# Impact of Grid Processing on Signal Cross-Correlation

Rhea Senthil Kumar, Nathan Simard, Jonathan Mathews, Jeremy Kepner, Timothy Collard
MIT Lincoln Laboratory

***Abstract*—Systems involving signal processing produce large amounts of data, requiring an efficient computing architecture to measure data characteristics on feasible time frames. Grid processing offers scalability and user-controlled resource optimization, making it a viable method for large-scale computing applications, such as machine-learning, financial modeling, and data analytics. In this paper, we focus on grid processing of collected signal data. In this work, we adapted a cross-correlation algorithm to be compatible with grid processing, improving runtime by several orders of magnitude. This paper measures the performance of the MIT Lincoln Laboratory Supercomputing Center (LLSC) cluster using Intel Xeon 64-core processors for signal cross-correlation. To the best of our knowledge, this is the first successful implementation of fast Fourier transform (FFT)-based signal cross-correlation utilizing CPU-based grid processing.**

***Index Terms*—grid computing, signal processing, cross-correlation**

## I. Introduction

Cross-correlation is a mathematical technique used to measure the degree to which two signal sequences are similar as a function of their time lag. By systematically shifting one signal relative to a reference signal, cross-correlation can be used to extract similar features or delays between the two signals, even in the presence of noise. Cross-correlation is often encountered in applications such as radar, sonar, digital communications, image and video processing, as well as various other areas of science and engineering [1].

This study investigates the runtime performance of a cross-correlation algorithm designed to measure temporal alignment between two signals. The authors measured real signal data with the objective of collecting cross-correlation metrics. This required computing multiple cross-ambiguity function (CAF) surfaces per minute of data. However, initial attempts to generate each CAF took several hours on an Intel Xeon w7-2495X processor, leading to an intractable sequential processing time for the entire collect.



To solve this computational challenge, the authors designed a parallelized FFT-based cross-correlation pipeline using the LLSC cluster's multi-core, multi-threaded architecture to significantly increase runtime efficiency. The calculation of correlation for each signal shift is computationally expensive for large signals, scaling as $O(N^2)$, where N is the length of the signal. By using an FFT-based correlation algorithm, this time complexity can be reduced to $O(NlogN)$. The goal of this parallelized pipeline is to demonstrate the feasibility of cross-correlating large volumes of signal data.

## II. Processing

The CAF serves to generalize the correlation process for signal feature estimation. The CAF can be expressed as:

$$A(\tau, f) = \int_0^T s_1(t) * s_2(t+\tau) \exp(-j2\pi ft)dt \quad (1)$$

Where $s_1(t)$ and $s_2(t)$ are complex signals that have similar envelopes, and $\tau$ and $f$ are parameters that represent time lag and frequency offset, respectively. For aperiodic signal features within the integration period $T$, a maximum occurs in the CAF surface when the parameters $\tau$ and $f$ compensate exactly for the offset in time and frequency between the envelope features of $s_1(t)$ and $s_2(t)$.

Each $\tau$ in the CAF surface is represented by the difference frequency components extracted from the product of $s_1(t)$ and $s_2(t)$, and may be represented as a mixing product $r(t;\tau)$:

$$r(t;\tau) = s_1(t) * s_2(t+\tau) \quad (2)$$

Taking the FFT of this mixing product yields a spectrum where the maximum indicates the frequency offset between the two signals. Collating the FFTs of the mixing product at varying time lags builds up the full CAF surface. The computational complexity of this operation may be mitigated by reducing FFT bin size and restricting the range of lags to be evaluated. However, these techniques can severely constrain the utility of this function, especially for cases where large time lags or frequency offsets are expected.

Although other techniques exist to improve computational efficiency under certain signal conditions [2], parallelization is a particularly effective solution for this paradigm. The composite CAF construction can be naturally subdivided and distributed across myriad cores.

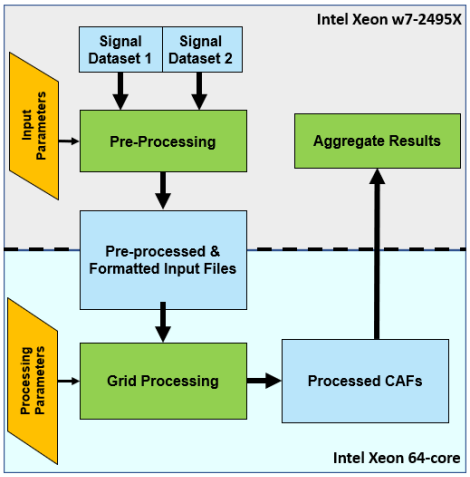


Fig. 1. Cross-correlation pipeline integrated with LLSC cluster.

