
To save storage space, Doris uses dictionary encoding when storing string-type data in the storage layer if the cardinality is relatively low. Dictionary encoding involves mapping string values to integer values using a dictionary. The data can be stored directly as integers, and the dictionary information is stored separately. When reading the data, the integers are converted back to their corresponding string values based on the dictionary.

The storage layer doesn’t know whether a column has low or high cardinality when the data comes in. Currently, the implementation encodes the first page using dictionary encoding, and if the dictionary becomes too large, it indicates a column with high cardinality. Subsequent pages will not use dictionary encoding. However, even for columns with high cardinality, a dictionary page is still retained, which doesn’t save storage space and adds additional memory overhead during reading as well as extra CPU overhead during decoding.
Optimizations can be made to improve the memory and CPU overhead caused by dictionary encoding.

Recommended Skills
Familiar with C++ programming
Familiar with the storage layer of Doris

Mentor
Mentor: Xin Liao, Apache Doris Committer, liaoxinbit@gmail.com
Mentor: YongQiang Yang, Apache Doris PMC Member, dataroaring@gmail.com
Mailing List: dev@doris.apache.org
Website: https://doris.apache.org
Source Code: https://github.com/apache/doris

OpenDAL
Apache OpenDAL ofs, OpenDAL File System via FUSE

Background
OpenDAL is a data access layer that allows users to easily and efficiently retrieve data from various storage services in a unified way. ofs can expose OpenDAL power in fuse way that allow users to mount storage services locally.

Objectives
Implement ofs, allowing users to mount storage services locally for read and write.

Features
In Scope:

- Continuous reading
- Continuous writing
- Random reading
- List dir
- Stat file

Out Scope:

- Random Write
- Xattrs
- Permissions

Tasks

- Implement features that in scope
- Implement tests suite

Recommended Skills

- Familiar with Rust
- Familiar with basic ideas of file system and fuse
- Familiar with OpenDAL Rust Core

Mentor

Mailing List: dev@opendal.apache.org

Mentor: junouyang, Apache OpenDAL PMC Member, junouyang@apache.org

Please leave comments if you want to be a mentor

**Difficulty:** Major
**Project size:** ~350 hour (large)
**Potential mentors:**
- Hao Ding, mail: xuanwo (at) apache.org
- Project Devas, mail: dev (at) opendal.apache.org

Apache OpenDAL OVFS Project Proposal

1 Project Abstract

Virtio is an open standard designed to enhance I/O performance between virtual machines (VMs) and host systems in virtualized environments. VirtioFS is an extension of the Virtio standard specifically crafted for file system sharing between VMs and the host. This is particularly beneficial in scenarios where seamless access to shared files and data between VMs and the host is essential. VirtioFS has been widely adopted in virtualization technologies such as QEMU and Kata Container.

Apache OpenDAL is a data access layer that allows users to easily and efficiently retrieve data from various storage services in a unified manner. In this project, our goal is to reference virtiofsd (a standard vhost-user backend, a pure Rust implementation of VirtioFS based on the local file system) and implement VirtioFS based on OpenDAL.

This storage-system-as-a-service approach conceals the details of the distributed storage system's file system from VMs. This ensures the security of storage services, as VMs do not need to be aware of the information, configuration and permission credentials of the accessed storage service. Additionally, it enables the utilization of a new backend storage system without reconfiguring all VMs. Through this project, VMs can access numerous data services through the file system interface with the assistance of the OpenDAL service deployed on the host, all without their awareness. Furthermore, it ensures the efficiency of file system reading and writing in VMs through VirtioFS support.

2 Project Detailed Description
This chapter serves as an introduction to the overall structure of the project, outlining the design ideas and principles of critical components. It covers the OVFS architecture, interaction principles, design philosophy, metadata operations beyond various storage backend, cache pool design, configuration support, the expected POSIX interface support, and potential usage scenarios of OVFS.

2.1 The Architecture of OVFS

The picture above is the OVFS architecture diagram. OVFS is a file system implementation based on the VirtioFS protocol and OpenDAL. It serves as a bridge for semantic access to file system interfaces between VMs and external storage systems. Leveraging the multiple service access capabilities and unified abstraction provided by OpenDAL, OVFS can conveniently mount shared directories in VMs on various existing distributed storage services.

The complete OVFS architecture consists of three crucial components:

1) VMs FUSE client that supports the VirtioFS protocol and implements the VirtioFS Virtio device specification. An appropriately configured Linux 5.4 or later can be used for OVFS. The VirtioFS protocol is built on FUSE and utilizes the VirtioFS Virtio device to transmit FUSE messages. In contrast to traditional FUSE, where the file system daemon runs in the guest user space, the VirtioFS protocol supports forwarding file system requests from the guest to the host, enabling related processes on the host to function as the guest's local file system.

2) A hypervisor that implements the VirtioFS Virtio device specification, such as QEMU. The hypervisor needs to adhere to the VirtioFS Virtio device specification, supporting devices used during the operation of VMs, managing the file system operations of the VMs, and delegating these operations to a specific vhost-user device backend implementation.

3) A vhost-user backend implementation, namely OVFSD (OVFS daemon). This is a crucial aspect that requires particular attention in this project. This backend is a file system daemon running on the host side, responsible for handling all file system operations from VMs to access the shared directory. virtiofsd offers a practical example of a vhost-user backend implementation, based on pure Rust, forwarding VMs’ file system requests to the local file system on the host side.

2.2 How OVFSD Interacts with VMs and Hypervisor

The Virtio specification defines device emulation and communication between VMs and the hypervisor. Among these, the virtio queue is a core component of the communication mechanism in the Virtio specification and a key mechanism for achieving efficient communication between VMs and the hypervisor. The virtio queue is essentially a shared memory area called vring between VMs and the hypervisor, through which the guest sends and receives data to the host.

Simultaneously, the Virtio specification provides various forms of Virtio device models and data interaction support. The vhost-user backend implemented by OVFSD achieves information transmission through the vhost-user protocol. The vhost-user protocol enables the sharing of virtio queues through communication over Unix domain sockets. Interaction with VMs and the hypervisor is accomplished by listening on the corresponding sockets provided by the hypervisor.

In terms of specific implementation, the vm-memory crate, virtio-queue crate and vhost-user-backend crate play crucial roles in managing the interaction between OVFSD, VMs, and the hypervisor.

The vm-memory crate provides encapsulation of VMs memory and achieves decoupling of memory usage. Through the vm-memory crate, OVFSD can access relevant memory without knowing the implementation details of the VMs memory. Two formats of virtio queues are defined in the Virtio specification: split virtio queue and packed virtio queue. The virtio-queue crate provides support for the split virtio queue. Through the DescriptorChain package provided by the virtio-queue crate, OVFSD can parse the corresponding virtio queue structure from the original vring data. The vhost-user-backend crate provides a way to start and stop the file system demon, as well as encapsulation of vring access. OVFSD implements the vhost-user backend service based on the framework provided by the vhost-user-backend crate and implements the event loop for the file system process to handle requests through this crate.

2.3 OVFS Design Philosophy

In this section, we will present the design philosophy of the OVFS project. The concepts introduced here will permeate throughout the entire design and implementation of OVFS, fully manifesting in other sections of the proposal.
Stateless Services

The mission of OVFS is to provide efficient and flexible data access methods for VMs using Virtio and VirtioFS technologies. Through a stateless services design, OVFS can easily facilitate large-scale deployment, expansion, restarts, and error recovery in a cluster environment running multiple VMs. This seamless integration into existing distributed cluster environments means that users do not need to perceive or maintain additional stateful services because of OVFS.

