SoftwareX 12 (2020) 100549

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

AMBER: A real-time pipeline for the detection of single pulse astronomical transients

A. Sclocco *, S. Heldens, B. van Werkhoven

Netherlands eScience Center, Science Park 140, 1098 XG Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 13 December 2018
Received in revised form 25 March 2020
Accepted 12 June 2020

Keywords:
Radio astronomy
Fast radio bursts
Transients

ABSTRACT

Detecting single-pulse astronomical transients, such as pulsars or fast radio bursts, requires collecting and processing enormous amounts of data. AMBER is a software processing pipeline that makes it possible to detect these single-pulse phenomena in real-time. The software achieves this by offloading compute-intensive kernels to many-core accelerators. Additionally, AMBER automatically tunes these kernels to achieve high performance on a variety of different platforms. We therefore see AMBER as an important tool in the search for new and interesting astronomical transients.

Code metadata

Current code version	3.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2018_248
Legal Code License	Apache License, Version 2.0
Code versioning system used	git
Software code languages, tools, and services used	C++, OpenCL, OpenMP
Compilation requirements, operating environments & dependencies	Linux, OpenCL library, CMake
If available Link to developer documentation/manual	
Support email for questions	a.sclocco@esciencecenter.nl

1. Motivation and significance

Transient astrophysical phenomena, such as pulsars [1] and fast radio bursts (FRBs) [2] are important probes for space–time. In the years since their first discovery, pulsars have been used as proof for Einstein's theory of gravitation [3], and there have been proposals to use them for the navigation of spacecrafts [4]. More recently, both pulsars [5] and FRBs [6] have become tools in the hunt for gravitational waves.

However, finding new and interesting transients is a challenging endeavor. To find them, it is necessary to monitor large areas of the sky, collecting and processing an enormous amount of data [7]. Moreover, for mostly non-repeating events such as FRBs, all this processing must be done in real-time, in order to store high-resolution data and trigger follow-up observations with other instruments operating at different frequencies.

Although today multiple FRB searching pipelines are available, including ALFABURST [8], bonsai [9], and FREDDA [10], this was not the case at the time of the Apertif Radio Transient System (ARTS) [11] design, when the only available pipeline was Heimdall [12]. Because of concerns regarding Heimdall's performance and portability, and based on previous work on dedispersion [13], the ARTS team began the design and development of what will later become AMBER.

AMBER, or the "Apertif Monitor for Bursts Encountered in Real-time", is a software pipeline designed and developed to make it possible to detect single-pulse transients in real-time. AMBER is open-source, fully parallelized, and runs on many-core

* Corresponding author.
E-mail addresses: a.sclocco@esciencecenter.nl (A. Sclocco), s.heldens@esciencecenter.nl (S. Heldens), b.vanwerkhoven@esciencecenter.nl (B. van Werkhoven).

https://doi.org/10.1016/j.softx.2020.100549

accelerators to achieve high performance. AMBER's high performance is not only the result of parallelization, but also of the extensive use of auto-tuning. In fact, all of AMBER's computational kernels can be tuned to achieve the best possible performance on the hardware the pipeline is executed on; this application of auto-tuning also makes AMBER performance portable among different platforms, such as Graphics Processing Units (GPUs) from different vendors.

Although AMBER has initially been developed in the context of the ARTS survey, it is generic enough to be used for other surveys. In this paper, we present AMBER's functionalities and its software architecture in Section 2, while describing its use in the ARTS survey in Section 3. Section 4 highlights some of AMBER's results, and the differences between this and other existing FRB pipelines; finally, we draw our conclusions in Section 5.

2. Software description

AMBER is a software pipeline for real-time detection of single pulse astronomical transients in the time domain. The pipeline's input is a multi-beam data stream containing channelized time series, one per beam; a beam represents one pointing of the telescope. During the processing of this stream, a two dimensional search space is explored to look for single pulses with a Signal to Noise Ratio (SNR) higher than a user specified threshold. The dimensions of this search space are (1) Dispersion Measure (DM) and (2) pulse width.

Currently, the AMBER's pipeline is composed of four computing stages: (1) *dedispersion*, (2) *integration*, (3) *SNR computation*, and (4) *thresholding*. Communication between these different stages happens through memory buffers, and Fig. 1 shows the application's data path from input to output. Each stage takes as input the output of the previous stage, applies some transformation to the data, and makes its output available for the next stage.

