

Meson: Workflow Orchestration for Netflix Recommendations

Netflix Technology Blog [Follow](#)

May 31, 2016 · 8 min read

At Netflix, our goal is to predict what you want to watch before you watch it. To do this, we run a large number of machine learning (ML) workflows every day. In order to support the creation of these workflows and make efficient use of resources, we created Meson.

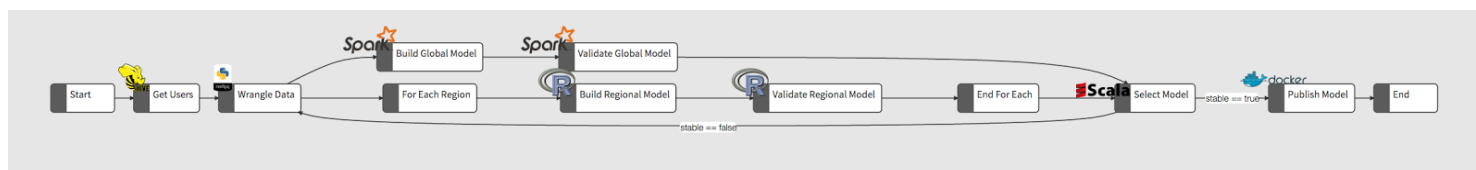
Meson is a general purpose workflow orchestration and scheduling framework that we built to manage ML pipelines that execute workloads across heterogeneous systems. It manages the lifecycle of several ML pipelines that build, train, and validate personalization algorithms that drive video recommendations.

One of the primary goals of Meson is to increase the velocity, reliability, and repeatability of algorithmic experiments while allowing engineers to use the technology of their choice for each of the steps themselves.

Powering Machine Learning Pipelines

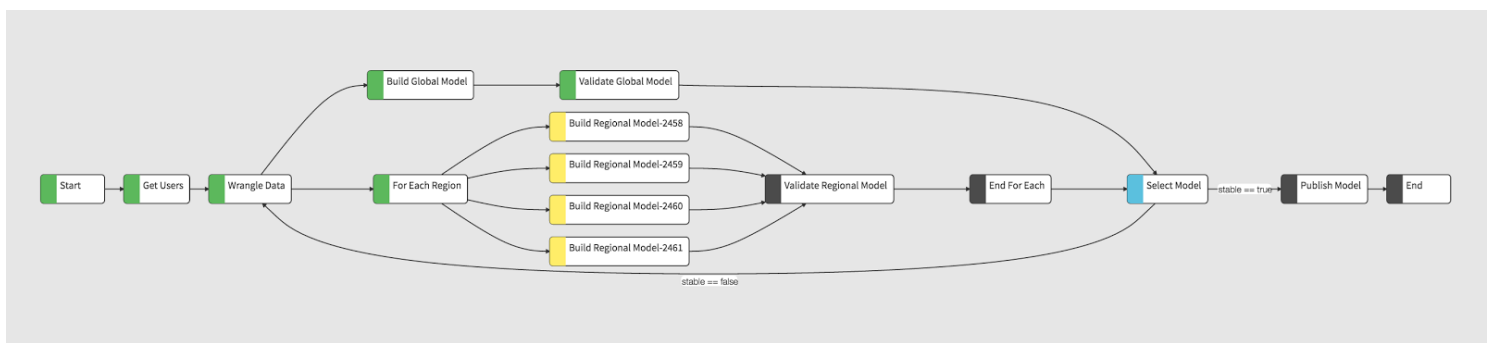
Spark, MLlib, Python, R, and Docker play an important role in several current generation machine learning pipelines within Netflix.

Let's take a look at a typical machine learning pipeline that drives video recommendations and how it is represented and handled in Meson.



