PennyToolLibrary

Penny Tool Library

Here are some tools that we have already developed using the Penny framework. Many of them are somewhat rudimentary and could be improved (volunteers accepted!). Don't see the tool you were hoping to find? Build your own – it's easy! – see PennyCreateYourOwnTool.

name	description	how to run	limitations
crash investigator	Determine which record(s) might be causing your pig script to crash.	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.ci.Main <pig_script> (pig_script = your pig script, e.g. foo.pig)	Narrows it down to a small set of records, but can't pinpoint the exact record due to pipeline and partition parallelism.
row-level integrity alerts	Throw an alert if a particular field of a particular intermediate data record contains a NULL. (Should be easy to generalize to arbitrary predicates by supplying a code fragment that returns a boolean pass/no-pass decision – anyone want to volunteer?)	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.ri.Main <pig_script> <alias> <field#> (alias = pig script alias you want to monitor, field# = field # to check for NULLs)	
table-level integrity alerts	Throw an alert if a particular intermediate table (i.e. the set of records passing between steps i and j) is too small. (Any volunteers to generalize this to general checks? Again, shouldn't be very hard.)	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.ti.Main <pig_script> <alias> <minimum size>	
data samples	Print a few records from each intermediate data set, as the pig script is running – allows you to get a feel for the transformations being performed, and do some basic sanity checks.	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.ds.Main <pig_script>	
data histograms	Print a histogram of a particular field of a particular intermediate data set.	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.dh.Main <pig_script> <alias> <field#> <min_val> <max_val> <bucket_size>	
forward tracing	Trace a particular record as it flows through the pig script and gets transformed by the various steps.	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.ft.Main <pig_script> <alias> <field#> <value> (alias = alias from which start forward tracing; field# = field to inspect to decide when to trace a record; value = value that triggers tracing – e.g. if I set alias=foo, field#=2, value=bar it will trace all records emitted by script alias "foo" that have "bar" in field #2	Script must use positional notation for group-by keys (i.e. instead of "group X by url" you have to write "group X by $2". Currently does not support scripts that use JOIN or ORDER – waiting on parsing support from Pig for those.
backward tracing	Trace a particular record backward through the pig script, to find out where it came from (i.e. trace its "lineage" or "provenance").	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.bt.Main query_analysis.pig <alias> <record> (alias = alias of record to trace; record = record to trace, in quotes, e.g. "(texas,berets)")	Script must use positional notation for group-by keys (i.e. instead of "group X by url" you have to write "group X by $2". Currently does not support scripts that use JOIN or ORDER – waiting on parsing support from Pig for those. Will not perform well on large data sets. Can be improved by implementing an initial "weak inversion" analysis phase – T.B.D.
golden logic testing	Compare a "golden" piece of logic (one that you're pretty sure is correct) against the logic performed by Pig, to see if there might be a bug.	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.gl.Main <pig_script> <alias> <sample_rate> <golden_logic_class> (sample_rate = what fraction of records to check; golden_logic_class = your golden logic class, which must implement the org.apache.pig.penny.apps.gl.GoldenLogic interface)	Script must use positional notation for group-by keys (i.e. instead of "group X by url" you have to write "group X by $2". Currently does not support scripts that use JOIN or ORDER – waiting on parsing support from Pig for those.
latency alerts	Throw an alert if a given record takes much longer to process than the average record.	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.la.Main <pig_script>	
latency profiling	Trace records as they flow through the pig steps, and see how long it takes the record to reach each step.	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.lp.Main <pig_script>	Script must use positional notation for group-by keys (i.e. instead of "group X by url" you have to write "group X by $2". Currently does not support scripts that use JOIN or ORDER – waiting on parsing support from Pig for those.
overhead profiling	Determine how much time is spent on each step in your pig script.	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.op.Main <pig_script>	Currently only works on pig scripts that have a linear chain structure (no joins or splits).
trial runs	Run your pig script on a small sample of the input data. Use this the first time you run a new pig scripts to catch certain bugs quickly.	java -cp penny.jar:pig.jar org.apache.pig.penny.apps.tr.Main <pig_script>	

As your script runs, tasks will communicate back to Penny. For this to work, you will need to open port 33335 to your cluster on the machine where you ran Penny.