Astronomy and Computing 12 (2015) 181–190

Contents lists available at ScienceDirect

Astronomy and Computing

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

Full length article

A compression scheme for radio data in high performance computing

K. Masui [a,b,*], M. Amiri [a], L. Connor [c,d,e], M. Deng [a], M. Fandino [a], C. Höfer [a], M. Halpern [a], D. Hanna [f], A.D. Hincks [a], G. Hinshaw [a], J.M. Parra [f], L.B. Newburgh [d], J.R. Shaw [a,c], K. Vanderlinde [e,d]

[a] Department of Physics and Astronomy, University of British Columbia, 6224 Agricultural Rd., Vancouver, V6T 1Z1, Canada
[b] Canadian Institute for Advanced Research, CIFAR Program in Cosmology and Gravity, Toronto, ON, M5G 1Z8, Canada
[c] Canadian Institute for Theoretical Astrophysics, 60 St George St, Toronto, ON, M5S 3H8, Canada
[d] Dunlap Institute for Astronomy & Astrophysics, University of Toronto, 50 St George St, Toronto, ON, M5S 3H4, Canada
[e] Department of Astronomy & Astrophysics, University of Toronto, 50 St George St, Toronto, ON, M5S 3H4, Canada
[f] Department of Physics, McGill University, 3600 University St, Montreal, Canada

ARTICLE INFO

Article history:
Received 2 March 2015
Received in revised form
2 July 2015
Accepted 3 July 2015
Available online 5 August 2015

Keywords:
Radio astronomy
Data compression
HDF5
High performance computing

ABSTRACT

We present a procedure for efficiently compressing astronomical radio data for high performance applications. Integrated, post-correlation data are first passed through a nearly lossless rounding step which compares the precision of the data to a generalized and calibration-independent form of the radiometer equation. This allows the precision of the data to be reduced in a way that has an insignificant impact on the data. The newly developed `Bitshuffle` lossless compression algorithm is subsequently applied. When the algorithm is used in conjunction with the HDF5 library and data format, data produced by the CHIME Pathfinder telescope is compressed to 28% of its original size and decompression throughputs in excess of 1 GB/s are obtained on a single core.

1. Introduction

The simultaneous drives to wider fields and higher sensitivity have led radio astronomy to the cusp of a big-data revolution. There is a multitude of instruments, including 21 cm cosmology experiments (Pober et al., 2013; Battye et al., 2013; Canadian Hydrogen Intensity Mapping Experiment, CHIME; Pober et al., 2014; Greenhili et al., 2012; van Haarlem et al., 2013; Zheng et al., 2013; Parsons et al., 2010; Chen, 2012), Square Kilometer Array Precursors (Johnston et al., 2008; Lonsdale et al., 2009; Booth et al., 2009), and ultimately the Square Kilometer Array (SKA Organization, 2015), whose rate of data production will be orders of magnitude higher than any existing radio telescope. An early example is the CHIME Pathfinder (Bandura et al., 2014; Newburgh et al., 2014) which will soon be producing data at a steady rate of over 4 TB per day. The cost associated with storing and handling these data can be considerable and therefore it is desirable to reduce the size of the data as much as possible using compression. At the same time, these data volumes produce a significant data processing challenge. Any data compression/decompression scheme must be fast enough as to not hinder data processing, and would ideally lead to a net increase in performance due to the reduced time required to read the data from disk.

Here, after discussing some general considerations for designing data storage formats in Section 2, we present a scheme for compressing astronomical radio data. Our procedure has two steps: a controlled (relative to thermal noise) reduction of the precision of the data which reduces its information entropy (Section 3), and a lossless compression algorithm—`Bitshuffle`[1]—which exploits this reduction in entropy to achieve a very high compression ratio (Section 4). These two steps are independent in that, while they work very well together, either of them can be used without the other. When we evaluate our method in Section 5 we show that the precision reduction improves compression ratios for most lossless compressors. Likewise, `Bitshuffle` outperforms most other lossless compressors even in the absence of precision reduction.

* Corresponding author at: Department of Physics and Astronomy, University of British Columbia, 6224 Agricultural Rd., Vancouver, V6T 1Z1, Canada.
E-mail address: kiyo@physics.ubc.ca (K. Masui).

[1] https://github.com/kiyo-masui/bitshuffle.

http://dx.doi.org/10.1016/j.ascom.2015.07.002
2213-1337/

2. Considerations for designing data storage formats

2.1. Characteristics of radio-astronomy data and usage patterns

Integrated, post-correlation radio-astronomy data are typically at least three dimensional, containing axes representing spectral frequency, correlation product, and time.[2] The *correlation product* refers to the correlation of all antenna input pairs, including auto-correlations and cross-correlations between different polarizations from the same antenna. In a single dish these form the polarization channels for each beam and in an interferometer these are the visibilities. This also applies to beam forming interferometers, where linear combinations of antenna inputs are formed (either in analog or digitally) before correlation.

