Pig Journal

Pig Journal

This document is a successor to the ProposedRoadMap. Rather than simply propose the work going forward for Pig, it also summarizes work done in the past (back
to Pig moving from a research project at Yahoo Labs to being a part of the Yahoo grid team, which was approximately the time Pig was first released to open
source), current work, and proposed future work. Note that proposed future work is exactly that, **proposed**. There is no guarantee that it will be done, and the
project is still open to input on whether and when such work should be done.

Completed Work

The following table contains a list of features that have been completed, as of Pig 0.9

Feature	Available in Release	Comments
Describe Schema	0.1	
Explain Plan	0.1	
Add log4j to Pig Latin	0.1	
Parameterized Queries	0.1	
Streaming	0.1	
Documentation	0.2	Docs are never really done of course, but Pig now has a setup document, tutorial, Pig Latin users and reference guides, a cookbook, a UDF writers guide, and API javadocs.
Early error detection and failure	0.2	When this was originally added to the !ProposedRoadMap it referred to being able to do type checking and other basic semantic checks.
Remove automatic string encoding	0.2	
Add ORDER BY DESC	0.2	
Add LIMIT	0.2	
Add support for NULL values	0.2	
Types beyond String	0.2	
Multiquery support	0.3	
Add skewed join	0.4	
Add merge join	0.4	
Add Zebra as contrib project	0.4	
Support Hadoop 0.20	0.5	
Improved Sampling	0.6	There is still room for improvement for order by sampling
Change bags to spill after reaching fixed size	0.6	Also created bag backed by Hadoop iterator for single UDF cases
Add Accumulator interface for UDFs	0.6	
Switch local mode to Hadoop local mode	0.6	
Outer join for default, fragment-replicate, skewed	0.6	
Make configuration available to UDFs	0.6	
Load Store Redesign	0.7	

Pig Mix 2.0	0.7	
Rewrite Logical Optimizer	0.8	
Cleanup of javadocs	0.8	
UDFs in scripting languages	0.8	
Ability to specify a custom partitioner	0.8	
Pig usage stats collection	0.8	
Make Pig available via Maven	0.8	
Standard UDFs Pig Should Provide	0.8	
Add Scalars To Pig Latin	0.8	
Run Map Reduce Jobs Directly From Pig	0.8	
Make Illustrate Work	0.9	
Better Parser and Scanner Technology	0.9	
Clarify Pig Latin Semantics	0.9	
Extending Pig to Include Branching, Looping, and Functions	0.9	
Typed maps	0.9	
Make Pig work with hadoop 23	0.10	
boolean datatype	0.10	
nested cross/foreach	0.10	
JRuby udf	0.10	
limit by expression	0.10	
split default destination	0.10	
tuple/bag/map syntax support	0.10	
map-side aggregation	0.10	
Rank operator	0.11	
Cube, Rollup operator	0.11	
Datetime datatype	0.11	
Groovy UDFs	0.11	
Schema-based Tuples	0.11	
ASSERT operator	0.12	
Streaming UDF	0.12	
New AvroStorage	0.12	
IN/CASE operator	0.12	
BigInteger/BigDecimal datatype	0.12	
Native Windows OS support	0.12	
Pluggable execution engines	0.13	
auto-local mode	0.13	
fetch optimization	0.13	

user level jar cache	0.13	
Pig command blacklist and whitelist	0.13	
Pig on Tez	0.14	
OrcStorage	0.14	
predicate push down	0.14	
constant calculation	0.14	
UDF auto-ship jars	0.14	
Tez Grace auto-parallelism	0.15	
Support Hive UDF	0.15	

Work in Progress

This covers work that is currently being done. For each entry the main JIRA for the work is referenced.

Feature	JIRA	Comments
Move GruntParser to Antlr	PIG-2597	
LIMIT inside nested foreach should have combiner optimization	PIG-4536	
Optimize the case of Order by + Limit in nested foreach	PIG-4449	
Support for Tez speculative execution	PIG-4411	
Jython algebraic udfs	PIG-1804	
local scope set statement	PIG-4424	
Error handling		
Pig on Spark	PIG-4059	

Proposed Future Work

Work that the Pig project proposes to do in the future is further broken into three categories:

1. Work that we are agreed needs to be done, and also the approach to the work is generally agreed upon, but we have not gotten to it yet
2. Work that we are agreed needs to be done, but the approach is not yet clear or there is not general agreement as to which approach is best
3. Experimental, which includes features that may not yet be agreed upon or where we do not know if they will be beneficial or not

For each of these proposed features, a brief description is given plus the following information:

- Category - what type of feature is this; categories include:
    - Performance
    - Usability
    - Integration with other Hadoop Subprojects
    - New Functionality
    - Development - that is proposed features that will be of interest to development but may not make noticeable changes for users
- Dependencies - any other feature or proposed feature that this depends on
- References - any relevant JIRAs, wiki pages, white papers, etc.
- Estimated Development Effort - a **guess** at how long this will take, with the following three categories:
    - small - less than 1 person month
    - medium - 1-3 person months
    - large - more than 3 person months

Within each subsection order is alphabetical and does not imply priority.

