# QARTOD in Practice

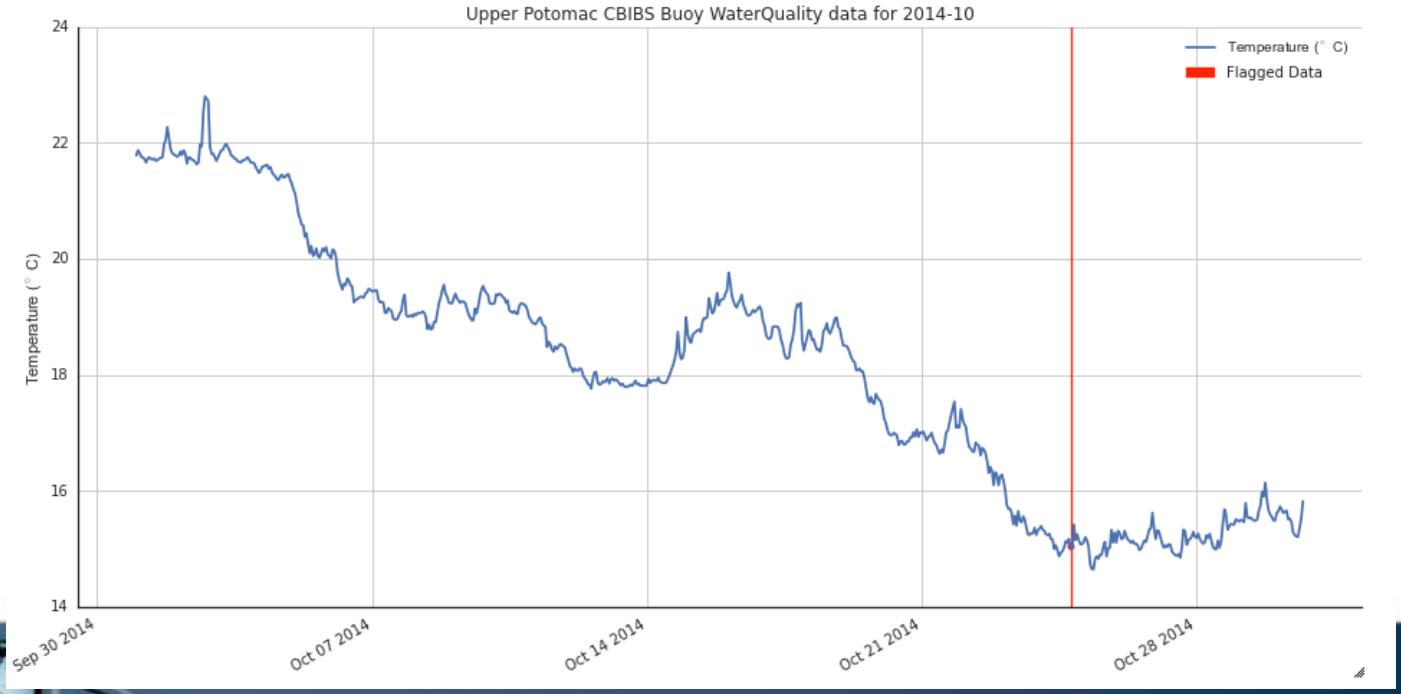
Luke Campbell, Software Engineer, RPS ASA

# Background

- We were asked to design, and implement software that would perform near realtime QC for the Chesapeake Bay Interpretive Buoy System (CBIBS).
- The first step was understanding QARTOD and then implementing it as a software module or service.
- We've successfully implemented and are applying QARTOD to CBIBS Station Data
  - Minus waves (currently in progress, and scheduled to be done June)

# Background

- We were asked to design, and implement software that would perform near realtime QC for the Chesapeake Bay Interpretive Buoy System (CBIBS).
- The first step was understanding QARTOD and then implementing it as a software module or service.
- We've successfully implemented and are applying QARTOD to CBIBS Station Data
  - Minus waves (currently in progress, and scheduled to be done June)