To achieve stateless services, OVFS refrains from persisting any metadata information. Instead, it maintains and synchronizes all state information of the OVFS file system during operation through the backend storage system. There are two implications here: OVFS doesn't need to retain additional operational status during runtime, and it doesn't require the maintenance of additional file system metadata when retrieving data from the backend storage system. Consequently, OVFS doesn't necessitate exclusive access to the storage system. It permits any other application to read and write data to the storage system when it serves as the storage backend for OVFS. Furthermore, OVFS ensures that the usage semantics of data in the storage system remain unchanged. All data in the storage system is visible and interpretable to other external applications.

Under this design, OVFS alleviates concerns regarding synchronization overhead and potential consistency issues stemming from data alterations in the storage system due to external operations, thereby reducing the threshold and risks associated with OVFS usage.

Storage System As A Service

We aspire for OVFS to serve as a fundamental storage layer within a VM cluster. With OVFS's assistance, VMs can flexibly and conveniently execute data read and write operations through existing distributed storage system clusters. OVFS enables the creation of distinct mount points for various storage systems under the VMs' mount point. This service design pattern facilitates mounting once to access multiple existing storage systems. By accessing different sub-mount points beneath the root mount point of the file system, VMs can seamlessly access various storage services, imperceptible to users.

This design pattern allows users to customize the data access pipeline of VMs in distributed clusters according to their needs and standardizes the data reading, writing, and synchronization processes of VMs. In case of a network or internal error in a mounted storage system, it will not disrupt the normal operation of other storage systems under different mount points.

User-Friendly Interface

OVFS must offer users a user-friendly operating interface. This entails ensuring that OVFS is easy to configure, intuitive, and controllable in terms of behavior. OVFS accomplishes this through the following aspects:

1) It's essential to offer configurations for different storage systems that align with OpenDAL. For users familiar with OpenDAL, there's no additional learning curve.

2) OVFS is deployed using a formatted configuration file format. The operation and maintenance of OVFS only require a TOML file with clear content.

3) Offer clear documentation, including usage and deployment instructions, along with relevant scenario descriptions.

2.4 Metadata Operations Beyond Various Storage Backend

OVFS implements a file system model based on OpenDAL. A file system model that provides POSIX semantics should include access to file data and metadata, maintenance of directory trees (hierarchical relationships between files), and additional POSIX interfaces.

Lazy Metadata Fetch In OVFS
OpenDAL natively supports various storage systems, including object storage, file storage, key-value storage, and more. However, not all storage systems directly offer an abstraction of file systems. Take AWS S3 as an example, which provides object storage services. It abstracts the concepts of buckets and objects, enabling users to create multiple buckets and multiple objects within each bucket. Representing this classic two-level relationship in object storage directly within the nested structure of a file system directory tree poses a challenge.

To enable OVFS to support various storage systems as file data storage backends, OVFS will offer different assumptions for constructing directory tree semantics for different types of storage systems to achieve file system semantics. This design approach allows OVFS to lazily obtain metadata information without the need to store and maintain additional metadata. Additional metadata not only leads to synchronization and consistency issues that are challenging to handle but also complicated OVFS's implementation of stateless services. Stateful services are difficult to maintain and expand, and they are not suitable for potential virtualization scenarios of OVFS.

**Metadata Operations Based On Object Storage Backend**

The working principle of OVFS based on the object storage backend is to translate the storage names of buckets and objects in object storage into files and directory systems in the file system. A comprehensive directory tree architecture is realized by treating the bucket name as a full path in the file system and treating the slash character "/" in the bucket name as a directory delimiter. All objects in each bucket are considered as files in the corresponding directory. File system operations in the VMs can interact with the object storage system through similar escape operations to achieve file system-based data reading and writing. The following table lists the mapping of some file system operations in the object storage system.

<table>
<thead>
<tr>
<th>Metadata Operations</th>
<th>Object Storage Backend Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>create a directory with the full path &quot;xxx/yyy&quot;</td>
<td>create a bucket named &quot;xxx/yyy&quot;</td>
</tr>
<tr>
<td>remove a directory with the full path &quot;xxx/yyy&quot;</td>
<td>remove a bucket named &quot;xxx/yyy&quot;</td>
</tr>
<tr>
<td>read all directory entries under the directory with the full path &quot;/xxx/yyy&quot;</td>
<td>list all objects under the bucket named &quot;xxx/yyy&quot; and the buckets whose names are prefixed with &quot;xxx/yyy/&quot;</td>
</tr>
<tr>
<td>create a file named &quot;zzz&quot; in a directory with the full path &quot;xxx /yyy&quot;</td>
<td>create an object named &quot;zzz&quot; under the bucket named &quot;xxx/yyy&quot;</td>
</tr>
<tr>
<td>remove a file named &quot;zzz&quot; in a directory with the full path &quot;xxx /yyy&quot;</td>
<td>remove an object named &quot;zzz&quot; under the bucket named &quot;xxx/yyy&quot;</td>
</tr>
</tbody>
</table>

**Metadata Operations Based On File Storage Backend**

Unlike distributed object storage systems, distributed file systems already offer operational support for file system semantics. Therefore, OVFS based on a distributed file system doesn't require additional processing of file system requests and can achieve file system semantics simply by forwarding requests.

**Limitations Under OVFS Metadata Management**

While OVFS strives to implement a unified file system access interface for various storage system backends, users still need to be aware of its limitations and potential differences. OVFS supports a range of file system interfaces, but this doesn't imply POSIX standard compliance. OVFS cannot support some file system calls specified in the POSIX standard.

**2.5 Multi Granular Object Size Cache Pool**

In order to improve data read and write performance and avoid the significant overhead caused by repeated transmission of hot data between the storage system and the host, OVFS needs to build a data cache in the memory on the host side.

**Cache Pool Based On Multi Linked List**

OVFS will create a memory pool to cache file data during the file system read and write process. This huge memory pool is divided into object sizes of different granularities (such as 4 kb, 16 kb, 64 kb, etc.) to adapt to different sizes of data file data blocks.
Unused cache blocks of the same size in the memory pool are organized through a linked list. When a cache block needs to be allocated, the unused cache block can be obtained directly from the head of the linked list. When a cache block that is no longer used needs to be recycled, the cache block is added to the tail of the linked list. By using linked lists, not only can the algorithmic complexity of allocation and recycling be O(1), but furthermore, lock-free concurrency can be achieved by using CAS operations.

Write Back Strategy

OVFSD manages the data reading and writing process through the write back strategy. Specifically, when writing data, the data is first written to the cache, and the dirty data will be gradually synchronized to the backend storage system in an asynchronous manner. When reading the file data, the data will be requested from the backend storage system after a cache miss or expiration, and the new data will be updated to the cache, and its expiration time will be set.

OVFSD will update the dirty data in the cache to the storage system in these cases:

1) When VMs called ffsync, fdatasync, or used related flags during data writing.

2) The cache pool is full, and dirty data needs to be written to make space in the cache. This is also known as cache eviction, and the eviction order can be maintained using the LRU algorithm.

3) Cleaned by threads that regularly clean dirty data or expired data.

