VLBA Sensitivity Upgrade MEMO 16

The Cost of Software Correlation

Walter Brisken (National Radio Astronomy Observatory)

Adam Deller (Swinburne University)

October 9, 2007

Abstract

Software correlators are becoming practical alternatives to their hardware equivalents. In this memo, the merits of software correlators are discussed and a scaling law is derived that allows the processing requirements to be determined for a given correlation task. In the case of recorded media VLBI it is shown that software correlation is affordable. Conventional computer clusters are assumed throughout.

1 Introduction

A cross correlator in radio astronomical terms is any device which computes the correlation function of pairs of voltage signals received by radio telescopes. Typically past correlators have been a hardware devices built from either analog or digital components with the chosen technology dictated primarily by economics. When including engineering resources, the economics now favors, the use of clusters of general purpose computers running optimized software over special purpose digital machines in the construction of new VLBI correlators. Aside from economics, the reasons for embracing such a transformation are manyfold, including:

1. **Flexibility.** Unlike most hardware implementations, the maximum number of antennas, total bandwidth, output data rate, etc. do not need to be fixed.
2. **Efficiency.** All processing resources can be fully occupied at all time, even for very low bandwidth experiments. This is compared to the VLBA correlator where we estimate that on average only about 30% of its processing power is used because of the hard-wired nature of the data pathways and synchronous processing. Additionally some less common processing modes such as multiple phase centers and pulsar binning can be done without a high level of redundancy.
3. **Upgradability.** Perhaps the most significant advantage is that incremental or large upgrades can be made to the processing power with very little engineering effort. Software correlators can scale with increasing recording capacity.
4. **Compatibility** Supporting new recording formats becomes a small software effort rather than a hardware engineering effort. Data recorded in different formats can be readily cross-correlated.
5. **Special processing** Addition of special processing capabilities is not difficult. Flexible pulsar binning, for example, can be added without construction of special hardware. Automatic flagging of strong RFI using very high spectral resolution could be performed before averaging down to a reasonable number of output channels. Arbitrary time and spectral averaging will likely allow new possibilities.

2 Processing requirements

In this section the theoretical processing requirements for a software correlator are calculated. Quantities used in this document are described in Table 1. To clarify in advance, "BBC" in all cases in this document is intended to mean "Base Band Channel', implying a portion of spectrum in single polarization which can have in its digital representation an arbitrary number of bits[1].

[1]This is in contrast to a "Base Band Converter", which is a device which may play a role in the construction of one or more Base Band Channels.

2.1 Model algorithm

In order to evaluate the computational needs, one must first adopt a model algorithm to study. In this case a standard FX architecture is used, but this model is not based on any particular implementation. It is expected that any practical implementation would differ from the model only by constant factors of order unity. In this model the processing steps are as follows:

1. Load data from disc into a large RAM buffer, apply integer sample delay correction and unpack the appropriate time range of coarsely quantised (typically one or two bit precision) data into a floating point representation.
2. Apply geometric delay model to data (i.e., fringe rotate). The data are now complex values.
3. Perform a complex-to-complex Fast Fourier Transform (FFT).
4. For each spectral component, multiply by appropriate phase factor to account for fractional sample delay at center of FFT interval.
5. For each desired correlator product multiply and accumulate to form visibility spectra.
6. Write out visibility data in a standard format.

The above algorithm is similar (but with definite differences) to the DiFX software correlator.[2]

