A STREAMING APPROACH TO RADIO ASTRONOMY IMAGING

Alain Biem, Bruce Elmegreen, Olivier Verscheure, Deepak Turaga, Henrique Andrade, Tim Cornwell[1]

IBM T.J. Watson Research Center, NY, USA
[1]CSIRO Australia Telescope National Facility, PO Box 76, Epping,NSW, 1710, Australia
{biem,bge,ov1,turaga,hcma}@us.ibm.com

ABSTRACT

The emergence of large-field radio telescopes has generated the need to process huge amounts of data "on the fly" in order to avoid substantial costs in storage and signal processing. In this paper, we describe a streaming software approach to radio astronomy, and in particular, show how the IBM-developed stream computing architecture *System S* enables seamless "on the fly" radio astronomy imaging. This approach overcomes pertinent issues of memory limitation and latencies related to data processing, and it allows autonomous processor allocation to tasks as the need arises.

Index Terms— Radioastronomy, imaging, convolutional resampling, System S

1. INTRODUCTION

Capturing radio signals has conventionally been done with a standard radio telescope that uses a large parabolic dish antenna connected to a central processing location where analog electronic signals are digitized [1]. There is a shift towards fixed phased arrays and focal plane arrays on dish telescopes, where arrays consist of antennas connected by high speed communication links. The large number of antennas in a typical array, and the large number of beams for each antenna, lead to an enormous rate of data collection. For instance, LOFAR's projected 77 antenna stations spread across a 10000 square-km area will generate data at a rate of around 37 Tbit/sec [2] . This requires 116 Tfops for local processing and 43 Tflops at the central processing unit. Such a large amount of data needs to be processed on the fly. Although techniques exist to handle such volumes, including new calibration approaches, data replay buffers, and grid integrations, this is done at the cost of storage, processing time, and infrastructure. Even the modest goal of continuous cosmic ray detection and analysis is not fully within the reach of current computing systems. Such problems call for a new approach in dealing with high-volume, analysis-intensive space data.

In this paper, we argue for a streaming approach to radio astronomy and demonstrate the validity of the technique to a core application, imaging. Radio astronomy imaging maps radio emission from the sky over a wide range of frequencies. Imaging with streaming data is of particular importance to next-generation radio telescopes like ASKAP, MeerKAT and the Square Kilometer Array [3]. We show that a streaming approach to imaging enables one to overcome latencies and memory limitations, and permits the non-stop use of omni-directional antennas for instantaneous imaging of wide fields of view. This work further adds to the recent trend of using streaming techniques to earth science applications [4, 5].

2. RADIO ASTRONOMY IMAGING

A synthesis imaging radio telescope is usually a large number of radio elements (e.g. antennas, amplifiers, beam formers, analog-to-digital converters, processors and software), which build up an image over time as the Earth turns [1]. Signals from the antennas are multiplexed by the interferometer. The output of the interferometer $V(u,v,w)$, called visibility, is the time-averaged cross-correlation of the electric fields measured at two separate antenna locations; the coordinates (u,v) measure the distances in wavelengths between the two antenna projected onto a plane perpendicular to the direction of the source and the coordinate w measures the component of distance in the direction of the source. Visibility data are obtained by taking measurements for all possible B pairs. The number B is refered to as a baseline.

The image, $I(l,m)$, or sky brightness, is related to visibility by the following equation [1]:

$$V(u,v,w) = \int \frac{I(l,m)}{\sqrt{(1-l^2-m^2)}} G_w(l,m) e^{-2\pi i[ul+vm]} dldm \quad (1)$$

where $G_w(l,m) = e^{-2i\pi[w(\sqrt{1-l^2-m^2}-1)]}$. Equation (1) is the Fourier transform of the product of $G_w(l,m)$ and $I(l,m)/\sqrt{(1-l^2-m^2)}$, which is the Fourier Transform of $V(u,v,w=0)$. (l,m) are direction cosines relative to the direction of the sources. Radio astronomy imaging consists of estimating $I(l,m)$ using a finite set, $k = 1,..,N_s$, of measured visibilities $V(u_k,v_k,w_k)$, which leads to two key problems: (a) inverting Eq. (1) to get I from V and (b) minimizing the bias introduced by using a finite number N_s of measured samples.

These are the two key issues affecting the quality of the imaging process. For a small field of view, solving problem (a) is relatively straightforward because the w-dependent term, $\sqrt{1-l^2-m^2}$), is ≈ 1. Eq (1) then becomes a two-dimensional Fourier transform of the brightness, and image-making becomes equivalent to taking an inverse Fourier Transform of the visibility.