DAX Window Support (Experimental)

The VirtioFS protocol extends the DAX window experimental features based on the FUSE protocol. This feature allows memory mapping of file contents to be supported in virtualization scenarios. The mapping is set up by issuing a FUSE request to OVFSD, which then communicates with QEMU to establish the VMs memory map. VMs can delete mapping in a similar manner. The size of the DAX window can be configured based on available VM address space and memory mapping requirements.

By using the mmap and memfd mechanisms, OVFSD can use the data in the cache to create an anonymous memory mapping area and share this memory mapping with VMs to implement the DAX Window. The best performance is achieved when the file contents are fully mapped, eliminating the need for file I/O communication with OVFSD. It is possible to use a small DAX window, but this incurs more memory map setup/removal overhead.

2.6 Flexible Configuration Support

Running QEMU With OVFSD

As described in the architecture, deploying OVFS involves three parts: a guest kernel with VirtioFS support, QEMU with VirtioFS support, and the VirtioFS daemon (OVFSD). Here is an example of running QEMU with OVFSD:

```
host# ovfsd --config-file=./config.toml

_host# qemu-system __
__ ___ _-blockdev file,node-name=hdd,filename=<image file> _
__ ___ _device virtio-blk,drive=hdd_  
__ ___ _chardev socket,id=char0,path=/tmp/vfsd.sock _
__ ___ _device vhost-user-fs-pci,queue-size=1024,chardev=char0,tag=<fs tag> _
__ ___ _object memory-backend-memfd,id=mem,size=4G,share=on _
```
The configurations above will generate two devices for the VMs in QEMU. The block device named hdd serves as the backend for the virtio-blk device within the VMs. It functions to store the VMs' disk image files and acts as the primary device within the VMs. Another character device named char0 is implemented as the backend for the vhost-user-fs-pci device using the VirtioFS protocol in the VMs. This character device is of socket type and is connected to the file system daemon in OVFS using the socket path to forward file system messages and requests to OVFSD.

It is worth noting that the configuration method largely refers to the configuration in virtiofsd, and ignores many VMs configurations related file system access permissions or boundary handling methods.

Enable Different Distributed Storage Systems

In order for OVFS to utilize the extensive service support provided by OpenDAL, the corresponding service configuration file needs to be provided when running OVFSD. The parameters in the configuration file are used to support access to the storage system, including data root address and permission authentication. Below is an example of a configuration file, using a toml format similar to oli (a command line tool based on OpenDAL):

```toml
[ovfsd_settings]
socket_path = "/tmp/vfsd.sock"
enabled_services = "s3,hdfs"
enabled_cache = true
enabled_cache_write_back = false
enabled_cache_expiration = true
cache_expiration_time = "60s"

[profiles.s3]
type = "s3"
mount_point = "s3_fs"
bucket = "<bucket>"
endpoint = "https://s3.amazonaws.com"
access_key_id = "<access_key_id>"
secret_access_key = "<secret_access_key>"

[profiles.swift]
type = "swift"
mount_point = "swift_fs"
endpoint = "https://openstack-controller.example.com:8080/v1/account"
container = "container"
token = "access_token"

[profiles.hdfs]
```

OVFS can achieve hot reloading by monitoring changes in the configuration file. This approach allows OVFS to avoid restarting the entire service when modifying certain storage system access configurations and mounting conditions, thus preventing the blocking of correct request processing for all file systems in the virtual machine.

2.7 Expected POSIX Interface Support

Finally, the table below lists the expected POSIX system call support to be provided by OVFS, along with the corresponding types of distributed storage systems used by OpenDAL.

<table>
<thead>
<tr>
<th>System Call</th>
<th>Object Storage</th>
<th>File Storage</th>
<th>Key-Value Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>getattr</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>mknod/unlink</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>mkdir/mkdir</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>open/release</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>read/write</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>truncate</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>opendir/releasedir</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>readdir</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>rename</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>flush/fsync</td>
<td>Support</td>
<td>Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>getxattr/setxattr</td>
<td>Not Support</td>
<td>Not Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>chmod/chown</td>
<td>Not Support</td>
<td>Not Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>access</td>
<td>Not Support</td>
<td>Not Support</td>
<td>Not Support</td>
</tr>
</tbody>
</table>

Since the data volume of an individual file may be substantial, contradicting the design of key-value storage, we do not intend to include support for key-value storage in this project. The complex permission system control of Linux is not within the scope of this project. Users can restrict file system access behavior based on the configuration of storage system access permissions in the OVFS configuration file.

2.8 Potential Usage Scenarios

In this section, we list some potential OVFS usage scenarios and application areas through the detailed description of the OVFS project in the proposal. It's worth mentioning that as the project progresses, more application scenarios and areas of advantage may expand, leading to a deeper understanding of the positioning of the OVFS project.

1) Unified data management basic software within distributed clusters.

2) The OVFS project could prove highly beneficial for large-scale data analysis applications and machine learning training projects. It offers a means for applications within VM clusters to read and write data, models, checkpoints, and logs through common file system interfaces across various distributed storage systems.

3 Deliverables

This chapter describes the items that the OVFS project needs to deliver during the implementation cycle of GSoC 2024.
1) A code repository that implements the functions described in the project details. The services implemented by OVFS in the code repository need to meet the following requirements: (1) VirtioFS implementation, well integrated with VMs and QEMU, able to correctly handle VMs read and write requests to the file system. (2) Supports the use of distributed object storage systems and distributed file systems as storage backends, and provides complete and correct support for at least one specific storage service type for each storage system type. S3 can be used as the target for object storage systems, and HDFS can be used as the target for distributed file systems. (3) Supports related configurations of various storage systems. Users can configure storage system access and use according to actual needs. When an error occurs, users can use the configuration file to restart services.

2) Form an OVFS related test suite. Testing about the project should consist of two parts: (1) Unit testing in code components. Unit testing is the guarantee that the code and related functions are implemented correctly. This test implementation accompanies the entire code implementation process. (2) CI testing based on github actions. The OpenDAL project integrates a large number of CI tests to ensure the correct behavior of OpenDAL under various storage backends. OVFS needs to use good CI testing to check potential errors during code submission.

3) A performance test report of OVFS. The report needs to perform basic metadata operations, data reading and writing performance tests on the VMs mounted with OVFS, and summarize the performance of OVFS through the test results. Reports can be based on file system performance testing tools such as fio, sysbench and mtdtest, and compared with virtiofsd when necessary.

4) Documentation on the introduction and use of OVFS, and promote the inclusion of OVFS documentation into the official OpenDAL documentation when the GSoC project is completed.

Mentor

Mentor: Xuanwo, Apache Apache PMC Member Chair, xuanwo@apache.org
Mailing List: dev@opendal.apache.org

Major Difficulty: ~350 hour (large)
Potential mentors:
Hao Ding, mail: xuanwo (at) apache.org
Project Devs, mail: dev (at) opendal.apache.org

Apache OpenDAL oftp, OpenDAL FTP Server

Proposal for GSoC

- Organization: Apache
- Project: OpenDAL oftp
- Contributor: George Miao

About me

I'm George Miao, an undergraduate student at Syracuse University with over 6 years of experience in software development. I have a strong background in Rust programming language and web development.

>Description>

OpenDAL is a data access layer that allows users to easily and efficiently retrieve data from various storage services in a unified way. oftp can expose OpenDAL power in FTP way that allows users to access storage services via FTP protocol.

>Results for the Apache community
The OpenDAL project will benefit from the addition of oftp, as it will allow users to access storage services via FTP protocol.

