URSI AP-RASC, Sydney, Australia, 17 – 22 August 2025

GPU-accelerated Scan Statistics With A Case Study In Radio Astronomy

Nimalan Mahadevan(1), Wasim Raja*(2), and Harshal Hayatnagarkar(1)
(1) Engineering for Research (e4r), Thoughtworks, Pune, India
(2) CSIRO Space & Astronomy, P.O. Box 76, Epping, NSW 1710, Australia

Abstract

Outlier detection in data with high variability and with clustering present in multiple scales requires using scan statistics —a process where statistical properties of the data are computed within appropriately sized moving windows that allows detection of outliers (signal) in the presence of bias or background. While this technique has been effectively employed in diverse areas, in this paper we address the computationally expensive case of deriving scan statistics (in particular, median filters) when detecting and cataloging astronomical sources in big-data radio astronomy images such as those produced by the Australian Square Kilometre Array Pathfinder (ASKAP) telescope. We present a GPU-optimized implementation where we introduce a new versatile abstract machine namely **Network for Scan Statistics**. While high performance computing (HPC) strategies have been adopted in advanced source-finding software packages like **Selavy**, we demonstrate significant additional improvement (3x) in processing times using our novel methods that employ hardware acceleration, while using fewer resources. This speed-up, compared to the widely used single CPU approach provided in the core libraries of the Common Astronomy Software Applications (CASA), is 229x. We believe that our approach could find applications beyond source-finding and radio astronomy.

1 Introduction

Modern telescopes such as the Australian Square Kilometre Array Pathfinder (ASKAP) [1], and future telescopes like the Square Kilometre Array (SKA) produce wide-field of view images. The ASKAP source finder **Selavy** [2] runs several complex compute and data-intensive operations on these large images to reliably detect and characterize astronomical sources in images in the presence of spatially varying background noise, and instrumental and imaging artifacts. This process involves the usage of scan statistics where robust measures such as median and Median Absolute Deviation from Median (MADFM) are used. These are memory intensive, especially for large wide-field images. Efficient ways of calculating median filters have been studied, but most of such work has been done for small-windowed filters. The diverse range of algorithms used in Selavy makes it an ideal case for studying performance bottlenecks, and exploring performance enhancement strategies in the context of big data in the SKA era. In this paper, we first describe the optimizations we implemented for a CPU based large window median filter from Casacore [3]. To scale this further and to optimize for resource utilization, we explore hardware accelerated approaches. We describe an abstract device **Network for Scan Statistics** implemented as a high-performance GPU-accelerated algorithm with high memory-throughput while performing large-windowed median based scan statistics. In comparison to the original CPU based algorithm we were able to achieve an efficiency enhancement of **229x** on a data center grade NVIDIA Tesla A100 GPU (see Figure 2).

2 Source Finding

Selavy adapts the `slidingArrayMath` method provided by Casacore to compute the required scan statistics, along with Casacore `MaskedArray` to mask unwanted pixels in the image. `slidingArrayMath` faces challenges in scaling for large array sizes and window sizes. For example, computing both maps for a 1024×1024 `MaskedArray` (with 75% valid pixels) takes **498 seconds** on an AMD EPYC 7551 CPU. For comparison of the execution times across different array sizes see Table 1. Casacore's `slidingArrayMath` scans the input image twice —Once to compute the mean map, and a second time to compute the mean-removed noise maps. Our approach simplified this to using a single scan by reusing the mean map to compute the noise map[1]. This was then specialized for computing Median and MADFM for 2D arrays, while also optimizing the way `MaskedArray` is used. Since windows are independent of one another, the algorithm is highly parallel in nature. We used **OpenMP** [5] for declarative multi-threading on multi-core architectures. This gave a performance improvement of **4.8x** for two threads, and **9.6x** for four threads. These results led us to explore the problem on a GPU.

