Efficient calculation of quantiles for large datasets in an astronomical image viewer

Adrianna Pińska[1]

May 3, 2017

[1]adrianna.pinska@uct.ac.za

Who am I?

- A research software developer at IDIA
- Not an astronomer!
  - I know just enough astronomy to be dangerous.

Talk summary

- What is CARTA? Why was it written?
- What are quantiles? How are they used in CARTA?
- Why is this calculation so expensive?
- How can we fix this?

The CARTA viewer

- Replacement for older viewer used by CyberSKA[2]
- Developed in cooperation with NRAO and ASIAA
- Design goal: scaling to **very large images** (multi-terabyte – SKA)
  - Important to identify and eliminate **bottlenecks**

[2] E. Rosolowsky et al. *The Cube Analysis and Rendering Tool for Astronomy*, 2015

Colour maps

- How to visualise sky data as an image?
- Numeric values **mapped to colours**
- Various colour maps can be used
- Usually a gradient
- Limited number of colours available
- **Clipping** of extreme values is desirable
  - more colours available for middle of the range
  - improves visibility of features

Quantiles

- Top and bottom clipping values = **quantiles**
- Partition data into intervals of a particular size
  - like median, quartiles, etc., but much smaller fractions at both extremes
- Calculation of quantiles = **selection problem**
  - related to sorting
  - solved by similar class of algorithms

Quantile calculation in CARTA

- CARTA displays 2D images, but a cube may have multiple frames
  - e.g. different frequencies
- Clipping values are sometimes calculated over all frames
  - e.g. when user is playing frames like a movie
  - consistent clipping values needed for smooth viewing experience

The current solution

- `nth_element` – existing function in C++ standard library
- Returns element which would have rank N if data were sorted
- Fine for small images, but doesn't scale well to *medium* images, let alone "big data".
- Both **slow** and **memory-intensive**.

Why?

- The implementation scales linearly with data size, *but*...
  - data size scales quadratically with increases in image resolution
  - it further increases when we need to calculate value for all frames
- The implementation is an in-place partial sort
  - ...so we have to copy the whole image in memory to preserve the original data

Mitigating the expense

- Three complementary approaches:
- **Caching** values on disk
- Calculating **multiple quantiles** in a single pass
- Calculating **approximations** instead of the true values

Caching

- Calculated values are individual scalars
  - Very *'inexpensive* to store
  - Storage requirements remain constant as image size increases
- Reading precalculated values from disk is much **faster** than recalculating them
- Precalculation of common values may be performed pre-emptively
  - when image is first uploaded to platform, before it is ever opened
  - UI implications – make it easier for user to pick discrete options?

Multiple quantiles in one pass

- In many algorithms, the same initial setup can be **reused** to find multiple quantiles
- This can be substantially cheaper than repeating the entire process for each quantile
- Some calculations in the viewer could be merged
- This could also make precalculation of common values more efficient

Approximate algorithms

- High precision only requred if values are used for analysis
- For visual inspection by a human, a **close value** is good enough
- Fast approximate solutions could be replaced by slow exact solutions later
- Approximate algorithms provide various strategies to sample the data.
  - Tradeoffs between e.g. memory usage and accuracy

Algorithm criteria

- Speed
- Suitability for parallelisation
- Memory usage
- One pass over data?
- Prior knowledge of data required?
- Error bounds
- Different performance for different quantiles?

Current prototype

- G. S. Manku et al. *Approximate Medians and other Quantiles in One Pass and with Limited Memory*, 1998
  - Stores elements in a fixed number of buffers of fixed size
  - Fills empty buffers by iterating over dataset
  - Merges multiple full buffers by weighted sampling of ordered elements
  - Repeats until all data is consumed
  - Calculates quantiles using final set of samples
- Initial results from unoptimised, serial prototype:
  - 0.5% - 10% memory usage
  - 1% - 2% error
  - 3x - 5x speedup
- More extensive testing required.

Future work

- G. S. Manku et al. *Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets*, 1999
  - Uses less memory to calculate extreme quantile values – which are the ones we want.
- Algorithms which perform better on specific kinds of data?

Questions?