-{Deliverables}-

- The ftp server itself
- Unit test for each ftp command and some integration tests
- User guide and configuration guide

-{Scheduling}-

- Week 1-2: Implement the core FTP server
- Week 3-4: Test core functionality, fix bugs
- Week 5-6: Implement auxiliary features (configuration, command line, etc.)
- Week 7-8: Test auxiliary features, fix bugs
- Week 9-10: Write documentation and user guide
- Rest of the time: Tracking, buffer for bugfixes, more features, etc.

Mentor

Mentor: PsiACE, Apache Apache Member, psiace@apache.org
Mailing List: dev@opendal.apache.org

Please leave comments if you want to be a mentor

**Mentor**

Mentor: PsiACE, Apache Apache Member, psiace@apache.org
Mailing List: dev@opendal.apache.org

Apache OpenDAL Ofs via CloudFilter Project

Apache OpenDAL Ofs via CloudFilter Project Proposal

**Contributor information**

- Name: Feihan Huang
- Email: ho229v3666@gmail.com
- GitHub: https://github.com/ho-229
- Location: Tianjin, China (GMT+8:00)

**Project information**

- Name: OpenDAL Ofs via CloudFilter
- Related Issues: https://github.com/apache/opendal/issues/4130
- Project Mentors: Xuanwo xuanwo@apache.org, Jun Ouyang junouyang@apache.org
- Project Community: Apache OpenDAL
- Project Size: Medium, ~175 hours

**Project abstract**

OpenDAL is a data access layer that allows users to easily and efficiently retrieve data from various storage services in a unified way. Currently, ofs can expose OpenDAL power in a fuse way that allows users to mount storage services locally.

But fuse is only supported on Linux and some UNIX platforms which limits the usage scenarios of ofs. So we need to support other popular platforms, i.e., Windows, to extend its usage scenarios.

Windows has a number of options that allow user-mode applications to project hierarchical data from the backing data store into the file system, such as ProjFs and CloudFilter. Considering ofs are majorly used in cloud storage, we need to support CloudFilter as well.

**Goals**

**In scope**

- fetch data
- cancel fetch data
• validate data
• fetch placeholders (directory entries)
• cancel fetch placeholders
• open, close, delete, rename

Out scope
• dehydrate (deprecate local cache)

Timeline

Before April 30
• Familiarize CloudFilter and its APIs.
• Investigate existing CloudFilter Rust bindings, and get in touch with the bindings maintainer if needed.
• Determine specific goals to implement.

May 1 - May 26
• Work closely with CloudFilter Rust bindings, make sure it can be used in ofs.

May 27 - June 20
• Write the major implementation of ofs via CloudFilter.

June 21 - June 30
• Test the implementation of ofs via CloudFilter.
• Update docs of ofs
• Prepare midterm evaluation with mentor.

July 1 - July 19
• Complete tests and fix bugs.
• Prepare final evaluation with mentor.

Links
[1] ho229v3666@gmail.com: _ho229v3666@gmail.com_ blocked URL
[2] xuanwo@apache.org: _xuanwo@apache.org_ blocked URL
[3] junouyang@apache.org: _junouyang@apache.org_ blocked URL

Major Difficulty:
~350 hour (large)
Project size:
~350 hour (large)
Potential mentors:
Hao Ding, mail: xuanwo (at) apache.org
Project Devs, mail: dev (at) opendal.apache.org

EventMesh

Apache EventMesh Enhance the serverless ability for EventMesh

Apache EventMesh
Apache EventMesh is a new generation serverless event middleware for building distributed event-driven applications.

Website: https://eventmesh.apache.org
GitHub: https://github.com/apache/eventmesh
Upstream Issue: https://github.com/apache/eventmesh/issues/4765

Background
EventMesh currently has Eventing capabilities in the serverless field, but it should also improve and supplement the automatic expansion and contraction capabilities of eventmesh's own services and access services. This service is the coordinator responsible for automatically scaling services connected to EventMesh, supporting automatic scaling from 0 to n and scaling down from n to 0 based on event traffic or other user conditions.

Task
1. Discuss with the mentors what you need to do
2. Learn the details of the Apache EventMesh project
3. Implement the auto scaling service for eventmesh, which can support different auto scaling strategies by default, or naive and keda can be selected as plugin services for automatic expansion and contraction of the service.

**Recommended Skills**
1. Familiar with go and K8S
2. Familiar with Knative/KEDA

**Difficulty:** Major  
**Project size:** ~350 hour (large)

**Mentor**
Eason Chen, PMC of Apache EventMesh, [https://github.com/qqeasonchen](https://github.com/qqeasonchen), chenguangsheng@apache.org
Mike Xue, PMC of Apache EventMesh, [https://github.com/xwm1992](https://github.com/xwm1992), mikexue@apache.org

**Difficulty:** Major  
**Project size:** ~350 hour (large)

**Potential mentors:**  
Xue Weiming, mail: mikexue (at) apache.org
Project Devs, mail: dev (at) eventmesh.apache.org

---

**ShardingSphere**

**[GSOC] [ShardingSphere] Support for PostgreSQL Search Path Feature**

**Apache ShardingSphere**

Apache ShardingSphere is positioned as Database Plus and aims at building a standard layer and ecosystem above heterogeneous databases. It focuses on how to reuse existing databases and their respective upper layer, rather than creating a new database. The goal is to minimize or eliminate the challenges caused by underlying databases fragmentation.

**Page:** [https://shardingsphere.apache.org](https://shardingsphere.apache.org)

**Github:** [https://github.com/apache/shardingsphere](https://github.com/apache/shardingsphere)

**Discussion:** [https://github.com/apache/shardingsphere/discussions/30252](https://github.com/apache/shardingsphere/discussions/30252)

**Background**

Apache ShardingSphere is positioned as Database Plus and aims to build a standard layer and ecosystem on top of heterogeneous databases. Therefore, ShardingSphere can be used with storage nodes like PostgreSQL. When users use PostgreSQL, they may have set the priority of query schema in system variables in advance. For example, when executing the SQL statement: `SELECT * FROM order`, if no schema is specified, the logical execution on PostgreSQL is as follows: scan through schemas in search_path one by one until the order table is found. On ShardingSphere, the current logic is to always query the public schema. Therefore, it is planned to support the search_path functionality for PostgreSQL in ShardingSphere to enhance the user experience.

**Task**

1. Support configuring search_path in ShardingSphere.
   - Support setting the search_path using PostgreSQL's "SET search_path" syntax;
   - Store the user-configured search_path information in the metadata of ShardingSphereSchema;
   - Support querying the search_path using PostgreSQL's "SHOW search_path" syntax.

2. Support matching the corresponding schema through search_path configuration.
   - During the SQL Binder phase, for a given SQL statement such as "select * from t_order", identify the schema corresponding to the table "t_order" based on the search_path and record it.

3. Support propagating the matched schema to the storage node.
   - During the rewriting phase, add the schema information matched from the search_path to the actual SQL statement.
Relevant Skills

1. Proficiency in Java.
2. Familiarity with PostgreSQL databases.
3. Familiarity with ShardingSphere.

Deliverables

Related code + unit tests + E2E tests.

