Submarine workflow

Xun Liu

UPDATE DATE	UPDATE CONTENT
February 25, 2019	Draft system design
February 26, 2019	Add Typical use case
March 9, 2019	Add FlowView
March 10, 2019	Add JOB
March 10, 2019	Add Condition
March 10, 2019	Add Scheduler
March 11, 2019	Add Development Plan
March 18, 2019	Add Development Module

This is a system design to support workflow in submarine. It needs a lot of discussion and modification. You can make comments or modify it directly.

Because there are several interpreters in submarine, it is really helpful for data analysts. A lot of data development is interdependent. For example, the development of machine learning algorithms requires relying on spark to preprocess data, and so on.

submarine should have built-in workflow capabilities. Instead of relying on other software for scheduling, For several reasons:

1. Now we have upgraded from the age of data processing to the age of algorithms, After submarine has own workflow, Will have a complete ecosystem of complete data processing and algorithms operations.
2. Submarine have powerful interactive processing capabilities help algorithms engineers improve productivity and work. submarine should give the algorithm engineer more direct control. Instead of just doing the algorithm, let other teams(or software) handle the workflow.

3. Submarine knows the data processing status better than Azkaban or Airflow. So the built-in workflow will have better performance, Give users better experience and control.
4. The most important thing is to form a closed loop of data.

Typical use case

Especially in machine learning, Because machine learning has a long workload.

A typical example is as follows:

1. First, obtain data from HDFS through spark;
2. Clean and transform data through sparksql;
3. Extract feature data through spark;
4. Write Tensorflow Machine Learnning algorithms through python;
5. Distributed model training through YARN or K8S;
6. Publish the model training and provide online prediction services;
7. Model prediction through streaming processing;
8. Receive incremental data through flink for incremental update of the model;

Therefore, submarine needs provide the ability to support workflow.

Flow view

Flow view includes three web pages: Graph, Configure and Executions.

Flow Graph

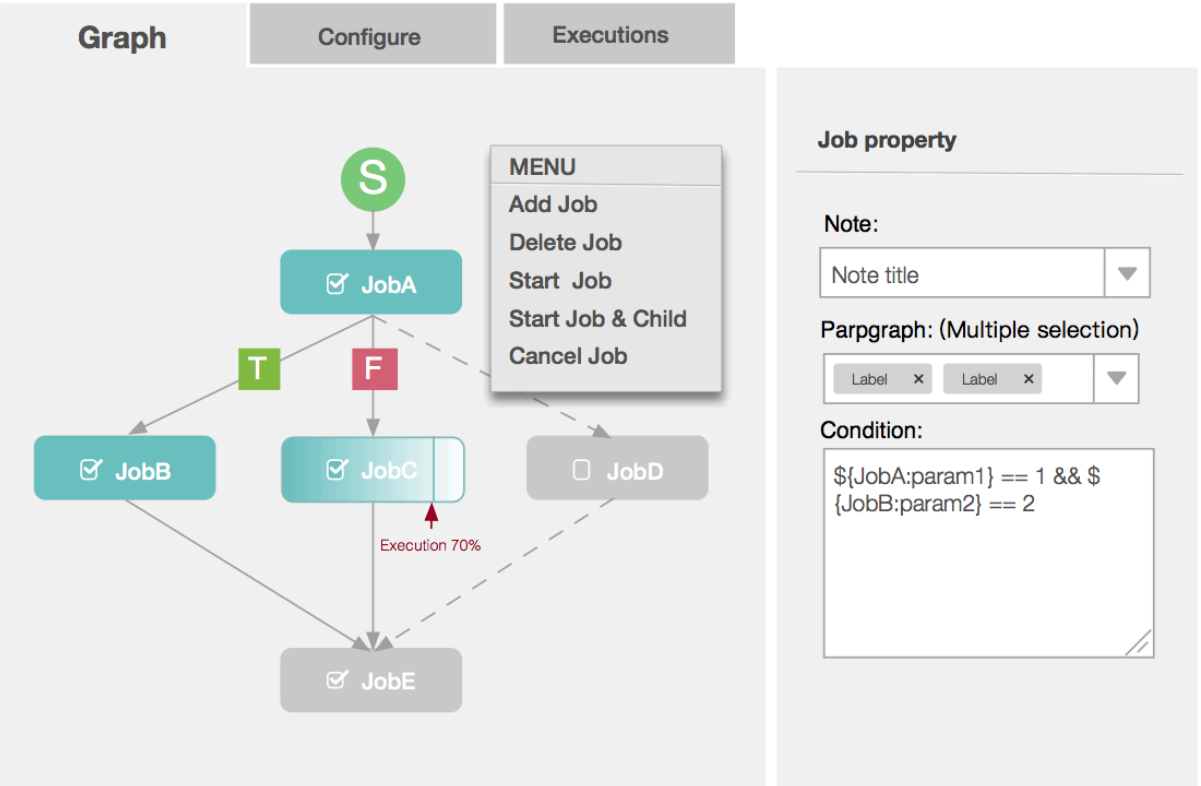
Parpgraph Text area

```
%system.workflow
nodes:
  - name: jobC
    note: nid1(note-title)
      parpgraph: pid1(p1-title), pid2(p2-title)
    retry: 3 # Default configurable
    depends:
      - jobA
      - jobB
    condition: ${JobA:param1} == 1 && ${JobB:param2} == 2

  - name: jobA
    note: nid2
      parpgraph: pid1, pid2

  - name: jobB
    note: n3
      parpgraph: pid1, pid2
```