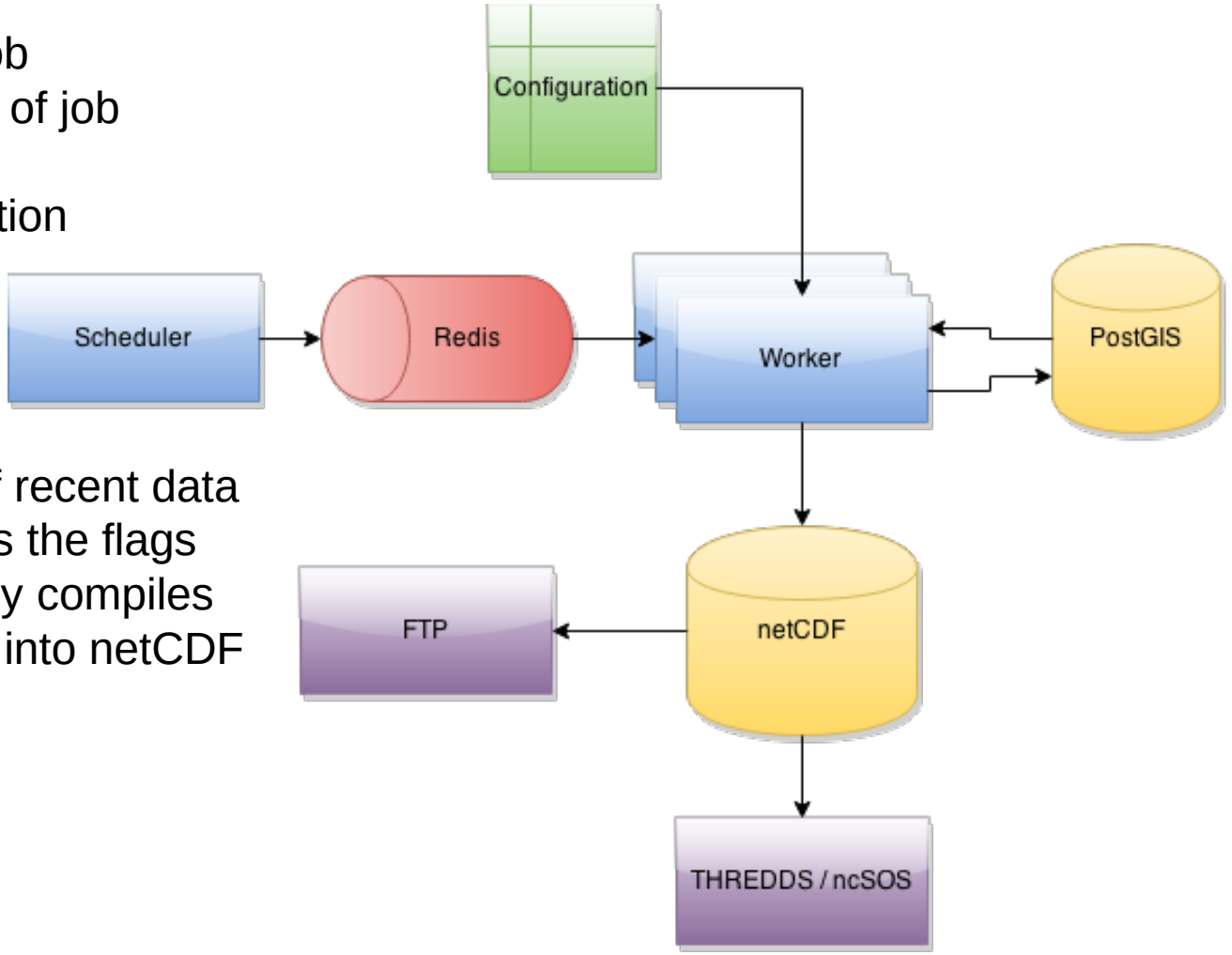


# QARTOD Tests in Practice

Here's a small demonstration of us downloading and running QC on the Toledo Low Se

# Automating QC

1. Scheduler Creates a Job
2. Worker retrieves job off of job queue.
3. Worker loads configuration information
   - Station and Sensor Arrangements
   - Test parameters
4. Retrieves a small set of recent data
5. Performs QC and saves the flags
6. Another worker regularly compiles the flags and metadata into netCDF files



# Tests

The tests we have currently published and are available as a pure python package:
- Location Check
- Gross Range
- Climatology
- Spike
- Rate of Change / Gradient
- Flat Line
- Attenuated Signal

We're currently in the middle of testing the tests for Waves.