The CHIME collaboration determined that its data are most commonly accessed along the time axis. That is, it is generally most efficient for the axis representing time to be the fastest varying once loaded into memory. This is the case for noise characterization, radio-frequency interference (RFI) flagging and system-health monitoring, to name a few. Most importantly, the map-making pipeline typically produces maps on a per-frequency basis and is most efficient at processing time-contiguous data. Though it is sometimes necessary to work with spectra (slices along the frequency axis) or 'correlation triangles' (slices along the correlation product axis), we find that these use cases normally only involve a few slices and large I/O operations in these spaces are rare.

Of course, the CHIME collaboration's preference for the time axis to be the fastest varying will not apply to all consumers of radio data. One expects that access patterns might vary considerably for the diverse applications of radio data, including spectroscopy, synthesis imaging, and pulsar timing. But as discussed below, arranging data with time as the fastest varying index is beneficial for data compression.

2.2. Compression: benefits and requirements

Compression can greatly ease the burden of storing and handling large data sets, but there are also performance benefits. Compression algorithms exist whose decompression cost is negligible compared to the cost of reading from disk. As we will show, data may be compressed by up to a factor of four in some cases. As such, the time required to load a dataset from disk into memory may be reduced by a factor of four using compression.

We previously stated that ordering data with the axis representing time as the fastest varying is most efficient for the majority of I/O operations. This ordering is also beneficial for compression, since adjacent data points are likely to be highly correlated, presuming that the cadence is such that the spatially-smoothly varying sky is Nyquist sampled. On the other hand, it is most natural to record data with time as the *slowest* varying index since that is the order in which they are generated by the instrument. To have time as the fastest varying index, the data must either be buffered in memory (which is impractical), written with strided writes (which is inefficient) or reordered after acquisition. Since the data are acquired and written only once but read many times, it is logical to prioritize read-performance over write-performance. Thus, the CHIME collaboration deemed a post-acquisition reordering step to be worthwhile.

The same argument can be used to prioritize data decompression speed over compression speed. Compression is sufficiently cheap computationally that even a modest number of processors should be able to keep up with the acquisition rate of CHIME Pathfinder data (which will be ∼50 MiB/s depending on runtime parameters) for almost any compression algorithm. Even if this were not the case, data could be compressed in parallel post acquisition. On the other hand, one might wish to load several days of acquired data at once for analysis, and ideally, this would be bound only by disk read times, not decompression speed.

The decompression cost may not be negligible compared to read times for files that are cached in memory or stored on high-performance parallel file systems. This makes it desirable to have as fast a decompression scheme as possible as the benefits of speed are not always limited by hard drive access. A multi-threaded implementation of the decompression algorithm can thereby result in a significant speed up on multi-core systems.

To summarize all of the foregoing, the following requirements for a compression scheme emerge:

Unbiased Any lossy compression must not bias the data in any way.

Nearly lossless Any lossy compression employed must be controlled in a manner that is *guaranteed* not to significantly decrease the sensitivity of the data.

Time minor In the multi-dimensional dataset, the axis representing time should be the fastest varying. This allows for the efficient reading of small subsets of spectral frequencies and correlation products but for large periods of time.

Fast decompression To realize the performance gains associated with compressing the data, we require the time to decompress the data to be small compared to the time required to read the data from disk. At the time of writing, a single hard drive can typically be read at a rate of ∼100 MiB/s. As such, a compression algorithm with throughput of ∼1 GiB/s on a single processor is desirable.

Threaded When using a parallel file system, or when the file is cached in system memory, reading throughputs can be much higher compared to when using a single hard disk. For decompression to not degrade performance in these cases, the compression library should be threaded.

Thread-safe While the HDF5 library (see Section 2.3) is not internally threaded, it may become so in the future. In addition, programs may attempt to hide the cost of IO operations by putting them in a separate thread. The compression library must therefore be thread-safe.

2.3. HDF5 and chunked storage

The Hierarchical Data Format 5 (HDF5) (HDF Group, 1997–2015a) is a widely used data format in astronomy, capable of storing and organizing large amounts of numerical data. In the context of this paper, it also has the benefit of allowing for 'chunked storage'. That is, an HDF5 'dataset'—a multidimensional array of homogeneous type—can be broken up into subsets of fixed size, called chunks, which are stored in any order on disk, with locations recorded in a look-up table. This is in contrast to contiguous storage, in which random access is obviously trivial. The advantage of chunked storage for our purposes is that though the number of elements in a chunk is fixed, its size is not, and as such it may readily be compressed.

The primary drawback of chunked storage is that full chunks must be read from disk at a time. As such, to read a single array element, the full chunk containing thousands of elements must be read. In practice, this is mitigated by the fact that hard disk latencies are very long compared to the time required to read data

[2] A fourth axis is often introduced when data are 'folded' or 'gated'—i.e., if data from the on- and off-periods of a switched, calibration noise source are accumulated separately, or pulsar data is folded on the pulsar's period which is divided into many gates.