Summary: Building a High-level Dataflow System on top of MapReduce: The Pig Experience

Pig is a high-level dataflow system that aims at a sweet spot between SQL and Map-Reduce. Pig offers SQL-style high-level data manipulation constructs, which can be assembled in an explicit dataflow and interleaved with custom Map- and Reduce-style functions or executables. Consequently, users end up stitching together Map-Reduce dataflows by hand, hacking multi-input flows, and repeatedly implementing standard operations inside black-box functions. These practices slow down data analysis, introduce mistakes, make data processing programs difficult to read, and impede automated optimization. Pig seems to give the necessary parallel programming constructs (FOREACH, FLATTEN, COGROUP etc.) and also give sufficient control back to the programmer (which a purely declarative approach like [SQL on top of Map-Reduce] doesn't)." Pig compiles these dataflow programs, which are written in a language called Pig Latin, into sets of Hadoop Map-Reduce jobs, and coordinates their execution. The Pig system takes a Pig Latin program as input, compiles it into one or more Map-Reduce jobs, and then executes those jobs on a given Hadoop cluster.

```
urls = LOAD `dataset' AS (url, category, pagerank);

groups = GROUP urls BY category;

bigGroups = FILTER groups BY COUNT(urls)>1000000;

result = FOREACH bigGroups GENERATE group, top10(urls);

STORE result INTO `myOutput';
```

Pig allows three modes of user interaction:

1. **Interactive mode:** In this mode, the user is presented with an interactive shell (called Grunt), which accepts Pig commands.

2**. Batch mode:** In this mode, a user submits a prewritten script containing a series of Pig commands, typically ending with STORE.

3. **Embedded mode:** Pig is also provided as a Java library allowing Pig Latin commands to be submitted via method invocations from a Java program.

Pig has a nested data model, thus supporting complex, non-normalized data. Standard scalar types of int, long, double, and chararray (string) are supported. Pig supports three complex types: map, tuple, and bag. Map is an associative array, where the key is a chararray and the value is of any type. It uses tabs to delimit data values and carriage returns to delimit tuples and left/right delimiters like {} to encode nested complex types. Data stored in Hadoop may or may not have schema information stored with it. For this reason, Pig supports three options for declaring the data types of fields.

1- The first option is that no data types are declared. In this case the default is to treat all fields as bytearray.

2-- The second option for declaring types in Pig is to provide them explicitly as part of the AS clause during the LOAD.

3-- The third option for declaring types is for the load function itself to provide the schema information, which accommodates self-describing data formats such as JSON.

Lazy Conversion of Types: When Pig does need to cast a bytearray to another type because the program applies a type specific operator, it delays that cast to the point where it is actually necessary.

Pig currently performs a limited suite of logical optimizations to transform the logical plan, before the compilation into a Map-Reduce plan. The map stage processes the raw input data, one data item at a time, and produces a stream of data items annotated with keys. A subsequent local sort stage orders the data produced by each machine's map stage by key. The locally-ordered data is then passed to an (optional) combiner stage for partial aggregation by key. The shuffle stage then redistributes data among machines to achieve a global organization of data by key (e.g. globally hashed or ordered). All data received at a particular machine is combined into a single ordered stream in the merge stage. If the number of incoming streams is large (relative to a configured threshold), a multi-pass merge operation is employed; if applicable, the combiner is invoked after each intermediate merge step. Lastly, a reduce stage processes the data associated with each key in turn, often performing some sort of aggregation.

Pig uses Java's MemoryPoolMXBean class to be notified of a low memory situation. This class allows the program to register a handler which gets notified when a configurable memory threshold is exceeded.

The Pig features included in the Pig Mix benchmark were selected by developers, based on inspecting Pig programs collected from users. The aim was to include features that are representative of typical use cases. Some key features included in Pig Mix at present include:

- Joins (distributed hash-sort-merge join as well as fragment-replicate join).
- Grouping and co-grouping, including GROUP ALL in which all tuples are collected into a single group (e.g. to enable summing of values across all tuples).
- Distinct aggregation.
- Nested statements inside a FOREACH expression.
- ORDER BY, both on single and multiple fields.
- UNION, DISTINCT and SPLIT.
- Data containing nested types.

FUTUREWORK

There are a number of areas under consideration for future development:

- Query optimization.
- Non-Java UDFs.
- SQL interface.
- Grouping and joining of pre-partitioned/sorted data.
- Skew handling.
- Code generation.