PROCEEDINGS OF SCIENCE

Performance testing of the DiFX correlator

Chris Phillips*
CSIRO ATNF
E-mail: Chris.Phillips@csiro.au

Adam Deller
NRAO
E-mail: adeller@nrao.edu

The DiFX software correlator is an easy to install, powerful VLBI correlator designed to run on standard PC clusters. This paper discusses a benchmarking approach using a fake eVLBI data stream to eliminate disk i/o as a bottle neck. Specific tests on how DiFX scales across multiple compute nodes and multicore CPUs are presented as well as the affect on throughput with number of spectral channels.

Science and Technology of Long Baseline Real-Time Interferometry:
The 8th International e-VLBI Workshop, EXPReS09
June 22 - 26 2009
Madrid, Spain

*Speaker.

PoS(EXPReS09)044

1. Introduction

DiFX is a software correlator for VLBI[2]. It is an FX-style correlator, implemented in C++, making heavy use of the Intel Integrated Performance Primitives (IPP) library[1] for optimization. It is designed to run on a cluster of commodity computers (a Beowulf style cluster) and uses MPI (Message Passing Interface) for parallelization. DiFX was written with the aim of maximum performance without compromising generality or ease of maintenance. A modular design allows for relatively easy "3rd party" contributions to the code.

DiFX was written to replace the hardware correlator[1] of the Australian Long Baseline Array (LBA) and has been used exclusively for LBA correlation since 2007. It currently has a large install base across the world and is an integral part of the VLBA sensitivity upgrade[6]. The VLBA hardware correlator is planned to be decommissioned at the end of 2009 to be replaced by DiFX.

The pace of commodity computing equipment development has given software correlators a number of advantages over specifically designed hardware correlators. These include logistical advantages such as shortened development time, improved maintainability and upgradeability, and lower total cost, and also scientific advantages such as greater time and frequency resolution and advanced pulsar binning.

A further advantage is a "non-clocked" processing model. The correlator does not have to run at a fixed data rate, so correlation speed happens as fast as the processing cluster can handle - which may be faster, or slower, than the data was recorded. While this generally is a good thing it means for any given cluster it is difficult a-priori to know the number of processing nodes needed for a specific experiment (ie number of stations, spectral resolution, bandwidth etc). The only accurate way to judge this is with benchmarking.

2. Benchmarking Methodology

Typically benchmarking of software correlators is done by running some disk-based data through the correlator, timing how long it takes to process. In itself this is a valid test to test how long a specific data set will take to process on a cluster. However, a relatively small cluster can possess sufficient computational power to exceed the capacity of its interconnect, which in turn generally exceeds the speeds which can be obtained from hard disks. In such circumstances, the correlator becomes data-starved and it is difficult to ascertain the true performance of the code. This requires a work-around to the problem of (relatively) slow hard disks.

For the tests presented here, DiFX was run in eVLBI mode exploiting the non-clocked processing mode. Real eVLBI data formatters could not be used because they run at a fixed data rate. For a true benchmark, we wish to see how fast a specific cluster can run. To resolve this a simple eVLBI data generator was written. This opens a network connection and sends eVLBI formatted data (as either LBADR[3], Mark5b[4] or VDIF[5]) as fast as possible. Only the VLBI headers are created - no attempt is made to create valid baseband data. For benchmarking, a TCP protocol was used to stream the eVLBI formatted data; the DiFX correlator can accept either TCP or UDP packet streams, but UDP is unsuited for this benchmarking application (due to the lack of any flow control in a UDP data stream). The VLBI data generation program was run on the same node that

[1] http://software.intel.com/en-us/intel-ipp

received the eVLBI data stream to avoid (external) network bandwidth as a possible bottle neck (using the "loopback" network interface).

All testing was done using a set of Bourne shell scripts to start DiFX and the VLBI data generation program. The scripts would run DiFX multiple times changing the values to be tested, such as number of processing nodes, number of spectral points etc. Because DiFX has to be started before the eVLBI data generators start, simply measuring the execution time would not give a good estimation of the sustained processing speed. The VLBI data generation program reports the data transfer speed every couple of seconds. These data are logged and the median data rate calculated (the mean was not used to prevent contamination from outlying values at the start and end of the correlation, when the data buffers within DiFX are being filled and emptied respectively).

For all tests 8 channel 2 bit data was used. If the bandwidth of these baseband channels is assumed to be 16 MHz, this is equivalent to a recording rate of 512 Mbps. However for the purposes of these tests this is not particularly important. The data rates reported give the recording data rate the cluster could sustain in realtime. So, for example, a reported data rate of 256 Mbps would mean the cluster could process a total of 64 MHz of bandwidth in realtime (4x16 MHz at 2 bit).