Because the different frequencies emitted by a radio source are dispersed over time by the interaction of the emission photons with the free electrons present in the interstellar medium, detecting single pulse signals from Earth is challenging, unless the distance of the emitting source from the receiver is known in advance. Unfortunately, this distance is only known for known emitting sources, and must be guessed for new objects. The first stage of AMBER, called *dedispersion*, is the one responsible for guessing this parameter, i.e. the dispersion measure (DM). An arbitrary, and user controlled, number of these DMs can be searched at this stage, and AMBER implements two different search strategies: *direct* and two-steps *subband* dedispersion. The implementation of this stage is described in further detail in [13].

The second stage of AMBER, *integration*, is the one responsible to explore the second dimension of the search space, i.e. the width of the emitted pulse in the time dimension. In this stage, the dedispersed time series produced by *dedispersion* are integrated and averaged over time, for a set of user-defined integration steps. Each integration step is a guess at the width of the emitted pulse, and the closer the guess is to the real width, the higher the SNR of the corresponding integrated time series will be. Another advantage of this stage is that, by downsampling data in the time dimensions, it is possible to get rid of single sample outliers that are probably caused by Radio Frequency Interference (RFI) or system malfunctions.

In the *SNR computation* stage we look for statistically significant peaks in the time series produced by the previous stages, and do so by computing the SNR of each time series. In more detail, for each dedispersed and integrated time series, this stage computes a single value defined as $(x - \mu)/\sigma$, where x is the value of the sample with the highest intensity in the series, and μ and σ are, respectively, the mean and standard deviation.

The last stage of AMBER is *thresholding*. In this stage, the two dimensional matrix containing all SNR values produced in the previous stage is scanned to select all values that exceed some user-specified threshold. These values, together with the coordinates necessary to unequivocally identify them, such as beam, DM, width, and arrival time, are then written to disk in AMBER's main output file. It is also possible to cluster these candidate detections in both DM and width dimensions, to reduce the number of entries in the output file.

2.1. Software architecture

AMBER is implemented in C++, with each stage being a separate library with its own repository and development process. The computational kernels of the first three stages, i.e. *dedispersion*, *integration*, and *SNR computation*, are implemented in both C++ and OpenCL, with the OpenCL implementation designed to run efficiently on GPUs and other many-core accelerators; the last stage, i.e. *thresholding*, is only implemented in C++. Fig. 2 shows AMBER's main loop and the standard assignment of the computing stages to execution platforms, with the first three stages running on the GPU, and the last and least computationally intensive one running on the CPU. With this partitioning, before each iteration of AMBER's main loop one memory transfer from host to GPU is performed to copy the necessary data to the input buffer. After the third computing stage, the matrix containing the SNR values is copied from the GPU to the host.

2.1.1. Auto-tuning

The OpenCL kernels require many low-level parameters, such as the OpenCL work-group size, the usage of local memory, or the amount of serial processing per thread, in order to function properly. These parameters also fundamentally impact the execution time of each stage, and a proper selection of their values is necessary to achieve high performance. However, each OpenCL kernel is likely to have a different optimal configuration, and this configuration depends on the kernel itself, its execution platform, and characteristics of the input, such as the input dimensions. For this purpose, each module contains an auto-tuning utility that, provided a description of the observational scenario, tests all valid configurations for the given kernel, on the desired execution platform, and selects the fastest. A more detailed description of the auto-tuning process for *dedispersion* can be found in [13]. Because of auto-tuning, the computational stages of AMBER achieve high-performance on a variety of many-core accelerators, making AMBER both portable and high-performance.

2.2. Software functionalities

This section explains how the user can interact with AMBER.

2.2.1. Input

AMBER is controlled through its command line interface, with the data types being the only parameter that must be set before the compile phase. The software supports three distinct input sources: (1) filterbank [14] files, (2) HDF5 [15] files, and (3) PSRDADA[1] buffers. Moreover, AMBER also provides a simple data generator that can be used to test and benchmark the pipeline. Filterbank files and PSRDADA buffers can be processed in a real-time streaming mode, while for HDF5 files only batch processing is supported. The tuned configurations of all computational kernels are passed to AMBER through configuration files, one per kernel. Two more configuration files are necessary for the pipeline to work correctly: one file containing a list of integration steps, and a file containing a list of tainted input frequency channels to exclude from computation. This last feature is used to remove previously known sources of RFI.

[1] http://psrdada.sourceforge.net.

