# Multicore Acceleration of the Complex Ambiguity Function

Douglas P. Enright, Eric M. Dashofy, Michael AuYeung, R. Scott Boughton, J. Matt Clark, Ronald Scrofano, Jr.
The Aerospace Corporation
{Douglas.P.Enright, edashofy, mauyeung, RScott.Boughton, JMatt.Clark, Ronald.Scrofano}@aero.org

## Abstract

Results from a multicore parallelization study of a target application, the Complex Ambiguity Function (CAF)[1], are presented. The CAF is a key algorithm in performing time and frequency delay of arrival calculations for actively sensed objects. The main algorithmic component of the CAF involves computing multiple cross-correlations over a range of frequency and time shifts with Fast Fourier Transforms. Also additional pre-processing of the input signals with Hilbert transforms and FIR filters using FFTs is performed. To assess the efficacy of our parallelization effort two workload-driven metrics were calculated, a fixed workload-constrained (WC) metric and a modified linear-scaled workload time-constrained (MLSWTC) metric. By taking advantage of the loop-level parallelism present within the CAF Processing Chain via OpenMP parallel directives, parallel speedups of 75% and greater for a dual-quad core Intel Xeon system were achieved.

## Hardware and Software

A dual-quad core system comprised of Intel Xeon E5335 "Clovertown" processors was used for our workload-driven evaluation. Each "Clovertown" processor runs at a 2.0GHz clock rate with a 2x4MB shared L2 cache and split 32KB L1 instruction and data caches along with a quad-pumped 1333 MHz Front-Side Bus interface resulting in a peak 10.6 GB/sec bandwidth between the processors. Both processors shared 8GB of system memory and the OS used was 64-bit CentOS5. Results reported used the Intel ICC v.10.0 compiler along with Intel's OpenMP runtime, OpenMP*, based upon the OpenMP v.2.5 API standard and the FFT library supplied with v. 10.0.3.20 of Intel's MKL package.

## CAF Processing Chain and Workload Parameterization

Figure 1 illustrates the required processing steps in order to calculate the time and frequency delay of arrival for two input streams. Specifically, there are four steps involved. Each real-valued input stream is transformed into a complex-valued stream by a Hilbert transform. Then each transformed complex-valued stream is broken up into $k$ frequency channels via an FIR filter process. Pairs of frequency channels are then combined in a time-shifted fashion to form "caf surfaces" from which the time and frequency delay of arrival values of objects of interest correspond to peaks on the caf surface. Mathematically, a caf surface formed from a pair of transformed streams can be represented as

$$A_k(m,\nu) = \sum_{l=0}^{L-1} S_{a,k}(l+\nu) \times S_{b,k}^{*}(l) \times e^{-j2\pi ml/L},$$

**Equation 1: caf surface** $A_k(m,\nu)$

where $m$ is the frequency shift and ν is the time shift for the caf surface corresponding to the $k$th frequency band. 256 channels were used in our implementation. The time length $L$ of the portion of the signal used in Equation 1 is bounded by the amount of available data and the minimum signal-to-noise ratio (SNR) one wishes to detect. For the results reported here, the SNR is -10dB, corresponding to a length or block size of 4096.