https://github.com/asascience-open/qartod

# Representing the Flags as Binary Elements

In the past, we looked at encoding each combination of test outcomes as a bit-string.
After further research, I think we should take a simpler approach.

| | 15-14 | 13-12 | 11-10 | 9-8 | |
|---|---|---|---|---|---|
| **Winds** | Multi-Variate | Attenuated Signal | Neighbor | N/A | N |
| **Water Level** | Multi-Variate | Attenuated Signal | Neighbor | N/A | N |
| **In-situ Temperature and Salinity** | Multi-Variate | Attenuated Signal | Neighbor | TS Curve/Space | D |
| **Dissolved Oxygen (Version 1)** | Multi-Variate | Attenuated Signal | Neighbor | N/A | N |
| **Waves** | N/A | N/A | Neighbor | N/A | N |
| **Currents** | N/A | N/A | N/A | N/A | N |
| **Ocean Optics (Draft/In Review)** | Multi-Variate | Attenuated Signal | Neighbor | Radiance/ Irradiance | N |

# Encoding the Flags

Concerns that drive the design:

- Space
- Complexity
- Support for future tests
- Number of test output combinations
- Compression

## Space

For station, and time series data, we won't see a scale of data that will be cause for alarm on modern architectures.

AWS charges $17.00 per month for 200GB SSD for a directly mounted EBS Volume. AWS S3 charges $47.77 per month for a TB of data.

For a station with 30 sensors that each measure one floating point observation every second and is deployed year round, the uncompressed file will be 7.18 GB.

In summary, for modern architecture and managing station and time-series data, incorporating additional QC flags with the data won't introduce an enormous burden.

## Complexity

I've always preached for a long time that complex solutions have a real cost associated with them. Simpler solutions allow for an easier adoption.

A key insight is that code is read much more often than it is written.

By using independent variables for each test, we gain:

- Simplicity
- No ambiguity in the results
- Easy to implement
- Easy to scale

## Support for future tests

I made some changes to the encoding scheme to support the potential for future tests by extending all of the flag encoding sizes to unsigned 64 bit integers. This extension allowed for every test to have 8 flags but limited our ability to safely grow the number of tests without exceeding 64bits or introducing new variables.

If we just include individual variables, we can have as many flags and as many tests as we want, there's unlimited scalability.

## Number of test output combinations

Individual variables can allow for up to 2^64 flags, although I would recommend avoiding that many independent flags.