Agreed Work, Agreed Approach

Combiner Not Used with Limit or Filter

Pig Scripts that have a foreach with a nested limit or filter do not use the combiner even when they could. Not all filters can use the combiner, but in some cases
they can. I think all limits could at least apply the limit in the combiner, though the UDF itself may only be executed in the reducer.

Category: Performance

Dependency: Map Reduce Optimizer

References:

Estimated Development Effort: small

Error Handling

Pig's error handling is not good. There are two parts to this problem. First, users frequently complain that the error messages give no useful information as to what
the problem is or how to fix it. Work needs to be done to assure that error messages are meaningful to users and that an error resolution guide exists to help
users understand what an error message means and actions they should take to remedy the situation. Second, Map Reduce does not reliably return error messages
that occur during Map Reduce execution. Since the error is returned to Pig as one long Java String (rather than as an Exception object) Pig is left to attempt
to decipher which portion of the error message is meaningful to the user and which is not. Pig is not always successful in this attempt. Map Reduce needs to
return the error to Pig in an object format so it can more easily determine the relevant part of the error.

Category: Usability

Dependency: Exceptions from Map Reduce as Exception, not String; Standardize on Parser and Scanner Technology because many of the bad error messages come from
the parser

References:

Estimated Development Effort: medium

Outer Join for Merge Join

Merge join is the last join type to not support outer join. Right outer join is doable in the current infrastructure. Left and full outer join will require an index (either in the data or built by a preliminary MR job, just as index required of the right side is now).

Category: New Functionality

Dependency:

References:

Estimated Development Effort: small

Extend Load and Store Functions to be in Scripting Languages

In 0.8 we added the ability to write EvalFuncs and FilterFuncs in scripting languages. We should extend this capability to load and store functions.

Category: New Functionality

Dependency:

References: PIG-1777

Estimated Development Effort: small

Extend UDFs in Scripting Languages to Allow Algebraic and Accumulator

In 0.8 we added the ability to write EvalFuncs and FilterFuncs in scripting languages. However, these cannot use the Accumulator or Algebraic interfaces, both of which can provide significant performance and scalability benefits.

Category: New Functionality

Dependency:

References: PIG-1804

Estimated Development Effort: medium

Mavenization

Switch Pig build system from ant to maven. Would like to modularize Pig so we will have module pig-core, mr, tez. Also need to switch the build system for e2e tests

Category: Build

Dependency:

References: PIG-1804

Estimated Development Effort: medium

Summary query

For some file format such as Orc, we have stats build into the data file, so we can get the data summary such as min/max/sum/avg quickly without scanning the data,

Category: Performance

Dependency:

References:

Estimated Development Effort: small

Replicated cross

Pig should be able to do map side cross. Currently, user can emulate it with replicated join. But it would be better to add native support

Category: Performance

Dependency:

References:

Estimated Development Effort: small

PMML support

Able to consume PMML model and score input data.

Category: New functionality

Dependency:

References: **https://github.com/Netflix/Surus**

Estimated Development Effort: small

Performance benchmark

Adding TPC/DI, TPC/DS query into Pig.

Category: Performance

Dependency:

References:

Estimated Development Effort: medium

Staged replicated join

Currently for replicated join, right table must fit memory. We can borrow idea from Hive staged map join to spill right table to disk if not fit, and process the overflow in map cleanup.

Category: Performance

Dependency:

References:

Estimated Development Effort: medium

Agreed Work, Unknown Approach

Make Use of HBase

Pig can do bulk reads and writes from HBase. But it cannot use HBase in operators like a hash join. We need operators that make use of HBase where it makes sense. Also, we may need to provide support so that UDFs can efficiently access HBase themselves.

Category: Integration, Performance

Dependency:

References:

Estimated Development Effort: medium

Runtime Optimizations

Currently Pig does all of its optimizations up front before beginning any execution. In a multi-job pipeline information will be learned in initial jobs that could be used in later jobs to make optimization decisions. For example, a join later in the pipeline may turn out to have inputs of a size such that fragment replicate makes sense as a join strategy. Being able to rewrite the plan midway through the execution will provide the ability to optimize for these types of situations.

