GSoC 2021 Ideas list

Contents

- Syncope
  - Debezium integration for live sync
  - Allow export for individual items in XML
- Synapse
  - Open Telemetry based Tracing for Apache Synapse
  - Containerization of Integration Framework
- StreamPipes
  - Introduce event windowing to the StreamPipes core/sdk
  - More powerful real-time visualizations for StreamPipes
  - New Python Wrapper
- Spatial Information Systems
  - Create metadata, CRS and tabular data editors in JavaFX
  - Coordinate operation methods to implement
- Solr
  - Refactor test infra to work with a managed SolrClient; ditch TestHarness
- Pulsar
  - Integration with Apache Ranger
  - Throttle the ledger rollover for the broker
  - Support reset cursor by message index
  - Support publish and consume avro objects in pulsar-perf
  - Improve the message written count metrics for the topic
  - Improve the message backlogs for the topic
  - Expose the broker level message metadata to the client.
- OODT
  - Improve OPSUI React.js UI with advanced functionalities
- James Server
  - [GSOC-2021] Implement Thread support for JMAP
- Fineract Cloud Native
  - Machine Learning Scorecard for Credit Risk Assessment Phase 4
  - Create Open Banking Layer for Fineract CN Self-Service App
  - Functional Enhancements to Fineract CN Mobile
- SkyWalking
  - Apache SkyWalking: Python agent supports profiling
  - Apache SkyWalking: Python agent collects and reports PVM metrics to backend
- ShardingSphere
  - Apache ShardingSphere: Proofread the DDL/TCL SQL definitions for ShardingSphere Parser
  - Apache ShardingSphere: Add unit test for example
  - Apache ShardingSphere: Proofread the DML SQL definitions for ShardingSphere Parser
- IoTDB
  - Implement PISA index in Apache IoTDB
  - Apache IoTDB Integration Test
  - Apache IoTDB C# library
  - Apache IoTDB: Metadata (Schema) Storage Engine
  - Apache IoTDB: GUI workbench
  - Apache IoTDB: Complex Arithmetic Operations in SELECT Clauses
  - Apache IoTDB: integration with Chaos Mesh
  - Apache IoTDB: use netty as the memory buffer pool to reduce GC problem and take full use of memory
  - Integrating Apache IoTDB and Apache Superset
- TrafficControl
  - GSOC: Varnish Cache support in Apache Traffic Control
- DolphinScheduler
  - Apache DolphinScheduler-Parameter coverage
- CouchDB
  - GSOC: Apache CouchDB and Debezium integration
- CloudStack
  - CloudStack GSoC 2021 - Clone a Virtual Machine (with all the data disks)
  - CloudStack GSoC 2021 Ideas
  - Mapping existing configuration parameters to the APIs
  - Support Multiple SSH Keys for VMs
  - Add dynamism to Persistent Networks in CloudStack
  - CloudStack GSoC 2021 - UX improvements in the UI with Vue.js
  - CloudStack GSoC 2021 - CloudStack OAuth2 Plugin
- Clerezza
  - Implement a Secure Scuttlebutt Server
  - Library for Messaging App Storage
- Cassandra
  - Add Plugin Support for CQLSH
  - Prevent and fail-fast any attempts to incremental repair cdc/mv tables
  - Add ability to ttl snapshots
  - Add nodetool command to display or export the contents of a virtual table
  - Script to autogenerate cassandra.yaml
  - Allow table property defaults (e.g. compaction, compression) to be specified for a cluster/keystore
  - Global configuration parameter to reject repairs with anti-compaction
  - Expose application_name and application_version in virtual table system_views.clients
• Per-node overrides for table settings
• Add ability to disable schema changes, repairs, bootstraps, etc (during upgrades)

• Beam
  • Profile and improve performance for the local FnApiRunner

• Apache NuttX
  • NuttX Support for Rapid Control Applications Development with pysimCoder
  • NuttX NAND Flash Subsystem
  • Device Tree support for NuttX
  • Rust integration on NuttX
  • QA for NuttX with tests using QEMU, Renode and/or real hardware

• Apache Nemo
  • Hierarchical aggregation and fidelity control in geo-distributed datacenter streaming environments
  • Implement an accurate task execution simulator to predict distributed data processing execution
  • Dynamic Work Stealing on Nemo for handling skews

• Apache Hudi
  • Snowflake integration w/ Apache Hudi
  • Apache Airflow integration w/ Apache Hudi
  • Pandas/python) integration w/ Apache Hudi
  • [UMBRELLA] Survey indexing technique for better query performance
  • [UMBRELLA] Support Apache Calcite for writing/querying Hudi datasets
  • [UMBRELLA] Improve CLI features and usabilities
  • [UMBRELLA] Improve source ingestion support in DeltaStreamer
  • [UMBRELLA] Checkstyle, formatting, warnings, spotless
  • [UMBRELLA] Support Apache Beam for incremental tailing
  • [UMBRELLA] Support schema inference for unstructured data

• Apache Fineract
  • Fineract Credit Bureau Integration Phase 4
  • Live Fineract CN API Documentation (Swagger, etc.)
  • Create Open Banking Layer for Fineract 1.x Self-Service Apps
  • Create Open Banking Layer for Mobile Wallet Apps
  • Reference Open Banking Fintech App on Fineract
  • Improve Robustness of Milos X and Apache Fineract by Fixing Issues/Feature Requests in Backlog
  • Update Fineract 1.x Android Client SDK
  • Create Fineract CN mobile wallet app on top of wallet framework

• APISIX
  • APISIX supports obtaining etcd data information through plugin
  • APISIX support to fetch more useful information of client request
  • APISIX: Support apply certificates from Let's Encrypt or any other ACMEv2 service
  • APISIX: Support Nacos in a native way
  • APISIX: Enhanced verification for APISIX ingress controller
  • APISIX Dashboard: Enhancement plugin orchestration
  • APISIX: Support invoke an AWS Lambda function through plugin
  • Integrate Cert-Manager to APISIX Ingress Controller
  • Exposing more Prometheus Metrics for Apache APISIX Ingress Controller
  • APISIX: enhanced authentication for Dashboard
  • Record short videos about Apache APISIX
  • APISIX: improve the website
  • APISIX ingress controller integration with Knative Serving

---

**Syncope**

**Debezium integration for live sync**

Debezium provides a mean to transform changes from RDBMS, MongoDB and Cassandra into Kafka messages.

This tool can be leveraged to implement a listener-like approach to enable a "live-sync" mechanism from External Resources not requiring ConnId.

**Difficulty:** Major

**Potential mentors:**

Francesco Chicchiricò, mail: ilgrosso (at) apache.org

Project Devs, mail:

---

**Allow export for individual items in XML**

Provide a functionality in the admin console (and from REST) that allows one to view the XML configuration of an individual item. For example, an user would be interested in seeing the XML representation of the new configuration parameter, or it might be a user object, group, provisioning rules, etc.

**Difficulty:** Major

**Potential mentors:**

Andrea Patricelli, mail: andrea.patricelli (at) apache.org

Project Devs, mail:
Synapse

Open Telemetry based Tracing for Apache Synapse

Currently, Apache Synapse does not have sophisticated support for modern tracing standardized. Therefore this new feature is intended to implement OpenTelemetry based tracing implementation for apache synapse.

This feature will include request-response training and inbound/outbound tracing at the transport level and the orchestration layer. Further, this also needs a really good investigation on Opentelemetry specification[1] and the Apache synapse transport component [1].

Relevant Skills

1. JAVA language
2. Understanding about observability
3. Integration and Synapse configuration language.

[1] https://opentelemetry.io/

Difficulty: Major

Potential mentors:
Vanjikumaran Sivajothy, mail: vanjikumaran@gmail.com (at) apache.org
Project Devs, mail:

Containerization of Integration Framework

The world has been moved towards container friendly products, microservices, and cloud-native technologies. However, the integration problem solved by the ESB architecture still exists and is handled by different entities. Even though Synapse is an ESB, it has most of the qualities (ex: faster startup time, low resource consumption, etc.) that are important for containers and microservices integrations. So with a little effort, we could make Synapse a container friendly product that suits any architectural style for doing the integration. As a very first step, Making Synapse a container friendly framework is important.

Task

- Reduce memory footprint of Synapse
- Create docker, Kubernetes artifacts [https://issues.apache.org/jira/projects/SYNAPSE/issues/SYNAPSE-1111]
- Documentation
- Publish the containers in the docker-hub

Relevant Skills

- JAVA language
- Kubernetes
- Docker
- Integration and Synapse configuration language.

Possible Mentors
Vanjikumaran Sivajothy
Mohamad Anfar Mohamad Shafreen
Isuru Udana

Difficulty: Critical

Potential mentors:
Vanjikumaran Sivajothy, mail: vanjikumaran@gmail.com (at) apache.org
Project Devs, mail:

StreamPipes

Introduce event windowing to the StreamPipes core/sdk
Apache StreamPipes

Apache StreamPipes (incubating) is a self-service (Industrial) IoT toolbox to enable non-technical users to connect, analyze and explore IoT data streams. StreamPipes offers several modules including StreamPipes Connect to easily connect data from industrial IoT sources, the Pipeline Editor to quickly create processing pipelines and several visualization modules for live and historic data exploration. Under the hood, StreamPipes utilizes an event-driven microservice paradigm of standalone, so-called analytics microservices making the system easy to extend for individual needs.

Background

Currently, window logic can be individually defined per pipeline element. The whole windowing logic needs to be declared in the controller and runtime logic needs to be individually added based on the selected runtime wrapper (Java, Siddhi, Flink, etc...).

As many data processors benefit from using window-functions (i.e PE such as Event Counter, Count Aggregation, Rate Limiter), windowing logic is often duplicated as it needs to be implemented for every new pipeline element. In addition, the feature set of supported window operators differs (and often depends on the developer) as it is unclear which windows and parameters should/can be offered.

Therefore, adding support for explicit window semantics to the SDK/Core would make implementing data processors and sinks using windows much easier and less error-prone.

Tasks

1. Design and introduce new processor and controller classes for windowed event processors (e.g., WindowedDataProcessor) which handle the windowing logic internally and only expose the higher-level methods to users (i.e onCurrentEvent, onExpiredEvent, etc...).
2. Implement internal logic for few window functions (i.e TimeWindow, LengthWindow, TimeBatchWindow, LengthBatchWindow, etc...)  
3. Write a few sample pipeline-elements using your new API!

Relevant Skills

- Basic knowledge in StreamPipes core (cloning the repo, going through the codebase/documents would do).
- Basic knowledge of stream analytics window functions (this is not a must, but it's awesome if you know your way around analytics window functions).
- Some Java experience.

Learning Material

For StreamPipes:

- https://streampipes.apache.org/docs/
- https://streampipes.apache.org/media.html
- https://github.com/apache/incubator-streampipes
- https://github.com/apache/incubator-streampipes-extensions

For Streaming Analytics:

- https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.5/developing-storm-applications/content/understanding_sliding_and_tumbling_windows.html

For the context for the issue:

- https://www.mail-archive.com/dev@streampipes.apache.org/msg00868.html

Mentor


Difficulty: Major

Potential mentors:

Grainier Perera, mail: grainier [at] apache.org
Project Devs, mail: [email]

More powerful real-time visualizations for StreamPipes

Apache StreamPipes

Apache StreamPipes (incubating) is a self-service (Industrial) IoT toolbox to enable non-technical users to connect, analyze and explore IoT data streams. StreamPipes offers several modules including StreamPipes Connect to easily connect data from industrial IoT sources, the Pipeline Editor to quickly create processing pipelines and several visualization modules for live and historic data exploration. Under the hood, StreamPipes utilizes an event-driven microservice paradigm of standalone, so-called analytics microservices making the system easy to extend for individual needs.
Background
Currently, the live dashboard (implemented in Angular) offers an initial set of simple visualizations, such as line charts, gauges, tables and single values. More advanced visualizations, especially those relevant for condition monitoring tasks (e.g., monitoring sensor measurements from industrial machines) is easy. Visualizations can be flexibly created by users and there is an SDK that allows to express requirements (e.g., based on data type or semantic type) for visualizations to better guide users through the creation process.

Tasks
1. Extend the set of real-time visualizations in StreamPipes, e.g., by integrating existing visualizations from Apache ECharts.
2. Improve the existing dashboard, e.g., by introducing better filtering or more advanced customization options.

Relevant Skills
0. Don’t be afraid! We’ll guide you through your first steps with StreamPipes.
1. Angular
2. Basic knowledge of Apache ECharts

Mentor
Dominik Riemer, PPMC Apache StreamPipes (riemer@apache.org)

Difficult: Major
Potential mentors:
Dominik Riemer, mail: riemer (at) apache.org
Project Devs, mail:

New Python Wrapper

Apache StreamPipes
Apache StreamPipes (incubating) is a self-service (Industrial) IoT toolbox to enable non-technical users to connect, analyze and explore IoT data streams. StreamPipes offers several modules including StreamPipes Connect to easily connect data from industrial IoT sources, the Pipeline Editor to quickly create processing pipelines and several visualization modules for live and historic data exploration. Under the hood, StreamPipes utilizes an event-driven microservice paradigm of standalone, so-called analytics microservices making the system easy to extend for individual needs.

Background
Current wrappers such as standalone (JVM, Siddhi) or distributed (Flink) already allow to develop new processors in the given runtime environment. The idea is to extend the list of standalone runtime wrappers to also support pure Python processors. We already got a minimal working version that however is pretty inflexible and still relies on Java as a proxy to the pipeline management in the backend service for the model declaration in the setup phase (capabilities, requirements, static properties of a processor) as well as the actual invocation in the execution phase (receiving specific configuration from pipeline management when pipeline is started). This issue is to track the status of the development.

Tasks
1. Add API endpoints as an interface for registration/invocation (partly done)
2. Port relevant model classes over to Python (declaration + invocation descriptions)
3. Implement support for various transport protocols and transport formats
4. Implement dev friendly alternative to Java builder pattern for model declaration
5. Implement overall runtime logic for Python wrapper

Relevant Skills
0. Don’t be afraid! We’ll guide you through your first steps with StreamPipes.
1. Excellent Python skills
2. Excellent understanding of stream processing paradigm incl. message broker such as Kafka, MQTT, etc
3. Good Understanding of RESTful web services (HTTP, etc.)
4. Basic Java skills to understand existing wrapper logic

Info
- SIP-02 to collect design decisions: https://cwiki.apache.org/confluence/display/STREAMPIPES/SIP-02+Python+wrapper
- Current python runtime wrapper implementation: https://github.com/apache/incubator-streampipes/tree/dev/streampipes-wrapper-python
- POC example: https://github.com/apache/incubator-streampipes-examples/tree/dev/streampipes-pipeline-elements-examples-processors-jvm
  /src/main/java/org/apache/streampipes/pe/examples/jvm/python

Mentor
Patrick Wiener, PPMC Apache StreamPipes (wiener@apache.org)

Difficulty: Major
Potential mentors:
- Patrick Wiener, mail: wiener (at) apache.org
- Project Devs, mail:

Spatial Information Systems

Create metadata, CRS and tabular data editors in JavaFX

Creates the foundation of a GUI application for Apache SIS based on JavaFX. This application should leverage the functionalities available in Apache SIS 0.8. In particular:

- Read metadata from files in various formats (currently ISO 19139, GeoTIFF, NetCDF, LANDSAT, GPX, Moving Features)
- Get Coordinate Reference System from a registry or from GML or WKT definitions and apply coordinate transformations.
- Show vector data in a tabular format.