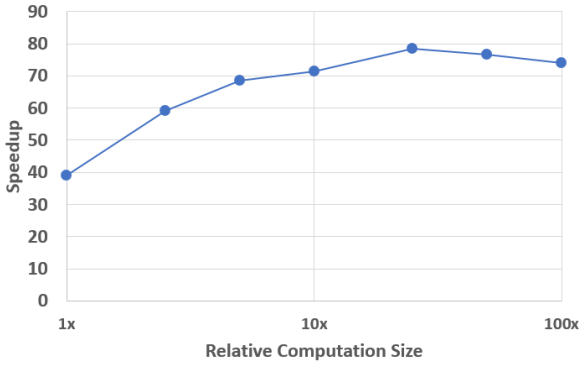


Fig. 2. Comparison of CAF speedup factor from local to grid-side across increasing CAF width.

## III. Implementation

The MIT LLSC cluster offers high-performance on-demand supercomputing capabilities. User-submitted jobs are automatically scheduled by the free open-source Simple Linux Utility for Resource Management (SLURM) scheduler. This study used the Intel Xeon 64-core processor compute nodes, which support the Intel AVX-512 instruction set. Each process on these nodes is allocated 2 cores, and each user is allocated a maximum of 8192 cores at once. Although these nodes are not the fastest computational resource on the cluster, such as the Intel Xeon Gold 6248 CPUs and NVIDIA Volta V100 GPUs, they were selected as the baseline for this work due to their large scalability. Additionally, job parameters can also be customized by the user to optimize multi-core and multi-processing parameters on the LLSC cluster [3].

The cross-correlation processing pipeline, displayed in Figure 1, was developed locally with MATLAB 2023b, then integrated with the LLSC MATLAB environment. Signal data was first pre-processed on Intel Xeon w7-2495X processors (hereafter referred as local processing), and CAF processing parameters were computed before being packaged as data structures and passed to the LLSC cluster. The CAF computation was then submitted as a job array on the Grid's SLURM scheduler, where each process was assigned to a batch of CAFs proportional to number of cores allocated. The computation, hereafter referred to as grid-side processing, was spread over 128 cores, allowing for 64 processes to be executed simultaneously. Parameters affecting runtime included signal integration time, CAF width, and FFT bin size. Each resulting CAF and its associated metadata were then packaged into individual data structures. These are passed back to sequential Intel Xeon w7-2495X processors and aggregated for further analysis of cross-correlation metrics.

## IV. Results

The comparison of CAF computing capacity shows that the LLSC cluster can generate CAFs at a significantly higher rate as compared to the Intel Xeon w7-2495X machine. Note that the CAF width greatly impacts the time to generate a CAF, as larger sizes reduce the the need for a high quality initial peak estimate. The authors compare relative computation size increasing from a baseline CAF width of 10 microseconds. As shown in Figure 2, the LLSC Intel Xeon 64-core processors demonstrate superior CAF computing efficiency across the entire search space, and reach a maximum speedup factor of 80x. The authors compared the worst-case runtime performance, assuming a 100 ms signal integration time, generation of 50 CAFs per minute of data, and 1 millisecond CAF width. The local Intel Xeon w7-2495X processor would require 1.5 years to complete the task, whereas the grid-side LLSC Intel Xeon 64-core processors would accomplish it in just over a week, approximately a 75x improvement in runtime.

## V. Conclusion

The runtime performance of a distributed signal cross-correlation algorithm was evaluated on the MIT LLSC cluster. By developing the algorithm to be compatible with a grid architecture, a massive performance increase on the order of several magnitudes was achieved. Grid computing has enabled tractable signal cross-correlation which has potential applications across communication systems, pattern recognition, signal processing, and other areas in science and engineering. To the best of our knowledge, this is the first successful implementation of a supercomputer-supported FFT-based cross-correlation algorithm.

## VI. Acknowledgments

The authors wish to acknowledge the following individuals: R. Haack, A. Reuther, J. Mullen, C. Byun.

## References

[1] J. G. Proakis and D. G. Manolakis, "Digital Signal Processing," Pearson Education Inc, 2021: pp. 87.

[2] Stein, Seymour, "Algorithms for ambiguity function processing," IEEE Transactions on Acoustics, Speech, and Signal Processing 29, no. 3 (1981): 588-599.

[3] MIT Lincoln Laboratory Supercomputing Center, "Welcome to the Lincoln Laboratory Supercomputing Center," 2023. [Online]. Available: https://www.ll.mit.edu/r-d/cyber-security-and-information-sciences/lincoln-laboratory-supercomputing-center