The workflow involves:

- Selecting a set of users—This is done via a Hive query to select the cohort for analysis
- Cleansing / preparing the data—A Python script that creates 2 sets of users for ensuring parallel paths
- In the parallel paths, one uses Spark to build and analyze a global model with HDFS as temporary storage.
The other uses R to build region (country) specific models. The number of regions is dynamic based on the cohort selected for analysis. The Build Regional Model and Validate Regional Model steps in the diagram are repeated for each region (country), expanded at runtime and executed with different set of parameters as shown below
- Validation—Scala code that tests for the stability of the models when the two paths converge. In this step we also go back and repeat the whole process if the model is not stable.
- Publish the new model—Fire off a Docker container to publish the new model to be picked up by other production systems



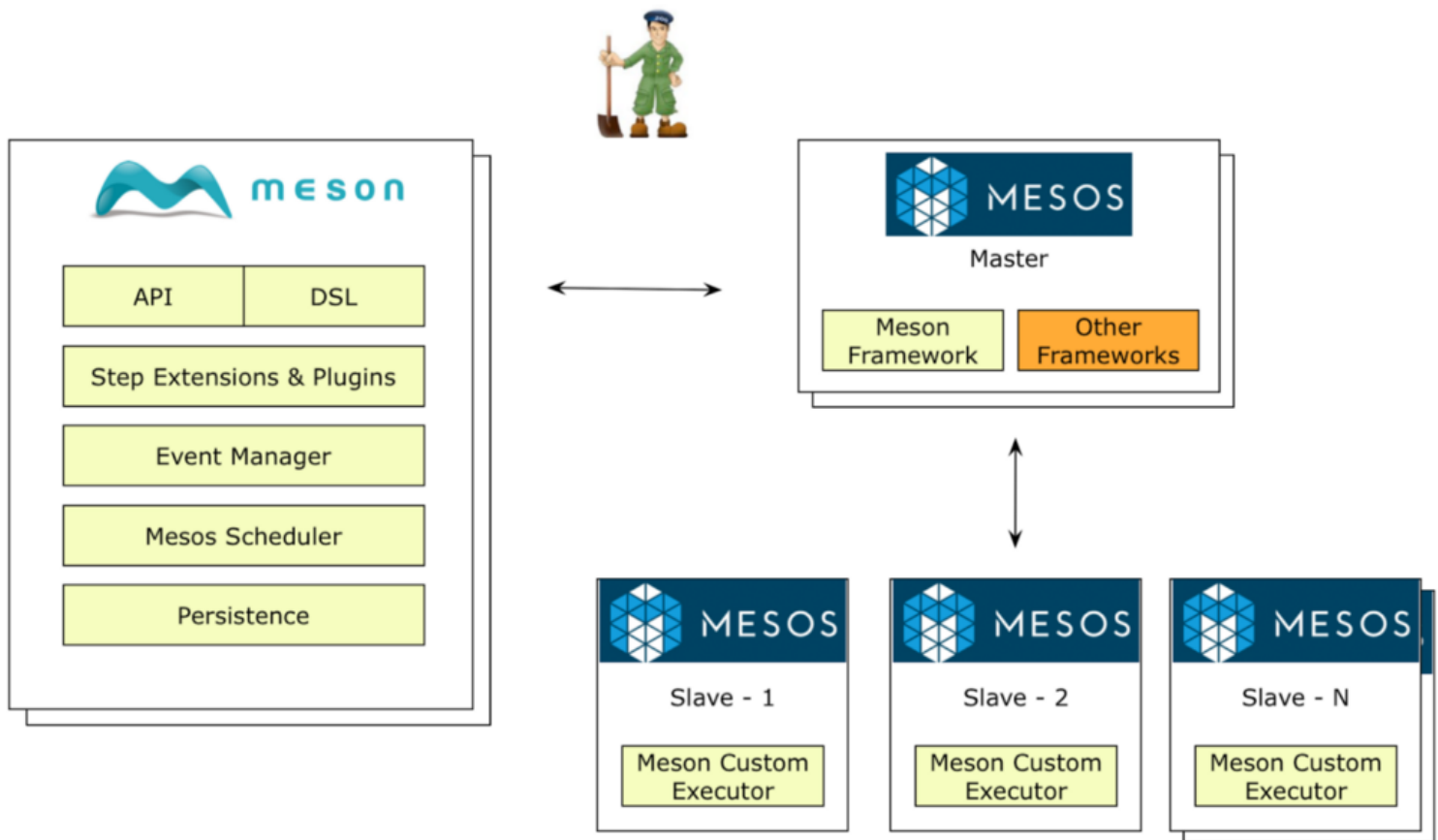
The above picture shows a run in progress for the workflow described above