Since SIS does not yet have a renderer engine, we can not yet show maps in the application. However the application should be designed with this goal in mind.

This project should create a metadata editor showing the ISO 19115 metadata. We should provide a simplified view with only the essential information, and an advanced view showing all information. The information to shown should be customizable. The user should be able to edit the metadata and save them in ISO 19139 format.

The project should also create the necessary widgets for showing a Coordinate Reference System (CRS) definition and allow the user to edit it. Another widget should use the CRS definitions for applying coordinate operations (map projections) using the existing Apache SIS referencing engine, and show the result in a table with information about accuracy and domain of validity.

Edit (March 2021): A JavaFX application has been created. It has widget for metadata and vector data, but we still need widget for Coordinate Reference System definitions. See SIS wiki for screenshots.

Difficulty: Major
Potential mentors:
- Martin Desruisseaux, mail: desruisseaux (at) apache.org
- Project Devs, mail:

Coordinate operation methods to implement

This is an umbrella task for some coordinate operation methods not yet supported in Apache SIS. Coordinate operations include map projections (e.g. Transverse Mercator, Lambert Conic Conformal, etc.), datum shifts (e.g. transformations from NAD27 to NAD83 in United States), transformation of vertical coordinates, etc. We can of course not list all possible formulas that we do not support, but this JIRA task lists at least some of the operations listed in the EPSG guidance notes.

The main material for this work is the EPSG guidance notes, which can be downloaded freely from the following site:

- IOGP Publication 373-7-2 – Geomatics Guidance Note number 7, part 2
  Coordinate Conversions and Transformations including Formulas
  http://www.epsg.org/GuidanceNotes

Google summer of code students interested in this work would need to be reasonably comfortable with the Java language (but not necessarily with the JDK library at large, since this work uses relatively few JDK classes outside Math), and in mathematic. In particular, this work requires a good understanding of affine transforms: their representation as a matrix, and how to map a term in a formula to a coefficient in the affine transform matrix.
Apache SIS has one advanced feature which is not easily found in popular geospatial software or text books: the capability to compute the derivative (or more precisely, the Jacobian) of a transformation at a given point. Implementation of this feature requires the capability to find the analytic derivative of a non-linear formula and to simplify it.

Implementations of those formulas take place in one of the org.apache.sis.referencing.operation sub-packages (projection or transform). Implementations of JUnit test happen partially in Apache SIS, and partially in the "conformance module" of the GeoAPI project, if possible through the Geospatial Integrity of Geoscience Software (GIGS) tests.

**Difficulty:** Major  
**Potential mentors:**  
- Martin Desruisseaux, mail: desruisseaux (at) apache.org  
- Project Devs, mail:

## Solr

### Refactor test infra to work with a managed SolrClient; ditch TestHarness

This is a proposal to substantially refactor SolrTestCaseJ4 and some of its intermediate subclasses in the hierarchy. In essence, I envision that tests should work with a SolrClient typed "solrClient" field managed by the test infrastructure. With only a few lines of code, a test should be able to pick between an instance based on EmbeddedSolrServer (lighter tests), HttpSolrClient (tests HTTP/Jetty behavior directly or indirectly), SolrCloud, and perhaps a special one for our distributed search tests. STCJ4 would refactor its methods to use the solrClient field instead of TestHarness. TestHarness would disappear as-such; bits of its existing code would migrate elsewhere, such as to manage an EmbeddedSolrServer for testing.

I think we can do a transition like this in stages and furthermore minimally affecting most tests by adding some deprecated shims. Perhaps STCJ4 should become the deprecated shim so that users can still use it during 7.x and to help us with the transition internally too. More specifically, we'd add a new superclass to STCJ4 that is the future – "SolrTestCase".

Additionally, there are a bunch of methods on SolrTestCaseJ4 that I question the design of, especially ones that return XML strings like delI (generates a delete-by-id XML string) and adoc. Perhaps that used to be a fine idea before there was a convenient SolrClient API but we've got one now and a test shouldn't be building XML unless it's trying to test exactly that.

For consulting work I once developed a JUnit4 TestRule managing a SolrClient that is declared in a test with an annotation of @ClassRule. I had a variation for SolrCloud and EmbeddedSolrServer that was easy for a test to choose. Since TestRule is an interface, I was able to make a special delegating SolrClient subclass that implements TestRule. This isn't essential but makes use of it easier since otherwise you'd be forced to call something like getSolrClient(). We could go the TestRule route here, which I prefer (with or without having it subclass SolrClient), or we could alternatively do TestCase subclassing to manage the lifecycle.

Initially I'm just looking for agreement and refinement of the approach. After that, sub-tasks ought to be added. I won't have time to work on this for some time.

**Difficulty:** Major  
**Potential mentors:**  
- David Smiley, mail: dsmiley (at) apache.org  
- Project Devs, mail:

## Pulsar

### Integration with Apache Ranger

Currently, Pulsar only supports store authorization policies under local-zookeeper. Is it possible to support (ranger) ([https://github.com/apache/ranger](https://github.com/apache/ranger)), apache ranger can provide a framework for central administration of security policies and monitoring of user access.

**Difficulty:** Major  
**Potential mentors:**  
- Penghui Li, mail: penghui (at) apache.org  
- Project Devs, mail:

### Throttle the ledger rollover for the broker

In Pulsar, the ledger rollover is split the data of a topic into multiple segments. For each ledger rollover operation, the metadata of the topic needs to be updated in the ZookKeeper. High ledger rollover frequency may lead to the ZookKeeper cluster in heavy workload. In order to make the ZookKeeper run more stable, we should limit the ledger rollover rate.

**Difficulty:** Major  
**Potential mentors:**  
- Penghui Li, mail: penghui (at) apache.org
Support reset cursor by message index

Currently, Pulsar supports resetting the cursor according to time and message-id, e.g., you can reset the cursor to 3 hours ago or reset the cursor to a specific message-id. For some cases that users want to reset to the 10,000 earlier messages, Pulsar has not supported this operation yet.

**PIP-70** [https://github.com/apache/pulsar/wiki/PIP-70%3A-Introduce-lightweight-raw-Message-metadata](https://github.com/apache/pulsar/wiki/PIP-70%3A-Introduce-lightweight-raw-Message-metadata) Introduced a broker level entry metadata which can support message index for a topic (or message offset of a topic), this will provide the ability to support reset cursor according to the message index.

**Difficulty:** Major  
**Potential mentors:**  
- Penghui Li, mail: penghui (at) apache.org  
- Project Devs, mail:

Support publish and consume avro objects in pulsar-perf

We should use perf tool to benchmark producing and consuming messages using Schema.

**Difficulty:** Major  
**Potential mentors:**  
- Penghui Li, mail: penghui (at) apache.org  
- Project Devs, mail:

Improve the message written count metrics for the topic

Currently, Pulsar exposes the message written count metrics though the Prometheus endpoint, and the metrics maintain in the broker, no been persistent. So if the topic ownership changes or restart broker, this will lead to reset the message written count of the topic to 0. This will confused users and not able to get the correct message written count metrics.

**PIP-70** [https://github.com/apache/pulsar/wiki/PIP-70%3A-Introduce-lightweight-broker-entry-metadata](https://github.com/apache/pulsar/wiki/PIP-70%3A-Introduce-lightweight-broker-entry-metadata) Introduced a broker level entry metadata which can support message index for a topic (or message offset of a topic), this will provide the ability to calculate the precise message written count for a topic. So we can leverage PIP-70 to improve the message written count metrics for the topic.

**Difficulty:** Major  
**Potential mentors:**  
- Penghui Li, mail: penghui (at) apache.org  
- Project Devs, mail:

Improve the message backlogs for the topic

In Pulsar, the client usually sends several messages with a batch. From the broker side, the broker receives a batch and write the batch message to the storage layer.

The message backlog is maintaining how many messages should be handled for a subscription. But unfortunately, the current backlog is based on the batches, not the messages. This will confuse users that they have pushed 1000 messages to the topic, but from the subscription side, when to check the backlog, will return a value that lower than 1000 messages such as 100 batches. Not able to get the message based backlog is it's so expensive to calculate the number of messages in each batch.

**PIP-70** [https://github.com/apache/pulsar/wiki/PIP-70%3A-Introduce-lightweight-raw-Message-metadata](https://github.com/apache/pulsar/wiki/PIP-70%3A-Introduce-lightweight-raw-Message-metadata) Introduced a broker level entry metadata which can support message index for a topic (or message offset of a topic). This will provide the ability to calculate the number of messages between a message index to another message index. So we can leverage PIP-70 to improve the message backlog implementation to able to get the message-based backlog.

For the Exclusive subscription or Failover subscription, it easy to implement by calculating the messages between the mark delete position and the LAC position. But for the Shared and Key_Shared subscription, the individual acknowledgment will bring some complexity. We can cache the individual acknowledgment count in the broker memory, so the way to calculate the message backlog for the Shared and Key_Shared subscription is “backlogOfTheMarkdeletePosition” - “IndividualAckCount”

**Difficulty:** Major  
**Potential mentors:**  
- Penghui Li, mail: penghui (at) apache.org  
- Project Devs, mail:
Expose the broker level message metadata to the client.


But currently, the client can't get the broker level message metadata since the broker skip this information when dispatching messages to the client. Provide a way to expose the broker level message metadata to the client.

**Difficulty:** Major  
**Potential mentors:** Penghui Li, mail: penghui (at) apache.org  
Project Devs, mail:

**OODT**

**Improve OPSUI React.js UI with advanced functionalities**

In GSoC 2019, we implemented a new OPSUI UI based on React.js. See the related blog posts [1] [2]. Several advanced features require to be implemented including:

- Implement querying functionality at OPSUI side (scope can be determined)
- Show progress of workflows and file ingestions
- Introduce a proper REST API for resource manager component
- Introduce proper packaging (with configurable external REST API URLs) and deployment mechanism (as a docker deployment or an npm package)

In this project, the student will have to work on the UI with React.js and will have to implement several REST APIs using JAX-RS. Furthermore, will have to work on making OPSUI easy to deploy.

The existing wicket based OPSUI will be replaced by the new React.js based OPSUI at the end of this project. And the linked blog posts will be a good start to understand what the new React.js based OPSUI is capable of doing.


**Difficulty:** Major  
**Potential mentors:** Imesha Sudasingha, mail: imesha (at) apache.org  
Project Devs, mail:

**James Server**

**[GSOC-2021] Implement Thread support for JMAP**

**Why?**

Mail user agents generally allow displaying emails grouped by conversations (replies, forward, etc...).

As part of JMAP RFC-8621 implementation, there is a dedicated concepts: threads. We did implement JMAP Threads in a rather naive way: each email is a thread of its own.

This naive implementation is specification compliant but defeat the overall purposes of threads.

I propose myself to mentor the implementation of Threads as part of the James JMAP implementation.

See: [https://jmap.io/spec-mail.html#threads](https://jmap.io/spec-mail.html#threads)

**Difficulty:** Major  
**Potential mentors:** Benoit Tellier, mail: btellier (at) apache.org  
Project Devs, mail:

**Fineract Cloud Native**
Machine Learning Scorecard for Credit Risk Assessment Phase 4

Mentors
- Lalit Mohan S
- VICTOR ROMERO

Overview & Objectives
Financial Organizations using Mifos/Fineract are depending on external agencies or their past experiences for evaluating credit scoring and identification of potential NPAs. Though information from external agencies is required, financial organizations can have an internal scorecard for evaluating loans so that preventive/proactive actions can be done along with external agencies reports. In industry, organizations are using rule based, Statistical and Machine learning methods for credit scoring, predicting potential NPAs, fraud detection and other activities. This project aims to implement a scorecard based on statistical and ML methods for credit scoring and identification of potential NPAs.

Description
The approach should factor and improve last year’s GSOC work (https://gist.github.com/SupreethSudhakaranMenon/a20251271adb341f949dbfeb035191f7) on Features/Characteristics, Criteria and evaluation (link). The design and implementation of the screens should follow Mifos Application standards. Should implement statistical and ML methods with explainability on decision making. Should also be extensible for adding other functionalities such as fraud detection, cross-sell and up-sell, etc.

Helpful Skills
JAVA, Integrating Backend Service, MIFOS X, Apache Fineract, AngularJS, ORM, ML, Statistical Methods, Django

Impact
Streamlined Operations, Better RISK Management, Automated Response Mechanism

Other Resources
2019 Progress: https://gist.github.com/SupreethSudhakaranMenon/a20251271adb341f949dbfeb035191f7
https://gist.github.com/lalitsanagavarapu

Difficulty: Major
Potential mentors:
- Ed Cable, mail: edcable (at) apache.org
- Project Devs, mail: dev (at) fineract.apache.org

Create Open Banking Layer for Fineract CN Self-Service App

Mentors
- Avik Ganguly
- Garvit Agarwal

Overview & Objectives
Across our ecosystem we’re seeing more and more adoption and innovation from fintechs. A huge democratizing force across the financial services sector is the Open Banking movement providing Open Banking APIs to enable third parties to directly interact with customers of financial institutions. We have recently started providing an Open Banking API layer that will allow financial institutions using Mifos and Fineract to offer third parties access to requesting account information and initiating payments via these APIs. Most recently the MojaLoop community, led by Google, has led the development of a centralized PISP API. We have chosen to follow the comprehensive UK Open Banking API standard which is being followed and adopted by a number of countries through Sub-Saharan Africa and Latin America.

Tremendous impact can be had at the Base of the Pyramid by enabling third parties to establish consent with customers to authorize transactions to be initiated or information to be accessed from accounts at their financial institution. This Open Banking API layer would enable any institution using Mifos or Fineract to provide a UK Open Banking API layer to third parties and fintechs.

The API Gateway to connect to is still being chosen (WS02, Gravitee, etc.)

Description
The APIs that are consumed by the reference Fineract 1.x mobile banking application have been documented in the spreadsheet below. The APIs have also been categorized according to whether they are an existing self-service API or back-office API and if they have an equivalent Open Banking API and if so, a link to the corresponding Open Banking API.