Mentor

Zhengqiang Duan, PMC Member of Apache ShardingSphere, duanzhengqiang@apache.org
Cheng Zhang, Committer of Apache ShardingSphere, chengzhang@apache.org

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Project size: ~350 hour (large)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential mentors</td>
<td>Cheng Zhang, mail: chengzhang (at) apache.org</td>
</tr>
<tr>
<td></td>
<td>Project Devs, mail: dev (at) shardingsphere.apache.org</td>
</tr>
</tbody>
</table>

[GSOC] [ShardingSphere] Scan and create issues for classes that have not implemented unit tests

Apache ShardingSphere

Apache ShardingSphere is positioned as a Database Plus, and aims at building a standard layer and ecosystem above heterogeneous databases. It focuses on how to reuse existing databases and their respective upper layer, rather than creating a new database. The goal is to minimize or eliminate the challenges caused by underlying databases fragmentation.

Page: https://shardingsphere.apache.org
Github: https://github.com/apache/shardingsphere
Discussion: https://github.com/apache/shardingsphere/discussions/30252

Background

The ShardingSphere community has been striving for high unit test coverage, so ShardingSphere community needs a scanning tool to identify all classes without implemented unit tests, then generate a report based on the results, and create corresponding issues in the ShardingSphere community.

Task

Here are the steps for the scanning tool:
1. Scan the ShardingSphere project for classes that have not implemented unit tests(exclude the enum, constant, entity, interface and so on) in GitHub action;
2. Generate report base on these scan result;
3. Create issues automatically.

Relevant Skills

1. Master Java language
2. Have a basic understanding of GitHub action

Mentor

Nianjun Sun, Committer of Apache ShardingSphere, sunnianjun@apache.org

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Project size: ~175 hour (medium)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential mentors</td>
<td>Nianjun Sun, mail: sunnianjun (at) apache.org</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Project size: ~350 hour (large)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential mentors</td>
<td>Nianjun Sun, mail: sunnianjun (at) apache.org</td>
</tr>
</tbody>
</table>
[GSOC] [ShardingSphere] Supports executing DistSQL via JDBC

Apache ShardingSphere

Apache ShardingSphere is positioned as a Database Plus, and aims at building a standard layer and ecosystem above heterogeneous databases. It focuses on how to reuse existing databases and their respective upper layer, rather than creating a new database. The goal is to minimize or eliminate the challenges caused by underlying databases fragmentation.

Page: https://shardingsphere.apache.org
Github: https://github.com/apache/shardingsphere
Discussion: https://github.com/apache/shardingsphere/discussions/30252

Background

DistSQL (Distributed SQL) is Apache ShardingSphere’s specific SQL, providing additional operation capabilities compared to standard SQL.

Currently DistSQL can only be executed in ShardingSphere-Proxy. We hope to enhance this capability so that users can also execute DistSQL through ShardingSphere-JDBC.

Task

1. Understand the current reason why ShardingSphere-JDBC cannot execute DistSQL, and design a refactoring plan.
2. After completing the refactoring, add necessary unit tests to ensure that basic functionality is available.
3. Update documentation.

Relevant Skills

1. Master Java language
2. Understand SQL and DistSQL

Mentor

Longtao Jiang, Committer of Apache ShardingSphere, jianglongtao@apache.org
Jiahao Chen, Committer of Apache ShardingSphere, chenjiahao@apache.org

Difficulty: Major
Project size: ~175 hour (medium)
Potential mentors:
  Longtao Jiang, mail: jianglongtao (at) apache.org
  Project Devs, mail: dev (at) shardingsphere.apache.org

RocketMQ

Optimizing Lock Mechanisms in Apache RocketMQ

Background
Apache RocketMQ is a cloud-native messaging and streaming platform, streamlining the process of creating event-driven applications. Over the years, with the iteration of RocketMQ, a significant amount of code has been written to leverage multicore processors, enhancing program efficiency through concurrency. Consequently, managing concurrent performance has become vitally important. Locks are essential for ensuring multiple execution threads synchronize safely when accessing shared resources. Although locks are indispensable in ensuring mutual exclusion in multicore systems, their use can also pose optimization challenges. As concurrent systems grow more complex internally, deploying effective lock management strategies is key to preserving performance.

The adoption of locks in the concurrent code of RocketMQ may have room for optimization. For instance, the current usage of locks, while critical for ensuring consistency and preventing race conditions, could potentially be refined to improve overall message throughput without significantly impacting performance. In practice, we have demonstrated that adjusting the lock strategy can impact the message-sending performance of RocketMQ. Merely altering the backoff strategy of SpinLock can result in a performance difference of 20% (or even more) between the best and worst cases. Therefore, we hope to delve deeper into exploring the potential for performance optimization in this area. The concept of an adaptive lock mechanism could be introduced to enhance these synchronization points.

An adaptive lock could dynamically adjust its behavior based on runtime conditions, such as lock contention levels and the number of threads competing for the same resource. This could lead to improved performance by minimizing the overhead associated with lock acquisition and release, especially in scenarios with high contention. By monitoring the system’s performance metrics in real time, an adaptive lock could switch between different locking strategies, such as spinning versus blocking or using a queue-based lock versus a contention-free mechanism.

To implement such a system, a lock profiling tool could be employed to analyze the lock's performance, provide insights into lock contention, and suggest the optimal lock configuration tailored to RocketMQ's specific workload patterns. This approach would ensure that the locking mechanism remains both efficient and responsive to the changing dynamics of the system, ultimately enhancing the performance of message passing while maintaining the necessary safety guarantees.

Relevant Skills

- **Java Concurrent Programming Skills:** Understanding how to write and optimize code that can run in parallel across multiple processor cores is a must. This includes knowledge of synchronization mechanisms, such as locks, semaphores, and barriers.

- **Familiarity with Locking Mechanisms:** A deep understanding of various locking strategies and their trade-offs. This includes mutexes, spinlocks, reader-writer locks, and potentially more advanced lock-free and wait-free algorithms.

- **Expertise in System Performance Analysis:** The ability to analyze system performance, identify bottlenecks, and interpret metrics such as lock contention, CPU utilization, and thread performance.

Tasks

- Examine the locking mechanism in RocketMQ and analyze any potential performance bottlenecks it may cause.
- Enhance the message sending and processing performance through flexible lock optimization strategies.
- Condense the research findings into a report or article, and submit our discoveries to academic journals or conferences.

Learning Material

- [RocketMQ HomePage](https://rocketmq.apache.org)
- Github: [https://github.com/apache/rocketmq](https://github.com/apache/rocketmq)

Mentor

Lei Ding, PMC Member of Apache RocketMQ, dinglei@apache.org
Rongtong Jin, PMC Member of Apache RocketMQ, jinrongtong@apache.org
Juntao Ji, Contributor of Apache RocketMQ, 3160102420@zju.edu.cn
Yinyou Gu, Contributor of Apache RocketMQ, guyinyou.gyy@alibaba-inc.com

Difficulty: Major
Project size: ~350 hour (large)
Potential mentors:
- Juntao Ji, mail: generousman (at) apache.org
- Project Devs, mail: dev (at) rocketmq.apache.org
RocketMQ Dashboard Supports RocketMQ 5.0 Architecture and Enhances Usability

Background

Apache RocketMQ is a cloud-native messaging and streaming platform, making it simple to build event-driven applications.

The state-of-the-art Dashboard of Apache RocketMQ provides excellent monitoring capabilities. It makes various graphs and statistics of events, performance, and system information of clients and applications evidently available to the user.