Symbol	Description
$N_{\rm ant}$	number of antennas
$N_{\rm FFT}$	FFT length in samples
$N_{\rm BBC}$	number of base band channels
$N_{\rm pol}$	number of polarization correlations per BBC (= 2 for full pol, 1 otherwise)
$N_{\rm bit}$	bits per sample, usually 1 or 2
$N_{\rm overlap}$	overlap factor (= 1 for non-overlapping FFTs)
$N_{\rm oversamp}$	oversample factor (= 1 for Nyquist sampling)
$R_{\rm BBC}$	BBC sample rate
Δt	integration (accumulation) time
$N_{\rm pass}$	number of correlation passes
$N_{\rm chan}$	number of channels ($= N_{\rm FFT} \div 2N_{\rm oversamp}$)
$R_{\rm samp}$	record sample rate ($= N_{\rm BBC} \cdot R_{\rm BBC}$)
$R_{\rm rec}$	record bit rate ($= N_{\rm bit} \cdot R_{\rm samp}$)
$\Delta t_{\rm samp}$	sample interval ($= 1 \div R_{\rm BBC}$)
$\Delta t_{\rm FFT}$	FFT interval ($= N_{\rm FFT} \cdot \Delta t_{\rm samp}$)
$N_{\rm ave}$	number of FFT intervals to average ($= N_{\rm overlap} \cdot \Delta t \div \Delta t_{\rm FFT}$)
Δf	frequency resolution ($= 2 \div \Delta t_{\rm FFT}$)
$F_{\rm bloat}$	data volume expansion factor after correlation (Eqn. 1; usually $\ll 1$)
$O_{\rm load}$	floating point operations per sample for data loading and unpacking (Eqn. 2)
$O_{\rm fringe}$	... for fringe rotation (Eqn. 3)
$O_{\rm FFT}$	... for FFTs (Eqn. 4)
$O_{\rm correct}$	... for fractional bit correction (Eqn. 5)
$O_{\rm MAC}$	... for multiply and accumulate (Eqn. 6)
$O_{\rm total}$	... for all operations (Eqn. 7)
$R_{\rm comp}$	total computation rate (Eqn. 8)

Table 1: Descriptions of variables used in this document.

[2]Deller et al., 2007, PASP 119, 318D; also astro-ph/0702141. The DiFX software correlator is an FX architecture correlator written in C++. It is designed to take advantage of large numbers of CPUs within a cluster with the use of the Message Passing Interface. It is currently in use by the Australian LBA and is being investigated for use by several other VLBI processing centers.

2.2 Operation counts

In counting needed operations it will be assumed that $N_{\rm ave} \gg 1$ so that data export and any calibration that may be applied does not substantially increase the operation count. This assumption is fair except in cases where very high time and frequency resolution are needed (i.e., $\Delta t \Delta f \sim 1$) or when the number of antennas is very large. In such cases the correlator no longer reduces data volume! To be more precise, the data expansion factor assuming the output data are represented by 32-bit complex floating point values is

$$F_{\rm bloat} = \frac{32}{N_{\rm bit}} \cdot \frac{N_{\rm ant}}{2} \cdot \frac{N_{\rm pass} \cdot N_{\rm overlap}}{N_{\rm ave} \cdot N_{\rm oversamp}}. \quad (1)$$

For the VLBA with $N_{\rm ant} = 10$, $N_{\rm bit} = 2$, and $N_{\rm overlap} = 2$ to have $F_{\rm bloat} < 1$ requires $N_{\rm ave} > 160$, which justifies the initial assumption for all but the most impractical projects. The following subsections justify estimates of operation counts (O) for each of the various major logical processing steps in an FX software correlator.

2.3 Data loading and unpacking

Data loading and unpacking is an I/O bound activity that can be effectively done in parallel during the correlation of the previos FFT interval. The load on the CPU is dominated by a lookup-table operation that assigns a floating point value to each bit pattern or a flag value for invalid data. The cost of this is probably equivalent to about two floating point operations per sample:

$$O_{\rm load} = 2. \quad (2)$$

2.4 Fringe rotation

Fringe rotation requires a few simple operations. Application of a linear gradient in phase can be done with a single add and a lookup table operation to determine the real and imaginary parts, followed by a real-complex multiplication. In total:

$$O_{\rm fringe} = 8. \quad (3)$$

A combined load and fringe rotation operation would likely spare a few operations per sample for a marginal overall speedup.