Output area



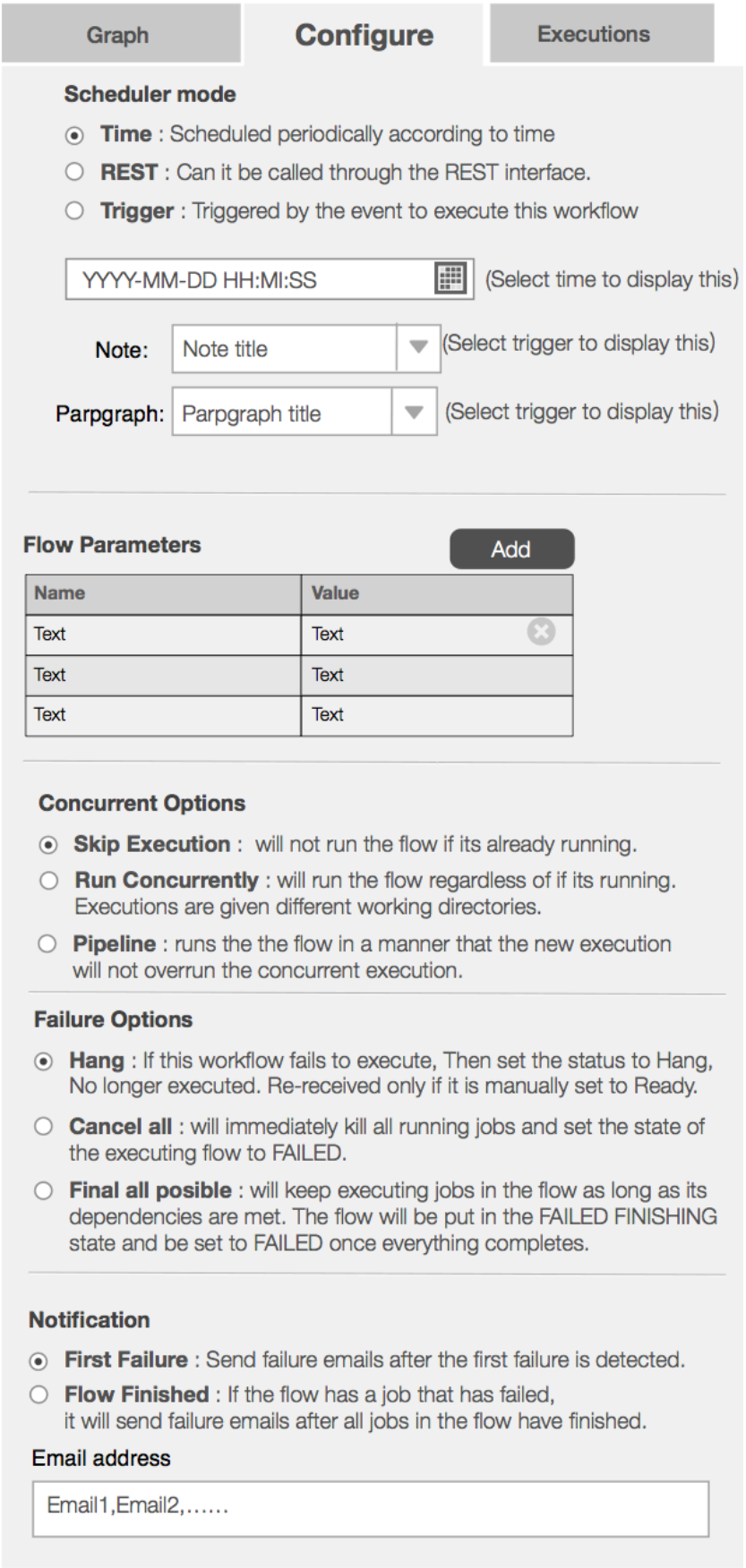
In the graph page, mainly perform the addition of the job and edit the dependencies between the jobs.