- The user set selection, and cleansing of the data has been completed as indicated by the steps in green.
- The parallel paths are in progress
- The Spark branch has completed the model generation and the validation

- The for-each branch has kicked off 4 different regional models and all of them are in progress (Yellow)
- The Scala step for model selection is activated (Blue). This indicates that one or more of the incoming branches have completed, but it is still not scheduled for execution because there are incoming branches that have either (a) not started or (b) are in progress
- Runtime context and parameters are passed along the workflow for business decisions

Under the Hood

Let's dive behind the scenes to understand how Meson orchestrates across disparate systems and look at the interplay within different components of the ecosystem. Workflows have a varying set of resource requirements and expectations on total run time. We rely on resource managers like [Apache Mesos](#) to satisfy these requirements. Mesos provides task isolation and excellent abstraction of CPU, memory, storage, and other compute resources. Meson leverages these features to achieve scale and fault tolerance for its tasks.



Meson Scheduler

Meson scheduler, which is registered as a Mesos framework, manages the launch, flow control and runtime of the various workflows. Meson delegates the actual resource scheduling to Mesos. Various requirements including memory and CPU are passed along to Mesos. While we do rely on Mesos for resource scheduling, the scheduler is designed to be pluggable, should one choose to use another framework for resource scheduling.

Once a step is ready to be scheduled, the Meson scheduler chooses the right resource offer from Mesos and ships off the task to the Mesos master.

Meson Executor

The Meson executor is a custom Mesos executor. Writing a custom executor allows us to maintain a communication channel with Meson. This is especially useful for long running tasks where framework

messages can be sent to the Meson scheduler. This also enables us to pass custom data that's richer than just exit codes or status messages.

Once Mesos schedules a Meson task, it launches a Meson executor on a slave after downloading all task dependencies. While the core task is being executed, the executor does housekeeping chores like sending heartbeats, percent complete, status messages etc.

DSL

Meson offers a Scala based DSL that allows for easy authoring of workflows. This makes it very easy for developers to use and create customized workflows. Here is how the aforementioned workflow may be defined using the DSL.

```
val getUsers = Step("Get Users", ...)
val wrangleData = Step("Wrangle Data", ...)
...
val regionSplit = Step("For Each Region", ...)
val regionJoin = Step("End For Each", ...)
val regions = Seq("US", "Canada", "UK_Ireland", "LatAm", ...)
val wf = start -> getUsers -> wrangleData ==> (
  trainGlobalModel -> validateGlobalModel,
  regionSplit ** (reg = regions) --> (trainRegModel,
  validateRegModel) >-- regionJoin
) >== selectModel -> validateModel -> end

// If verbs are preferred over operators
val wf = sequence(start, getUsers, wrangleData) parallel {
  sequence(trainGlobalModel, validateGlobalModel)
  sequence(regionSplit,
    foreach(reg = regions) sequence(trainRegModel,
    validateRegModel) foreach,
    regionJoin)
} parallel sequence(selectModel, validateModel, end)
```

Extension architecture

Meson was built from the ground up to be extensible to make it easy to add custom steps and extensions. Spark Submit Step, Hive Query Step, Netflix specific extensions that allow us to reach out to microservices or other systems like Cassandra are a some examples.

In the above workflow, we built a Netflix specific extension to call out to our Docker execution framework that enables developers to specify the

bare minimum parameters for their Docker images. The extension handles all communications like getting all the status URLs, the log messages and monitoring the state of the Docker process.