2.5 FFT

The complex-to-complex FFT algorithm can in principle be preformed using only $5N \log_2 N$ floating point operations. Given this, the number of operations per sample needed to compute the FFTs is:

$$O_{\rm FFT} = 5 \log_2 N_{\rm FFT} \cdot N_{\rm overlap}. \quad (4)$$

Actual implementations of this algorithm (such as FFTW: `http://www.fftw.org`) can approach this operation count for large FFTs but typically fall short for very small FFTs (below ~ 32 samples) where setup overhead is large. For large FFTs (more than ~ 4096 samples) cache misses within the CPU cause performance to drop by as much as a factor of 2.

Only half the output spectrum is used in further processing: the positive frequencies are retained for upper side band channels and the negative frequencies are retained for lower side band channels.

2.6 Fractional bit correction

The FFT output must be multiplied by a phase factor with a constant slope in frequency in order to compensate for the fractional sample period portion of the delay model. The computation of each phase factor can be done in several ways. An efficient algorithm may use repeated multiplication of phase factors, each requiring 6 operations. Each positive-frequency sample is multiplied by such a phase factor requiring another 6 operations. For oversampled data only part of the spectrum is wanted. Thus the number of operations required is:

$$O_{\rm correct} = 6 \frac{N_{\rm overlap}}{N_{\rm oversamp}}. \quad (5)$$

2.7 Multiply and accumulate

Every requested correlation product requires cross multiply and accumulate of a pair of FFT data streams. For correlation of a given BBC on all baselines in an array, the usual case, each FFT stream needs to be multiplied by every other stream. The number of operations required for each recorded sample for multiplication and accumulation is:

$$O_{\mathrm{MAC}} = 5\frac{N_{\mathrm{overlap}}}{N_{\mathrm{oversamp}}} \cdot \frac{N_{\mathrm{ant}}}{2} \cdot N_{\mathrm{pol}}. \tag{6}$$

2.8 Number of correlation passes

Considerable overall savings can be made by correlating multiple passes (i.e., multiple phase centers or pulsar gates) concurrently. Data loading, bulk fringe rotation, and FFT operations are the same for each pass. For additional passes a relative fringe rotation is still required, but in almost all cases this fringe rate is small enough to be done after the FFT without decorrelation. However, is typically too large to do after the multiply and accumulate operations. The relative fringe rotation can be incorporated into the fractional bit correction operation at virtually no cost. Thus for each additional pass an additional fractional bit correction operation and multiply and accumulate operation is needed.

2.9 Total operation count

Summing up the contributions from the above components results in a total number of floating point operations required for each recorded sample:

$$O_{\mathrm{total}} = 10 + N_{\mathrm{overlap}} \cdot \left(5\log_2 N_{\mathrm{FFT}} + N_{\mathrm{pass}} \cdot \frac{12 + 5N_{\mathrm{ant}} \cdot N_{\mathrm{pol}}}{2N_{\mathrm{oversamp}}} \right) \tag{7}$$

This equation breaks down for small FFT sizes due to loop overhead. Practical experience shows that the peak throughput occurs above or around $N_{\mathrm{FFT}} = 256$. Correlation into fewer spectral channels should then be achieved though spectral averaging. At this stage filters can be applied that improve the spectral response with essentially zero impact on compute time.

The total computation rate required to correlate at real-time speed, measured in FLoating point OPerations per Second (FLOPS) is:

$$R_{\mathrm{comp}} = N_{\mathrm{ant}} \cdot R_{\mathrm{samp}} \cdot O_{\mathrm{total}} \tag{8}$$

It should be noted that correlation has computation complexity $\mathcal{O}(N_{\mathrm{ant}}^2 \cdot R_{\mathrm{samp}})$. This model algorithm suggests that for typical FFT sizes of ~ 512 the quadratic term starts to dominate when correlating around 10 antennas. For DiFX it appears that the quadratic order calculations are better optimized and/or the complexity of station-based calculations is underestimated in this model moving the cross-over point to around 20 antennas. Table 2 tabulates R_{comp} for various typical VLBI observation types as well as for some VLA, EVLA, and GMRT modes. It can be seen that for most VLBI modes between 150 and 250 floating point operations are required per sample. The vast range of sample rates and array size makes the real-time equivalent compute rate vary by nearly a factor of 1000 for realistic VLBI observations, ranging from about 5 to over 4000 GFLOPS. For comparison, a fast desktop computer today can sustain about 3 to 5 GFLOPS.

