White Paper

HPCC Systems:
Performing in the Pig Pen

LexisNexis®

Performing in the Pig-Pen

The ECL performance advantage over Hadoop/PIG comes from two different areas: from the enhanced linguistic expressivity of ECL and from the advanced, mature engineering that underpins the platform. Clearly, in real use, an operating ECL system will benefit greatly from both features. Unfortunately it is almost impossible to measure the lift that comes from expressivity. 'Lift from expressivity' essentially means you will program better because the language is better; but for any given benchmark you will spend so much time focusing on the code that the lift is lost!

It can be equally difficult to fairly measure the combination of expressivity and engineering. By coding any benchmark 'well' it is often possible to get orders of magnitude improvement by avoiding some particularly nasty operation; but then one is open to the charge of cheating.

It is however possible to objectively measure something very different – just how well would ECL perform if it was programmed **as** you would program in PIG? In fact, if you gave PIG every advantage you could and stripped away every benefit of ECL that you could, just how well would ECL perform on identical hardware? This is not a meaningless metric; it gives some idea how ECL will behave when first used by an expert PIG programmer. This same metric also provides a good **lower bound** on just how much better the ECL engineering is compared to the PIG equivalent.

In order to perform such a measurement we decided to use the PigMix[1]. This is a set of 17 Pig programs within the community that have been used to measure the comparative performance of Pig and Java/Hadoop since 11/8/2008. This fits our criteria nicely: it has algorithms chosen by the Pig community, which the Pig community coded and which the Pig system developers have spent almost two years optimizing[2]. As such it ought to be representative of what Pig is used for and embody best practices for how to do it.

The next task was to produce the ECL versions of these Pig programs; we wanted to do so in a way which was reasonably good but which didn't sneak in any ECL 'tricks of the trade'. We therefore chose to use our automatic PIG->ECL converter (BACON). The exact translations this performs for each Pig statement is documented in "ECL for the PIGger". The result is that we now have the same 'PIG' program in 3 languages; PIG, Java/Hadoop and naïve ECL[3]. It should be noted that ECL and Pig are roughly line-for-line equivalents; the Java version required substantially more coding.

The test hardware was a 25 node system. Each node had 4GB of memory, a quad core CPU and 600GB of hard drive space. Nodes were interconnected through a non-blocking Gigabit switch backplane using 1Gbps connections. The data was generated using the Perl data generation script generate_data.pl found in issue PIG-200 on the Apache site[4]. The script allows varying sizes of the base data set page_views to be generated depending on the size of the available cluster with the default of 625M rows. The timings provided on the PigMix wiki page are based on only 10M page_view rows. The results presented here are using a page_views dataset with 156.25M rows which more effectively demonstrates the handling of big data on the 25-node cluster tested. Additional datasets generated by the script include page_views_sorted (156.25M rows), power_users (500 rows), power_users_samples (252 rows), users (1,599,555 rows), users_sorted (1,599,555 rows), widegroupbydata (156.25M rows), and widerow (10M rows). The generated data was translated from the Pig data model to the ECL data model where needed[5] for the ECL version of the benchmark but maintained the identical data content and size.