## Compression

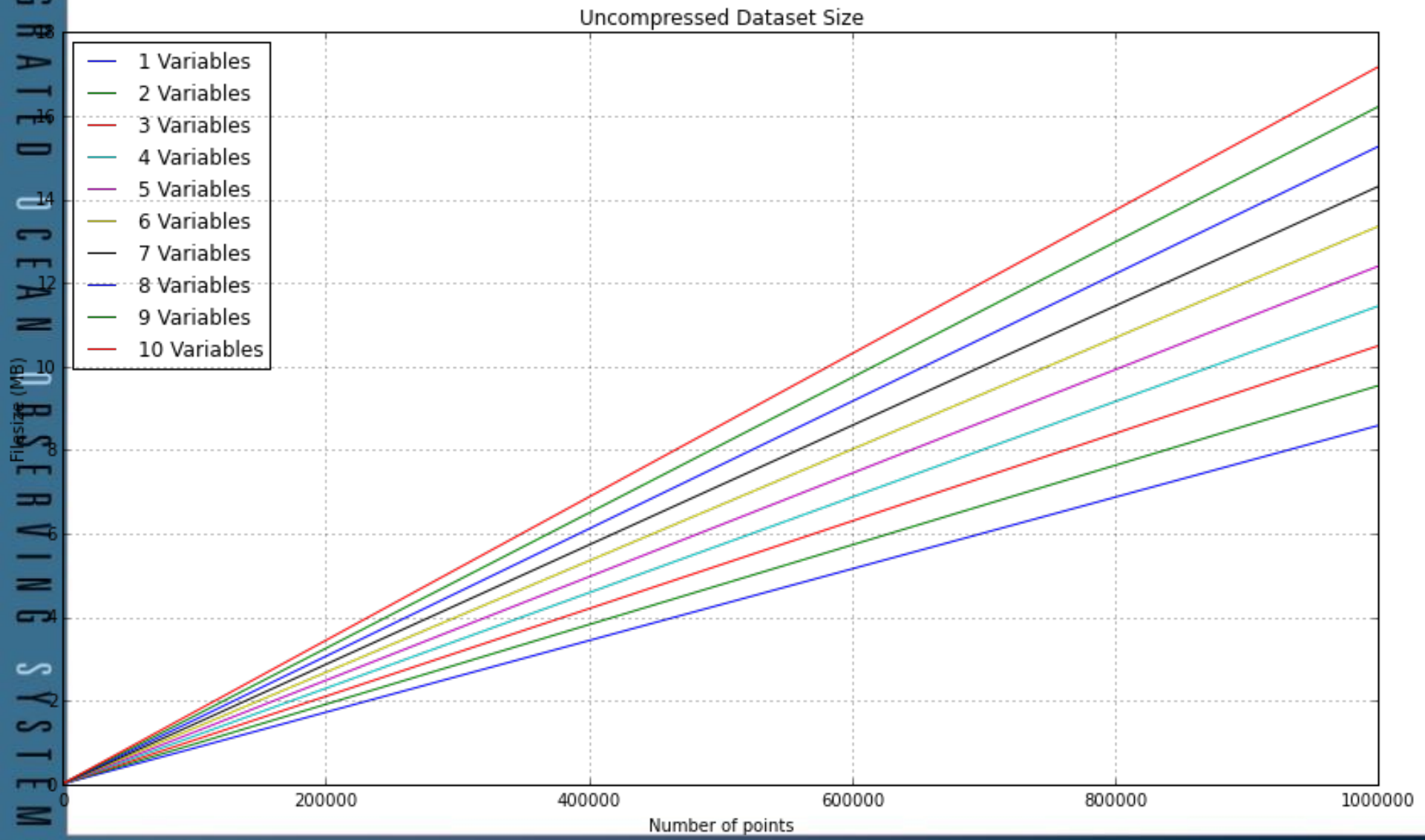
Using just zlib compression, I was able to see a fairly large reduction in file size compared to that of using a single variable with the encoding scheme.

A file with one QC variable using the encoding scheme took up 9MB for one million points.

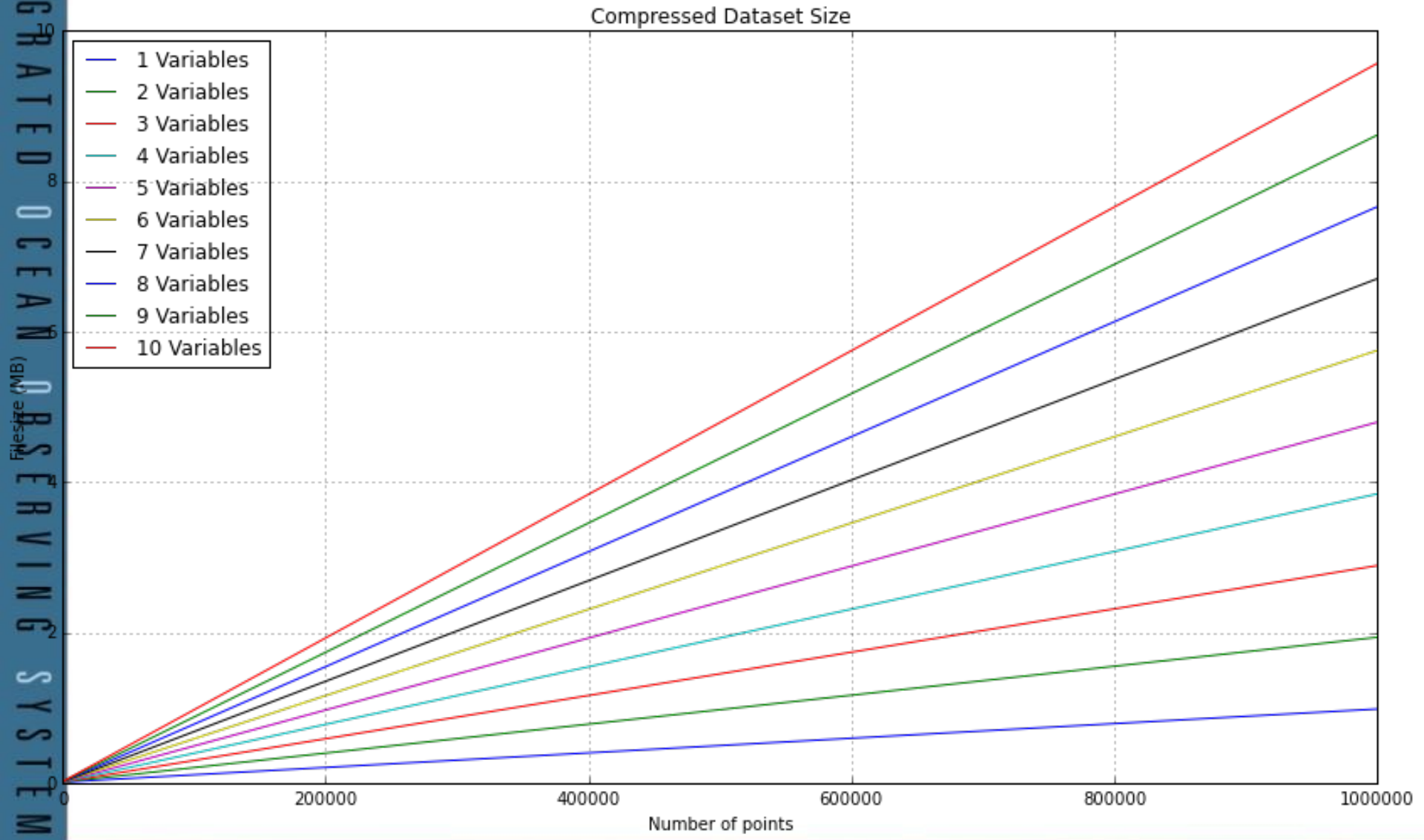
A file with 10 individual QC variables for all of the required, and recommended tests (water level) took up 9.2 MB