2.1 Our GPU-optimized Implementation

We investigated this problem using OpenMP offloading to target GPUs. This was slower than the serial version running on the CPU. The poor performance could be attributed to the following reasons. One, **Data fetch bottleneck** —The slowdown when offloading to a GPU was due to the instructions being stalled for several instruction cycles

[1] A similar approach was already employed for Selavy's predecessor **Duchamp** [4] but not in Selavy

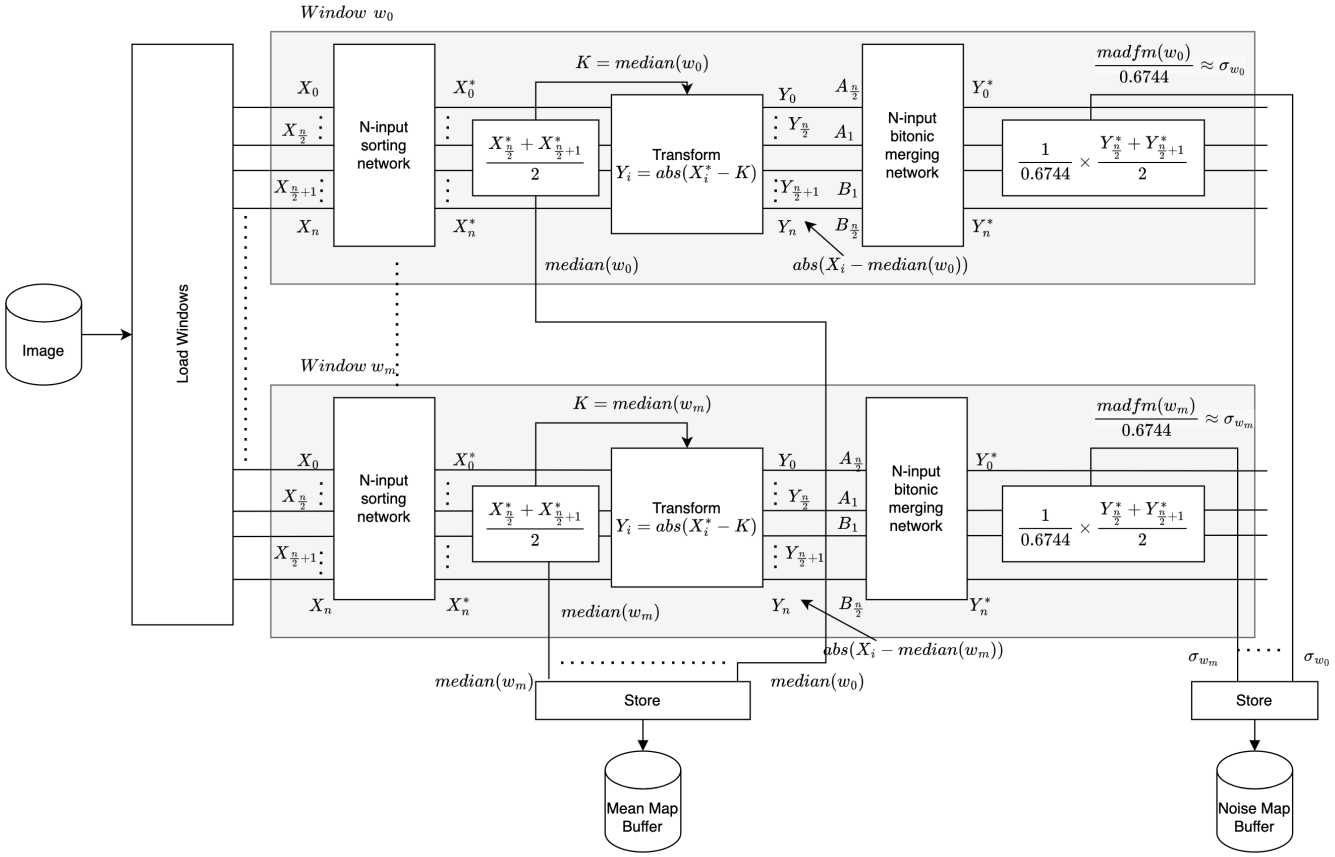


Figure 1. Scan statistical network for calculating mean map and noise map using robust statistical measures.

waiting for data to be fetched from device memory to the GPU core. Simultaneous access of cache by a lot of threads makes it a performance bottleneck. Reducing the number of threads would improve memory bandwidth but would cause a drop in the degree of parallelism or occupancy. Two, **Complexity of Sorting** —In the case of the median filter fitting multiple windows within the cache would not be possible, leading to cache evictions and read/write to the device memory which has a higher latency.