However, following the introduction of the RocketMQ 5.0 architecture, the RocketMQ Dashboard has not adapted well to RocketMQ 5.0. Issues such as the Dashboard's inability to create various types of topics for V5, lack of support for the Proxy component, and incorrect master-slave synchronization metrics have arisen. Therefore, we hope that you can adapt the RocketMQ Dashboard to the RocketMQ 5.0 architecture and enhance its usability in this project.

Relevant Skills

- Java development skills
- Familiarity with Spring Boot
- Front-end capabilities are preferred
- Having a good understanding of RocketMQ Anyway, the most important relevant skill is motivation and readiness to learn during the project!

Tasks

- Ability to start and try out RocketMQ Dashboard
- Add support for the RocketMQ 5.0 architecture to the RocketMQ Dashboard, including creating various 5.0 related resources, while ensuring compatibility with the 4.0 architecture
- Enhance the usability of the RocketMQ Dashboard, including error handling.

Learning Material

- RocketMQ HomePage (https://rocketmq.apache.org)
- Github: https://github.com/apache/rocketmq
- RocketMQ Dashboard Github: https://github.com/apache/rocketmq-dashboard

Mentor

Rongtong Jin, PMC Member of Apache RocketMQ, jinrongtong@apache.org
Lei Ding, PMC Member of Apache RocketMQ, dinglei@apache.org

Comdev GSOC

[GSoc][HugeGraph] Support Memory Management Module

Apache HugeGraph (incubating) is a fast-speed and highly-scalable graph database. Billions of vertices and edges can be easily stored into and queried from HugeGraph due to its excellent OLTP ability.

- Website: https://hugegraph.apache.org/
- GitHub: https://github.com/apache/incubator-hugegraph/
Description

When the JVM GC performs a large amount of garbage collection, the latency of the request is often high, and the response time becomes uncontrollable. To reduce request latency and response time jitter, the hugegraph-server graph query engine has already used off-heap memory in mostOLTP algorithms.

However, at present, hugegraph cannot control memory based on a single request Query, so a Query may exhaust the memory of the entire process and cause OOM, or even cause the service to be unable to respond to other requests. To solve this problem, we can implement a memory management module based on a single Query. Applicants will work with community developers to complete this task, and the specific implementation plan and division of labor/priority can be adjusted as needed.

Overall, it can be divided into 3 modules:

1. Memory management implementation module. Implement the life cycle management of memory objects, memory capacity restrictions and other functions, and provide the Allocator interface (including allocation, release interface). This is a relatively independent module.
2. Integrate the Allocator module into the HugeGraph context and provide a unified interface for memory transformation.
3. Transform the places where a large amount of memory is occupied, and adapt to use Allocator for object allocation and release.

Recommended Skills

1. Java/JVM Basics: Deep understanding of Java's memory model, including the management and operation of heap memory and off-heap memory.
2. Java NIO: Java NIO library provides an interface for operating off-heap memory, which needs to be mastered. (Familiarity with Netty or other memory management basic libraries is preferred)
3. Concurrent Programming: Since memory management involves multi-thread concurrent operations, it is necessary to have knowledge of concurrent programming and multi-thread safety.
4. Data Structures: Understand and apply appropriate data structures to manage memory, such as using queues, stacks, etc., to manage memory blocks.
5. Operating System: Understand the memory management mechanism of the operating system in order to better understand and optimize Java's off-heap memory management.

Task List

- Implement a unified memory pool, independently manage JVM off-heap memory, and adapt the memory allocation methods of various native collections, so that the memory mainly used by the algorithm comes from the unified memory pool, and it is returned to the memory pool after release.
- Each request corresponds to a unified memory pool, and the memory usage of a request can be controlled by counting the memory usage of a request.
- Complete related unit tests UT and basic documentation (better with the perf diff).

PS: More tech details could refer: hugegraph/wiki/MemoryManagement

- Difficulty: Hard
- Project size: ~350 hour (full-time/large)

Potential Mentor

- Jermy Li: jermy@apache.org (Apache HugeGraph PPMC)
- Imba Jin: jin@apache.org (Apache HugeGraph PPMC)
- Yan Zhang: vaughn@apache.org (Apache HugeGraph PPMC)

Difficulty: Major
Project size: ~350 hour (large)
Potential mentors:

Imba Jin, mail: jin (at) apache.org
Project Devs, mail:

[GSOC][HugeGraph] Support Graph Traversal Expansion API

Description

Subtitle: Building New Edge Types Based on Path Start and End Vertex

In big data processing scenarios, to better utilize the graph database HugeGraph, we propose a new graph processing/extension function. This function aims to perform path traversal based on specified rules, and then build a new edge based on the start and end points of this path.
For example, in a family tree, starting from each vertex, following the path father->person(male)->father to the third node, a new edge is established, and the type of edge is grandfather.

Requirement Standards

1. The starting point can be defined by rules.
1. Implement path traversal based on the specified path.
1. Implement the logic of building a new edge from the start to the end of this path. (API)
1. Complete relevant unit tests (UT).
1. Complete the writing of user documentation.

Technical Requirements

- Familiarity with graph databases/HugeGraph is preferred.
- Possess Java/Scala development capabilities & familiar with basic Linux usage.
- Understand data structures and algorithms, especially those related to graphs.
- Have good problem-solving abilities and teamwork skills.

PS: More details refer: hugegraph/wiki/Graph-Traversal-Expansion

Time Nodes

1. Start to use and understand HugeGraph preliminarily, and can complete the setting and debugging of the local environment.
1. Design the implementation plan for this function.
1. Develop functions and test cases, optimize performance.
1. Improve user documentation.

**Difficulty:** Easy~Medium

**Project size:** ~130 hour (part-time/medium)

Potential Mentor

- Imba Jin: jin@apache.org blocked URL (Apache HugeGraph PPMC)
- Coderzc: zhaocong@apache.org blocked URL (Apache HugeGraph PPMC)

**Difficulty:** Major

**Project size:** ~175 hour (medium)

**Potential mentors:**
- Imba Jin, mail: jin (at) apache.org
- Project Devs, mail:

SkyWalking

[GSOC] [SkyWalking] Add Overview page in BanyanDB UI

Background

SkyWalking BanyanDB is an observability database, aims to ingest, analyze and store Metrics, Tracing and Logging data.

The BanyanDB UI is a web interface provided BanyanDB server. It's developed with Vue3 and Vite3

Objectives
The UI should have a user-friendly Overview page. The Overview page must display a list of nodes running in a cluster. For each node in the list, the following information must be shown:

- Node ID or name
- Uptime
- CPU usage (percentage)
- Memory usage (percentage)
- Disk usage (percentage)
- Ports (gRPC and HTTP)

The web app must automatically refresh the node data at a configurable interval to show the most recent information.

**Recommended Skills**

1. Familiar with Vue and Vite
2. Have a basic understanding of RESTful
3. Have an experience of Apache SkyWalking

**Difficulty**: Major  
**Project size**: ~350 hour (large)  
**Potential mentors:**
- Hongtao Gao, mail: hanahmily (at) apache.org
- Project Devs, mail: dev (at) skywalking.apache.org

---

**[GSOC] [SkyWalking] Self-Observability of the query subsystem in BanyanDB**

**Background**

SkyWalking BanyanDB is an observability database, aims to ingest, analyze and store Metrics, Tracing and Logging data.