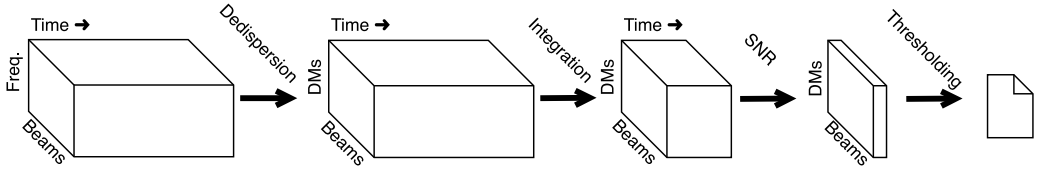

Fig. 1. Overview of AMBER's stages and the data path between them.

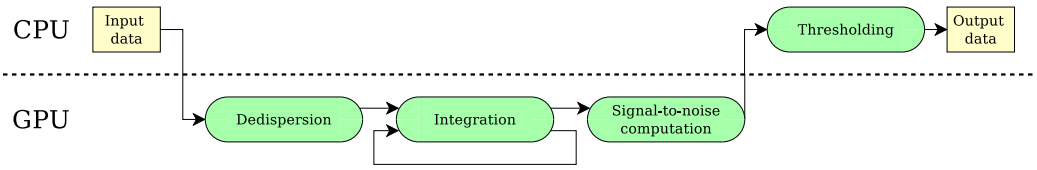

Fig. 2. Processing stages of AMBER and their execution platform: CPU or GPU.

2.2.2. Output

AMBER produces two output files after each successful execution: (1) a file containing candidate detections, and (2) a file containing performance statistics on the run. The file containing the candidate detections contains, for each entry, all the coordinates necessary to identify the detection, such as beam, DM, time of detection, and pulse width, together with the SNR of the detected signal. If the results compacting mode is enabled, AMBER clusters the results in two dimensions (i.e. DM and pulse width), and outputs only the candidate with the highest SNR of each cluster. The file containing performance statistics includes the execution time of many of AMBER components, and can be used to monitor the performance of each run of the pipeline.

2.2.3. Scripts

A script[2] is available to help users install and maintain a working AMBER installation. Provided with a description of the observational and search parameters, the script can be used to install and update AMBER, and more importantly it can be used to generate and test auto-tuned configurations for all computational stages. The documentation of the script includes a demo showing how to install, tune, and use AMBER to search a Filterbank file.

3. Illustrative examples

AMBER is currently being used to perform real-time FRB searches within the ARTS survey [16]. Time-domain observations are scheduled every three weeks, and each round of observations is normally divided into runs of 3 h, each pointing in a different direction. The input data rate during operations amounts to 9.2 GBs per second, for a total of 99 TBs for a single run, and almost 4 PBs during a standard observing week of 120 h.

The search space is composed by 5120 DMs and 6 integration steps. These observations are being processed in real-time on the 40 nodes of the ARTS cluster, using 3 of the onboard GPUs; each node processes 12 of the 480 input beams, while each GPU processes roughly one third of the DM space, and all the integration steps. The remaining GPU is used to run a post-processing pipeline that ingests AMBER's output and sends alerts to astronomers and follow-up instruments.

4. Impact

AMBER is already enabling the astronomers involved in the ARTS survey to perform real-time FRB searches, as described in Section 3. More interestingly, ARTS observations are already producing scientific results, such as multiple detections of the repeating source FRB 121102 [18,19], and the detection and localization of FRB 191108 [17]; a plot of this FRB is provided in Fig. 3. And while AMBER is deployed in production, new features are being developed and new developers are getting involved. We believe that the modular structure of AMBER will facilitate its maintenance and evolution, and we already know that two new modules are being currently developed and will be added to the pipeline in a future release: a new SNR computing stage, and RFI mitigation [20].

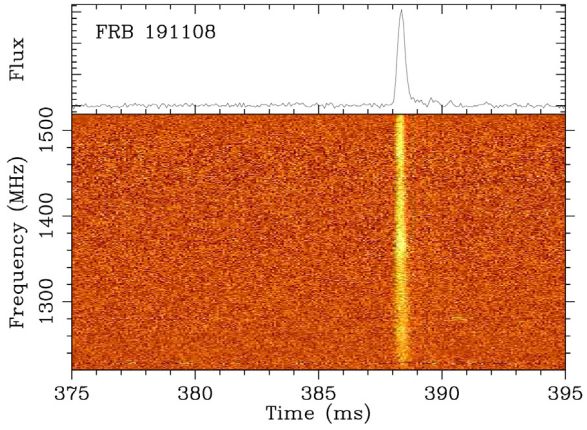

Fig. 3. FRB 191108 (courtesy of Connor et al. [17]), one of the new FRBs discovered using AMBER.