1. Add workflow paragraph
    - Add a paragraph in the node, use the `%system.workflow` keyword, after execution will display a `Graph view` page.

    - In the Text editing area of the note interface, you can directly write the workflow content in YAML format.
    - The drog and drop job node can update the text content.
    - The workflow text content can be visualized in the graph view.
    - Workflow Text and graph can be synchronized by variable binding.
2. workflow orchestration
    - There is a `[scheduler]` starting point in the Graph View by default, The `[scheduler]` is the scheduled configuration trigger point
    - Click the `[Add node]` button to add a `Job block` to the `Flow view` .
    - In the `Job block`, you can select note id and parpgraph id through the drop-down box, and note title and paragraph title are displayed in node.
    - There is a checkbox in front of the `Job block`, Indicates whether this node is executed when the workflow is run.
    - In `Job block`, you can use a single arrow segment to connect to other nodes. For example: `A -> B` means B depends on A. After A is executed, B can be executed.
    - The dependencies between nodes can be multiple dependencies, for example: `A -> B, C-> B` means B depends on A, and B also depends on C.
    - `Job block` can be dragged in the workflow view.
3. Job
    - Each Job represents a paragraph in which the user has execute permission
    - Each Job can set the Condition field, the Condition is established, and the Node is executed.
4. Menu
    - Add Job : Add a job node in the workflow.
    - Delete Job : Delete a job node in the workflow.
    - Start Job : Run the current job node in the workflow. A progress bar is displayed on the running job node graph.
    - Start Job & child: Run the job node and all child jobs in the workflow.
    - Cancel Job : Cancel running current job and all child jobs.
5. Job property
    - When you select a Job node, the properties of the Job are displayed. Each Job node can choose to execute the note and paragraph.
    - If only note is selected, this means that the job will execute all the paragraphs in the entire note in turn.
    - If one or more paragraphs are selected, it means that the job will execute the paragraphs in the specified note in turn.
    - Condition : Set the execution conditions of the current job.
    - For example, the execution condition of JOB-b is that JOB-A execution is equal to true.

- For example, the execution condition of JOB-C is that JOB-A execution is equal to false.
- The connection between the jobs will show T or this F Flag.