3 Implementation

Benchmarking for FFTW suggests that a powerful 3 GHz processor today will perform at about 3 GFLOPS in some real life situations (such as doing FFTs), so for the sake of simplicity this number will be assumed. The one-to-one correspondence between floating point operations and CPU clocks does not imply one floating point operation per cycle, but more likely suggests that at times multiple operations are done simultaneously in the Single Instruction Multiple Data (SIMD) floating point unit. The majority of the remaining clock cycles are consumed moving data and and in overhead associated with function calls. Moderm CPUs and compilers make it difficult to trace exactly what happens on each clock cycle,

Observation Type	$N_{\rm FFT}$	$N_{\rm ant}$	$N_{\rm pol}$	$N_{\rm BBC}$	$N_{\rm bit}$	$T_{\rm int}$ (sec.)	$R_{\rm rec}$ (Mbps)	$F_{\rm bloat}$	$O_{\rm total}$	$R_{\rm comp}$ (GFLOPS)
OH Maser	4096	10	2	2	2	5	8	0.002	137	5.5
Spacecraft	1024	10	1	8	2	1	128	0.01	172	110
Pulsar scattering	65536	4	2	8	2	1.25	256	0.1	222	114
Simple imaging	32	10	1	8	2	2	256	0.00008	122	156
... 2 passes	32	10	1	8	2	2	256	0.00016	184	236
RDV	16	20	1	8	2	2	256	0.00008	162	415
HSA (512 Mbit)	128	14	2	16	2	2	512	0.0005	232	824
... 2 passes	128	14	2	16	2	2	512	0.0009	384	1380
Widefield global	1024	25	2	16	2	0.5	512	0.025	372	2380
VLBA (1 Gbps)	256	10	2	16	2	2	1024	0.00032	202	1033
VLBA (4 Gbps)	512	10	2	32	2	2	4096	0.00032	212	4340
VLBA (16 Gbps)	2048	10	2	32	2	2	16384	0.00032	232	19000
GMRT	128	30	2	4	4	10	512	0.00005	392	1500
VLA (2AC mode)	16	27	1	2	$\log_2 3$	10	80	0.00002	197	266
EVLA (8-bit)	65536	27	2	4	8	5	65536	0.0004	452	100000
EVLA (3-bit)	65536	27	2	8	3	5	98304	0.0005	452	400000

Table 2: Various observe modes and their computation requirements. For all of these modes $N_{\rm overlap} = 2$. For the first example, $N_{\rm oversamp} = 16$. For all others $N_{\rm oversamp} = 1$. The first group of examples demonstrates that for a very wide variety of extreme experiments the number of floating point operations per sample varies by only a modest amount. The second group lists is representative of typical continuum imaging experiments using the VLBA at 1, 4, and 16 Gbps. Finally examples for the GMRT, VLA and EVLA are shown for comparison, assuming these were to use the same software correlator discussed here. The EVLA cases demomstrate that software correlation is not yet a viable alternative for all applications.

and consequently it is difficult to directly compare CPUs of different generations or families as each has its strenghts and weaknesses.

In order to correlate typical VLBA projects at 256 Mbps record rate in real time at least 200 GFLOPS will be needed. In order to replace the full power of the VLBA correlator about 4 times this would be required. Since most projects use less than 30% of the VLBA correlator's computational capacity, and the increased efficiency of additional correlator passes and pulsar modes in software correlators, a 200 GFLOPS machine would probably be sufficient to replace the existing correlator.