One of the main challenges for future high-resolution surveys will be processing an ever increasing amount of input data in real-time. In this respect, the high-performance provided by AMBER's accelerated kernels will play a key role not only for ARTS, but for future surveys as well. Fig. 4 shows a comparison of the execution time of the *dedispersion* stage, one of the most time consuming operations in transient searches, in AMBER and Heimdall; the experiment consists in the processing of a single input beam composed of 20,000 time samples and 1024 channels, with 300 MHz of bandwidth centered around a frequency of 1425 MHz. Not only AMBER's *dedispersion* stage is 2.5 times faster than Heimdall when running on the same hardware (a NVIDIA K20X GPU in this case), but AMBER can also run on different hardware and achieve an execution time that is one order of magnitude lower. This is the result of the inherent portability of AMBER, and we believe that this feature will not only extend the lifespan of this pipeline, but will also make AMBER an option for other surveys.

[2] https://github.com/AA-ALERT/AMBER_setup.

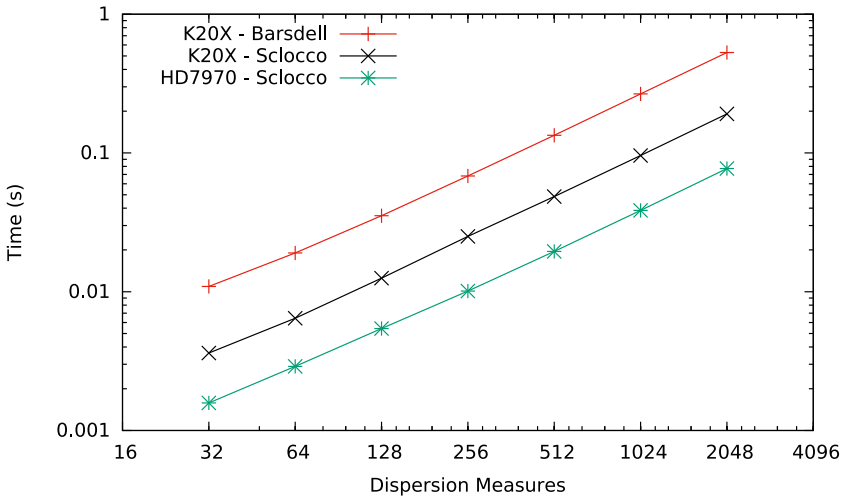

Fig. 4. Execution time of AMBER and Heimdall *dedispersion* stages (lower is better).

5. Conclusions

In this work, we presented AMBER: a software pipeline for detection of single pulse astronomical transients such as FRBs. AMBER achieves high performance by offloading work to many-core accelerators and makes extensive use of auto-tuning to obtain the best performance on different hardware platforms.

While AMBER is being used in production within the ARTS survey, and it is already contributing to the discovery of new FRBs, we also expect it to be adopted by other future surveys, and to contribute to the discovery of new and interesting astronomical phenomena.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the Netherlands eScience Center under the project AA-ALERT: Access and Acceleration of the Apertif Legacy Exploration of the Radio Transient Sky, filenumber 027.015.G09. We would like to thank the whole ARTS survey team, and in particular Joeri van Leeuwen, Leon Oostrum, Liam Connor, and Dany Vohl, for their valuable contributions to this paper and AMBER.

References

[1] Hewish A, Bell SJ, Pilkington JDH, Scott PF, Collins RA. Observation of a rapidly pulsating radio source. Nature 1968;217(5130):709–13, URL http://dx.doi.org/10.1038/217709a0.

[2] Lorimer DR, Bailes M, McLaughlin MA, Narkevic DJ, Crawford F. A bright millisecond radio burst of extragalactic origin. Science 2007;318(5851):777–80. http://dx.doi.org/10.1126/science.1147532, arXiv:http://www.sciencemag.org/content/318/5851/777.full.pdf, URL http://www.sciencemag.org/content/318/5851/777.abstract.

[3] Hulse R. The discovery of the binary pulsar. In: Bulletin of the American astronomical society, Vol. 26. 1994, p. 971–2.