Category: Performance

Dependency:

References:

Estimated Development Effort: large

Support Append in Pig

Appending to HDFS files is supported in Hadoop 0.21. None of Pig's standard load functions support append. We need to decide if append is added to the language itself (is there an APPEND modifier to the STORE command?) or if each store function needs to decide how to indicate or allow appending on its own. !PigStorage
should support append as users are likely to want it.

Category: New Functionality

Dependency: Hadoop 0.21 or later

References:

Estimated Development Effort: small

IDE for Pig

!PigPen was developed and released for Pig with 0.2. However, it has not been kept up to date. Users have consistently expressed interest
in an IDE for Pig. Ideally this would also include tools for writing UDFs, not just Pig Latin scripts. One option is to bring !PigPen up to date and maintain it. Another option is to build a browser based IDE. Some have suggested that this would be better than an Eclipse based one.

Category: New Functionality

Dependency:

References:

Estimated Development Effort: large and ongoing

Vectorization

Pig shall process operator in a batch manner. One possibility is to use Hive vectorization library.

Category: Performance

Dependency:

References:

Estimated Development Effort: large

Sparse tuple support

Implement sparse tuple and expose to Pig syntax to make the memory footprint small for sparse data.

Category: New Functionality

Dependency:

References:

Estimated Development Effort: small

Experimental

Add List Datatype

Pig has tuples (roughly equivalent to structs or records in many languages). Bags, which are roughly equivalent to lists, have the restriction that they can only
contain tuples. This means that users have modeled lists as bags of tuples of a single element. This is confusing to users and wastes memory. Changing bags to
take any type would be very disruptive, since much existing Pig code is built around the assumption that bags only contain tuples. Additionally bags contain extensive functionality to handle memory management, spilling, etc. A list type need not offer all these features. Therefore the best route to adding this functionality may be to add a list type to Pig Latin.

Category: New Feature

Dependency:

References:

Estimated Development Effort: Medium

Automated Hadoop Tuning

Hadoop has many configuration parameters that can affect the latency and scalability of a job. For different types of jobs, different configurations will yield optimal results. For example, a job with no memory intensive operations in the map phase but with a combine phase will want to set Hadoop's io.sort.mb quite
high, to minimize the number of spills from the map. But a job with a memory intensive operation in the map and no combine phase will want to set io.sort.mb low
to allocate more memory to the memory intensive operator and less to the combiner. Adding this feature will greatly increase the utility of Pig for Hadoop users,
as it will free them from needing to understand Hadoop well enough to tune it themselves for their particular jobs.

Category: Usability

Dependency:

References:

Estimated Development Effort: large

Generated Execution Code

Currently Pig has a set of Physical Operators that contain the logic to execute Pig programs. To execute a given program a pipeline of these physical operators
is constructed, split into Map Reduce jobs, and shipped to Hadoop. We need to investigate changing the physical operators to instead understand how to generate
Java code. Pig can then generate Java code, compile it, and pass that to Hadoop. Some sources we have read suggest that a significant performance improvement
could be gained. Also this would allow Pig to use pre-compiled tuples specific to a given script, which should improve memory usage and performance. This would
make the code more complex to develop and maintain. It would also make is more complex to install as it would require a Java compiler as part of the Pig deployment.

Category: New Functionality

Dependency:

References:

Estimated Development Effort: large

Integration with Oozie

It has been suggested that Pig should be able to generate Oozie jobs in addition to (or perhaps instead of) directly generating Map Reduce jobs. It has also been
suggested that Pig Latin should include commands to control Oozie, thus allowing Pig Latin to be a language for workflows on Hadoop. The Pig team needs to consider
these options and decide how Pig and Oozie should be integrated.

Category: Integration

Dependency:

References:

Estimated Development Effort: depends on what type of integration is chosen

Physical Operators Take List of Tuples

Currently tuples are passed one at a time between physical operators. Moving all the way through the pipeline for each tuple causes a lot of context switching. We
need to investigate batching tuples and passing a list between operators instead. In the map phase this would be likely to help, though we would want to re-implement our map implementation to take control from Map Reduce so we get multiple records at a time. In reduce it is less clear, since tuples in reduce can
tend to be large (since they already contain the group) and thus batching them may cause memory problems.

Category: Performance

Dependency:

References: PIG-688

Estimated Development Effort: medium (involves rewrite of many physical operators)