For each API with an equivalent Open Banking API, the interns must: Take rest api, upload swagger definition, do transformation in OpenBanking Adapter, and publish on API gateway.

For back-office and/or self-service APIs with no equivalent Open Banking API, the process is: Take rest api, upload swagger definition, and publish on API gateway.

For example:

- Submit Loan Application (Self-Service API with Equivalent Open Banking API)
  - https://demo.mifos.io/api-docs/apiLive.htm#loans_create
  - Used by Fineract 1.x Self-Service App

- Images API (Back-Office API with No Equivalent Open Banking API)
  - https://demo.mifos.io/api-docs/apiLive.htm#client_images
  - Used by Mifos Mobile and Mobile Wallet

- Fetch Identification Card API (Fineract CN API with no equivalent Open Banking API)
  - https://docs.google.com/document/d/15LbxVoQQRo4uU7QjV7FpJFVjkyyNb9_HJwFvS47O4I/edit?pli=1#heading=h.xfl6jxdpcpy1

Sample APIs to be Documented

-----------------------------
Mifos Mobile CN API Matrix (completed by Garvit)
https://docs.google.com/spreadsheets/d/1-HrfPKhh1kO7ojK15Ylf6uzejQmaz?2eXf5MzCBCE3M/edit?gid=0
Mobile Wallet API Matrix (completed by Devansh)
https://docs.google.com/spreadsheets/d/1VgpIwN2JsljWWyt_kWwh6xaIoRgMNIAv3g/edit?gid=0

Helpful Skills

Android development, SQL, Java, Javascript, Git, Spring, OpenJPA, Rest, Kotlin, Gravitee, WSO2

Impact

By providing a standard UK Open Banking API layer we can provide both a secure way for our trusted first party apps to allow customers to authenticate and access their accounts as well as an API layer for third party fintechs to securely access Fineract and request information or initiate transactions with the consent of customers.

Other Resources

CGAP Research on Open Banking: https://www.cgap.org/research/publication/open-banking-how-design-financial-inclusion
Docs: https://mifos.gitbook.io/docs/wso2-1/setup-openbanking-apis
Self-Service APIs: https://demo.mifos.io/api-docs/apiLive.htm#selfserviceth

- https://cwiki.apache.org/confluence/display/FINERACT/Customer+Self-Service+Phase+2
  - Open Banking Adapter: https://github.com/openMF/openbanking-adapter
  - Transforms Open Banking API to Fineract API
  - Works with both Fineract 1.x and Fineract CN
  - Can connect to different API gateways and can transform against different API standards.

Reference Open Banking Fintech App:

- Backend: https://github.com/openMF/openbanking-tpp-server
- GUI: https://github.com/openMF/openbanking-tpp-client

UK Open Banking API Standard: https://standards.openbanking.org.uk/

Open Banking Developer Zone: https://openbanking.atlassian.net/wiki/spaces/DZ/overview

Examples of Open Banking Apps: https://www.ft.com/content/a5f0af78-133e-11e9-a581-4ff78404524e

See https://openmf.github.io/mobileapps.github.io/

Difficulty: Major
Potential mentors:

Ed Cable, mail: edcable (at) apache.org
Project Devs, mail: dev (at) fineract.apache.org
Functional Enhancements to Fineract CN Mobile

Mentors

- Rajan Maurya
- Ahmad Jawid Muhammadi

Overview & Objectives

Just as we have a mobile field operations app on Apache Fineract 1.x, we have recently built out on top of the brand new Apache Fineract CN microservices architecture, an initial version of a mobile field operations app with an MVP architecture and material design. Given the flexibility of the new architecture and its ability to support different methodologies - MFIs, credit unions, cooperatives, savings groups, agent banking, etc - this mobile app will have different flavors and workflows and functionalities.

Description

In 2020, our Google Summer of Code intern worked on additional functionality in the Fineract CN mobile app. In 2021, the student will work on the following tasks:

- Integrate with Payment Hub to enable disbursement via Mobile Money API
- Improve Task management features into the app.
- Create UI for creating new account and displaying account details
- Create UI for creating tellers and displaying tellers details
- Improve GIS features like location tracking, dropping of pin into the app
- Improve offline mode via Couchbase support
- Write Unit Test, Integration Test and UI tests

Helpful Skills

Android Development, Kotlin, Java, Git, OpenJPA, Rest API

Impact

Allows staff to go directly into the field to connect to the client. Reduces cost of operations by enabling organizations to go paperless and be more efficient.

Other Resources

1. Repo on Github: https://github.com/apache/fineract-cn-mobile
2. Fineract CN API documentation: https://izakey.github.io/fineract-cn-api-docs-site/
4. https://cwiki.apache.org/confluence/display/FINERACT/Fineract+CN
5. GSMA mobile money API: https://developer.mobilemoneyapi.io/1.1/oas3/22466
7. Some UI designs: https://www.figma.com/file/KHX1ZPdlpC3Tqvd1VZu8CW/fineract-cn-mobile?node-id=0%3A1

1. 2020 GSoc progress report: https://gist.github.com/jawidMuhammadi/9fa91d37b1cbe43d9cdefe165ad8f2102

Difficulty: Major
Potential mentors: Ed Cable, mail: edcable (at) apache.org
Project Devis, mail: dev (at) fineract.apache.org

SkyWalking

Apache SkyWalking: Python agent supports profiling
Apache SkyWalking [1] is an application performance monitor (APM) tool for distributed systems, especially designed for microservices, cloud native and container-based (Docker, K8s, Mesos) architectures.

SkyWalking is based on agent to instrument (automatically) monitored services, for now, we have many agents for different languages, Python agent [2] is one of them, which supports automatic instrumentations.

The goal of this project is to extend the agent's features by supporting profiling [3] a function's invocation stack, help the users to analyze which method costs the most major time in a cross-services call.

To complete this task, you must be comfortable with Python, have some knowledge of tracing system, otherwise you'll have a hard time coming up to speed..


Difficulty: Major
Potential mentors:
  Zhenxu Ke, mail: kezhenxu94 (at) apache.org
  Project Devs, mail: dev (at) skywalking.apache.org

Apache SkyWalking: Python agent collects and reports PVM metrics to backend

Apache SkyWalking [1] is an application performance monitor (APM) tool for distributed systems, especially designed for microservices, cloud native and container-based (Docker, K8s, Mesos) architectures.

Tracing distributed systems is one of the main features of SkyWalking, with those traces, it can analyze some service metrics such as CPM, success rate, error rate, apdex, etc. SkyWalking also supports receiving metrics from the agent side directly.

In this task, we expect the Python agent to report its Python Virtual Machine (PVM) metrics, including (but not limited to, whatever metrics useful are also acceptable) CPU usage (%), memory used (MB), (active) thread/coroutine counts, garbage collection count, etc.

To complete this task, you must be comfortable with Python and gRPC, otherwise you'll have a hard time coming up to speed.

Live demo to play around: [http://122.112.182.72:8080](http://122.112.182.72:8080) (under reconstruction, maybe unavailable but latest demo address can be found at the GitHub index page [http://github.com/apache/skywalking](http://github.com/apache/skywalking))


Difficulty: Major
Potential mentors:
  Zhenxu Ke, mail: kezhenxu94 (at) apache.org
  Project Devs, mail: dev (at) skywalking.apache.org

ShardingSphere

Apache ShardingSphere: Proofread the DDL/TCL SQL definitions for ShardingSphere Parser

Apache ShardingSphere

Apache ShardingSphere is a distributed database middleware ecosystem, including 2 independent products, ShardingSphere JDBC and ShardingSphere Proxy presently. They all provide functions of data sharding, distributed transaction, and database orchestration.

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

Background

ShardingSphere parser engine helps users parse a SQL to get the AST (Abstract Syntax Tree) and visit this tree to get SQLStatement (Java Object). At present, this parser engine can handle SQLs for ‘MySQL’, ‘PostgreSQL’, ‘SQLServer’ and ‘Oracle’, which means we have to understand different database dialect SQLs.


Task

This issue is to proofread the following definitions,

- All the DDL SQL definitions for Oracle except for ALTER, DROP, CREATE and TRUNCATE.
- All the TCL (Transaction Control Language) SQL definitions for Oracle

You can learn more here.
As we have a basic Oracle SQL syntax definitions but do not keep in line with Oracle DOC, we need you to find out the vague SQL grammar definitions and correct them referring to Oracle DOC.

**Notice**, when you review these target SQLs above, you will find that these definitions will involve some basic elements of Oracle SQL. No doubt, these elements are included in this task as well.

**Relevant Skills**
1. Master JAVA language
2. Have a basic understanding of Antlr g4 file
3. Be familiar with Oracle SQLs

**Targets files**
1. DDL SQLs g4 file: https://github.com/apache/shardingsphere/blob/master/shardingsphere-sql-parser/shardingsphere-sql-parser-dialect/shardingsphere-sql-parser-oracle/src/main/antlr4/imports/oracle/DDLStatement.g4
2. TCL SQLs g4 file: https://github.com/apache/shardingsphere/blob/master/shardingsphere-sql-parser/shardingsphere-sql-parser-dialect/shardingsphere-sql-parser-oracle/src/main/antlr4/imports/oracle/TCLStatement.g4

**References**

**Mentor**
Juan Pan, PMC of Apache ShardingSphere, panjuan@apache.org

**Difficulty:** Major  
**Potential mentors:**  
Juan Pan, mail: panjuan (at) apache.org  
Project Devs, mail: dev (at) shardingsphere.apache.org

**Apache ShardingSphere: Add unit test for example**

**Apache ShardingSphere**
Apache ShardingSphere is a distributed database middleware ecosystem, including 2 independent products, ShardingSphere JDBC and ShardingSphere Proxy presently. They all provide functions of data sharding, distributed transaction, and database orchestration.  
Page: https://shardingsphere.apache.org  
Github: https://github.com/apache/shardingsphere

**Background**
The examples of ShardingSphere do not have test cases. After mvn install, developer can know compiling success only, but the can not guarantee code correct, especially config for YAML, spring namespace and spring boot starter.

**Task**
This issue is to add auto test cases with JUnit to assert startup success and code logic correct.

**Notice**, the code of current example may need to be refactor to make it easy for test.

**Relevant Skills**
1. Master JAVA language  
2. Be familiar with spring framework  
3. Have a basic understanding of JUnit

**Targets files**
Example repo: https://github.com/apache/shardingsphere/tree/master/examples

**Mentor**
Liang Zhang, PMC Chair of Apache ShardingSphere, zhangliang@apache.org
Apache ShardingSphere: Proofread the DML SQL definitions for ShardingSphere Parser

Apache ShardingSphere

Apache ShardingSphere is a distributed database middleware ecosystem, including 2 independent products, ShardingSphere JDBC and ShardingSphere Proxy presently. They all provide functions of data sharding, distributed transaction, and database orchestration.

Page: https://shardingsphere.apache.org
Github: https://github.com/apache/shardingsphere

Background

ShardingSphere parser engine helps users parse a SQL to get the AST (Abstract Syntax Tree) and visit this tree to get SQLStatement (Java Object). At present, this parser engine can handle SQLs for `MySQL`, `PostgreSQL`, `SQLServer` and `Oracle`, which means we have to understand different database dialect SQLs.


Task

This issue is to proofread the DML(SELECT/UPDATE/DELETE/INSERT) SQL definitions for Oracle. As we have a basic Oracle SQL syntax definitions but do not keep in line with Oracle DOC, we need you to find out the vague SQL grammar definitions and correct them referring to Oracle DOC.

Notice, when you review these DML(SELECT/UPDATE/DELETE/INSERT) SQLs, you will find that these definitions will involve some basic elements of Oracle SQL. No doubt, these elements are included in this task as well.

Relevant Skills

1. Master JAVA language
2. Have a basic understanding of Antlr g4 file
3. Be familiar with Oracle SQLs

Targets files

1. DML SQLs g4 file: https://github.com/apache/shardingsphere/blob/master/shardingsphere-sql-parser/shardingsphere-sql-parser-dialect/shardingsphere-sql-parser-oracle/src/main/antlr4/imports/oracle/DMLStatement.g4
2. Basic elements g4 file: https://github.com/apache/shardingsphere/blob/master/shardingsphere-sql-parser/shardingsphere-sql-parser-dialect/shardingsphere-sql-parser-oracle/src/main/antlr4/imports/oracle/BaseRule.g4

References


Mentor

Juan Pan, PMC of Apache ShardingSphere, panjuan@apache.org

Difficulty: Major
Potential mentors:
Juan Pan, mail: panjuan (at) apache.org
Project Devs, mail: dev (at) shardingsphere.apache.org

IoTDB

Implement PISA index in Apache IoTDB

Apache IoTDB is a highly efficient time series database, which supports high speed query process, including aggregation query.

Currently, IoTDB pre-calculates the aggregation info, or called the summary info, (sum, count, max_time, min_time, max_value, min_value) for each page and each Chunk. The info is helpful for aggregation operations and some query filters. For example, if the query filter is value >10 and the max value of a page is 9, we can skip the page. For another example, if the query is select max(value) and the max value of 3 chunks are 5, 10, 20, then the max(value) is 20.
However, there are two drawbacks:

1. The summary info actually reduces the data that needs to be scanned as \(1/k\) (suppose each page has \(k\) data points). However, the time complexity is still \(O(N)\). If we store a long historical data, e.g., storing 2 years data with 500KHz, then the aggregation operation may be still time-consuming. So, a tree-based index to reduce the time complexity from \(O(N)\) to \(O(\log N)\) is a good choice. Some basic ideas have been published in [1], while it can just handle data with fix frequency. So, improving it and implementing it into IoTDB is a good choice.

2. The summary info is helpless for evaluating the query like where value > 8 if the max value = 10. If we can enrich the summary info, e.g., storing the data histogram, we can use the histogram to evaluate how many points we can return.

This proposal is mainly for adding an index for speeding up the aggregation query. Besides, if we can let the summary info be more useful, it could be better.

Notice that the premise is that the insertion speed should not be slow down too much!

By the way, IoTDB provides an index framework already. So, the PISA index should be compatible with the index framework.

You should know:
- IoTDB query process
- TsFile structure and organization
- Basic index knowledge
- Java

mentors: Major
hxd@apache.org

Reference:

---

**Apache IoTDB Integration Test**

Apache IoTDB is an Open Source IoT database designed to meet the rigorous data, storage, and analytics requirements of large-scale Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications.

Now, IoTDB uses JUnit for its UT/IT test.

However, there are two drawbacks:

1. There are many singleton class instances in IoTDB. Therefore, modifying something in a test may impact others, and it requires us do many cleanup work after a test.

Especially, after we open an serversocket (by Thrift), though we have called the socket.close, the socket may be not closed quickly (controlled by Thrift). But, if the next test begins, then a "the port is already used" error will occur.