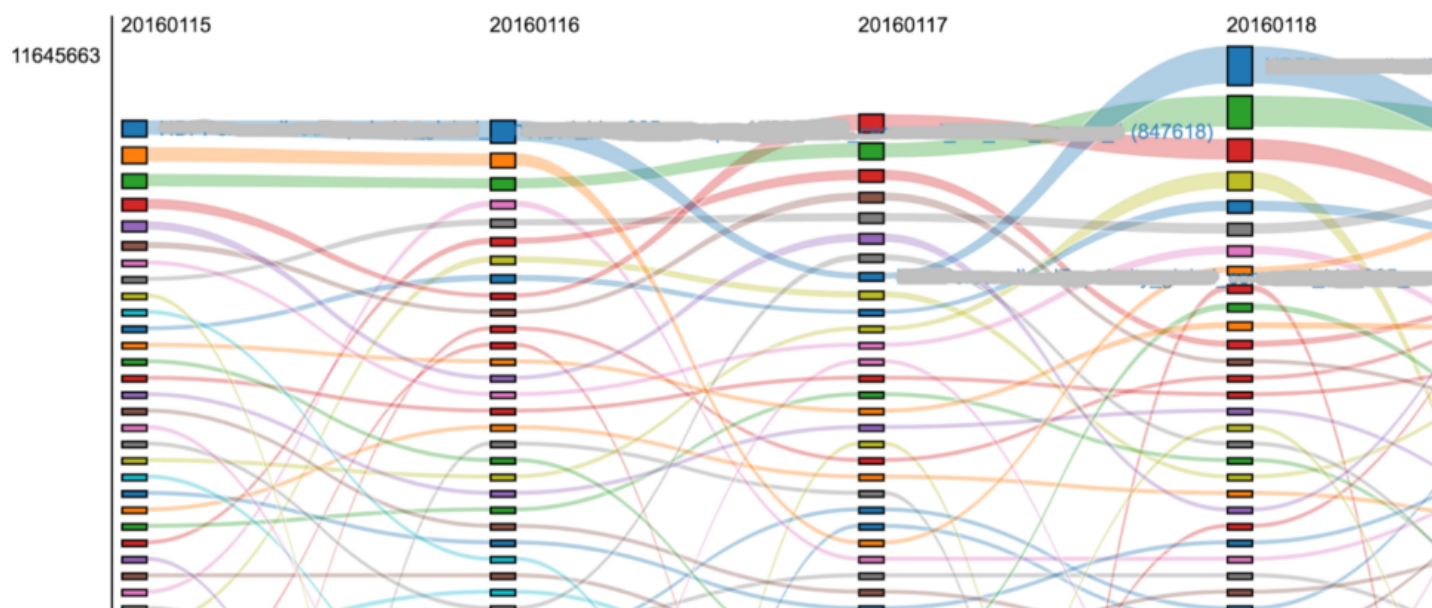
Artifacts

Outputs of steps can be treated as first class citizens within Meson and are stored as Artifacts. Retries of a workflow step can be skipped based on the presence or absence of an artifact id. We can also have custom visualization of artifacts within the Meson UI. For e.g. if we store feature importance as an artifact as part of a pipeline, we can plug in custom visualizations that allow us to compare the past n days of the feature importance.

Feature Importance as Function of Date

Thickness at each node is proportional to the absolute value of the predictionDecrease from that feature on that date.

The total height is proportional to the sum of predictionDecrease from all features on that date.



Mesos Master / Slave

Mesos is used for resource scheduling with Meson registered as the core framework. Meson's custom Mesos executors are deployed across the slaves. These are responsible for downloading all the jars and custom artifacts and send messages / context / heartbeats back to the Meson

scheduler. Spark jobs submitted from Meson share the same Mesos slaves to run the tasks launched by the Spark job.