[1] *Data given here taken from http://wiki.apache.org/pig/PigMix*

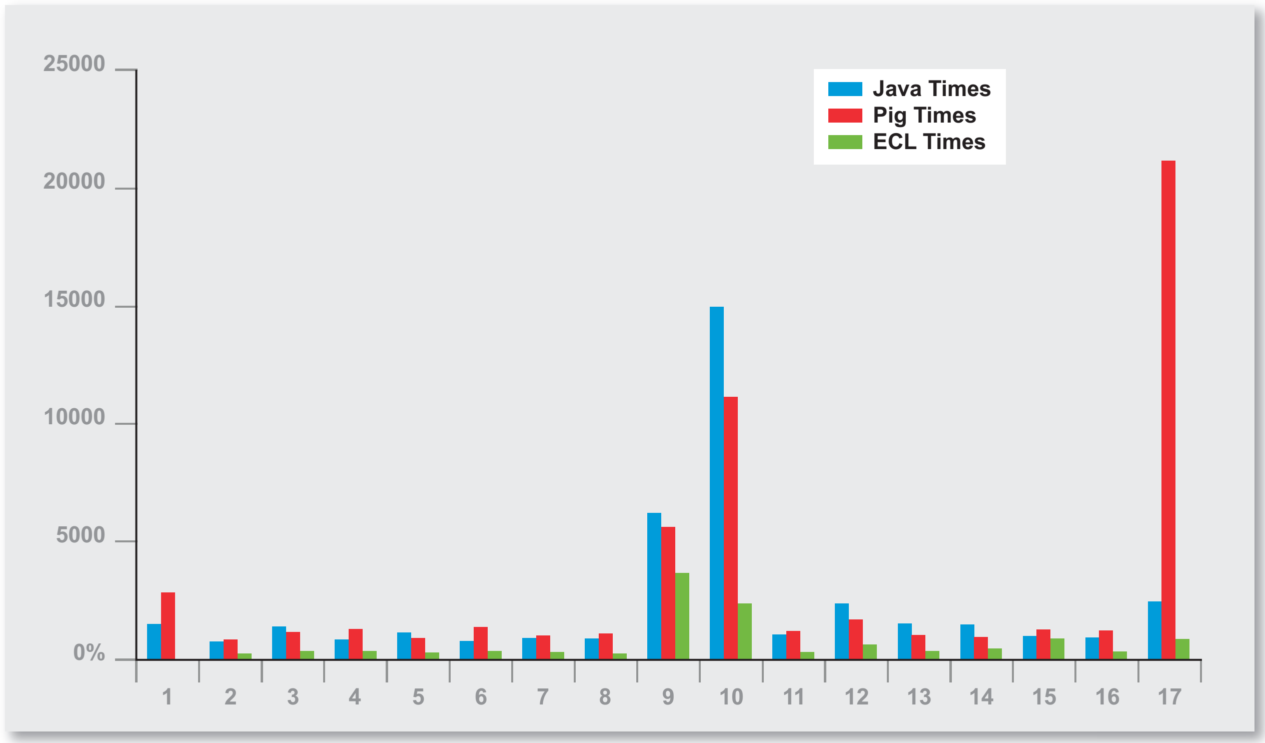
[2] *On the first run they were 7.6x slower than the Java baseline, as of 5/29/10 that number was down to 1.1*

[3] *It should be noted that PigMix L1 tests the Pig MAP capability for which BACON has no direct mapping; thus our test does NOT include a timing for the ECL version of L1*

[4] *PigMix data generation script generate_data.pl was taken from https://issues.apache.org/jira/browse/PIG-200*

[5] *Pig inner bags were mapped to ECL child datasets*

The results were as follows:



It can be seen that there is significant variation from test to test. The outlier is L17 where the Pig time is much worse than Java and worse than on the official Pig website. This is a new test so it could indicate a new optimization has not yet made it into release code and that Pig number will improve. L9 and L10 are also interesting in that it shows that the Pig code (which generates Java) substantially beats the original Java baseline. This demonstrates the work that has been put into enhancing the Pig performance on these benchmarks.

Comparing ECL to Pig we see that it substantially wins every single comparison. The weakest win is L15 where it is 1.46x faster than Pig. The strongest win (ignoring the outlier 24 fold improvement on L17) is the 4.74x improvement granted by our patented SORT algorithm on L10. Across all tests[6] ECL was an average 4.45x faster than Pig.

Comparing ECL to Java we note that the gap shrinks a little but ECL still wins every single test. The weakest is again L15 with a 1.175x speedup and the strongest is L10 with a 6.35x speedup. Across all tests ECL is an average 3.23x faster than native coded Java/Hadoop.

Conclusion

Under conditions set by and for Pig programmers, naively coded ECL was able to win every single one of the 16 benchmarks that had direct equivalents. The winning factor ranged from 1.46x to 4.74x with an average of 4.45x.

[6] *Excluding L1 as noted previously*

For more information:
Website: http://hpccsystems.com/
Email: info@hpccsystems.com
US inquiries: 1.877.316.9669
International inquiries: 1.678.694.2200

About HPCC Systems

HPCC Systems from LexisNexis® Risk Solutions offers a proven, data-intensive supercomputing platform designed for the enterprise to solve big data problems. As an alternative to Hadoop, HPCC Systems offers a consistent data-centric programming language, two processing platforms and a single architecture for efficient processing. Customers, such as financial institutions, insurance carriers, insurance companies, law enforcement agencies, federal government and other enterprise-class organizations leverage the HPCC Systems technology through LexisNexis® products and services. For more information, visit http://hpccsystems.com.

About LexisNexis Risk Solutions

LexisNexis® Risk Solutions (http://lexisnexis.com/risk/) is a leader in providing essential information that helps customers across all industries and government predict, assess and manage risk. Combining cutting-edge technology, unique data and advanced scoring analytics, Risk Solutions provides products and services that address evolving client needs in the risk sector while upholding the highest standards of security and privacy. LexisNexis Risk Solutions is headquartered in Alpharetta, Georgia, United States.

LexisNexis®