2.1. W-projection

For a large field of view, the w-component in the equation cannot be ignored. Various algorithms have been proposed to solve this issue [6]. In this paper, we are concerned with the W-projection algorithm. This is an elegant solution to the inversion problem and requires less computation time than alternatives, but it involves a high cost in terms of memory [6]. $V(u,v,w)$ is viewed as the convolution of the two terms, namely

$$V(u,v,w) = g_w(u,v) \otimes V(u,v,w=0), \qquad (2)$$

where $g_w(u,v)$ is the inverse Fourier transform of $G_w(l,m)$.

This decomposition allows the estimation of visibility for non zero w from the visibility computed at $w=0$ by convolution with the kernel $g_w(u,v)$. For small angles, $g_w(u,v)$ is approximated by $g_w(u,v) = \frac{i}{w} e^{-\pi i(\frac{(u^2+v^2)}{w})}$. The W-projection enables one to map the 3D problem into a 2D solution, where a 2D FFT can be used for computational efficiency.

2.2. Convolution resampling

In practice, the kernel g_w is partitioned into *planes* over which g_w is roughly constant. The number P_w of these w-planes is a parameter of the system. The convolution matrix $\mathbf{C}$ of the system can then be viewed as a number of P_w juxtaposed 2D convolution matrices $\mathbf{C}_{w_j}$. A larger P_w yields a higher resolution along the w-planes axis, and a better estimate of Eq. (1), but also a require a large memory footprint for the convolution matrix.

Convolution resampling in the framework of the W-projection algorithm enables one to transform the 3-dimensional measured visibilities, $V(u_j,v_j,w_j)$ into 2-dimensional gridded visibilities, $V_g(u_k,v_k)$. $V_g(u_k,v_k)$ is computed by convolving the data with one of the kernel using a square support of size M within each $\mathbf{C}_{w_j}$ convolution matrix as:

$$V_g(u_k,v_k) = \sum_{j=1}^{M} \mathbf{C}_{w_j}(u_k-u_j, v_k-v_j) V(u_j,v_j,w_j). \quad (3)$$

The size M of the support region is typically seven or nine pixels but can reach up to 100 pixels for extra wide-field imaging.

The $\mathbf{C}_{w_j}$ matrix depends on the w_j coordinate of the data and realizes a projection of a w-dependent plane unto the $w=0$ as

$$\mathbf{C}_w = \mathbf{C}_{w=o} \otimes g_w$$

where g_w is the w-correction that compensates for non coplanar planes, as introduced in Eq (2).

From the gridded visibilities, the Fourier transform is performed, deriving an estimate $I^D(l,m)$ of the image. Further processing is needed to ensure that $I^D(l,m)$ is a reasonable estimate of $I(l,m)$ [1].

3. SYSTEM S

System S[1] [7] was developed at the IBM T. J. Watson Research Center for large-scale data-stream processing. It supports structured as well as unstructured data stream processing and can be scaled to a large collection of compute nodes, simultaneously hosting multiple applications. A data-flow graph consists of a set of processing elements (PEs) connected by streams, where each stream carries a series of stream data objects (SDOs). A PE implements data stream analytics and is the basic execution container that is distributed over computational hosts.

SPADE [8] is a stream-centric, operator-based programming language for System S.

3.1. SPADE Operators

SPADE is a scripting language that was conceived around the idea of providing toolkits of operators. Currently, a relational operator toolkit is available. The operators currently supported in the stream-relational toolkit are: **Source** A Source operator is used for creating a stream of data flowing from an external source. This operator is capable of performing parsing and tuple creation as well as interacting with external devices. **Sink**: A Sink operator is used for converting a stream into a flow of tuples that can be used by components that are not part of System S. **Functor**: A Functor operator is used for performing tuple-level manipulations such as filtering, projection, mapping, attribute creation and transformation. **Aggregate**: An Aggregate operator is used for grouping and summarization of incoming tuples. **Join**: A Join operation is used for correlating two streams. **Sort**: A Sort operator is used for imposing an order on incoming tuples in a stream. **Barrier**: A Barrier operator is used as a synchronization point.

In addition to the existing relational algebra toolkit, the language was designed to support extensions. The most important method of extension is to create User-Defined Operators (UDOPs), which are operators specialized to both wrap legacy libraries and provide customized processing. UDOPs are specialized for application-specific stream schemas.

[1] System S is being released as a product under the name IBM InfoSphere Streams or Streams.