Native Spark Support

Supporting Spark natively within Meson was a key requirement and goal. The Spark Submit within Meson allows for monitoring of the Spark job progress from within Meson, has the ability to retry failed spark steps or kill Spark jobs that may have gone astray. Meson also supports the ability to target specific Spark versions—thus, supporting innovation for users that want to be on the latest version of Spark.

Supporting Spark in a multi-tenant environment via Meson came with an interesting set of challenges. Workflows have a varying set of resource requirements and expectations on total run time. Meson efficiently utilizes the available resources by matching the resource requirements and SLA expectation to a set of Mesos slaves that have the potential to meet the criteria. This is achieved by setting up labels for groups of Mesos slaves and using the Mesos resource attributes feature to target a job to a set of slaves.

ML Constructs

As adoption increased for Meson, a class of large scale parallelization problems like parameters sweeping, complex bootstraps and cross validation emerged.

Meson offers a simple ‘for-loop’ construct that allows data scientists and researchers to express parameter sweeps allowing them to run tens of thousands of docker containers across the parameter values. Users of this construct can monitor progress across the thousands of tasks in real time, find failed tasks via the UI and have logs streamed back to a single place within Meson making managing such parallel tasks simple.


Screenshots

Here are some screenshots of the Meson UI:

Search for workflows by name, owner, status
Group:

▶ Fictitious ML Pipeline : fictitious_ml_pipeline	Today at 2:50 PM	RUN	INSTANCES	OVERVIEW	DESIGN	DELETE	DASHBOARD
<p>Owner: aarokaisamy@netflix.com</p> <p>Description: -</p> <p># of Runs: -</p> <p>Last Run: May 18th 2016, 2:50:08 pm</p> <p>Last Run Status: COMPLETED</p>							
▶ da [redacted]	Today at 2:00 PM	RUN	INSTANCES	OVERVIEW	DESIGN	DELETE	DASHBOARD
▶ AR [redacted]	Today at 2:00 PM	RUN	INSTANCES	OVERVIEW	DESIGN	DELETE	DASHBOARD
▶ AR [redacted]	Today at 1:55 PM	RUN	INSTANCES	OVERVIEW	DESIGN	DELETE	DASHBOARD
▶ AR [redacted]	Today at 1:55 PM	RUN	INSTANCES	OVERVIEW	DESIGN	DELETE	DASHBOARD
▶ da [redacted]	Today at 1:17 PM	RUN	INSTANCES	OVERVIEW	DESIGN	DELETE	DASHBOARD
▶ La [redacted]	Today at 1:03 PM	RUN	INSTANCES	OVERVIEW	DESIGN	DELETE	DASHBOARD
▶ da [redacted]	Today at 12:59 PM	RUN	INSTANCES	OVERVIEW	DESIGN	DELETE	DASHBOARD
▶ AB [redacted]	Today at 12:15 PM	RUN	INSTANCES	OVERVIEW	DESIGN	DELETE	DASHBOARD




 **meson** Workflows Composer About

Fictitious ML Pipeline:7

INSTANCESOVERVIEWDESIGNDASHBOARD

Add Step



Build Global Model

AdvancedDelete

NameBuild Global Model

StepId4

Description

TYPE

TypeSpark Submit

Jar *

STRINGFetch(IN)

SparkVersion?

STRING(IN)

JobClass *?

STRING(IN)