2. when testing IoTDB's cluster module, we may need to start at least 3 IoTDB instances in one server.

Using JUnit, the 3 instances will be in one JVM, which is conflicted with the reality "IoTDB has many singleton instances".

So, next, we want to use TestContainer, which is a combiner of Docker and JUnit.

This task is for:

1. using TestContainer to re-implement all IT codes of IoTDB;
2. using TestContainer to add some IT codes for IoTDB's cluster module.

Needed skills:
- Java
- Docker (Docker-Compose better)
- Know or learn Junit and TestContainer

[1] iotdb.apache.org

mentors: Major
Xiangdong Huang, mail: hxd (at) apache.org
Project Devs, mail: dev (at) iotdb.apache.org
Apache IoTDB C# library

Apache IoTDB [1] is an Open Source IoT database designed to meet the rigorous data, storage, and analytics requirements of large-scale Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications.

IoTDB has two kinds of client interfaces: SQL and native API (also called as session API.)

This task is for the native API.

IoTDB uses Apache Thrift[2] as its RPC framework, so all native API can be generated by Thrift. However, to accelerate the performance, we may use some byte array in Thrift, rather than a Struct, which is not quite friendly to users.

That is why we provide our session API. Session API just wraps the interfaces of the generated thrift codes. Now we have Java[4], Python and c++ version[3]. The C# version is left.

This task hopes you can provide a c# library for IoTDB.

Needed skills:

- Thrift
- C#
- know Java

[1] iotdb.apache.org

Apache IoTDB: Metadata (Schema) Storage Engine

Apache IoTDB [1] is an Open Source IoT database designed to meet the rigorous data, storage, and analytics requirements of large-scale Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications.

Different with traditional relational databases, IoTDB uses tree-based structure in memory to manage the schema (a.k.a, metadata), and use a Write-Ahead-Log-like file structure to persist the schema.

Now, each time series will take about 300 Bytes in memory. However, an IoTDB instance may manage more than 100 million time series, which may take more than 30GB memory.

Therefore, we'd like to re-design the schema management module.
1. File: Persist the tree on disk like a b-tree.
2. WAL: implement the WAL of the metadata. So we can update the tree on disk in batch, rather than one operation by one.
3. Cache: we may have no memory to load the whole tree. So a cache is needed and query from the tree on disk is needed.

What knowledge you need to know:
1. Java
2. Basic design idea about Database [2]

[1] https://iotdb.apache.org

Difficulty: Major
Potential mentors:
Xiandong Huang, mail: hxd (at) apache.org
Project Devs, mail: dev (at) iotdb.apache.org

Apache IoTDB: GUI workbench

Apache IoTDB [1] is an Open Source IoT database designed to meet the rigorous data, storage, and analytics requirements of large-scale Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications.

As a database, it is good to have a workbench to operate IoTDB using a GUI.

For example, there is a 3rd-part web-based workbench for Apache Cassandra [2]. MySQL supports a more complex workbench application [3].

We also want to IoTDB has a workbench.
Task:
1. execute SQL and show results in Table or Chart.
2. view the schema of IoTDB (how many Storage groups, how many time series etc.)
3. View and modify IoTDB’s configuration
4. View IoTDB’s dynamic status (e.g., info that JMX can get)

(As we have integrated IOTDB with Apache Zeppelin, task 1 has done. So, we hope this workbench can be more lightweight than using Zeppelin.)

Better to use Java. (Python or some others are also ok).

Needed Skills:
- Java
- Web application development

[1] iotdb.apache.org

Apache IoTDB: Complex Arithmetic Operations in SELECT Clauses

Apache IoTDB [1] is an Open Source IoT database designed to meet the rigorous data, storage, and analytics requirements of large-scale Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications.

We have recently been working to improve the ease of use of IoTDB. For queries, we hope that IoTDB can provide more powerful analysis capabilities.

IOTDB supports many types of queries: raw data queries, function queries (including UDF queries), and so on. However, currently there is no easy way to combine the results of multiple queries. Therefore, we hope that IoTDB can support complex arithmetic operations in the SELECT clause, which will greatly improve the analysis capabilities.

Function description:
Applied to: raw time series, literal numbers and function outputs.
Applicable data types: all types except TIMESTAMP and TEXT.
Applicable operators: at least five binary operators (+, -, *, /, %) and two unary operator (+, -).

Usage examples:

1. raw queries
   SELECT -a FROM root.sg.d;
   SELECT a, b, c * b - 4 * a * c FROM root.sg.d WHERE b > 0;
   SELECT a, b, -(bool_value * (a - b)) FROM root.sg.d;
   SELECT 3.14 + a / 15 + 926 FROM root.sg.d;
   SELECT +a % 3.14 FROM root.sg.d WHERE a < 0;

2. function queries
   SELECT a + abs(a), sin(a) * cos(a) FROM root.sg.d;
   SELECT a, b, sqrt(a) * sqrt(b) / (a * b) FROM root.sg.d WHERE a < 0;

3. nested queries
   select a, b + udf(sin(a) * sin(b), cos(a) * cos(b)) FROM root.sg.d;
   select a, a + a, sin(sin(sin(a + a))) FROM root.sg.d WHERE a < 0;

Additional requirements:
1. For performance reasons, it's better to perform as few disk read operations as possible.
   Example:
   SELECT a, sin(a + a) FROM root.sg.d WHERE a < 0;
   The series root.sg.d.a should be read only once during the query.

2. For performance reasons, it's better to reuse intermediate calculation results as much as possible.
   Example:
   SELECT a + a, sin(a + a) FROM root.sg.d WHERE a < 0;
   The intermediate calculation result a + a should only be evaluated once during the query.

3. Need to consider memory-constrained scenarios.

What knowledge you need to know:
1. Java
2. Basic database knowledge (such as SQL, etc.)
3. ANTLR
4. IoTDB query process
Apache IoTDB: integration with Chaos Mesh

Apache IoTDB [1] is an Open Source IoT database designed to meet the rigorous data, storage, and analytics requirements of large-scale Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications.

Chaos Mesh [2] is a versatile chaos engineering solution that features all-around fault injection methods for complex systems on Kubernetes [3], covering faults in Pod, network, file system, and even the kernel.

We hope that Chaos Mesh can be used as a versatile chaos test tool for the IoTDB cluster module, so as to verify the reliability of the IoTDB cluster module in production environment.

You should define a series of failure simulations for the cluster using Chaos Mesh, such as Network partition, Network packet loss and Node collapse, and then define a series of operations and the expected results of those operations.

This task hopes that you can set up an automated framework for IoTDB cluster module chaos testing, so that we can detect potential problems of cluster module and iteratively fix them.

Needed skills:
- Java
- Go
- Kubernetes
- Chaos mesh
- Know iotdb-benchmark [4]

[1] https://iotdb.apache.org
[3] https://kubernetes.io

Apache IoTDB: use netty as the memory buffer pool to reduce GC problem and take full use of memory

Apache IoTDB [1] is an Open Source IoT database designed to meet the rigorous data, storage, and analytics requirements of large-scale Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications.

Memory control is very very important for DBMS.

Currently, we are using a customized memory buffer pool, which contains a pool of int[], a pool of long[], a pool of boolean[] and as well as float[] and double[].
However, there are two things left:

1. It is complex to implement a buffer pool for `String[]` or `byte[][]`, as the size of `String` and `byte[]` are variable.
2. We are using `HeapByteBuf`, while in many cases, `directByteBuf` is more efficient.

As Netty has provided a high efficient buffer pool, we'd like to try to migrate the current buffer pool to the Netty implementation.

Things you should know:

1. Know Java well
2. Know Netty well
3. Read codes of IoTDB (mainly in `StorageEngine`, `StorageGroupProcessor` and related classes)

---

**Integrating Apache IoTDB and Apache Superset**

Apache IoTDB [1] is an Open Source IoT database designed to meet the rigorous data, storage, and analytics requirements of large-scale Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications.

Apache Superset [2] is fast, lightweight, intuitive, and loaded with options that make it easy for users of all skill sets to explore and visualize their data, from simple line charts to highly detailed geospatial charts.

We hope that Superset can be used as a data display and analysis tool of IoTDB, which will bring great convenience to analysts of the IoT and IIoT.

For a database engine to be supported in Superset, it requires having a Python compliant SQLAlchemy dialect [3] as well as a DBAPI driver [4] defined. The current Python client of IoTDB is packaged by Apache Thrift generated code and does not follow a certain interface specification. Therefore, the first thing you need to do is to implement a standard SQLAlchemy connector based on the current Python client (or some new interfaces defined and generated by Thrift).

Next, you need to explore how to integrate IoTDB and Superset and document the usage in a user-friendly way. The integration documentation for Apache Kylin and Superset is here [5] for your reference.

What knowledge you need to know:

- Basic database knowledge (SQL)
- Python

---

[1] https://iotdb.apache.org

**Difficulty:** Major

**Potential mentors:**

Xiangdong Huang, mail: hxd (at) apache.org

Project Devs, mail: dev (at) iotdb.apache.org
Traffic Control

**GSOC: Varnish Cache support in Apache Traffic Control**

**Background**
Apache Traffic Control is a Content Delivery Network (CDN) control plane for large scale content distribution.

Traffic Control currently requires Apache Traffic Server as the underlying cache. Help us expand the scope by integrating with the very popular Varnish Cache.

There are multiple aspects to this project:

- **Configuration Generation**: Write software to build Varnish configuration files (VCL). This code will be implemented in our Traffic Ops and cache client side utilities, both written in Go.
- **Health Monitoring**: Implement monitoring of the Varnish cache health and performance. This code will run both in the Traffic Monitor component and within Varnish. Traffic Monitor is written in Go and Varnish is written in C.
- **Testing**: Adding automated tests for new code

**Skills:**
- Proficiency in Go is required
- A basic knowledge of HTTP and caching is preferred, but not required for this project.

**Difficulty:** Major

**Potential mentors:**
- Eric Friedrich, mail: friede (at) apache.org
- Project Devs, mail: dev (at) trafficcontrol.apache.org

Dolphin Scheduler

**Apache DolphinScheduler-Parameter coverage**

**Apache DolphinScheduler**
Apache DolphinScheduler is a distributed and extensible workflow scheduler platform with powerful DAG visual interfaces, dedicated to solving complex job dependencies in the data pipeline and providing various types of jobs available out of the box.

Page: https://dolphinscheduler.apache.org
GitHub: https://github.com/apache/incubator-dolphinscheduler

**Background:**
Configuration parameter override

At present, our parameter configuration is mainly based on configuration files: you can refer to PropertiesUtils,

But usually important parameters will be injected through the form of Java runtime virtual machine parameters, so we need to support this way of parameter injection. At the same time, because different ways of parameter injection have different priorities, we need to achieve configuration coverage. There are two main situations at present, SystemProperties and LocalFile. The priority of SystemProperties should be the highest, followed by LocalFile (that is, our various configuration files, such as master.properties).

issue: https://github.com/apache/incubator-dolphinscheduler/issues/5164

for example:
1: Configure master.max.cpuload.avg=-1 in master.properties
2: Java runtime virtual machine parameters -Dmaster.max.cpuload.avg=1
3: PropertiesUtils.get("master.max.cpuload.avg") = 1

**Task**: realize configuration parameter coverage

**Mentor**: CalvinKirs kirs@apache.org

**Difficulty**: Major
CouchDB

**GSoC: Apache CouchDB and Debezium integration**

Apache CouchDB software is a document-oriented database that can be queried and indexed in a MapReduce fashion using JavaScript. CouchDB also offers incremental replication with bi-directional conflict detection and resolution.

Debezium is an open source distributed platform for change data capture. Start it up, point it at your databases, and your apps can start responding to all of the inserts, updates, and deletes that other apps commit to your databases. Debezium is durable and fast, so your apps can respond quickly and never miss an event, even when things go wrong.

CouchDB has a change capture feed as a public HTTP API endpoint. Integrating with Debezium would provide an easy way to translate the _changes feed into a Kafka topic which plugs us into a much larger ecosystem of tools and alleviates the need for every consumer of data in CouchDB to build a bespoke "follower" of the _changes feed.

The project for GSoC 2021 here is to design, implement and test a CouchDB connector for Debezium.

**Required skills:**
- Java

**Nice-to-have skills:**
- Erlang

**Difficulty:** Major

Potentially mentors:
- Balázs Donát Bessenyei, mail: bessbd (at) apache.org
- Project Devs, mail: dev (at) couchdb.apache.org

CloudStack

**CloudStack GSoC 2021 - Clone a Virtual Machine (with all the data disks)**

Hi there,

Here is the background of the proposed improvement in the CloudStack.

Currently, there is no straight way to clone / create a copy of the VM (with all the data disks) in CloudStack. Operator/Admin requires a series of steps /API cmds to be followed to achieve that in CloudStack, and also it takes considerable time (to wait and check each cmd response before proceeding to next step). Some hypervisors (Eg. VMware) already supports clone VM operation, and CloudStack can leverage that.

The support for this new functionality, can be integrated by introducing a new (admin-only) API to clone the VM, something like `cloneVirtualMachine`, which facilitates direct way to clone / create a copy of the VM (with all the data disks) can be. CloudStack internally performs all the required operations to create the copy of the VM (leverages the relevant hypervisor(s) operations if necessary), and returns the VM as response when success, otherwise throws the relevant error message.

This improvement will be a good addition to the VM operations supported in the CloudStack. It requires some virtualization/cloud domain knowledge & usage.

More details here: https://github.com/apache/cloudstack/issues/4818

**Skills Required:**
- Java and Python
- Vue.js (for UI integration)

**Difficulty:** Major
CloudStack GSoC 2021 Ideas

Hello Students! We are the Apache CloudStack project. From our project website: "Apache CloudStack is open source software designed to deploy and manage large networks of virtual machines, as a highly available, highly scalable Infrastructure as a Service (IaaS) cloud computing platform. CloudStack is used by a number of service providers to offer public cloud services, and by many companies to provide an on-premises (private) cloud offering, or as part of a hybrid cloud solution."