4. A STREAMING APPROACH TO CONVOLUTIONAL RESAMPLING

In this section, we describe the use of System S in implementing the convolution resampling algorithm. We used 10000 computer-generated samples for all of our experiments but with the ASKAP radio telescope in mind. The baseline is 2km and the final grid is 512 by 512. The support size M is chosen to be consistent with values from earlier simulations [9] and fixed at 22 pixels within a convolution matrix of size 45 by 45 for a single plane and an oversampling factor of 8.

The focus here is to minimize the run time in performing the gridding itself, which is the optimal measure in term of cost and efficiency.

4.1. Conventional application

Conventionally, data are collected from the antennas after long hours. These data then undergo various signal conditioning. For imaging purposes, data go through the processes:

1. **Collection**: The visibility data are collected from the antennas after long hours of observations in the form of complex-valued $V_i = V(u_i, v_i, w_i)$ for $i = 0...N-1$ with real-valued coordinates (u_i, v_i, w_i) in meters.

2. **Frequency mapping**: The coordinates (u_i, v_i, w_i) are transformed from meter-unit into wavelength-unit $(u_i(\lambda), v_i(\lambda), w_i(\lambda))$ for a particular frequency of analysis λ. Each recorded visibility V_i is mapped to multiple $V_i(\lambda)$ where λ spans thousands of frequencies.

3. **Indexing**. Two types of indexing are considered: Indexing to the convolution matrix and indexing to the final grid. As each visibility datum uses a specific support area of size $M \times M$ within the convolution matrix, an index $\iota^c_{i,\lambda}$ to that area is precomputed prior the convolution process. This index points to both the w-plane whose kernel should be used but also the support area within that plane where convolution should be performed. Similarly, the convolution affects a specific area in the final grid where the integrated sample should be accumulated; this yields an index $\iota^g_{i,\lambda}$ to that region.

4. **Convolution and aggregation**: The data are convoluted with the convolution matrix and aggregated in the final grid as follows:

$$V_g[\iota^g_{i,\lambda} + s] += \mathbf{C}_w[\iota^c_{i,\lambda} + s] * V_i(\lambda)) \quad (4)$$

for $s = \{0, .., M\}$.

Conventionally, each of these steps is done separately. Data are gathered to a predetermined batch size. Then, the whole batch is scaled and indexed. Once the convolution matrix is computed, the gridding process is a simple series of muladd operations with index look-up. This scenario meets two key challenges: (a) memory storage of the convolution matrix as the number P_w of w-planes increases and (b) time to grid all the data across all the channels as the number of channels increases.

The first problem is primarily due to the size of the convolution matrix, which increases with the resolution along the w-planes. The second problem is inherent in computing a more accurate image.

We address both problems using System S as described in the next sections.

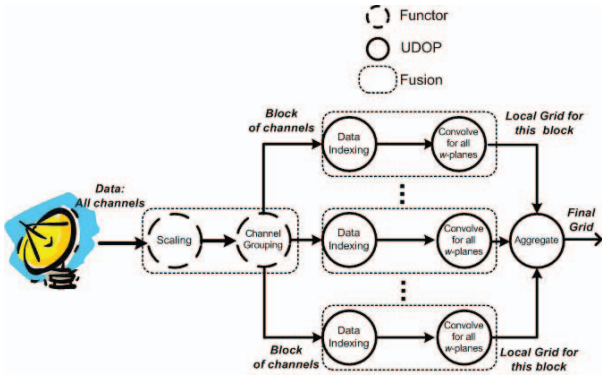

Fig. 1. The architecture for channel scalability using System S. Fusing operators minimizes transfer latencies between PEs

4.2. A streaming approach to channel scalability

The goal is to evaluate the potential of System S for improving the gridding time. Channel scalability is of prime importance in radio astronomy and being able to detect information available in a large number of channels will maximize the probability of event detection and discovery.

The proposed architecture system is shown in Fig (1). The data, sent from the source, are scaled and divided equally into blocks of channels by a slicing (or grouping) Functor. There is a set of fused operators devoted to processing each block of channels including indexing and convolution (convolutional PE or CPEs).

A Functor operator performs the frequency mapping using vectorized operation available in SPADE for all channels. The frequency are then grouped per block and all block are processed in parallel for indexing and the convolution.