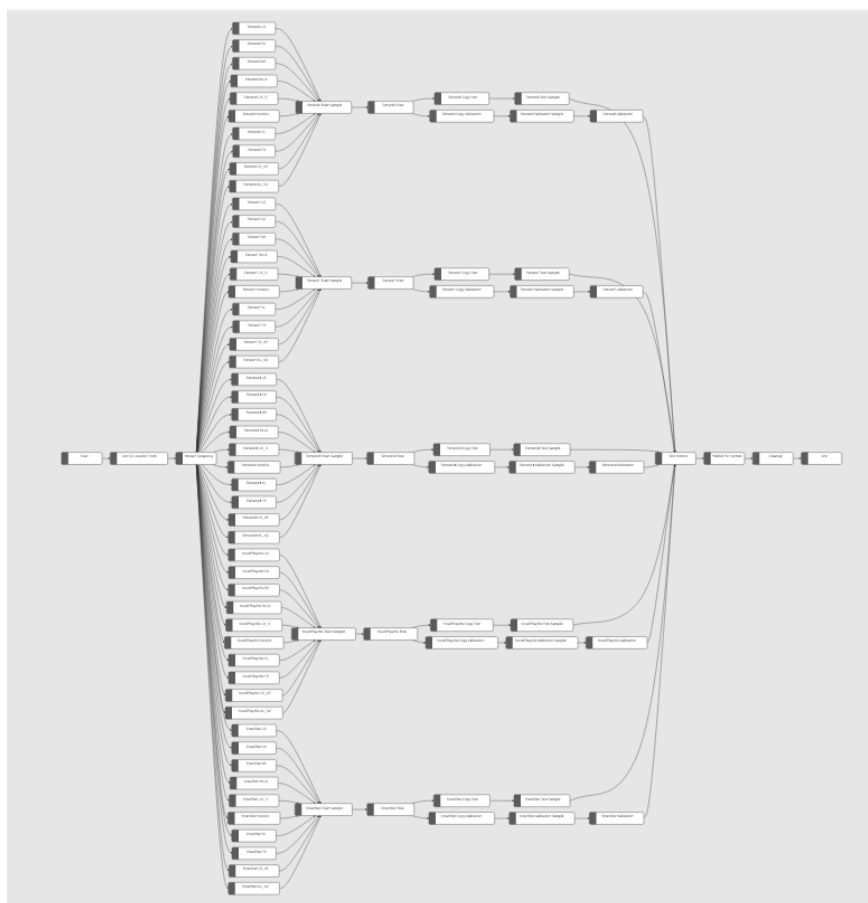
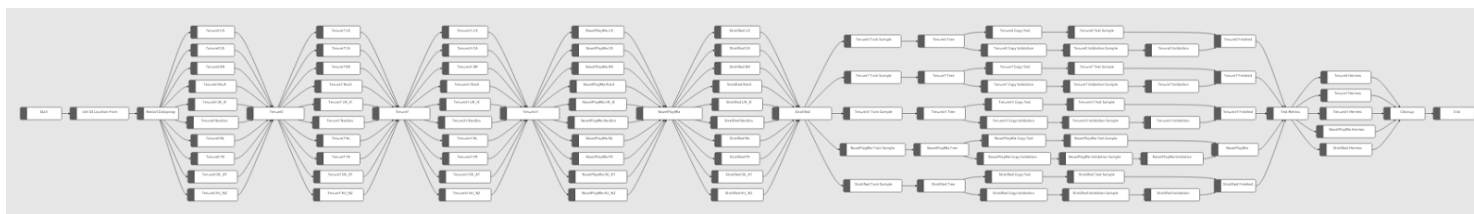
SparkSubmitOptions?

+

JobArgs?

+

And here are couple of interesting workflows in production:



Conclusion

Meson has been powering hundreds of concurrent jobs across multiple ML pipelines for the past year. It has been a catalyst in enabling innovation for our algorithmic teams thus improving overall recommendations to our members.

We plan to open source Meson in the coming months and build a community around it. If you want to help accelerate the pace of innovation and the open source efforts, [join us to help make Meson better](#).

— [Antony Arokiasamy](#), [Kedar Sadekar](#), [Raju Uppalapati](#), [Sathish Sridharan](#), [Prasanna Padmanabhan](#), [Prashanth Raghavan](#), [Faisal Zakaria Siddiqi](#), [Elliot Chow](#) and “a man has no linkedin” (aka Davis Shepherd) for the Meson Team

. . .

Originally published at techblog.netflix.com on May 31, 2016.