2-min video on the Apache CloudStack project - https://www.youtube.com/watch?v=oJ4b8HFmFTc

Here’s about an hour-long intro to what is CloudStack - https://www.youtube.com/watch?v=4qFFwyK9hos

The general skills student would need are - Java, Python, JavaScript/Vue. Idea-specific requirements are mentioned on the idea issue. We’re a diverse and welcoming community and we encourage interested students to join the dev ML: http://cloudstack.apache.org/mailing-lists.html (dev@cloudstack.apache.org)

All our Apache CloudStack GSoC2021 ideas are tracked on the project's Github issue: https://github.com/apache/cloudstack/issues?q=is%3Aissue+is%3Aopen+label%3Agsoc2021

<table>
<thead>
<tr>
<th>Feature</th>
<th>Skills Required</th>
<th>Difficulty Level</th>
<th>Potential Mentor(s)</th>
<th>Details and Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Multiple SSH Keys for VMs</td>
<td>Java, Javascript/Vue</td>
<td>Medium</td>
<td>David Jumani <a href="mailto:david.jumani@shapeblue.com">david.jumani@shapeblue.com</a></td>
<td><a href="https://github.com/apache/cloudstack/issues/4813">https://github.com/apache/cloudstack/issues/4813</a></td>
</tr>
<tr>
<td>Clone a Virtual Machine</td>
<td>Java, Javascript/Vue</td>
<td>Medium</td>
<td>Suresh Anaparti <a href="mailto:sureshanaparti@apache.org">sureshanaparti@apache.org</a></td>
<td><a href="https://github.com/apache/cloudstack/issues/4818">https://github.com/apache/cloudstack/issues/4818</a></td>
</tr>
<tr>
<td>UI Shortcuts (UX improvements in the UI)</td>
<td>Javascript, Vue</td>
<td>Easy</td>
<td>Boris Stoyanov <a href="mailto:boris.stoyanov@shapeblue.com">boris.stoyanov@shapeblue.com</a>, David Jumani <a href="mailto:david.jumani@shapeblue.com">david.jumani@shapeblue.com</a></td>
<td><a href="https://github.com/apache/cloudstack/issues/4798">https://github.com/apache/cloudstack/issues/4798</a></td>
</tr>
<tr>
<td>CloudStack OAuth2 Plugin</td>
<td>Java, Javascript/Vue</td>
<td>Medium</td>
<td>Nicolas Vazquez <a href="mailto:nicovazquez90@gmail.com">nicovazquez90@gmail.com</a>, Rohit Yadav <a href="mailto:rohit@apache.org">rohit@apache.org</a></td>
<td><a href="https://github.com/apache/cloudstack/issues/4834">https://github.com/apache/cloudstack/issues/4834</a></td>
</tr>
<tr>
<td>Synchronization of network devices on newly added hosts for Persistent Networks</td>
<td>Java</td>
<td>Medium</td>
<td>Pearl Dsilva <a href="mailto:pearl.dsilva@shapeblue.com">pearl.dsilva@shapeblue.com</a></td>
<td><a href="https://github.com/apache/cloudstack/issues/4814">https://github.com/apache/cloudstack/issues/4814</a></td>
</tr>
<tr>
<td>Add SPICE console for vms on KVM/XenServer</td>
<td>Java, Python, Javascript</td>
<td>Hard</td>
<td>Wei Zhou <a href="mailto:ustweizhou@gmail.com">ustweizhou@gmail.com</a></td>
<td><a href="https://github.com/apache/cloudstack/issues/4803">https://github.com/apache/cloudstack/issues/4803</a></td>
</tr>
<tr>
<td>Configuration parameters and APIs mappings</td>
<td>Java, Python</td>
<td>Hard</td>
<td>Harikrishna Patnala <a href="mailto:harikrishna@apache.org">harikrishna@apache.org</a></td>
<td><a href="https://github.com/apache/cloudstack/issues/4825">https://github.com/apache/cloudstack/issues/4825</a></td>
</tr>
<tr>
<td>Add virt-v2v support in CloudStack for VM import to KVM</td>
<td>Java, Python, libvirt, libguestfs</td>
<td>Hard</td>
<td>Rohit Yadav <a href="mailto:rohit@apache.org">rohit@apache.org</a></td>
<td><a href="https://github.com/apache/cloudstack/issues/4696">https://github.com/apache/cloudstack/issues/4696</a></td>
</tr>
</tbody>
</table>

We have an onboarding course for students to learn and get started with CloudStack: https://github.com/shapeblue/hackerbook

Project wiki and other resources:
https://cwiki.apache.org/confluence/display/CLOUDSTACK
https://github.com/apache/cloudstack
http://docs.cloudstack.apache.org/

**Difficulty:** Major
Mapping existing configuration parameters to the APIs

Hello students,

The following improvement will be a good addition to CloudStack and helps you learning a good product on a broad view.

Background:

Cloudstack has global settings in which there are around 672 configuration parameters. Using these parameters one can adjust the values according to the need of the environment. For example “allow.duplicate.networkname” default value of this parameter is true, which means networks can be created with same name in an account. If the value is set to false then duplicate network names are not allowed.

Problem statement:

When admin wants to change the API behaviour or during debugging an issue, admin has to search for the corresponding configuration parameter. The only way currently everyone is using is by using string search or looking into corresponding documentation. Searching over 672 configuration parameters which is a huge list is not straight forward or may lead to missing few parameters.

Solution:

To address this problem I would like to propose the solution which maps the configurations parameters to the corresponding APIs. One can know what are the configuration parameters involved for a specific API. For example “createNetwork” API will be mapped to “allow.duplicate.networkname”. When admin wants to see what are the configuration parameters used for “createNetwork” API , this mapping will help. In the final result we will have a table with APIs in one column and configuration parameters in another column.

Skills Required:

- Java
- Python

More details at https://github.com/apache/cloudstack/issues/4825

Good luck.

Support Multiple SSH Keys for VMs

To provide easy access to VMs without the need for password-based authentication, Cloudstack provides users with the ability to set or reset an SSH key for their VMs. These SSH keys can either be uploaded to, or generated by cloudstack.

As of now, it is limited to just a single SSH key. This requires the key to be shared amongst users with access to the VM and can be quite cumbersome when there are several VMs (perhaps across different projects), each with a different SSH key. It can also cause issues when a user is removed and the key needs to be reset since it’s a common key, and shared to the remaining users all over again.

This feature proposes extending the functionality to allow multiple SSH keys to be set / reset on a VM. This will allow users to add their own personal SSH key to the VM, thereby allowing them to directly access the VM without the need to manage multiple shared keys. New users can add their own keys to the list and access the VM instantly. It also solves the problem when a user is removed as their key can be removed from the list, thereby revoking their access without the need for a new key to be regenerated and shared.

It proposes the following changes :

1. Modify the API to accept multiple SSH keys
2. Change the service layer that handles the request
3. Alter the database accordingly
4. Update the UI to align with the new API

It requires the following relevant skills:

- Java (basic)
- SQL (basic)
- Javascript (basic)
- VueJS (Learning on the fly)

Further details can be found here
https://github.com/apache/cloudstack/issues/4813
Add dynamism to Persistent Networks in CloudStack

Hey There!

To give you a brief overview: Users often may want to manage resources like Virtual Machines, physical devices, like routers, switches, etc, outside the scope of CloudStack. To ensure that such devices can be easily managed and provisioned, CloudStack offers Persistent Networks, which ascertains that the network gets provisioned at the time of its creation, unlike the usual networks, which get provisioned only after deployment of VMs on that network.

However, today, we do not have a mechanism to facilitate automatic creation of the network on hosts that have been added to a cluster or transitioned from disabled/maintenance to Enabled state post creation of the network. Under such circumstances, users would need to either manually setup the network or deploy a VM via CloudStack to provision the network on specific hosts.

It would be a nice feature to incorporate some amount of dynamism into CloudStack such that we can introduce a sort of a Listener to

- Scan and identify if new hosts have been added
- Scan for hosts that have transitioned to Active state post creation of the network and enable implementing the resources on such Hosts.

And at the end of the day, end users are happy that everything required is already setup for them to deploy their resources - VMs.

Skills required:

- Java
- MySQL

More info can be got at: https://github.com/apache/cloudstack/issues/4814

CloudStack GSoC 2021 - UX improvements in the UI with Vue.js

Hi there student, this could be interesting task to any UI/UX engineer with some knowledge in Vue.js or relevant tech, it not required huge domain or java knowledge.

Here’s some of the main objectives in a brief description:

1. Introduce shortkeys navigation - Being able to navigate without mouse and use just keyboard – for example when on the dashboard when the user presses T, he could be directly transitioned to Templates. Or I for infrastructure, G – global settings.

2. Dialogue confirmations - Confirm dialogues with key – This is not working consistently, some dialogue windows have it some do not. But surely it’ll be great if we can confirm by pressing Enter or Space and cancel by Esc.

3. Forms submission improvements – For example, when deploying instance if you simply click ok without selected network it will prompt you with error and then you’ll need to scroll back to the network place and add it.

More info about the task here: https://github.com/apache/cloudstack/issues/4798

CloudStack GSoc 2021 - CloudStack OAuth2 Plugin
This can be an interesting task for an engineer with domain knowledge on backend services in Java and some knowledge in Vue.js or relevant tech. Also, domain knowledge of OAuth authentication is desirable.

The main objectives of this task are:

- Create a new CloudStack authentication plugin: this plugin will allow authentication to third-party libraries such as Google, Facebook, Github, etc.
- Extend CloudStack configurations: allow administrators to enable/disable the plugin and configure the auth provider

More information about the task on: https://github.com/apache/cloudstack/issues/4834

**Difficulty:** Major  
**Potential mentors:**  
Nicolás Vázquez, mail: nvazquez (at) apache.org  
Project Devs, mail: dev (at) cloudstack.apache.org

---

### Clerezza

#### Implement a Secure Scuttlebutt Server

Secure Scuttlebutt (SSB) is a peer-to-peer communication protocol, mesh network, and self-hosted social media ecosystem. The Scuttlebutt Protocol Guide can be found here: https://ssbc.github.io/scuttlebutt-protocol-guide/. Starting from this link https://handbook.scuttlebutt.nz/ you can acquire quite a lot knowledge of the SSB protocol and orientations from the conceptual to the practical aspects of writing and using SSB based applications.

The task of this issue is to implement the functionality of an SSB server as done in JavaScript by https://github.com/ssbc/ssb-server, but using Apache Tuweni (Incubating) for the handshake and an Apache Clerezza based triple store for storing and querying the messages. Apache Tuweni (https://tueni.apache.org/) is a set of libraries and other tools to aid development of blockchain and other decentralized software in Java and other JVM languages.

For those who are interested in getting more detail about SSB's features and its operations, see also the paper titled "Secure Scuttlebutt: An Identity-Centric Protocol for Subjective and Decentralized Applications" attached in this issue.

**Difficulty:** Major  
**Potential mentors:**  
Furkan Kamaci, mail: kamaci (at) apache.org  
Project Devs, mail:

---

### Library for Messaging App Storage

Nowadays, more and more companies are using a messaging app to communicate with their customers. The conversations contain not only information about customer engagements and requests, but may reveal interesting facts relevant for product marketing when analysed using business intelligent tools. Thus, having those conversations modelled appropriately is of great importance. RDF is surely one suitable format for modelling and storing those conversations.

The purpose of this task is to define an ontology for conversations and to create a Java library based on Apache Clerezza to store them. Signal (https://signal.org/docs/) is one of many messaging apps which is open source. This task recommends to integrate the library to be developed into Signal (http://github.com/signalapp) for testing.

By resolving this task you will learn

- RDF  
- Apache Clerezza  
- Signal API

**Difficulty:** Major  
**Potential mentors:**  
Furkan Kamaci, mail: kamaci (at) apache.org  
Project Devs, mail:

---

### Cassandra

#### Add Plugin Support for CQLSH
Currently the Cassandra drivers offer a plugin authenticator architecture for the support of different authentication methods. This has been leveraged to provide support for LDAP, Kerberos, and Sigv4 authentication. Unfortunately, cqlsh, the included CLI tool, does not offer such support. Switching to a new enhanced authentication scheme thus means being cut off from using cqlsh in normal operation.

We should have a means of using the same plugins and authentication providers as the Python Cassandra driver.

Here's a link to an initial draft of CEP.

**Difficulty:** Normal
**Potential mentors:**
- paulo, mail: paulo (at) apache.org
- Project Devs, mail: dev (at) cassandra.apache.org

---

**Prevent and fail-fast any attempts to incremental repair cdc/mv tables**

Running incremental repairs on CDC or MV tables breaks them.

Attempting to run incremental repair on such should fail-fast and be prevented, with a clear error message.

**Difficulty:** Normal
**Potential mentors:**
- paulo, mail: paulo (at) apache.org
- Project Devs, mail: dev (at) cassandra.apache.org

---

**Add ability to ttl snapshots**

It should be possible to add a TTL to snapshots, after which it automatically cleans itself up.

This will be useful together with the auto_snapshot option, where you want to keep an emergency snapshot in case of accidental drop or truncation but automatically remove it after a specified period when it's no longer useful. So in addition to allowing a user to specify a snapshot ttl on nodetool snapshot we should have a auto_snapshot_ttl option that allows a user to set a ttl for automatic snapshots on drop/truncate.

**Difficulty:** Normal
**Potential mentors:**
- paulo, mail: paulo (at) apache.org
- Project Devs, mail: dev (at) cassandra.apache.org

---

**Add nodetool command to display or export the contents of a virtual table**

Several virtual tables were recently added, but they're currently only accessible via cqlsh or programmatically. While this is valuable for many use cases, operators are accustomed with the convenience of querying system metrics with a simple nodetool command.

In addition to that, a relatively common request is to provide nodetool output in different formats (JSON, YAML and even XML) (CASSANDRA-5977, CASSANDRA-12035, CASSANDRA-12486, CASSANDRA-12698, CASSANDRA-12503). However this requires lots of manual labor as each nodetool subcommand needs to be adapted to support new output formats.

I propose adding a new nodetool command that will consistently print to the standard output the contents of a virtual table. By default the command will print the output in a human-readable tabular format similar to cqlsh, but a "--format" parameter can be specified to modify the output to some other format like JSON or YAML.

It should be possible to add a limit to the amount of rows displayed and filter to display only rows from a specific keyspace or table. The command should be flexible and provide simple hooks for registration and customization of new virtual tables.

I propose calling this command nodetool show <virtualtable> (naming bikeshedding welcome), for example:
nodetool show --list
  caches
  clients
  internode_inbound
  internode_outbound
  settings
  sstable_tasks
  system_properties
  thread_pools

nodetool show clients --format yaml
...
nodetool show internode_outbound --format json
...
nodetool show sstabletasks --keyspace my_ks --table -my_table
...

**Script to autogenerate cassandra.yaml**

It would be useful to have a script that can ask the user a few questions and generate a recommended cassandra.yaml based on their answers. This will help solve issues like selecting num_tokens. It can also be integrated into OS specific packaging tools such as debconf[1]. Rather than just documenting on the website, it is best to provide a simple script to auto-generate configuration based on common use-cases.


**Allow table property defaults (e.g. compaction, compression) to be specified for a cluster/keyspace**

During an IRC discussion in cassandra-dev it was proposed that we could have table property defaults stored on a Keyspace or globally within the cluster. For example, this would allow users to specify "All new tables on this cluster should default to LCS with SSTable size of 320MiB" or "all new tables in Keyspace XYZ should have Zstd compression with a 8 KiB block size" or "default_time_to_live should default to 3 days" etc ... This way operators can choose the default that makes sense for their organization once (e.g. LCS if they are running on fast SSDs), rather than requiring developers creating the Keyspaces/Tables to make the decision on every creation (often without context of which choices are right).