To address both concerns, we looked at the sorting networks as an effective solution. In the context of median filters, the use of optimized separable sorting networks [6] has been explored for window sizes up to 41×41. However, for source finding, a range of large window sizes (101×101 and beyond) are needed depending on the nature of the source and there is a need compute multiple statistics. Hence, the approach described in [6] is not directly suitable for source finding. We extend the idea of a sorting network ([7], [8]) to define a **Network for Scan Statistics** or **Scan Statistics Network**. It is an abstract machine that takes windows $\{W_0, W_1 ... W_n\}$ of a scan, flattens it as an input sequence for a network, and computes statistics using statistics modules present in it. Networks consist of input wires/lines and modules. Modules contain combinational and sequential logic, which act on input lines to produce intermediate results and/or output sequences, and that can be extracted from the network. When the windows of a scan are flattened, then the problem of computing scan statistics is simplified to calculating piecewise statistics on sequential groups of input lines.

Figure 1 shows our implementation of Selavy's threshold calculation with *slidingArrayMath* as a scan statistic network. First, we flatten the data in all the 2D windows of the sky image as a 1D array. This array is loaded as input lines to the network. A group of n input lines $\{X_0, X_1, .. X_n\}$ represent a window. Each group is passed through a series of modules, with independent groups processed in parallel. Second, a piecewise bitonic sorter is used to sort the input lines. From the sorted input lines of the window $\{X_0^*, X_1^*, .. X_n^*\}$, the input lines corresponding to $X_{\frac{n}{2}}^*$ and $X_{\frac{n}{2}+1}^*$ of each group are averaged[2] to get the median, which is then extracted to the *mean map* buffer (see Figure 1). Third, the median derived in the preceding step is used to calculate the absolute deviation from the median of the input lines. The resultant sequence $\{Y_0, Y_1, ... Y_n\}$ is a circular shifted bitonic sequence which could also be represented as $A_{\frac{n}{2}} \geq .. \geq A_1 \leq B_1 \leq .. \leq B_{\frac{n}{2}}$. A piecewise merging network is used to merge the bitonic sequence. This simplifies the network as the merging network needs only half the number of stages needed by a sorting network. Averaging[3] $Y_{\frac{n}{2}}^*$

[2] If the number of input lines in a window is odd, then we skip the intermediate averaging module and extract the median directly

[3] See footnote 2

and $Y^*_{\frac{n}{2}+1}$ gives the MADFM.

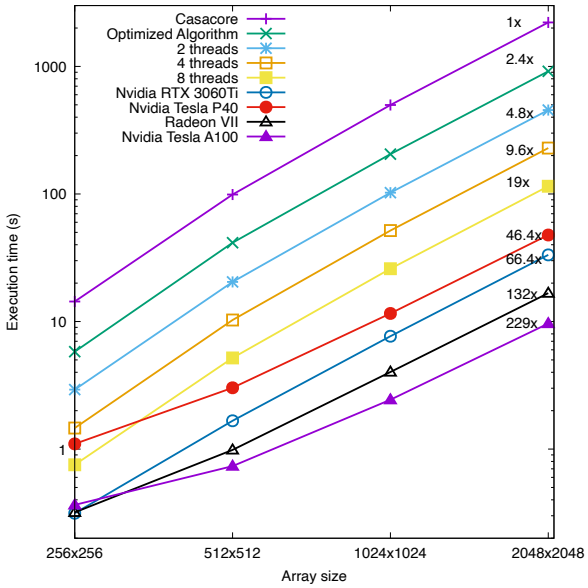


Figure 2. Performance of GPU implementation compared with Casacore and optimized multi-threaded version (in log scale).

3 Experiments and Results

Sliding window median and MADFM were calculated on a Casacore `MaskedArray` with 75%[4] valid pixels. Figure 2 overlays the performance curves for all the runs. The summary of the comparison is shown in Table 1. The key insights are —One, for a 2048×2048 grid using a single scan for both statistics is about **2.4x** faster than the base performance. This scales with better performance for additional threads used. The speedup for 2, 4, and 8 threads linearly improves by **4.8x**, **9.6x**, and **19x** respectively. Two, the data center-grade GPUs NVIDIA Tesla P40 and NVIDIA Tesla A100 show performance enhancements of **46.4x** and **229.4x** respectively. Consumer grade NVIDIA RTX 3060 Ti and AMD Radeon VII perform **66.4x** and **132x** better than the baseline. Three, the latency of launching kernels dominates for small grid sizes (for example 256×256) over computing time. As the size of the grid grows larger, the computation time starts to compensate for the latency of the kernels. Four, the performance of the GPU-optimized implementation is limited by the memory bandwidth. GPUs with memory based on **High Bandwidth Memory (HBM)** technology outperform GPUs with **Graphics Double Data Rate (GDDR)** technology.

