Summary: Nova: Continuous Pig/Hadoop Workflows

This paper describes a workflow manager developed and deployed at Yahoo called Nova, which pushes continually arriving data through graphs of Pig programs executing on Hadoop clusters. (Pig is a structured dataflow language and runtime for the Hadoop map-reduce system.) Nova is like data stream managers in its support for stateful incremental processing, but unlike them in that it deals with data in large batches using disk-based processing. Batched incremental processing is a good fit for a large fraction of Yahoo's data processing use-cases, which deal with continually-arriving data and benefit from incremental algorithms, but do not require ultra-low-latency processing. Despite the success of Pig/Hadoop, it is becoming apparent that a new, higher, layer is needed: a workflow manager that deals with a graph of interconnected Pig Latin programs, with data passed among them in a continuous fashion.

It turns out that this two-layer programming model enables key scheduling and data handling capabilities:

- Continuous processing.
- Independent scheduling.
- Cross-module optimization.
- Manageability features.

A workflow is a directed graph with two kinds of vertices: tasks and channels. Tasks are processing steps. Channels are data containers. Edges connect tasks to channels and channels to tasks; no edge can go from a task to a task or from a channel to a channel.

Data and Update Model

A channel's data is divided into blocks, each of which contains a set of data records or record transformations that have been generated by a single task invocation. Blocks may vary in size from kilobytes to terabytes. For the purpose of workflow management, a block is an atomic unit of data. Blocks also constitute atomic units of processing: a task invocation consumes zero or more input blocks and processes them in their entirety; partial processing of blocks is not permitted. Data blocks are immutable, and so as channels accumulate data blocks their space footprint can grow without bound.

Task/Data Interface

A task must declare, for each incoming edge from a data channel, its consumption mode: one of all or new.

Workflow Programming and Scheduling

Workflows are programmed bottom-up, starting with individual task definitions, and then composing them into workflow fragments called workflowettes. Workflowettes are abstract processing components that are not attached to specific named data channels instead they have ports to which input and output channels may be connected. Channels are defined via a separate registration process from workflowettes. Once workflowettes and channels have been registered, a third process, called binding, attaches channels to the input and output ports of a workflowette, resulting in a bound workflowette. The last step is for the user to communicate scheduling requirements, by associating one or more triggers with a bound workflowette. There are three basic types of triggers:

- Data-based triggers. Execute whenever new data arrives on a particular channel (typically a channel bound to one of the workflowettes input ports).

- Time-based triggers. Execute periodically, every t time units.
- Cascade triggers. Execute whenever the execution of another bound workflowette reaches a certain status (e.g. launched, completed successfully, failed).

TYING THE MODEL TO PIG/HADOOP

As mentioned earlier, Nova implements the data and computation model on top of Pig/Hadoop. The content of each data block resides in an HDFS file (or perhaps a directory of \part files, which is the unit of data output by a Hadoop map-reduce job). Nova maintains the mapping from data blocks to HDFS files/directories in its metadata. HDFS file names are hidden from users, and Nova generates unique _le names by incrementing a global counter, e.g. /nova/block_0, /nova/block_1, etc. The notion of a channel exists solely in Nova's metadata.

For example, the Pig Latin code for the \template tagging" task in our news de-duplication workflow (Figure 1) might look like this:

```
register news_processing_udfs.jar;
articles = $RAW_ARTICLES;
templates = $TEMPLATES;
joined = join articles by site, templates by site;
tagged = foreach joined generate TagTemplates(*);
store tagged into $TAGGED_ARTICLES;
```

where TagTemplates() is a user-defined function whose code resides in the JAR _le imported in the first line.

WORKFLOW MANAGER ARCHITECTURE

Most of Nova's modules are part of a Nova server instance process. The modules in a server instance are stateless; they keep their state externally, in a metadata database. Nova supports two types of clients. Human clients have access to a command-line interface and a web interface. Web-service clients interact with Nova via a SOAP web-services API. At Yahoo, Nova is deployed as part of a larger software environment that includes systems for data on-boarding, data storage and processing (Nova), and data serving. The onboarding and serving systems interact with Nova using web services.

The core Nova server modules are:

- User interface: This module provides API methods for registering (and deregistering) channels and workflowettes, and for binding workflowettes to channels to produce bound workflowettes.

- Process manager: This module keeps track of registered workflowettes and bound workflowettes.

- Data manager: The data manager maintains a list of blocks associated with each channel, as well as the mapping from blocks to underlying HDFS files/directories. It also maintains the task input cursors.

- Process optimizer: This is a placeholder for various performance optimizations on workflowette execution.

- Process executor: This module forwards workflowette execution requests to Oozie (which in turn runs the constituent Pig jobs, which in turn spawns Hadoop map reduce jobs), tracks their status, and reports the status back to the Nova process manager.

- Trigger manager: This module runs in its own thread, and fires triggers.

Cross-cluster Replication

Nova includes a module, called data & metadata replicator, used to replicate Nova data and state onto other clusters, generally running in other data centers. This cross-data-center replication is asynchronous and one-way, i.e. the recipient is not an active Nova instance, but rather a (slightly lagging) \stand-by" instance. Cross-data-center replication is used for two purposes: (1) fail-over in case the primary data center becomes unavailable; (2) migration to a new data center.

SUMMARY AND FUTURE WORK

We have described a workflow manager called Nova that supports continuous, large-scale data processing on top of Pig/Hadoop. Nova is a relatively new system, and leaves open many avenues for future work such as:

- Arbitrary workflow nesting, rather than the current, rigidly tiered model.
- Better schema migration support.
- Investigating H-Base, or a similar BigTable inspired storage system, as the underlying storage and merging infrastructure for data channels.
- Scheduling data compaction automatically, based on some optimization formulation such as minimizing total cost while bounding wasted space.
- Automatically rewriting non-incremental workflows to execute in an incremental fashion, and perhaps even dynamically switching between incremental and non-incremental execution based on input data sizes and other factors.
- Keeping track of temporal data inconsistencies that result from delayed or unsynchronized execution of workflow component and perhaps even incorporating consistency goals/bounds into a workflow scheduler.
- Dealing with (minor) changes in workflow structure and task logic using incremental processing strategies.