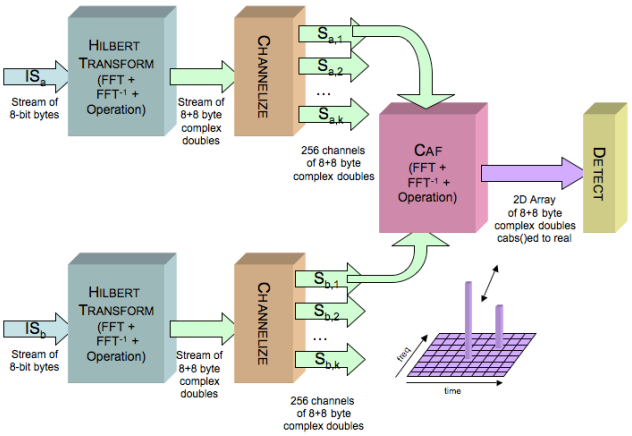



**Figure 1: CAF Processing Chain**

We parameterized the computational workload of the CAF Processing Chain in two ways. First, usual implementations of the CAF Processing Chain use a signal power culling optimization to avoid computing the computationally expensive caf surfaces when the input signal power is not large enough. We term this the Nominal Surface processing case. We also chose to compute all the possible caf surfaces that could be formed, providing for an upper bound on the overall CAF Processing Chain runtime. This case is termed the All Surface processing case. Second, the nominal input stream size was 6.25 Megasamples (MS) where 1MS = $2^{20}$ samples. For a 6.25 MS input signal, there were a total 32768 blocks. We varied the input size in steps of 6.25 MS up to a maximum of 31.25MS. For physical plausibility, the maximum input signal stream was not scaled more than 5 times the original input size. It was observed that the serial runtime scaled linearly with linear scaling in input signal stream size.

## Parallelization

The majority of work within the HILBERT TRANSFORM and CHANNELIZE modules are each contained within a for loop iterating through all the blocks within the input stream.

Both for loops span the length of the module and contain FFT calls. Also, there are no data dependencies between different iterations of the loop. This type of loop structure is ideal to be performed in parallel using a `#pragma omp parallel for` iterative workshare construct. In addition, the CAF and DETECT modules are contained within a doubly nested for loop, which iterates over all possible channels and blocks per channel. Again, there are no data dependencies between different loop iterations allowing for the use of a `parallel for` construct. One could parallelize within the CAF module solely instead of outside the module, however we found that the best parallel scaling was obtained when parallelizing at the outermost level.

## Workload-Driven Parallelization Metrics[2]

To assess the efficacy our parallelization effort and the ability of the dual-quad core Xeon system to scale, two workload-driven metrics were calculated. The first, a fixed workload-constrained (WC) metric, examines the ability of a parallel system to minimize overall wall-clock time through the use of multiple cores. A speedup value is calculated as shown in Equation 2.

$$\text{Speedup}_{\text{WC}}(p \text{ cores}) = \frac{\text{Time}(1 \text{ core})}{\text{Time}(p \text{ cores})}$$

**Equation 2: Workload-Constrained (WC) Speedup**

For the dual-quad core system used, a best-case parallel scaling over 8 cores would result in a speedup of 8.0. A worst-case scaling would result in a speedup of 1.0, i.e. the use of additional cores did not decrease overall wall-clock time.

A second workload-driven metric examines the ability of a parallel system to maintain a uni-core runtime as the workload is increased in proportion to the number of cores used. This is a time-constrained scaling, or "scaled-speedup" model as first proposed by Gustafson[3]. Since the dual-quad core system used has 8 cores while the input signal stream could only scale by a factor of 5, a modified linear-scaled workload time-constrained (MLSWTC) speedup metric is proposed. The speedup metric is

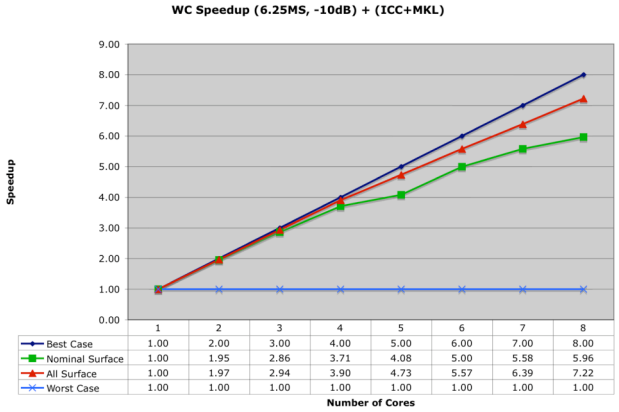
$$\text{Speedup}_{\text{MLSWTC}}(p \text{ cores}, \min(p, ubism) \text{ input size}) = \frac{\text{Time}(1 \text{ core}, 1 \text{ input size})}{\text{Time}(p \text{ cores}, \min(p, ubism) \text{ input size})},$$

**Equation 3: MLSWTC Speedup**

where *ubism* is the upper-bound input size multiple of the base input size, which for the purposes of this study is 5. A best-case MLSWTC speedup for (5 cores, 5 input size) is 1.0, indicating that 5 times the workload can be processed in the same amount of time as the base input workload when using 5 cores. For a (8 cores, 5 input size) case, the best-case MLSWTC speedup is 1.6. A worst-case MLSWTC speedup for (8 cores, 5 input size) is .2 .