3. DiFX Architecture

A detailed description of the DiFX architecture is given in [2]. For benchmarking purposes, the most important process run by DiFX is the Core, which receives baseband data via MPI and does the actual correlation (conversion to floating point, fringe rotation, FFT, baseline multiplication etc). Each Core can run multiple threads. For maximum efficiency the number of threads should be roughly the number of CPU cores on the node. The Core processes are fed baseband data by DataStream processes, which read the data from disk or off a network. After processing, the Core nodes send accumulated visibilities using MPI to an FxManager process, which further accumulates and writes to disk.

4. Tests Setup

The tests were run on a cluster operated by Curtin University of Technology. This cluster is a 20 node system of dual CPU quadcore processors (Intel Xeon X5355, 2.66GHz). Tests were run for a 6 station array with up to 12 processing nodes. The main testing was to see how DiFX scaled with number of processing threads/node and the number of processing nodes. For these tests each DataStream process was run on an independent node from the compute nodes.

5. Compute Node and Thread Scaling

Tests were run changing the number of compute threads per node from 1 to 8 (each node has 8 CPU cores) and the number of compute nodes from 1 to 12. This consisted of a total of 96 individual tests. The results are shown in Figures 1 and 2. These two figures show the same data but with the axis swapped to highlight the node and thread scaling.

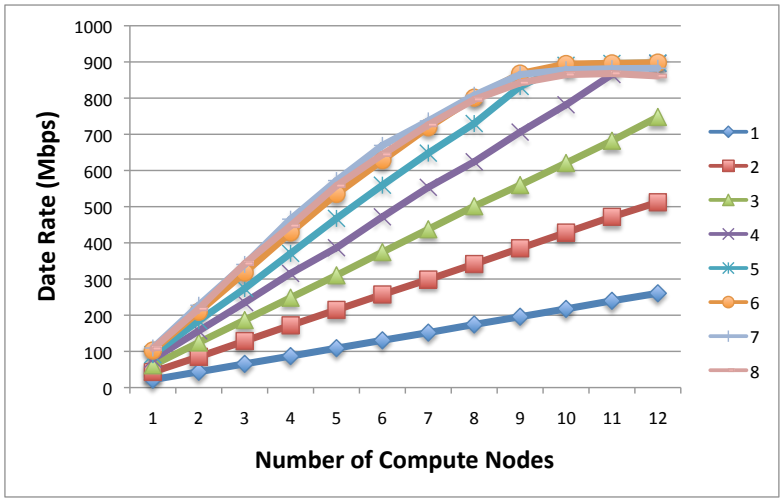

Figure 1: DiFX scaling over computer nodes (for a 6 station experiment) on the Curtin DIFX installation. The 8 coloured lines represent running the test with 1 to 8 processing threads per node.

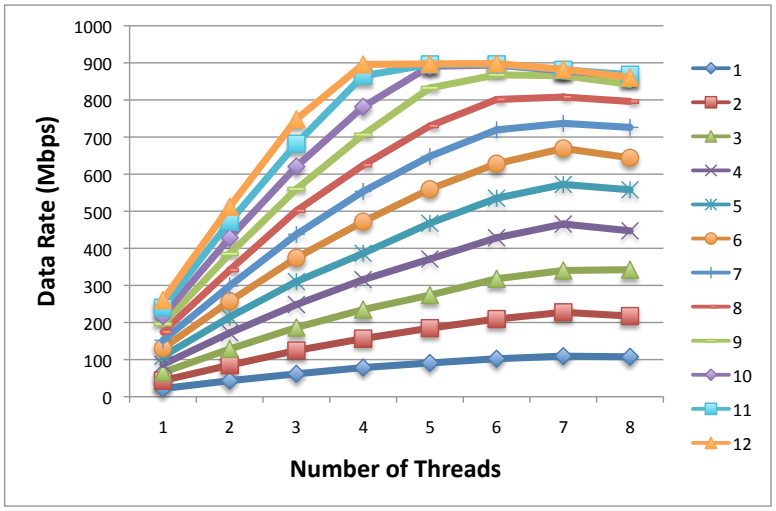

Figure 2: DiFX scaling over threads (for a 6 station experiment) on the Curtin DIFX installation. The 12 coloured lines represent running the test with 1 to 12 processing nodes.

From Figure 1 we see that DiFX scales perfectly across nodes. Doubling the number of processing nodes will halve the processing time. With 10 or more processing nodes and 5 or more processing threads we see data throughput saturating at around 900 Mbps. This seems to be an i/o bottle neck on the DataStream nodes. Given the cluster is connected with 1 Gbps Ethernet this is not much lower than the maximum possible throughput.