3.1 Beowulf Clusters

Beowulf Clusters (`http://www.beowulf.org/`) are clusters of individual computers running linux all connected by a fast network. These systems have become a proven cost-effective means of achieving processing power well into the TeraFLOPS range (e.g., the Swinburne supercomputer runs at 2.0 TFLOPS). Software correlation is an "embarrassingly parallelizable"[3] problem. This is because the calculations being performed on one node have no dependence on the results from the others, thus data buffers storing data to be processed allow each processing node to be fully utilized. It will be assumed through this document that such a cluster will be used. Since these clusters scale quite well with number of CPUs the same design principles should be used to guide the construction of a demonsration or operational cluster. Bang-for-the-buck systems, such as an array of inexpensive PCs, may appear economically most attractive, however all advice availble through first hand contact (i.e., the Swinburne supercomputer group) or online resources (such as `http://fscked.org/writings/clusters/cluster.html`) caution against this as the cost of a cluster can be much more than the hardware purchase price, especially for large clusters made with inferior components. Many large successful Beowulf Clusters have at least one full time

[3]This term is becoming an industry standard for problems that scale linearly with number of processing nodes without extreme consideration on the algorithm used.

operator, however a well built, moderate sized cluster optimized for a single application should require minimal maintenance. Additional general information about cluster computing can be found on the web. Two rather complete sites are: `http://www.clustermonkey.net` and `http://www.linuxhpc.org`.

3.2 Dividing the work

How is the problem of cross correlation be best divided over a number of CPUs? to minimize intra-process communication and redundancy of computation it is essential to perform all FFTs for a given integration period for a given BBC (or BBC pair for full polarization) on the same CPU. This hints at two schemes: BBC division and time division. Time division can be simply generalized without penalty to an arbitrary number of processing nodes, even in a heterogeneous cluster of computers. This is the scheme used by the DiFX software correlator.

4 Cost

In this section we estimate the cost to own and operate a software correlator capable of processing 10 stations at 1 Gbps, i.e. a reasonable upgrade over the current VLBA capability. To first order this cost estimate will scale with bandwidth. All prices quoted are subject to change and even for purchases on the reference days is probably accurate to only about 30%.

4.1 Cluster cost

To get an idea of the cost of this cluster, we went to a low-cost online rack server vendor (`http://www.siliconmechanics.com`) and configured a 1U server with 4 quad-core Intel XEON CPUs at 2 GHz for a total cost of just under $4000 (on 09 Oct 2007). The 1 TFLOPS required could be met with 40 of these systems at a cost of $160k. Getting to 1 Gbps per station is not challenging from a network point of view. Three stackable 48-port Gbit ethernet switches (such as the Netgear GS748TS at ∼$1000 each) would be sufficient. Cabling, power conditioning, and a rack would probably add another $4000 to the overall cost. Such a cluster would consume about 25kW of power, costing about $22k per year in electricity. Compute power as a function of cost and power consumption will most certainly continue dropping with time. Coincidentally, the total cost of this cluster is not much different than the cost of 10 Mark5 units that would be used to serve data to the software correlator.

4.2 Media cost

The VLBA operates with a 30 day media turn-around period. This means that for 10 stations at 1 Gbps mean record rate a 3200 TB media pool is required. Using 750 GB SATA discs today (09 Aug 2007) one can assemble a 6TB module for about $2000 for a media pool cost of $1.1M. Shipping of modules at our current mean record rate of about 160 Mbps costs around $100k annually. This number is not likely to decrease with increased recording bandwidth, even if much larger disc modules are used.

4.3 Software correlation is affordable

The estimates above show clearly that the price of a cluster for software correlation is a fraction (perhaps about 25%) of the cost of the media pool. Additionally the cost of powering and cooling the computer cluster would be a comparable fraction of media shipping expenses. It hard to predict exactly how all of the prices used in this analysis will change with time, but it is my guess that any changes that occur will be in favor of software correlation.