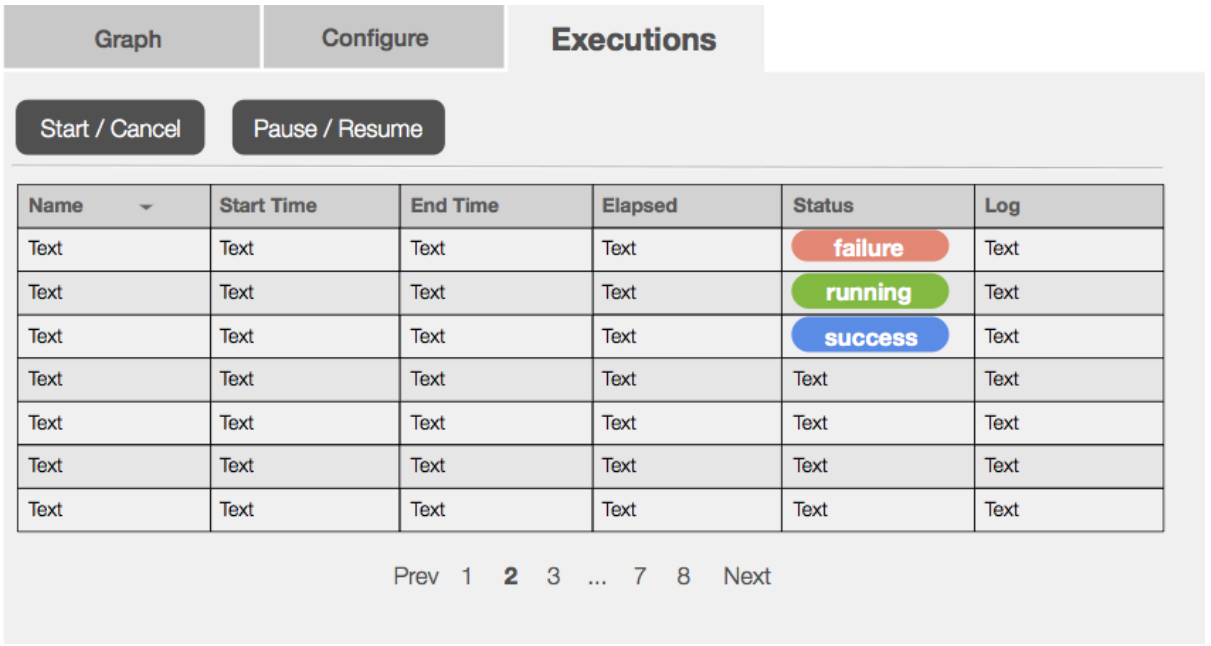
Flow Configure



1. Scheduler mode
    - TimeScheduler : Scheduled periodically according to time, similar to Crontab.
    - RESTScheduler : Can it be called through the REST interface.
    - TriggerScheduler : Triggered by the event to execute this workflow
        - FlinkTrigger：Choose a parpgraph written in `%flink`, In this paragraph, Flink receives external data. When this paragraph receives the data, Will trigger FlinkTrigger to make workflow execute.
    - Other modes ...
2. Flow Parameters（Optional）
3. Allows users to override flow parameters. The flow parameters override the global properties for a job, but not the properties of the job itself.
    - Workflow can run multiple instances at the same time. Each instance has a completely separate environment variable. Workflow execution environment variable release.
    - Environment variables can be used as run parameters and return values for each paragraph throughout the workflow lifecycle.
    - The format is: `%{var_name}=value`, When executing the workflow, The `%{var_name}` in your code for all paragraphs will be replaced by the value. Execute this paragraph.
4. Failure Options
    - **Hang**：If this workflow fails to execute, Then set the status to Hang, No longer executed. Re-received only if it is manually set to Ready.
    - **Cancel All**：will immediately kill all running jobs and set the state of the executing flow to FAILED.
    - **Finish All Possible**： will keep executing jobs in the flow as long as its dependencies are met. The flow will be put in the `FAILED FINISHING` state and be set to FAILED once everything completes.
5. Concurrent Options
6. If the flow execution is invoked while the flow is concurrently executing, several options can be set.
    - Skip Execution : will not run the flow if its already running.
    - Run Concurrently : will run the flow regardless of if its running. Executions are given different working directories.
    - Pipeline : runs the the flow in a manner that the new execution will not overrun the concurrent execution.
7. Notification
8. The notification options allow users to change the flow’s success or failure notification behavior.
    - Notify on Failure

        - First Failure - Send failure emails after the first failure is detected.
        - Flow Finished - If the flow has a job that has failed, it will send failure emails after all jobs in the flow have finished.
    - Email address
    - will use the default notification emails set in the final job in the flow. If overridden, a user can change the email addresses where failure or success emails are sent. The list can be delimited by commas, whitespace or a semi-colon.

Flow Executions



1. Toolbar
    - Cancel - kills all running jobs and fails the flow immediately. The flow state will be KILLED.
    - Pause - prevents new jobs from running. Currently running jobs proceed as usual.
    - Resume - resume a paused execution.
2. Job List

    Display all execution records of this workflow in a list

JOB

Job Dependencies

Jobs can have dependencies on each other. You can use `depends` section to list all the parent jobs. In the below example, after jobA and jobB run successfully, jobC will start to run.

```
nodes:
  - name: jobC
    note: nid1(note-title)
      paragraph: pid1(p1-title), pid2(p2-title)
    retry: 3 # Default configurable
    depends:
      - jobA
      - jobB
    condition: ${JobA:param1} == 1 && ${JobB:param2} == 2

  - name: jobA
    note: nid2
      paragraph: pid1, pid2

  - name: jobB
    note: n3
      paragraph: pid1, pid2
```

When paragraph is not set, it means that all paragraphs in the entire note are executed. paragraph can be set to one or more, separated by commas.

Condition

Conditional workflow feature allows users to specify whether to run certain jobs based on conditions.

Users can run or disable jobs based on runtime parameters like the output from previous jobs. submarine provides users with some predefined macros to specify the condition based on previous jobs' status. With those conditions, users can obtain more flexibility in deciding the job execution logic. For example, if the parent job runs successfully, the child jobA is run. if the parent job fails, Then the child jobB is run. They can implement branching logic in their workflow.