## Results

The WC speedup results obtained for an input size of 6.25MS, -10dB SNR and both the Nominal Surface and All Surface cases are presented below.

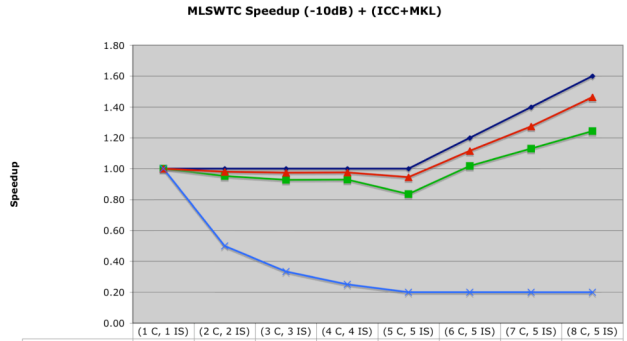



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Best Case | 1.00 | 2.00 | 3.00 | 4.00 | 5.00 | 6.00 | 7.00 | 8.00 |
| Nominal Surface | 1.00 | 1.95 | 2.86 | 3.71 | 4.08 | 5.00 | 5.58 | 5.96 |
| All Surface | 1.00 | 1.97 | 2.94 | 3.90 | 4.73 | 5.57 | 6.39 | 7.22 |
| Worst Case | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

**Figure 2: Nominal Surface and All Surface WC Speedup**

As seen in Figure 2, a speedup of 5.96 for 8 cores was obtained for the Nominal Surface case and 7.22 for 8 cores with the All Surface cases. The regularity of processing with the All Surface case allows for a better overall speedup. The uni-core wall-clock time was 5.36 seconds for the Nominal Surface case and 37.38 seconds for the All Surface case.

The MLSWTC speedup results for both surface processing cases and a SNR of -10dB is shown below.




| | (1 C, 1 IS) | (2 C, 2 IS) | (3 C, 3 IS) | (4 C, 4 IS) | (5 C, 5 IS) | (6 C, 5 IS) | (7 C, 5 IS) | (8 C, 5 IS) |
|---|---|---|---|---|---|---|---|---|
| Best Case | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.20 | 1.40 | 1.60 |
| Nominal Surface | 1.00 | 0.95 | 0.93 | 0.93 | 0.84 | 1.02 | 1.13 | 1.24 |
| All Surface | 1.00 | 0.98 | 0.97 | 0.98 | 0.95 | 1.12 | 1.27 | 1.46 |
| Worst Case | 1.00 | 0.50 | 0.33 | 0.25 | 0.20 | 0.20 | 0.20 | 0.20 |

**Figure 3: Nominal Surface and All Surface MLSWTC Speedup**

For both cases, the 31.25MS input size is processed with 6 cores in less time than the 6.25MS uni-processor time. Again, the regularity of processing with the All Surface case provides for better overall scalability.

## References


[1] Stein, S., "Algorithms for Ambiguity Function Processing", *IEEE Trans. Acoustics, Speech, and Signal Proc.* v. ASSP-29, p. 588-599 (1981).

[2] Culler., D.E. and Singh, J.P. with A. Gupta, "Parallel Computer Architecture – A Hardware/Software Approach", Morgan Kaufmann Publishers, Inc. (1999).

[3] Gustafson, J.L., "Reevaluating Amdahl's Law", *Comm. ACM*, v. 31, no. 5, p. 532-533 (1988).