A few implementation options were discussed including:

- A YAML option
- Schema provided at the Keyspace level that would be inherited by any tables automatically
- Schema provided at the Cluster level that would be inherited by any Keyspaces or Tables automatically

In IRC it appears that rough consensus was found in having global -> keyspace -> table defaults which would be stored in schema (no YAML configuration since this isn't node level really, it's a cluster level config).

**Global configuration parameter to reject repairs with anti-compaction**

We have moved from Cassandra 2.1 to 3.0 and from an operational aspect, the Cassandra repair area changed significantly / got more complex. Beside incremental repairs not working reliably, also full repairs (-full command-line option) are running into anti-compaction code paths, splitting repaired / non-repaired data into separate SSTables, even with full repairs.
Casandra 4.x (with repair enhancements) is quite away for us (for production usage), thus we want to avoid anti-compactions with Cassandra 3.x at any cost. Especially for our on-premise installations at our customer sites, with less control over on how e.g. nodetool is used, we simply want to have a configuration parameter in e.g. cassandra.yaml, which we could use to reject any repair invocations that results in anti-compaction being active.

I know, such a flag still can be flipped then (by the customer), but as a first safety stage possibly sufficient enough to reject anti-compaction repairs, e.g. if someone executes nodetool repair ... the wrong way (accidentally).

**Difficulty:** Normal  
**Potential mentors:**  
paulo, mail: paulo (at) apache.org  
Project Devs, mail: dev (at) cassandra.apache.org

### Expose application_name and application_version in virtual table system_views.clients

Recent java-driver's com.datastax.oss.driver.api.core.session.SessionBuilder respects properties ApplicationName and ApplicationVersion.

It would be helpful to expose this information via virtual table system_views.clients and with nodetool clientstats.

**Difficulty:** Normal  
**Potential mentors:**  
paulo, mail: paulo (at) apache.org  
Project Devs, mail: dev (at) cassandra.apache.org

### Per-node overrides for table settings

There is a few cases where it's convenient to set some table parameters on only one of a few nodes. For instance, it's useful for experimenting with settings like caching options, compaction, compression, read repair chance, gcGrace ... Another case is when you want to completely migrate to a new setting, but want to do that node-per-node (mainly useful when switching compaction strategy, see CASSANDRA-10898).

I'll note that we can already do some of this through JMX for some of the settings as we have methods like ColumnFamilyStoreMBean.setCompactionParameters(), but:

1. parameters settings are initially set in CQL. Having to go to JMX for this sounds less consistent to me. The fact we have both a ColumnFamilyStoreMBean.setCompactionParameters() and a ColumnFamilyStoreMBean.setCompactionParametersJson() (as I assume the former one is inconvenient to use) is also proof to me than JMX ain't terribly appropriate.
2. I think this can be potentially useful for almost all table settings, but we don't expose JMX methods for all settings, and it would be annoying to have to. The method suggested below wouldn't have to be updated every time we add a new setting (if done right).
3. Changing options through JMX is not persistent across restarts. This may arguably be fine in some cases, but if you're trying to migrate your compaction strategy node per node, or want to experiment with a setting over a mediumish time period, it's mostly a pain.

So what I suggest would be add node overrides in the normal table setting (which would be part of the schema as any other setting). In other words, if you want to set LCS for only one specific node, you'd do:

```
ALTER TABLE foo WITH node_overrides = { '192.168.0.1' : { 'compaction' : { 'class' : 'LeveledCompactionStrategy' } } }
```

I'll note that I already suggested that idea on CASSANDRA-10898, but as it's more generic than what that latter ticket is about, so creating its own ticket.

**Difficulty:** Challenging  
**Potential mentors:**  
paulo, mail: paulo (at) apache.org  
Project Devs, mail: dev (at) cassandra.apache.org

### Add ability to disable schema changes, repairs, bootstraps, etc (during upgrades)

There are a lot of operations that aren't supposed to be run in a mixed version cluster: schema changes, repairs, topology changes, etc. However, it's easily possible for these operations to be accidentally run by a script, another user unaware of the upgrade, or an operator that's not aware of these rules.

We should make it easy to follow the rules by making it possible to prevent/disable all of these operations through nodetool commands. At the start of an upgrade, an operator can disable all of these until the upgrade has been completed.

**Difficulty:** Normal  
**Potential mentors:**
Beam

Profile and improve performance for the local FnApiRunner

The FnApiRunner is undergoing a series of changes to support streaming. These changes are altering its execution significantly, and may introduce inefficiencies.

This project has the following deliverables:

- A report with results from profiling the execution of a pipeline, and finding hotspots, and inefficiencies
- Code improvements to speed up the execution of the FnApiRunner
- Improvements to the FnApiRunner manual to instruct others on how to do profiling.

Tools that you may need to use:

- Benchmarks for the Python direct runner in http://metrics.beam.apache.org
- The FnApiRunner and TestStream microbenchmarks in sdks/python/apache_beam/tools/
- A profiling tool like: https://pypi.org/project/flameprof/ or https://github.com/uber-archive/pyflame
- Google Docs

Contact Pablo in dev@beam.apache.org to ask questions about this project.

Difficulty: P2
Potential mentors:
  Pablo Estrada, mail: pabloem (at) apache.org
  Project Devs, mail: dev (at) beam.apache.org

Apache NuttX

NuttX Support for Rapid Control Applications Development with pysimCoder

The main goal is integrating pysimCoder with NuttX to develop NuttX applications easily and faster for Engineering Students!

The pysimCoder is open-source Rapid (Control) Application Development Tool (RAD) which targets broad range of operating systems and target platforms. It has a graphical editor, which is able to translate a block diagram into C code. The C-code can be easily integrated into a main file for different embedded and linux systems. The design of the controller is performed using a Python script.

The NuttX is ideal real-time operating systems to combine with pysimCoder to prepare Rapid Control Prototyping (RCP) platforms based on small and mid-range embedded microcontrollers systems. NuttX POSIX standard programing model allows fast porting of applications and pysimCoder support between GNU/Linux for large designs to cheaper and smaller MCU based systems for faster and simpler applications which gives great potential to fill large gaps between hobby grade solutions supported by Arduino or microPython and professional systems requiring expensive software licenses (even for hobby grade hardware) and for professional use even expensive platforms.

The pysimCoder project repository is hosted here:

http://robertobucher.dti.supsi.ch/python/pysimcoder/

As the demonstration platform and BSP enhanced to support more sensors and actuators the student should use for example the iMX RT1050 that has support to CAN-FD.

Other boards like the STM32F4Discovery and ESP32/ESP32-C3 could be used, but these boards don't have CAN-FD support.

Difficulty: Normal
Potential mentors:
  Alan Carvalho de Assis, mail: acassis (at) apache.org
  Project Devs, mail: dev (at) nuttx.apache.org

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

**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](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).

**Difficulty:** Major

**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](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 an complement to C drivers.

**Difficulty:** Normal

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

---

**QA for NuttX with tests using QEMU, Renode and/or real hardware**

Currently NuttX supports more than 150 embedded boards and sometime some Pull Requests end up generating issues on some of these boards because the patch author is not able to test on all boards.

So to solve this issue we need to create some automatic board test (could be using QEMU, Renode or real hardware). This QA solution should be integrated on our current CI to automatically test each new submitted PR.

**Difficulty:** Normal

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

---

**Apache Nemo**

**Hierarchical aggregation and fidelity control in geo-distributed datacenter streaming environments**

Many widely-used distributed applications run on geo-distributed data centers (DCs). To timely understand and analyze the logs with low latency, it is required to control the communication costs. However, processing real time logs in a global scale is challenging due to its colossal volume and the expensive wide area networks (WANs) that unpredictably change over time, which makes it impractical to gather all data events into a single DC.
To resolve these challenges, we aim to perform aggregation operations in a decentralized and a hierarchical way within the data plane. We profile the network bandwidth and delay among the different executors of the node to perform clustering to create a tree of nodes based on their distance. With the profiled information, we aggregate and summarize the data along the hierarchy of the nodes, sorted by the distance within the nodes of the cluster, so that the data travelling through the WAN is minimized among the cluster. In the meanwhile, we also aim to control the fidelity of the aggregated data over the different distances to keep the network delays low.

In order to fine-tune the specific levels of the hierarchy, as well as to control the fidelity between the different levels of hierarchy to keep the bandwidth utilizations high and the network delays low, our system takes an automatic learning-based approach. With the profiled network metrics, our model looks for the most efficient number of levels for the hierarchically clustered tree of nodes, and finds the adequate level of fidelity to set for each of the levels of the hierarchy.

We aim to solve the following action items:

- Implementing and checking the correctness of the intermediate shuffle
- Evaluations for the throughput to confirm the performance improvement
- Adding the layer of fidelity control on the different levels of hierarchy
- A learning-based approach to automatically find the right levels of hierarchy and the level of fidelity for each level

**Difficulty:** Major  
**Potential mentors:**  
Won Wook Song, mail: wonook (at) apache.org  
Project Devs, mail: dev (at) nemo.apache.org

**Implement an accurate task execution simulator to predict distributed data processing execution**

We need to predict the system performance prior to the actual execution, as the execution often takes a very long time to complete. This will enable the exploration of the different search spaces in a shorter period of time, to find a better solution within the search space.

We can refer to EuroSys '12: Jockey: Guaranteed Job Latency in Data Parallel Clusters as a related work.

Some of the related TODOs are as follows:

- Aggregating task metrics and historical data/traces
- A mechanism for classifying the tasks and the relevant metrics & configurations that contribute to the resulting performance of the task
- Utilizing our implementation of the event-based simulator (implemented as a scheduler) to integrate the task time prediction mechanism into the existing components
- Experiments to confirm the accuracy of the simulator

**Difficulty:** Major  
**Potential mentors:**  
Won Wook Song, mail: wonook (at) apache.org  
Project Devs, mail: dev (at) nemo.apache.org

**Dynamic Work Stealing on Nemo for handling skews**

We aim to handle the problem on throttled resources (heterogeneous resources) and skewed input data. In order to solve this problem, we suggest dynamic work stealing that can dynamically track task statuses and steal workloads among each other. To do this, we have the following action items:

- Dynamically collecting task statistics during execution
- Detecting skewed tasks periodically
- Splitting the data allocated in skewed tasks and reallocating them into new tasks
- Synchronizing the optimization procedure
- Evaluation of the resulting implementations

**Difficulty:** Major  
**Potential mentors:**  
Won Wook Song, mail: wonook (at) apache.org  
Project Devs, mail: dev (at) nemo.apache.org

**Apache Hudi**

**Snowflake integration w/ Apache Hudi**
Apache Airflow integration w/ Apache Hudi

Pandas(python) integration w/ Apache Hudi

[UMBRELLA] Survey indexing technique for better query performance

[UMBRELLA] Support Apache Calcite for writing/querying Hudi datasets

[UMBRELLA] Improve CLI features and usabilities

[UMBRELLA] Improve source ingestion support in DeltaStreamer

[UMBRELLA] Checkstyle, formatting, warnings, spotless
Umbrella ticket to track all tickets related to checkstyle, spotless, warnings etc.

**Difficulty:** Major  
**Potential mentors:**  
sivabalan narayanan, mail: shivnarayan (at) apache.org  
Project Devs, mail: dev (at) hudi.apache.org

[UMBRELLA] Support Apache Beam for incremental tailing

(More details to be added)

**Difficulty:** Major  
**Potential mentors:**  
Vinoth Chandar, mail: vinoth (at) apache.org  
Project Devs, mail: dev (at) hudi.apache.org

[UMBRELLA] Support schema inference for unstructured data

(More details to be added)

**Difficulty:** Major  
**Potential mentors:**  
Raymond Xu, mail: xushiyan (at) apache.org  
Project Devs, mail: dev (at) hudi.apache.org

Apache Fineract

**Fineract Credit Bureau Integration Phase 4**

**Mentors**
- Nikhil Pawar
- Rahul Pawar
- Nayan Ambali, Ed Cable

**Overview & Objectives**
Because of regulatory reasons or to do background check of a client (risk management), MFIs depend on credit bureaus. As part of it, MFI must submit client details to credit bureau and also need to pull client information from credit bureau before approving any new loans to a client. Apache Fineract can be integrated with a popular CBs in India and from other regions (based on the demand).

**Description**
Building off the work that was kicked off in 2016, during the 2020 Google Summer of Code, Rahul Pawar, completed the credit bureau integration module with integrations for the credit bureau in Myanmar, MCIX. This project will continue extending the functionality of the module and work on integrations with the major credit bureaus in Latin America and Sub-Saharan Africa.

The major functionality will be sending the data to CBs on regular intervals in the format CB expects. And option to pull the client’s information from CB whenever loan officer/branch manager/user wants to view the information for a particular client.

**Helpful Skills**
SQL, Java, Javascript, Git, Web Services, Big Data (Hadoop, Hive)

**Impact**
The credit report shows account information such as repayment record, defaults, type of loan, amount of loan, etc. of the customer. This information facilitates prudent decision-making when the credit underwriter processes the loan application. This help MFI to reduce the risk of bad loans and reduces the multiple lendings to same person from different MFIs.

**Other Resources**
Documentation: https://cwiki.apache.org/confluence/display/FINERACT/Documentation+to+use+Integrated+Credit+Bureau
For the scope of this project, see https://jira.apache.org/jira/browse/FINERACT-734

Detailed requirements: https://goo.gl/aZWMZa

Source Code: https://github.com/apache/fineract/pulls?q=is%3Apr+is%3Aclosed+credit+bureau

**Live Fineract CN API Documentation (Swagger, etc.)**

**Mentors**
- Sanyam Goel
- Manthan Surkar

**Overview & Objectives**

The aim of this project is to provide a visual display of the Fineract CN API Documentation. We are now starting to use more of the Postman toolset for our developer portal and this project would focus on extending the existing work that was done.

**Description**

This project involves providing a visual display of the API Documentation of Apache Fineract CN. Student would have to optimize documentation snippets (.adoc), document any service which isn't completely documented like template and reporting, document failing unit tests too and develop a visual website where these html files will be hosted.

**Helpful Skills**

Java, PostgreSQL, MariaDB, Cassandra, TDD With JUnit 4, Spring REST Docs, Asciidoctor, HTML/CSS, Graphic Design

**Impact**

A visual presentation of the Fineract CN APIs will be a key building block for an enabling environment for developers working on Fineract CN.