We used Selavy to analyze three observations from the EMU Survey [9, 10], namely SBID33442[5], SBID40625, and SBID40905. Table 2 summarizes the characteristics of these observations. Figure 3 shows the overall runtime for the ASKAP source-finder Selavy versus the number of CPUs used. Key insights are —One, since Selavy is an MPI application, it achieves parallelism by sub-dividing the image into multiple regions, with fewer MPI workers, Selavy takes more time to process. The optimized version of Selavy when run single-threaded performs **1.4x —1.9x** times compared to the baseline. Two, we experimented with different MPI configurations such that each MPI process has two OpenMP threads (4, 9, 16, 36 processes using CPU cores 8, 18, 32, 72 respectively). The speed-up across these configurations when comparing against MPI-only configurations using a similar number of CPU cores (16, 32) ranges between **1.2x—1.9x**. Similarly, the speed-up for four threads per MPI worker (4, 9, 16 processes using CPU cores 16, 36, 64 respectively) when compared with MPI-only configurations ranges between **1.2x—1.8x**. Finally, for the GPU implementation a MPI + GPU setup was used with one NVIDIA A100 GPU shared amongst all the MPI workers. The speed-up in this case ranges from **2.4x—7.6x** across all the MPI configurations. When 16 workers + 1 GPU is compared with Selavy's common configuration of 36 workers, the speed-up ranges between **1.4x—3.0x**.

[4] 75% is an estimate from the average in different images.

[5] SBID - Unique observatory identifier for a scheduled observation

4 Conclusion and Future Work

We could demonstrate that our GPU-optimized implementation of the core scan statistics algorithm performs **229x** faster than the native serial implementation. We also demonstrate how this optimization improves the overall run-time by **3x**. Going forward, we intend to integrate these implementations in ASKAP operations. Selavy could be further optimized for the use of multiple GPUs to process very large images, for example the mosaics stitched out of multiple observations. We believe that this technique has applications beyond source finding and radio astronomy.

References

[1] D. R. DeBoer, R. G. Gough, J. D. Bunton, T. J. Cornwell, R. J. Beresford, S. Johnston, I. J. Feain, A. E. Schinckel, C. A. Jackson, M. J. Kesteven, A. Chippendale, G. A. Hampson, J. D. O'Sullivan, S. G. Hay, C. E. Jacka, T. W. Sweetnam, M. C. Storey, L. Ball, and B. J. Boyle, "Australian ska pathfinder: A high-dynamic range wide-field of view survey telescope," *Proceedings of the IEEE*, vol. 97, pp. 1507–1521, 2009.

[2] M. Whiting and B. Humphreys, "Source-finding for the australian square kilometre array pathfinder," *Publications of the Astronomical Society of Australia*, vol. 29, no. 3, pp. 371–381, 2012.

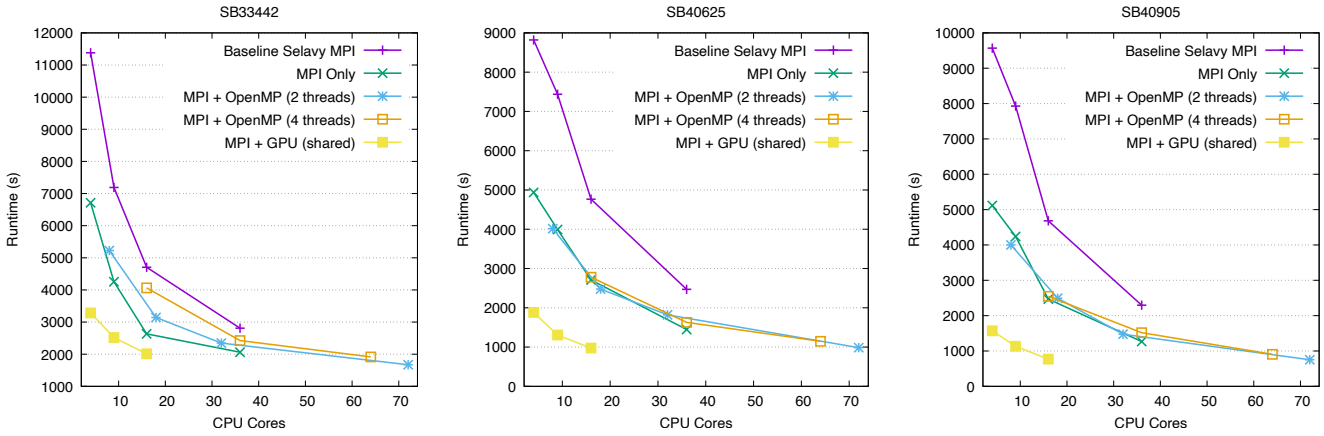
[3] Casacore Team, "casacore: Suite of C++ libraries for radio astronomy data processing." Astrophysics Source Code Library, record ascl:1912.002, Dec. 2019.