From Figure 2 we see scaling with number of processing threads per node is not so good. Running with 8 compute threads is only $\sim$5 times faster than a single thread.

It is unsurprising that 8 threads is less efficient than 7, since in addition to its computational threads each Core node runs a "main" thread to handle data sending and receiving. Since there are

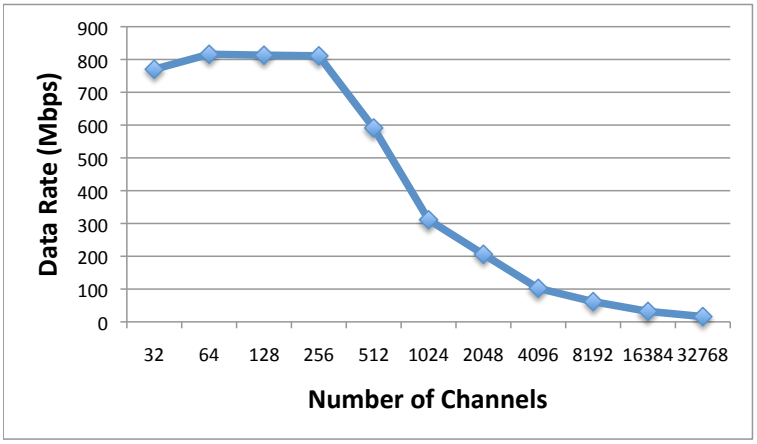

Figure 3: DiFX processing speed per station (for a 6 station experiment with 8 compute nodes) as a function of number of spectral points.

a total of 8 CPU cores, running 8 compute threads actually oversubscribes the available resources and is likely to lead to resource conflicts. However, the degree to which the multicore CPU fails to meet the theoretically available 7x speedup is surprising. The poor scaling with number of threads may have been due to I/O limitations, or the choice of buffering parameters for DiFX for this experiment. However, CPU bandwidth limitation is probably not the cause as CPU usage was monitored during the tests. When running a single processing thread the thread used 100% of the CPU. As more threads were added CPU utilization (of the active cores) dropped eventually to about 65%. As shown in Section 6, better scaling across multicore CPUs has been attained in other benchmarks.

Figures 1 and 2 show the data rate from each DataStream node. The maximum processing speed per compute node for these tests corresponds to $\sim$600 Mbps per node.

6. Scaling with Spectral Points

Further tests were run to benchmark DiFX processing speed as a function of spectral resolution. For these tests the number of processing threads per nodes were kept fixed (6 and 8 respectively). A 6 station experiment was used again. The results are shown in Figure 3. Up to 512 spectral points, the speed is relatively flat. Past this stage the throughput drops quite quickly. This is presumably the point when the FFT size means that the subsequent baseline computations no longer fit within the CPU cache, significantly slowing down the process. Alternatively, as with the CPU core scaling benchmarks, the choice of DiFX buffer parameters may be partly responsible. Testing of DiFX on a similar cluster of dual Intel quad-core nodes by the VLBA [6] have shown significantly improved scaling both across CPU cores (attaining a speed-up of 6.5 with 7 threads) and with increased number of channels (a factor of $<$2 overhead for 4096 channels).

7. Conclusion

DiFX scales perfectly across multiple nodes (up to the dozen tested here). The scaling across

multiple cores on a single node is disappointing. More investigation is needed to decide if this is a problem with how DiFX was configured for these tests, an MPI problem on the tested cluster or an intrinsic i/o limitation in DiFX.

References

[1] Cannon, W. H. and Baer, D. and Feil, G. and Feir, B. and Newby, P. and Novikov, A. and Dewdney, P. and Carlson, B. and Petrachenko, W. T. and Popelar, J. and Mathieu, P. and Wietfeldt, R. D. 1997, Vistas in Astronomy, 41, 297

[2] Deller, A. T., Tingay, S. J., Bailes, M., & West, C. 2007, PASP, 119, 318

[3] Phillips, C., Tzioumis, T., Tingay, A., Stevens, J., Lovell, J., Amy, S., West, C. & Dodson, R. *LBADR: The LBA Data Recorder*, in proceedings of *The 8th International e-VLBI Workshop, EXPReS09*, PoS(EXPReS09)099

[4] Whitney, A. & Cappallo, J., *Mark 5B design specifications* MIT Haystack Mark5 Memo series #019

[5] Whitney, A., *Introduction to VDIF and VTP*, in proceedings of *8th International e-VLBI Workshop EXPReS09*, PoS(EXPReS09)042

[6] http://www.vlba.nrao.edu/memos/sensi