**Other Resources**

- https://github.com/aasaru/fineract-cn-api-docs
- https://izakey.github.io/fineract-cn-api-docs-site/

**Create Open Banking Layer for Fineract 1.x Self-Service Apps**

**Mentors**
- Avik Ganguly
- Naman Dwivedi

**Overview & Objectives**

Across our ecosystem we’re seeing more and more adoption and innovation from fintechs. A huge democratizing force across the financial services sector is the Open Banking movement providing Open Banking APIs to enable third parties to directly interact with customers of financial institutions. We have recently started providing an Open Banking API layer that will allow financial institutions using Mifos and Fineract to offer third parties access to requesting account information and initiating payments via these APIs. Most recently the Mojaloop community, led by Google, has led the development of a centralized PISP API. We have chosen to follow the comprehensive UK Open Banking API standard which is being followed and adopted by a number of countries through Sub-Saharan Africa and Latin America.
Tremendous impact can be had at the Base of the Pyramid by enabling third parties to establish consent with customers to authorize transactions to be initiated or information to be accessed from accounts at their financial institution. This Open Banking API layer would enable any institution using Mifos or Fineract to provide a UK Open Banking API layer to third parties and fintechs.

The API Gateway to connect to is still being chosen (WS02, Gravitee, etc.)

**Description**

The APIs that are consumed by the the reference Fineract 1.x mobile banking application have been documented in the spreadsheet below. The APIs have also been categorized according to whether they are an existing self-service API or back-office API and if they have an equivalent Open Banking API and if so, a link to the corresponding Open Banking API.

For each API with an equivalent Open Banking API, the interns must: Take rest api, upload swagger definition, do transformation in OpenBanking Adapter, and publish on API gateway.

For back-office and/or self-service APIs with no equivalent Open Banking API, the process is: Take rest api, upload swagger definition, and publish on API gateway.

For example:

- Submit Loan Application (Self-Service API with Equivalent Open Banking API)
  - https://demo.mifos.io/api-docs/apiLive.htm#loans_create
  - Used by Fineract 1.x Self-Service App
- Images API (Back-Office API with No Equivalent Open Banking API)
  - https://demo.mifos.io/api-docs/apiLive.htm#client_images
  - Used by Mifos Mobile and Mobile Wallet
- Fetch Identification Card API (Fineract CN API with no equivalent Open Banking API)
  - https://docs.google.com/document/d/15LbxVoQRoa4uU7G7V7FpJFVjkyyN9b_HJwFvS47O4I/edit?pli=1#heading=h.xfl6jxdpicy1

**Sample APIs to be Documented**

Fineract 1.x Self Service App (Mifos Mobile) API Matrix (completed by Ashwin)
https://docs.google.com/spreadsheets/d/1gR84jZzLF-mM0Rw5JyeMAaHMk6RQPK0vyDmNAY9VhE/edit?pli=1

Fineract 1.x Self-Service App (Mifos Mobile) API Matrix (completed by Shivangi)
https://docs.google.com/spreadsheets/d/1exTv68v1IWidth7S7mSij0_ySFWGTjO6NhPZeNLjNly6Y/edit?pli=1#gid=0

**Helpful Skills**

Android development, SQL, Java, Javascript, Git, Spring, OpenJPA, Rest, Kotlin, Gravitee, WSO2

**Impact**

By providing a standard UK Open Banking API layer we can provide both a secure way for our trusted first party apps to allow customers to authenticate and access their accounts as well as an API layer for third party fintechs to securely access Fineract and request information or initiate transactions with the consent of customers.

**Other Resources**

CGAP Research on Open Banking: https://www.cgap.org/research/publication/open-banking-how-design-financial-inclusion
Docs: https://mifos.gitbook.io/docs/wso2-1/setup-openbanking-apis
Self-Service APIs: https://demo.mifos.io/api-docs/apiLive.htm#selfbasicauth
- https://cwiki.apache.org/confluence/display/FINERACT/Customer+Self-Service+Phase+2
  - Open Banking Adapter: https://github.com/openMF/openbanking-adapter
  - Transforms Open Banking API to Fineract API
  - Works with both Fineract 1.x and Fineract CN
  - Can connect to different API gateways and can transform against different API standards.

Reference Open Banking Fintech App:
- Backend: https://github.com/openMF/openbanking-tpp-server
- GUI: https://github.com/openMF/openbanking-tpp-client

Google Whitepaper on 3PPI: https://static.googleusercontent.com/media/nextbillionusers.google/en//tools/3PPI-2021-whitepaper.pdf

UK Open Banking API Standard: https://standards.openbanking.org.uk/

Open Banking Developer Zone: https://openbanking.atlassian.net/wiki/spaces/DZ/overview

Examples of Open Banking Apps: https://www.ft.com/content/a5f0af78-133e-11e9-a581-4ff78404524e

See https://openmf.github.io/mobileapps.github.io/
Create Open Banking Layer for Mobile Wallet Apps

Mentors

- Avik Ganguly
- Devansh Aggarwal

Overview & Objectives

Across our ecosystem we’re seeing more and more adoption and innovation from fintechs. A huge democratizing force across the financial services sector is the Open Banking movement providing Open Banking APIs to enable third parties to directly interact with customers of financial institutions. We have recently started providing an Open Banking API layer that will allow financial institutions using Mifos and Fineract to offer third parties access to requesting account information and initiating payments via these APIs. Most recently the Mojaloop community, led by Google, has led the development of a centralized PISP API. We have chosen to follow the comprehensive UK Open Banking API standard which is being followed and adopted by a number of countries through Sub-Saharan Africa and Latin America.

Tremendous impact can be had at the Base of the Pyramid by enabling third parties to establish consent with customers to authorize transactions to be initiated or information to be accessed from accounts at their financial institution. This Open Banking API layer would enable any institution using Mifos or Fineract to provide a UK Open Banking API layer to third parties and fintechs.

The API Gateway to connect to is still being chosen (WS02, Gravitee, etc.)

Description

The APIs that are consumed by the reference Fineract 1.x mobile banking application have been documented in the spreadsheet below. The APIs have also been categorized according to whether they are an existing self-service API or back-office API and if they have an equivalent Open Banking API and if so, a link to the corresponding Open Banking API.

For each API with an equivalent Open Banking API, the interns must: Take rest api, upload swagger definition, do transformation in OpenBanking Adapter, and publish on API gateway.

For back-office and/or self-service APIs with no equivalent Open Banking API, the process is: Take rest api, upload swagger definition, and publish on API gateway.

For example:

- Submit Loan Application (Self-Service API with Equivalent Open Banking API)
  - https://demo.mifos.io/api-docs/apilive.htm#loans_create
- Used by Fineract 1.x Self-Service App
- Images API (Back-Office API with No Equivalent Open Banking API)
  - https://demo.mifos.io/api-docs/apilive.htm#client_images
- Used by Mifos Mobile and Mobile Wallet
- Fetch Identification Card API (Fineract CNAPI with no equivalent Open Banking API)
  - https://docs.google.com/document/d/15LbxVoQRoa4uU7GI7FyVjkxyyNb9_HJwFvS47O4I/edit?pli=1#heading=h.xfl6jdpcpy1

Sample APIs to be Documented

---------------------------

Mobile Wallet API Matrix (completed by Devansh)
https://docs.google.com/spreadsheets/d/1VgpIwN2jJsljWWytk_Qb49kKzwmWw6xsa1oRgMNIAv3g/edit?#gid=0

Helpful Skills

Android development, SQL, Java, Javascript, Git, Spring, OpenJPA, Rest, Kotlin, Gravitee, WSO2

Impact

By providing a standard UK Open Banking API layer we can provide both a secure way for our trusted first party apps to allow customers to authenticate and access their accounts as well as an API layer for third party fintechs to securely access Fineract and request information or initiate transactions with the consent of customers.

Other Resources
Overview & Objectives

Across our ecosystem we’re seeing more and more adoption and innovation from fintechs. A huge democratizing force across the financial services sector is the Open Banking movement providing Open Banking APIs to enable third parties to directly interact with customers of financial institutions. We have recently started providing an Open Banking API layer that will allow financial institutions using Mifos and Fineract to offer third parties access to requesting account information and initiating payments via these APIs. Most recently the Mojaloop community, led by Google, has led the development of a centralized PISP API.

To demonstrate these Open Banking APIs and use cases that third parties and fintechs can provide we have developed a cross-platform reference mobile app on Kotlin to showcase a number of these features. It currently connects with the Open Bank Project that adheres to the UK Open Banking API standard. The API Gateway to connect to is still being chosen (WS02, Gravitee, etc.)

The breadth and variety of apps that could be built leveraging these APIs from region to region is endless. We would like this app to be built in an extensible and modular fashion such that core libraries and components could be re-used across different use cases with this framework as the foundation and multiple reference apps on top. Applications include personal financial management apps aggregating information from multiple bank accounts in one place, wallet apps allowing payments to be made from different banks, lending apps, leveraging data and insight from multiple accounts, savings apps, etc.

Description

Intern would work on refining the initial architecture of the framework, the UI and user experience, core use cases including customer authentication and onboarding that was implemented in 2020 and integrating with the Fineract Open Banking APIs and Mojaloop PISP APIs to demonstrate use cases around account information request and payment initiation.

Helpful Skills

Android development, SQL, Java, Javascript, Git, Spring, OpenJPA, Rest, Kotlin
Impact

By providing an extensible open banking fintech app framework, allow partners a complete stack of Open Banking APIs and reference front-end application to rapidly build innovation via Open Banking APIs.

Other Resources

2020 Progress: https://gist.github.com/ankurs287/4e7c3de462073bf36bd5247479cb176


UK Open Banking API Standard: https://standards.openbanking.org.uk/

Open Banking Developer Zone: https://openbanking.atlassian.net/wiki/spaces/DZ/overview

Examples of Open Banking Apps: https://www.ft.com/content/a5f0af78-133e-11e9-a581-4ff78404524e

Improve Robustness of Mifos X and Apache Fineract by Fixing Issues/Feature Requests in Backlog

Overview & Objectives

Mifos X and Apache Fineract is widely used by financial institutions of all different sizes and methodologies around the world. With that widespread user base there is a vast array of different processes and procedures that would like to be supported as slight modifications over the common functionality provided. Over the past several years, we have captured these minor enhancements in our issue tracker as feature requests. Also included in this backlog or additional minor and less critical bugs that have been reported but have not been fixed yet. This backlog has grown and it would be a very impactful project for an intern to work on completing as many of these bug fixes and minor enhancement as possible. The difficult level of these issues ranges from low to higher and touch all components of the platform - most don't require too much domain knowledge but some will.

Description

We have groomed the backlog and tagged issues and feature requests that are relevant for this project with the labels gsoc and/or Volunteer. Priority level of tasks is measured by p1 being the highest priority. Tasks with an assigned fix version of either 1.4.0 or 1.5.0 have a higher priority.

There are more than 120 tickets in the saved filter. You are not expected to complete all of the tasks in the backlog but throughout the internship you should fix as many issues/feature requests as possible. You will work with your mentor to deliver a plan for each sprint and adjust velocity as you get scaled up.

Issues to be worked on can be found under the blocked issue list below

Helpful Skills:

HTML, Spring, Hibernate, REST, Java, AngularJS, Javascript, SQL

Impact:

Better internal control and financial transparency

Other Resources:

Getting Started with Apache Fineract: https://cwiki.apache.org/confluence/display/FINERACT/Getting+Started+Docs

Update Fineract 1.x Android Client SDK

Mentors

- Chinmay Kulkarni
- Shashank Priyadarshi

Overview & Objectives

The goal of this project is to continue work on developing the Fineract 1.x Client Android SDK which will be used in other Mifos Mobile Applications (android-client, mifos-mobile, mobile-wallet). The project aims to remove a lot of repeated code in the mobile apps and help mobile apps easily migrate to newer versions of Apache Fineract 1.x
Description

The student will be working on implementing the following things:

- Generate and release latest Fineract Client SDK for Android using the Open API Specification of Apache Fineract
- Generate and publish documentation for the Fineract Android Client SDK
- Add support for RxJava + LiveData
- Migrate Mifos Android Field Officer App project to consume the new Fineract Client Android SDK
- Provide testing coverage throughout the SDK
- Implement CI/CD to automate steps 1 and 2

Helpful Skills

Java, Kotlin, Android, Swagger Specification, Open API Specification, Spring (Good to have)

Impact

Enabling other mifos mobile apps to easily migrate to latest versions of Fineract. More stable and error free codebase

Other Resources

- Last year progress - https://gist.github.com/Grandolf49/f79537436a467dac0baa9458a38290c5
- https://github.com/apache/fineract
- https://github.com/openMF/fineract-client

Difficulty: Major
Potential mentors:

- Ed Cable, mail: edcable (at) apache.org
- Project Devs, mail: dev (at) fineract.apache.org

Create Fineract CN mobile wallet app on top of wallet framework

Mentors

- Devansh Aggarwal
- Shivansh Tiwari

Overview & Objectives

We provide a reference mobile wallet application frameworks for consumers and merchants that has been developed by our Google Summer of Code interns from 2017 to 2020. The mobile wallet provides an extensible mobile wallet framework to support the basic use cases of a mobile wallet as documented in the Level One Project mobile wallet requirements. This extensible framework should support both merchant and client use cases as well as be capable of integrating with a Fineract or Fineract CN back-end.

Over time, we would like Fineract to be more generically a wallet management system and this reference application framework is a powerful tool to support that.

Currently there is no reference mobile wallet app that natively consumes APIs from Fineract CN. The current wallet utilizes a wrapper around Fineract APIs to facilitate such a connection.

Description

The initial mobile wallet framework along with 2 reference apps, PixieCollect and MifosPay, were developed in 2017. In 2019, these functionalities were extended further by Shivansh including including improving user experience and redesigning the app, support for Kotlin, integration with two Mojaloop transaction flows via the Paymeht Hub, adding improving Deeplinks, support for standing instructions and more well-rounded support for merchant transactions.

In 2020, Devansh Aggarwal added complete support for standing instructions, integrated with Fineract CN for core use cases by mapping Fineract back-office APIs to Fineract CN APIs, added multi-theme support, completed integration with Payment Hub EE for two use cases, added support for Hover, and converted Java code to Kotlin.

In 2021 we aim to complete a stand-alone MVP of a mobile wallet on top of the Fineract CN back-end. Once the requisite APIs are made available in Fineract CN, a new reference app will be built to consume these APIs. If there's enough time, some other major work beyond building the base Fineract CN mobile wallet app will be Payment Hub integration and improving peer to peer and merchant transactions (initiating transactions to merchants, maintaining history of users with which recent transactions took place, adding deeplink support for unique payment links, payment related notifications using FCM). Payment Hub and Mojaloop integration will allow us to make payments across tenants and fineract deployments. A very basic integration of payment hub with mobile wallet is already in place for Fineract but that will need to be extended to fully support all use cases on Fineract CN.
Helpful Skills

Android development, SQL, Java, Javascript, Git, Spring, OpenJPA, Rest, Kotlin

Impact

By providing an extensible mobile wallet framework, allow partners a complete reference stack of back and front-end applications to offer digital financial services to clients.