How to define a condition?

A valid condition is a combination of multiple conditions on job runtime parameters and one condition on job status macro. Comparison and logical operators can be used to connect individual condition components.

Supported operators are: `==, !=,  >, >=, <, <=, &&, ||, !`

Condition on job runtime parameter

Variable substitution `${jobName:param}` can be used to define the condition on job runtime parameter. `:` is used to separate the jobName and the parameter. The runtime parameter can be compared with a string or a number in the condition.

Condition on job status macro

This condition will be evaluated on all the parent jobs, i.e. the `depends` section in YAML file.

Currently supported macros:

- all_success (default)
- all_done
- all_failed
- one_success (at least one parent job succeeded)
- one_failed (at least one parent job failed)

Corresponding job status for each macro:

- all_done: `FAILED, KILLED, SUCCEEDED, SKIPPED, FAILED_SUCCEEDED, CANCELLED`
- all_success / one_success: `SUCCEEDED, SKIPPED, FAILED_SUCCEEDED`
- all_failed / one_failed: `FAILED, KILLED, CANCELLED`

Users are not allowed to combine multiple condition on job status macros in one single condition because they might have conflict with each other.

Flow scheduler

Support for starting workflows through Ajax APIs, timers, and triggers. Once all events are ready, the workflow will be triggered.

TimeScheduler

Triggering the execution of a workflow through a periodic timer, Just like corntab.

RestScheduler

Only when the REST call is accepted can it be called by the external system through the REST interface.

TriggerScheduler

Event trigger, This concept enables users to define the events on which the stream depends. Once the user-specified event is ready, the workflow will be triggered.

FlinkTrigger

Write a paragraph using %flink in the note, When this paragraph is executed by flink receiving stream data, The execution of the workflow will be triggered.
This is commonly used to perform, hadoop ETL, machine learning algorithm online detection service, machine learning algorithm model incremental update and other usage scenarios

Development Plan

Development method

Development language

I recommend using JAVA 8, because scala is not familiar to everyone, so you can maintain and extend the workflow function.

Develops Git repositories

https://github.com/apache/submarine.git

Clone Git repositories

```
git clone https://github.com/apache/submarine.git
```

Create develops branch for feature

```
git chckout -b JIRA-NUM
codeing …
test …
git commit …
```

Pull request in Github

If you find code conflicts, execute the following statement and resolve the conflict.

```
git fetch upstream/master
git rebase upstream/master
```

Development Module

Flow graph module (submarine-4075)

The flow graph view is where the user operates a lot. So you need a very good user experience.
description: Flow Graph

Example

1. https://mermaidjs.github.io/
2. https://modeling-languages.com/javascript-drawing-libraries-diagrams/
3. https://www.codeproject.com/Articles/709340/Implementing-a-Flowchart-with-SVG-and-AngularJS
4. https://github.com/ashleydavis/AngularJS-FlowChart
5. https://github.com/bennyrowland/ng-flow-chart
6. https://www.basicprimitives.com/index.php?option=com_content&view=article&id=71&Itemid=116&lang=en
7. https://flowchart.js.org/

These flow library, Have implemented some of the features we need, We need to master the implementation of these library, Transformed into the features we need.

TODO list

1. Workflow YAML file
   We can first test the file with a workflow Yaml file, Perform functional development and testing.
2. The Job block needs to support connections through unidirectional line segments.
   We need to search from the network, is there a closer to the angularJS library we need?
3. The Job block needs to support multiple colors, indicating different states of the JOB.
4. The job block needs to support the progress bar display, indicating the progress of the JOB execution.
5. In the blank of the graph view, click the right mouse button and a pop-up menu is required. The menu items are as follows:

MENU LIST
Add Job
Run workflow
Cancel workflow

6. Select a job block and click the right mouse button. A pop-up menu is required. The menu items are as follows:

MENU LIST
Delete Job
Start Job
Start Job & Child
Stop Job

7. Select a job block and the property items in the job property view need to correctly display the information that has been set.
8. The property item in the job property view The note drop-down box selects the note, and the paragraph drop-down box displays the list of paragraphs in the note.

Flow Engine

The flow engine is where the various tasks are executed in the workflow.
description: Flow Configure

TimeScheuder Module (submarine-4076)

Please add some simple design document in here …

RESTScheduler Module(submarine-XXXX)

Please add some simple design document in here …

TriggerScheduler Module(submarine-XXXX)

Please add some simple design document in here …

Other design modules are being supplemented.