Table 1. Comparison of execution time for GPU-optimized `slidingArrayMath` approaches for different array sizes.

Grid Size	Time in seconds (Lower is better)				
	Baseline Casacore	**Tesla P40**	**RTX 3060Ti**	**Radeon VII**	**Tesla A100**
256×256	14.35 s	1.09 s	0.31 s	0.31 s	0.36 s
512×512	99 s	3.02 s	1.66 s	0.98 s	0.73 s
1024×1024	498.42 s	11.56 s	7.66 s	4.02 s	2.43 s
2048×2048	2212.56 s	47.67 s	33.30 s	16.69 s	9.64 s

Table 2. Image characteristics of observations from the EMU survey used for the benchmark.

Observation	Image Dimensions	Size (MB)	Number of sources	Masked Pixels (%)
SB33442	15050×13148	754.85	15396	32.9
SB40625	15438×13526	796.57	11632	40.4
SB40905	14494×14491	801.22	16550	46.7



(a) Observation SB33442 **(b)** Observation SB40625 **(c)** Observation SB40905

Figure 3. Runtimes of baseline and optimized implementations of Selavy for observations SB33442, SB40625, and SB40905 from the EMU survey.

[4] M. T. Whiting, "duchamp: a 3d source finder for spectral-line data," *Monthly Notices of the Royal Astronomical Society*, vol. 421, pp. 3242–3256, mar 2012.

[5] L. Dagum and R. Menon, "Openmp: an industry standard api for shared-memory programming," *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[6] A. Adams, "Fast median filters using separable sorting networks," *ACM Transactions on Graphics (TOG)*, vol. 40, pp. 1 – 11, 2021.

[7] K. E. Batcher, "Sorting networks and their applications," in *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), (New York, NY, USA), p. 307–314, Association for Computing Machinery, 1968.

[8] D. E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 1998. Section 5.3.4: Networks for Sorting.

[9] R. P. Norris, J. D. Marvil, J. D. Collier, A. D. Kapińska, A. O'Brien, L. Rudnick, H. Andernach, J. Asorey, M. J. I. Brown, M. Brüggen, E. J. Crawford, J. English, S. F. ur Rahman, M. D. Filipović, Y. A. Gordon, G. Gürkan, C. Hale, A. M. Hopkins, M. T. Huynh, K. HyeongHan, M. J. Jee, B. S. Koribalski, E. Lenc, K. J. Luken, D. Parkinson, I. Prandoni, W. Raja, T. H. Reiprich, C. J. Riseley, S. S. Shabala, J. R. Sheil, T. Vernstrom, M. T. Whiting, J. R. Allison, C. S. Anderson, L. Ball, M. E. Bell, J. D. Bunton, T. J. Galvin, N. Gupta, A. Hotan, C. E. Jacka, P. J. Macgregor, E. K. Mahony, U. D. Maio, V. A. Moss, M. Pandey-Pommier, and M. A. Voronkov, "The evolutionary map of the universe pilot survey," *Publications of the Astronomical Society of Australia*, vol. 38, 2021.

[10] R. Norris, M. Filipovic, M. Huynh, A. Hotan, L. Rudnick, P. Manojlovic, G. Gurkanuygun, D. Parkinson, and P. Macgregor, 2019. CSIRO. Data Collection.