[4] Chester TJ, Butman SA. Navigation using x-ray pulsars. In: The telecommunication and data acquisition report. 1981.

[5] Kennefick D. Relativistic lighthouses: The role of the binary pulsar in proving the existence of gravitational waves. In: Rowe DE, Sauer T, Walter SA, editors. Beyond einstein: Perspectives on geometry, gravitation, and cosmology in the twentieth century. New York, NY: Springer New York; 2018, p. 111–36. http://dx.doi.org/10.1007/978-1-4939-7708-6_6.

[6] Yamasaki S, Totani T, Kiuchi K. Repeating and non-repeating fast radio bursts from binary neutron star mergers. Publ Astron Soc Japan 2018;70(3):39. http://dx.doi.org/10.1093/pasj/psy029.

[7] Broekema PC, van Nieuwpoort RV, Bal HE. Exascale high performance computing in the square kilometer array. In: Proceedings of the 2012 workshop on high-performance computing for astronomy. New York, NY, USA: ACM; 2012, p. 9–16. http://dx.doi.org/10.1145/2286976.2286982, URL http://doi.acm.org/10.1145/2286976.2286982.

[8] Chennamangalam J, Karastergiou A, MacMahon D, Armour W, Cobb J, Lorimer D, et al. ALFABURST: A realtime fast radio burst monitor for the Arecibo telescope. In: The fourteenth marcel grossmann meeting. World Scientific; 2017, p. 2872–6. http://dx.doi.org/10.1142/9789813226609_0359.

[9] Amiri M, Bandura K, Berger P, Bhardwaj M, Boyce MM, Boyle PJ, et al. The CHIME fast radio burst project: System overview. Astrophys J 2018;863(1):48, URL http://stacks.iop.org/0004-637X/863/i=1/a=48.

[10] Bannister K, Zackay B, Qiu H, James C, Shannon R. FREDDA: A fast, real-time engine for de-dispersing amplitudes. 2019, arXiv:1906.003.

[11] van Leeuwen J. ARTS – the apertif radio transient system. In: Wozniak PR, Graham MJ, Mahabal AA, Seaman R, editors. The third hot-wiring the transient universe workshop. 2014, p. 79.

[12] Barsdell B. Advanced architectures for astrophysical supercomputing (Ph.D. thesis), Swinburne University of Technology; 2012.

[13] Sclocco A, Bal HE, Hessels J, van Leeuwen J, van Nieuwpoort RV. Auto-tuning dedispersion for many-core accelerators. In: International parallel and distributed processing symposium (IPDPS). Los Alamitos, CA, USA: IEEE Computer Society; 2014.

[14] Lorimer DR. SIGPROC: Pulsar signal processing programs. Astrophysics Source Code Library; 2011, arXiv:1107.016.

[15] Folk M, Heber G, Koziol Q, Pourmal E, Robinson D. An overview of the HDF5 technology suite and its applications. In: Proceedings of the EDBT/ICDT 2011 workshop on array databases. AD '11, New York, NY, USA: ACM; 2011, p. 36–47. http://dx.doi.org/10.1145/1966895.1966900, URL http://doi.acm.org/10.1145/1966895.1966900.

[16] Maan Y, van Leeuwen J. Real-time searches for fast transients with Apertif and LOFAR. In: 2017 XXXIInd general assembly and scientific symposium of the international union of radio science (URSI GASS). 2017, p. 1–4. http://dx.doi.org/10.23919/URSIGASS.2017.8105320.

[17] Connor L, van Leeuwen J, Oostrum LC, Petroff E, Maan Y, Adams EAK, et al. A bright, high rotation-measure FRB that skewers the M33 halo. 2020, arXiv:2002.01399.

[18] Oostrum LC, van Leeuwen J, Attema J, van Cappellen W, Connor L, Hut B, et al. Detection of a bright burst from FRB 121102 with Apertif at the westerbork synthesis radio telescope, Vol. 10693. The Astronomer's Telegram; 2017.

[19] Oostrum LC, Maan Y, van Leeuwen J, Connor L, Petroff E, Attema JJ, et al. Repeating fast radio bursts with WSRT/Apertif. Astron Astrophys 2020;635:A61. http://dx.doi.org/10.1051/0004-6361/201937422.

[20] Sclocco A, Vohl D, van Nieuwpoort RV. Real-time RFI mitigation for the apertif radio transient system. In: 2019 RFI Workshop - Coexisting with Radio Frequency Interference (RFI). 2019, p. 1–8.