**Objectives**

1. Support EXPLAIN[1] for both measure query and stream query
2. Add self-observability including trace and metrics for query subsystem
3. Support EXPLAIN in the client SDK & CLI and add query plan visualization in the UI

[1]: EXPLAIN in MySQL

**Recommended Skills**

1. Familiar with Go
2. Have a basic understanding of database query engine
3. Have an experience of Apache SkyWalking or other APMs

**Mentor**

- Mentor: Jiajing Lu, Apache SkyWalking PMC, lujiaping@apache.org
- Mentor: Hongtao Gao, Apache SkyWalking PMC, Apache ShardingSphere PMC, hanahmily@apache.org
- Mailing List: dev@skywalking.apache.org

**Difficulty**: Major  
**Project size**: ~350 hour (large)  
**Potential mentors:**
- Jiajing Lu, mail: lujiaping (at) apache.org
- Project Devs, mail: dev (at) skywalking.apache.org

---

**Beam**

**[GSOC][Beam] Build out Beam Use Cases**

Apache Beam is a unified model for defining both batch and streaming data-parallel processing pipelines, as well as a set of language-specific SDKs for constructing pipelines and Runners for executing them on distributed processing backends. On top of providing lower level primitives, Beam has also introduced several higher level transforms used for machine learning and some general data processing use cases. This project focuses on identifying and implementing real world use cases that use these transforms.
Objectives:
1. Add real world use cases demonstrating Beam’s MLTransform for preprocessing data and generating embeddings
2. Add real world use cases demonstrating Beam’s Enrichment transform for enriching existing data with data from a slowly changing source.
3. (Stretch) Implement 1 or more additional “enrichment handlers” for interacting with currently unsupported sources

Useful links:
Apache Beam repo - https://github.com/apache/beam
MLTransform docs - https://beam.apache.org/documentation/transforms/python/elementwise/mltransform/
Enrichment docs (should be published soon) - https://github.com/apache/beam/pull/30187

Major Difficulty: ~350 hour (large)
Project size: Potential mentors:
Danny McCormick, mail: damccorm (at) apache.org
Project Devs, mail: dev (at) beam.apache.org

[GSOC][Beam] Add connectors to Beam ManagedIO

Apache Beam is a unified model for defining both batch and streaming data-parallel processing pipelines, as well as a set of language-specific SDKs for constructing pipelines and Runners for executing them on distributed processing backends. On top of providing lower level primitives, Beam has also introduced several higher level transforms used for machine learning and some general data processing use cases. One new transform that is being actively worked on is a unified ManagedIO transform which gives runners the ability to manage (upgrade, optimize, etc...) an IO (input source or output sink) without upgrading the whole pipeline. This project will be about adding one or more IO integrations to ManagedIO

Objectives:
1. Add a BigTable integration to ManagedIO
2. Add a Spanner integration to ManagedIO

Useful links:
Apache Beam repo - https://github.com/apache/beam
Docs on ManagedIO are relatively light since this is a new project, but here are some docs on existing IOs in Beam - https://beam.apache.org/documentation/io/connectors/

Major Difficulty: ~350 hour (large)
Project size: Potential mentors:
Danny McCormick, mail: damccorm (at) apache.org
Project Devs, mail: dev (at) beam.apache.org

[GSOC][Beam] Build out Beam Yaml features

Apache Beam is a unified model for defining both batch and streaming data-parallel processing pipelines, as well as a set of language-specific SDKs for constructing pipelines and Runners for executing them on distributed processing backends. Beam recently added support for launching jobs using Yaml on top of its other SDKs, this project would focus on adding more features and transforms to the Yaml SDK so that it can be the easiest way to define your data pipelines.

Objectives:
1. Add support for existing Beam transforms (IOs, Machine Learning transforms, and others) to the Yaml SDK
2. Add end to end pipeline use cases using the Yaml SDK
3. (stretch) Add Yaml SDK support to the Beam playground

Useful links:
Apache Beam repo - https://github.com/apache/beam
Yaml SDK code + docs - https://github.com/apache/beam/tree/master/sdks/python/apache_beam/yaml
Open issues for the Yaml SDK - https://github.com/apache/beam/issues?q=is%3Aopen+is%3Aissue+label%3Ayaml

Major Difficulty: ~350 hour (large)
Project size: Potential mentors:
Danny McCormick, mail: damccorm (at) apache.org
Project Devs, mail: dev (at) beam.apache.org

James Server

Implement RFC-8617 The Authenticated Received Chain (ARC) Protocol

What
The Authenticated Received Chain (ARC) protocol provides an authenticated "chain of custody" for a message, allowing each entity that handles the message to see what entities handled it before and what the message's authentication assessment was at each step in the handling.

IE secured and standard Received headers.

Example:

```
ARC-Seal: i=1; a=rsa-sha256; s=arcselector9901; d=microsoft.com; cv=none;

b=S4DQRvGLMQeGKq+iUkaq9DIPrecQA+tD+qrvDIxyuYoiqOtYWoleSyzajb6B7t9ceuFfCWWNmbez89vRt9bCc4KpcjEjzEzufoKoTo4He
vTzIn62DEqXnxK5n/sW3GEAhxAcX3w3R4oYeFCypKalcH3gjIMStBBKuMG2kX1jTk5vixrVqtcLx526AQ3XNGD2Ew1RMHyhjKDNhKtnjknJGLu
swLi5Y0hEmM4CDWvXsmbfhpBFQq1tFddi1+S4DQRvGLMQeGKq+iUkaq9DIPrecQA+tD+qrvDIxyuYoiqOtYWoleSyzajb6B7t9ceuFfCWWNmbez89v
Rt9bCc4KpcjEjzEzufoKoTo4He

ARC-Message-Signature: i=1; a=rsa-sha256; c=relaxed/relaxed; d=microsoft.com;

s=arcselector9901;

h=From:Date:Subject:Message-ID:Content-Type:MIME-Version:X-MS-Exchange-AntiSpam-MessageData-

ChunkCount:X-MS-Exchange-AntiSpam-Messagedata-0:X-MS-Exchange-AntiSpam-Messagedata-1;

bh=FrWLV42FszOMB/KTATCDQLYPHJhy7pWkvwAtd3ueFh8=

b=Ef

/prHbNynosGBK4s4Dxsdch6uPcErYd9R9h24lb9sH1BvycXb5yPjlswGrtnvqEO14+8MDExv41Yz1GHldjWhB5Pg6YHeKu+2k8zwy55at0
XXRgK1Rdedge2bFqtpB9RcvyV9kw plugged/c
內C1b5yTchzn3RyQ2IVCvZbE2tp0Bgvv4ytF9

/972AhZ2LvymgqZ2a17tRDrkVxkVj2hr7hm0B8f6aqI1Um820H35D2MbN50EmN9iclicV0kP0ld9m7szAazv5vybs/S52B4nd7ub7y5TKh/Enw

/tihr/E6HXX31HggOt8d9ypjT4vVjV8brv1AykBi6V5xPexKzea==

ARC-Authentication-Results: i=1; mx.microsoft.com 1; spf=pass

dmcaposte.fr; dmarc=pass action=none header.from=dmcaposte.fr;

dkim=pass header.d=dmcaposte.fr; arc=none
```

How

Implement a Mailet implementing ARC

Implement a Matcher validating ARC

Documentation (README)

If applicable, parsing ARC records shall be done as a separate maven module.

Definition of done

- Absence of ARC headers shall be nicely handled
- Failed ARC shall be rejected
- Able to send email to gmail (validates ARC)
- Passes the ARC test suite

GSOC notes

Presenting a 1 week POC on the topic (as a separate mailet) would greatly improve the submission.