Other Resources

2020 Mobile Wallet Progress: https://gist.github.com/devansh-299/e2041c07d9ab55a747391951e9090df4

Mobile Wallet Framework: Source Code | Issue Tracker | Gitter Chatroom

See https://openmf.github.io/mobileapps.github.io/

Difficulty: Major

Potential mentors:
- Ed Cable, mail: edcable (at) apache.org
- Project Devs, mail: dev (at) fineract.apache.org

APISIX

Apache APISIX: supports obtaining etcd data information through plugin

Apache APISIX

Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etcd.

APISIX provides rich traffic management features such as load balancing, dynamic upstream, canary release, circuit breaking, authentication, observability, and more.

You can use Apache APISIX to handle traditional north-south traffic, as well as east-west traffic between services. It can also be used as a k8s ingress controller.

Background

When we get the stored data of etcd, we need to manually execute the URI request method to get each piece of data, and we cannot monitor the changed data in etcd. This is not friendly to issues such as obtaining multiple etcd stored data and monitoring etcd data changes. Therefore, we need to design a method to solve this problem.

Related issue: https://github.com/apache/apisix/issues/2453

Task

In the Apache APISIX (https://github.com/apache/apisix) project, implement a plug-in with the following functions:

1. Find route based on URI;
2. Watch etcd to print out the object that has recently changed
3. Query the corresponding data based on ID (route, service, consumer, etc.).

Relevant Skills

1. Master Lua language;
2. Have a basic understanding of API Gateway or Web server;
3. Be familiar with ETCD.

Mentor

Yuelin Zheng, yuelinz99@gmail.com

Difficulty: Major

Potential mentors:
- Yuelin Zheng, mail: firstsawyou (at) apache.org
- Project Devs, mail: dev (at) apisix.apache.org

Apache APISIX: support to fetch more useful information of client request
What's Apache APISIX?

Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etcd. APISIX provides rich traffic management features such as load balancing, dynamic upstream, canary release, circuit breaking, authentication, observability, and more.

You can use Apache APISIX to handle traditional north-south traffic, as well as east-west traffic between services. It can also be used as a k8s ingress controller.

Background (route matching and run plugins)

When the client completes a request, there is a lot of useful information inside Apache APISIX.

Task

Needs a way to show it. It is convenient for callers to troubleshoot problems and understand the workflow of Apache APISIX.

The first version target can display:
1. Which route is matched.
2. Which plugins are loaded.

In subsequent versions, we will add more information that the caller cares about, such as:

- Whether the global plugin is executed
- Time consumption statistics
- The return value when the plugin is executed

Relevant Skills

1. Master Lua language
2. Have a basic understanding of API Gateway or Web server

Potential mentors:
YuanSheng Wang, mail: membphis (at) apache.org
Project Deva, mail: dev (at) apisix.apache.org

Apache APISIX: support apply certificates from Let’s Encrypt or any other ACMEv2 service

Apache APISIX

Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etcd. APISIX provides rich traffic management features such as load balancing, dynamic upstream, canary release, circuit breaking, authentication, observability, and more.

You can use Apache APISIX to handle traditional north-south traffic, as well as east-west traffic between services. It can also be used as a k8s ingress controller.

Background

The data plane of Apache APISIX supports dynamic loading of ssl certificates, but the control plane does not have the support of ACME. Although users can use other tools to obtain ACME certificates, and then call the admin API to write them into Apache APISIX, this is not convenient for many users.

Task

In the Apache APISIX dashboard ([https://github.com/apache/apisix-dashboard](https://github.com/apache/apisix-dashboard)) project, add support for ACME, which can automatically obtain and update certificates

Relevant Skills

- TypeScript
- Golang
  familiar with Apache APISIX’s admin API

Mentor
Ming Wen, PMC of Apache APISIX, wenming@apache.org

Difficulty: Major
Apache APISIX: Support Nacos in a native way

Apache APISIX is a dynamic, real-time, high-performance cloud-native API gateway, based on the Nginx library and etcd.

Page: https://apisix.apache.org
Github: https://github.com/apache/apisix

Background

To get the upstream information dynamically, APISIX need to be integrated with other service discovery systems. Currently we already support Eureka, and many people hope we can support Nacos too.

Nacos is a widely adopted service discovery system: https://nacos.io/en-us/index.html

Previously we try to support Nacos via DNS. Nacos provides a CoreDNS plugin to expose the information via DNS: https://github.com/nacos-group/nacos-coredns-plugin

However, this plugin seems to be unmaintained.

Therefore, it would be good if we can support Nacos natively via its API, which is expected to be maintained.

Task

Integrate Nacos with APISIX via Nacos's HTTP API.

Relevant Skills

1. Master Lua language and HTTP protocol
2. Have a basic understanding of APISIX / Nacos

Targets files


References


Mentor

Zexuan Luo, committer of Apache APISIX

spacewander@apache.org

Difficulty: Major

Potential mentors:

Zexuan Luo, mail: spacewander (at) apache.org
Project Devs, mail: dev (at) apisix.apache.org

Apache APISIX: Enhanced verification for APISIX ingress controller

Apache APISIX

Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etcd.

APISIX provides rich traffic management features such as load balancing, dynamic upstream, canary release, circuit breaking, authentication, observability, and more.
You can use Apache APISIX to handle traditional north-south traffic, as well as east-west traffic between services. It can also be used as a k8s ingress controller.

Background

We can use APISIX as kubernetes ingress. Use CRD (Custom Resource Definition) on kubernetes to define APISIX objects, such as route/service /upstream/plugins.

We have done basic structural verification of CRD, but we still need more verification. For example, plug-in schema verification, dependency verification between APISIX objects, rule conflict verification, etc. All these verifications need to be completed before CRD is applied.

Task

1. Implement a validating admission webhook.
2. Support plugins schema verification.

Relevant Skills

1. Golang
2. Be familiar with Apache APISIX's admin API
3. Be familiar with kubernetes

Mentor

Wei Jin, PMC of Apache APISIX, kvn@apache.org

Difficulty: Major
Potential mentors:
Wei Jin, mail: kvn (at) apache.org
Project Devs, mail: dev (at) apisix.apache.org

Apache APISIX Dashboard: Enhancement plugin orchestration

The Apache APISIX Dashboard is designed to make it as easy as possible for users to operate Apache APISIX through a frontend interface.

The Dashboard is the control plane and performs all parameter checks; Apache APISIX mixes data and control planes and will evolve to a pure data plane.

This project includes manager-api, which will gradually replace admin-api in Apache APISIX.

Background

The plugin orchestration feature allows users to define the order of plugins to meet their scenarios. At present, we have implemented the plugin scheduling feature, but there are still many points to be optimized.

Task

1. develop a new plugin, conditional judgment card style. Currently, both the conditional judgment card and the plug-in card are square shaped, which makes it difficult for users to distinguish them, so in this task the conditional judgment card needs to be changed to a diamond shape.
As shown in the figure below.
Add arrows for connecting lines. The connection lines in the current plugin orchestration are not directional, we need to add arrows to the connection lines as shown in the figure below.
3. Limit plugin orchestration operations. We need to restrict the connection of each card to ensure the proper use of the plugin orchestration, and the situation shown in the arrow below is not allowed.

**Relevant Skills**

1. Basic use of HTML, CSS, and JavaScript.
2. Basic use of React Framework.
Apache APISIX: Support invoke an AWS Lambda function through plugin

Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etc.

APISIX provides rich traffic management features such as load balancing, dynamic upstream, canary release, circuit breaking, authentication, observability, and more.

You can use Apache APISIX to handle traditional north-south traffic, as well as east-west traffic between services. It can also be used as a k8s ingress controller.

Background

AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers, creating workload-aware cluster scaling logic, maintaining event integrations, or managing runtimes: https://aws.amazon.com/lambda/

We can create a plug-in for AWS Lambda, including authentication with AWS, trigger lambda function, and get the return result of lambda function. So that we can trigger lambda more securely and more easily.

Task

In the Apache APISIX (https://github.com/apache/apisix) project, implement a plug-in with the following functions:

1. Authentication with AWS
2. Trigger lambda function
3. Get the response from AWS Lambda

Relevant Skills

1. Master Lua language;
2. Have a basic understanding of API Gateway or Web server;

Mentor

Xinxin Zhu, committer of Apache APISIX
starsz@apache.org

Integrate Cert-Manager to APISIX Ingress Controller

Apache APISIX

Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etc.

APISIX provides rich traffic management features such as load balancing, dynamic upstream, canary release, circuit breaking, authentication, observability, and more.

You can use Apache APISIX to handle traditional north-south traffic, as well as east-west traffic between services. It can also be used as a k8s ingress controller.

Background
Cert-Manager helps to deliver and rotate certificates dynamically; While apisix-ingress-controller has a CRD resource ApisixTls which supports configuring certificate from Kubernetes Secrets (https://kubernetes.io/docs/concepts/configuration/secret/) dynamically, the secret itself, still need to be created by administrators themselves.

So here has an opportunity to integrate Cert-Manager and apisix-ingress-controller, let the Cert-Manager maintains the Secret (update it once the certificate changed), and let apisix-ingress-controller watches the newest change and apply the newest certificate (and the private key). This integration reduces the complexity of the maintaining of the apisix-ingress-controller.

**Task**

- Research Cert-Manager and its ACME issuer
- Add new design in the CRD ApisixTls, let it support integrating Cert-Manager
- Coding on apisix-ingress-controller, implementing the new version ApisixTls and creates a certificate request when an ApisixTls resource was created.

**Relevant Skills**

- Golang
- Have basic learning about Kubernetes, Ingress, and ingress controller.

**Mentor**

Chao Zhang, PMC of Apache APISIX, tokers@apache.org

**Difficulty:** Major

**Potential mentors:**

Chao Zhang, mail: tokers (at) apache.org

Project Devs, mail: dev (at) apisix.apache.org

---

**Exposing more Prometheus Metrics for Apache APISIX Ingress Controller**

**Apache APISIX**

Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etc.

APISIX provides rich traffic management features such as load balancing, dynamic upstream, canary release, circuit breaking, authentication, observability, and more.

You can use Apache APISIX to handle traditional north-south traffic, as well as east-west traffic between services. It can also be used as a k8s ingress controller.

**Background**

Metrics in the apixis ingress controller is less, which results in the observability of the controller itself is bad (though the data plane (i.e. APISIX itself) has rich metrics), therefor we need more and more metrics for apixis ingress controller, and show them in Grafana.

**Task**

- Plan which metrics should be exported in apixis-ingress-controller
- Coding on apixis-ingress-controller, adding these hooks for the aggregation of metrics.
- Integrating with Grafana.

**Relevant Skills**

- Golang
- Be familiar with Prometheus
- Have basic understandings of Kubernetes, Ingress, and Ingress Controllers.

**Mentor**

Chao Zhang, PMC of Apache APISIX, tokers@apache.org
Apache APISIX: enhanced authentication for Dashboard

Apache APISIX
Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etcd. APISIX provides rich traffic management features such as load balancing, dynamic upstream, canary release, circuit breaking, authentication, observability, and more. You can use Apache APISIX to handle traditional north-south traffic, as well as east-west traffic between services. It can also be used as a k8s ingress controller.

Background
At present, Apache APISIX Dashboard only supports simple username and password login, we need a universal authentication mechanism that can connect to user’s existing identity provider.

Task
In the Apache APISIX dashboard (https://github.com/apache/apisix-dashboard) project
1. Implement a universal login class
2. Support OAuth2 connection

Relevant Skills
1. Golang
2. TypeScript
3. Be familiar with ETCD

Mentor
Junxu Chen, PMC of Apache APISIX, chenjunxu@apache.org

Record short videos about Apache APISIX

Apache APISIX
Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etcd, and we have a standalone website to let more people know about the Apache APISIX.

Background
Apache APISIX has the official website[1], we would like to record more videos about How Apache APISIX works, What Apache APISIX is, How to write plugins for Apache APISIX, etc, to help more users to know what & how Apache APISIX works 😊

Task
• Draft the video list outline (I will provide it);
• To read & try to use Apache APISIX docs, we must clearly know what it is;
• Record videos (30 videos IMO).

Relevant Skills
• Read Docs;
• will use FinalCut or PR;
• Know about Apache APISIX;

Mentor
Zhiyuan, PMC of Apache APISIX, juzhiyuan@apache.org

[1] https://apisix.apache.org/
Apache APISIX: improve the website

Apache APISIX

Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etcd, and we have a standalone website to let more people know about the Apache APISIX.

Background

The website of Apache APISIX is used for showing people what’s Apache APISIX is, and it will include up to date docs to let developers searching guides more easily, and so on.

Task

In the website[1] and its repo[2], we are going to refactor the homepage, improve those docs which include apisix’s docs and some like release guide.

Relevant Skills

TypeScript
React.js

Mentor

Zhiyuan, PMC of Apache APISIX, juzhiyuan@apache.org

[1] https://apisix.apache.org/

APISIX ingress controller integration with Knative Serving

Apache APISIX

Apache APISIX is a dynamic, real-time, high-performance API gateway, based on the Nginx library and etcd.

APISIX provides rich traffic management features such as load balancing, dynamic upstream, canary release, circuit breaking, authentication, observability, and more.

You can use Apache APISIX to handle traditional north-south traffic, as well as east-west traffic between services. It can also be used as a k8s ingress controller.

Background

Knative Serving is a platform allowing you to deploy serverless applications based on Kubernetes[1]. Especially compared to bare Kubernetes, Knative could do scale-to-zero without help with binding to cloud providers.

Currently, Knative has supported ingress gateway like Istio, Gloo and Kong as its network layer[2-3]. In this project, we hope you could support APISIX ingress controller as an alternative Knative network layer, and then we could use APISIX to manage ingress traffic, especially add APISIX plugins when we need, for serverless workloads in Knative.

Task

1. Get familiar with Knative serving, and current offers of network layers
2. Implement support for APISIX for Knative
3. Add test, docs, and preferably articles, for the new feature.
Relevant Skills

1. Golang
2. Familiar with Kubernetes
3. Familiar with Knative/APISIX-ingress-controller/other ingress is a plus

Mentor

Shuyang Wu, committer of Apache APISIX, shuyangw@apache.org

Ref

1. https://knative.dev/docs/serving/
1. https://knative.dev/docs/install/any-kubernetes-cluster/#installing-the-serving-component

Difficulty: Major
Potential mentors:

Shuyang Wu, mail: shuyangw (at) apache.org
Project Devs, mail: dev (at) apisix.apache.org