DiFX Benchmarking

There are many ways to benchmark DiFX. Obviously the most realistic benchmark it to setup an experiment with exactly the same disks/network connections etc and see how it runs. However if you wish to test how fast a specific cluster or setup could run, you will often hit a bottle neck of disk I/O speeds.

One approach, discussed here, is to use a program to generate fake VLBI data as fast as possible (using vlbi_fake) and see at what speed DiFX can process the data. Note that it is still easy to hit other bottlenecks, such as node interconnect speeds or diskspeed on the head node.

This approach also is easy to script and run batch test to explore setup parameter space (e.g. number of threads/nodes to use for correlation or changing the size of internal memory allocations etc). vlbi_fake is included in the full DiFX SVN repository in utilities/trunk/bench. It is not built by default.

Benchmarking

Sample scripts for benchmarking are included in the SVN bench directory in the subdirectory samples. Please note you will probably have to slightly (or significantly) edit this script to run on your local DiFX setup.

The first step is to create a DiFX include file and associated setup files (e.g. .delay etc). Typically using an existing experiment with the same number of stations as you want to test is the best approach.

Next, you need to copy samples/vstart.sh to the home directory of all nodes which will run mpifxcorr Datastream processes and edit the options and path etc to suite your local setup and time range the .input files is set for. Note by default vlbi_fake creates LBADR data format.

The example run.sh file tests processing speed varying the number of nodes and threads. To use this script you will need to copy the `threads1` ... `threads8` files to where you are going to run DiFX and edit the `iohosts`, which is where the DataStream processes will run. The array `nps` must also be set. The first values is the total number if mpi process to launch - ie it must start as 2 larger than the number of DataStream processes (telescopes) and increase up to the total number of machines to be included in the test (usually the size of the cluster). Depending on whether the i/o nodes run bash or csh, the line to launch xterm to run vlbi_fake needs to be commented/uncommented as appropriate.

Note that the run.sh script is set to terminate the mpifxcorr job after a specified time - otherwise you will find that you are running some tests for much longer than otherjobs.

Analysis

The time taken for mpifxcorr to run is not a good estimation of the run speed, as there is significant startup overheads (for starting MPI and then vlbi_fake) and mpifxcorr needs to be started before vlbi-fake - so will have been sitting idle initially. The example run.sh script keeps a log of the output from each time vlbi_fake runs. The average data rate from vlbi_fake is *not* a good estimate of the processign speed, as the during the first few seconds the internal buffers within mpifxcorr are being

filled, so vlbi_fake sends data much faster than the sustainable speed.

Each file vlbi_fake is run (via the example run.sh script) a log of its output is kept. The medianrate.pl script included in SVN will check this log and report the median data rate for the run - this is assumed to be the best estimate of the processing speed.

difxspeed

A tool called `difxspeed` was developed that tests DiFX performance using the 'FAKE' data source mode. In this mode the entire correlator is configured as it would be for normal processing, but instead of reading data from a file, Mark5 unit, or from the network, junk data will be produced at a datastream node with just enough formatting to fool the rest of the system. Everything else will proceed as it normally would, yielding fairly accurate performance ratings. A file similar to a simplified `.v2d` file should be prepared. In this file many typical `.v2d` parameters are accepted. Unlike the `.v2d` file syntax, there are no ANTENNA, SETUP, ... blocks (sections within curly braces). Instead all parameters are to be specified at the root level. 5 parameters are required: `cores` is a prioritized list of hostnames to run the core processes, `nCore` is the number of core processes to start, `datastreams` is a prioritized list of hostnames on which to run datastream processes, `nAnt` is the number of antennas, and `antennas` is a prioritized list of antennas to include in the correlation. Note that most simple parameters (including `nCore` and `nAnt`) can take a list of values. `difxspeed` will run every possible combination of the supplied value lists. This can be used to look at performance as a function of processing cores being used, values of certain buffer sizes, ...

Detailed information on difxspeed can be found.

Some additional information, including access to the difxspeed program itself, can be found at the NRAO wiki https://safe.nrao.edu/wiki/bin/view/HPC/UsnoDifxBenchmarking. This information will eventually be moved to the CIRA wiki.