How to write custom mailet / matcher: https://github.com/ValiMail/arc_test_suite

Apache james registered on https://arc-spec.org/?page_id=79

GSOC notes

Presenting a 1 week POC on the topic (as a separate mailet) would greatly improve the submission.

How to write custom mailet / matcher: https://github.com/ValiMail/arc_test_suite

Apache james registered on https://arc-spec.org/?page_id=79

Nutch

Overhaul the legacy Nutch plugin framework and replace it with PF4J
Motivation

Plugins provide a large part of the functionality of Nutch. Although the legacy plugin framework continues to offer lots of value i.e.,

1. [some aspects e.g. examples, are [fairly well documented](https://cwiki.apache.org/confluence/display/NUTCH/PluginCentral]
2. it is generally stable, and
3. offers reasonable test coverage (on a plugin-by-plugin basis)
4. … probably loads more positives which I am overlooking...

... there are also several aspects which could be improved

1. the [core framework is sparsely documented](https://cwiki.apache.org/confluence/display/NUTCH/WhichTechnicalConceptsAreBehindTheNutchPluginSystem], this extends to very important aspects like the plugin lifecycle, classloading, packaging, thread safety, and lots of other topics which are of intrinsic value to developers and maintainers.
2. the core framework is somewhat [sparsely tested](https://github.com/apache/nutch/blob/master/src/test/org/apache/nutch/plugin/TestPluginSystem.java), … currently 7 tests as of writing. Traditionally, developers have focused on providing unit tests on the plugin-level as opposed to the legacy plugin framework.
3. see’s very low maintenance/attention. It is my gut feeling (and I may be totally wrong here) but I think that not many people know much about the core legacy plugin framework.
4. writing plugins is clunky. This largely has to do with the legacy Ant + Ivy build and dependency management system, but that being said, it is clunky non-the-less.
5. generally speaking, any reduction of code in the Nutch codebase through careful selection and dependence of well maintained, well tested 3rd party libraries would be a good thing for the Nutch codebase.

This issue therefore proposes to overhaul the legacy Nutch plugin framework and replace it with Plugin Framework for Java (PF4J).

Task Breakdown

The following is a proposed breakdown of this overall initiative into Epics. These Epics should likely be decomposed further but that will be left down to the implementer(s).

1. **document the legacy Nutch plugin lifecycle**: taking inspiration from [PF4J’s plugin lifecycle documentation](https://pf4j.org/doc/plugin-lifecycle.html) provide both documentation and a diagram which clearly outline how the legacy plugin lifecycle works. Might also be a good idea to make a contribution to PF4J and provide them with a diagram to accompany their documentation [blocked URL]. Generally speaking just familiarize oneself with the legacy plugin framework and understand where the gaps are.

2. **study PF4J framework and perform feasibility study**: this will provide an opportunity to identify gaps between what the legacy plugin framework does (and what Nutch) needs Vs what PF4J provides. Touch base with the PF4J community, describe the intention to replace the legacy Nutch plugin framework with PF4J. Obtain guidance on how to proceed. Document this all in the Nutch wiki. Create mapping of [legacy Classes](https://github.com/apache/nutch/tree/master/src/java/org/apache/nutch/plugin) to [PF4J equivalents](https://github.com/pf4j/pf4j/tree/master/pf4j/src/main/java/org/pf4j).


4. **Restructure each plugin in the plugins directory**: https://github.com/apache/nutch/tree/master/src/plugin

5. **Update Nutch plugin documentation**

6. **Create/propose plugin utility toolings**: #4 in the motivation section states that developing plugins in clunky. A utility tool which streamlines the creation of new plugins would be ideal. For example, this could take the form of a [new bash script](https://github.com/apache/nutch/tree/master/src/bin) which prompts the developer for input and then generates the plugin skeleton. **This is a nice to have.**

Google Summer of Code Details

This initiative is being proposed as a GSoC 2024 project.

**Proposed Mentor:** lewismc

**Proposed Co-Mentor:**

**Difficulty:** Major

**Project size:** ~350 hour (large)

**Potential mentors:**

Lewis John McGibbney, mail: lewismc (at) apache.org

Openmeetings

**Add blur background filter options on video sharing - AI-ML**

OpenMeetings uses webRTC and HTML5 video to share audio video. Purely browser based.
One feature missing is the ability to blur your webcam's camera background.

There are multiple ways to achieve it, Google Meet seems to use: [https://www.tensorflow.org/](https://www.tensorflow.org/)

Tensorflow are AI/ML models, they provide precompiled models into JS, for detection of face/body it seems: [https://github.com/tensorflow/tfjs-models/tree/master/body-segmentation](https://github.com/tensorflow/tfjs-models/tree/master/body-segmentation) is the best model.

Since Chrome 14 there is also a Background Blur API (relying on operating system APIs): [https://developer.chrome.com/blog/background-blur](https://developer.chrome.com/blog/background-blur) - but that doesn't seem to be widely or reliably supported by operating systems yet.

The project would be about adding the background blur into a simple demo and then integrate into the OpenMeetings project. Additionally other types of backgrounds can be added.

Tensorflow TFJS is under the Apache 2.0 License (See LICENSE) and should be possible to redistribute with Apache OpenMeetings.

Other live demos and examples:

---

### UIMA

**Support Aggregate Engines in Apache UIMACPP**

UIMA is a framework for unstructured information management, built around the idea of heavy annotators interoperating using a common exchange format.

It has been in production use for about two decades.

The framework is mostly written in Java. It has a C++ counterpart that implements a subset of the framework.

The challenge for this GSOC is to work together with the mentor to implement the full framework.

More details on GitHub: [https://github.com/apache/uima-uimacpp/issues/6](https://github.com/apache/uima-uimacpp/issues/6)

**Benefits to the community**

This has been discussed as one of the main roadblocks in using the C++ version of the framework by its users: [https://lists.apache.org/thread/f1r3ghn2qohvzz27y26zg6j3olv8qq](https://lists.apache.org/thread/f1r3ghn2qohvzz27y26zg6j3olv8qq)

On a larger perspective, there is the question of why we need NLP frameworks in 2024. The field has moved to approaches where source text is consumed in a destructive tokenization process that generates subtoken indices over a fixed vocabulary. These are then fed as input to a deep /transformer neural network.

Now, when training said networks, particularly when building Large Language Models (LLMs), gargantuan amounts of texts are quickly tokenized and fed into the model being trained. Additional computational efforts at indexing time can help improve data quality, privacy and terms of use of the text. A high performant UIMA CPP can be the missing piece for quality input data to LLMs.

**Technical Skills**

Working on this problem requires intermediate knowledge of the C++ programming language.

A solution will most probably exercise this type of skills, which could be learned along the way parallel to the project (mentoring on these topics is not part of the project):

- Linux command-line and build systems
- XML parsing
- Docker (image creation, deployment, debugging)

**About the mentor**
Dr. Duboue has more than 25 years of experience in AI. He has a Ph.D. in Computer Science from Columbia University, and was a member of the IBM Watson team that beat the Jeopardy! Champions.

Aside from his consulting work, he has taught in three different countries and done joint research with more than fifty co-authors.

He has years of experience mentoring both students and employees.

**Difficulty:** Major  
**Project size:** ~350 hour (large)  
**Potential mentors:**  
- Pablo Duboue, mail: drdub (at) apache.org  
- Project Devs, mail: dev (at) uima.apache.org