We assume a fixed set of w-planes (65 in this case, including w=0 plane) and all CPEs are able to process data from any w-plane as long as those data belong to their block of channels. In addition, each CPE contains a local copy of a final grid for that data within their block of channels. This local copy of grid is passed on to the Aggregator operator, which

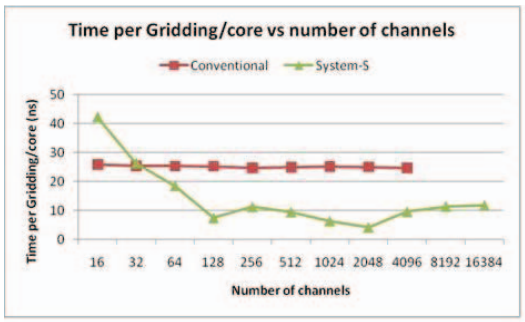

Fig. 2. Running time per grid addition for various numbers of channels per core. The number of w-planes is set to 65 (including w=0 plane)

does the final integration of all grids from all block of channels. This architecture enables one to process various channels at the same time as blocks of channels are processed in parallel; there is no idle time as each CPE is always busy processing data, which results in a significant gain in gridding rate. This is confirmed by the results shown in Fig (2), which plot gridding time as a function of the total number of channels, normalized by the number of cores.

The system was run on a rack of blade servers made of Dual-Core Xeon at 2.66 Ghz with a Linux OS. The conventional system was run on a single node of the rack while the System S approach was run on the rack. The number of nodes is automatically selected by the System S compiler under the constraint

The results for System S are best obtained after various trials. Typically, the best performance was obtained with the number of blocks of channels in the range from 2 to 8. The total number of PEs used equals to the number of blocks minus one for the PE dedicated to scaling and channel grouping and another dedicated to the Aggregator. This yields an average number of PEs of around 8.

Clearly, the streaming approach significantly decreases the gridding time: from around 25 ns for the conventional solution to 6 ns per grid addition per core in the System S solution. Furthermore, System S does not appear to have a limit in the total number of channels. As can be seen in this figure, the conventional system is unable to handle more than 4096 channels because all the work is done in a single PE and the memory in that PE is limited. Also, for the conventional system, gridding time is a linear function of the number of channels, which results in a constant curve for the time per gridding (i.e., the time normalized to the number of channels).

5. CONCLUSION

We have described a System S approach to solving a few issues pertaining to image synthesis in radio astronomy. System S architecture allows one to view data in a continuous fly-by fashion, without needing to read from or writ to enormous data files. Data is processed when it is available and only a final accumulated result needs to be stored. System S also allocates processors to the various tasks automatically, branching out to more processing elements as needed within the total computer system that is available. Using these attributes, we showed how System S can be programmed to overcome scalability issues. This is similar to the benefit obtained by parallelization with the same number of nodes, but with System S, such parallelization is done easily, on the fly as needed, and without user intervention or explicit MPI-type function calls. Timing results were discussed in details for the convolution resampling algorithm with a W-projection.

6. ACKNOWLEDGEMENTS

The authors would like to thank their colleagues Bugra Gedik for his help with the System S platform and Lisa Amini for her support in this research.

7. REFERENCES

[1] A. Thompson, J. Moran, and G Swenson, *Interferometry and synthesis in radio astronomy*, Wiley, New York, 2001.

[2] "http://www.lofar.org/," .

[3] "http://www.skatelescope.org/," .

[4] L.K.S Daldorff, Siavoush M Mohammadi, J. E. S. Bergman, B. Thide, A. Biem, B. Elmegreen, D. S. Turaga, Verscheure, and W O. Puccio, "Novel data stream techniques for real time HF radio weather statistics and forecasting," in *Proceedings of the International Conference on Ionospheric radio Systems and Techniques, 2009, IRST*, 04 2009.

[5] "IBM InfoSphere Streams and the Uppsala university space weather project," http://www.linuxjournal.com/article/10571.

[6] T.J Cornwell, K. Golap, and S Bhatnagar, "W Projection: A New Algorithm for Wide Field Imaging with Radio Synthesis Arrays," in *Proc. of Astronomical Data Analysis Software and Systems XIV ASP Conference Series*, 12 2005, vol. 347 of *2005ASPC..347...86C*, p. 86.

[7] Navendu Jain, Lisa Amini, Henrique Andrade, Richard King, Yoonho Park, Philipe Selo, and Chitra Venkatramani, "Design, implementation, and evaluation of the linear road benchmark on the stream processing core," in *International Conference on Management of Data, ACM SIGMOD*, Chicago, IL, 2006.

[8] Bugra Gedik, Henrique Andrade, Kun-Lung Wu, Philip S. Yu, and Myungcheol Doo, "SPADE: The System S declarative stream processing engine," in *International Conference on Management of Data, ACM SIGMOD*, Vancouver, Canada, 2008.

[9] T.J. Cornwell, "The Impact of Convolutional Resampling on ASKAP and SKA Phase 1 Computing Cost," Technical Report CONRAD-SW-0008, KAT-CSIRO, 2007.