So by sacrificing 0.2MB per one million observations we gain simplicity and scalability.

# Linear space growth for uncompressed datasets



# Linear space growth for compressed datasets



# Representing QC Flags: netCDF

```
byte sea_water_salinity_rate_of_change_test(time) ;
     sea_water_salinity_rate_of_change_test:_FillValue = 9b ;
     sea_water_salinity_rate_of_change_test:long_name = "Sea water salinity rate_of_change_test
     sea_water_salinity_rate_of_change_test:comments = "This test inspects the time series for
     sea_water_salinity_rate_of_change_test:coordinates = "time longitude latitude" ;
     sea_water_salinity_rate_of_change_test:flag_vals = 1b, 2b, 3b, 4b, 9b ;
     sea_water_salinity_rate_of_change_test:flag_meanings = "GOOD UNKNOWN SUSPECT BAD MISSING"
     sea_water_salinity_rate_of_change_test:references = "http://www.ioos.noaa.gov/qartod/tempe
     sea_water_salinity_rate_of_change_test:coverage_content_type = "qualityInformation" ;

byte sea_water_salinity_qc(time) ;
     sea_water_salinity_qc:standard_name = "sea_water_salinity status_flag" ;
     sea_water_salinity_qc:_FillValue = 9b ;
     sea_water_salinity_qc:flag_vals = 1b, 2b, 3b, 4b, 9b ;
     sea_water_salinity_qc:flag_meanings = "GOOD UNKNOWN SUSPECT BAD MISSING" ;
     sea_water_salinity_qc:references = "http://www.ioos.noaa.gov/qartod/temperature_salinity/q
```

# Representing QC Flags: SOS-SWE

```
<swe2:quality>
 <swe2:Category definition="http://mmisw.org/ont/ioos/swe_element_type/recordQuality">
   <swe2:constraint>
     <swe2:AllowedTokens>
       <swe2:value>1</swe2:value> <!-- Good, Passing -->
       <swe2:value>2</swe2:value> <!-- Not Evaluated -->
       <swe2:value>3</swe2:value> <!-- Suspect -->
       <swe2:value>4</swe2:value> <!-- Fail -->
       <swe2:value>9</swe2:value> <!-- Missing, theoretically you should never see this flag -->
     </swe2:AllowedTokens>
   </swe2:constraint>
 </swe2:Category>
</swe2:quality>


2009-05-23T00:00:00Z,1,wmo_41001_sensor1,2,0,1,359.0,x,10.0,3,2,352.0,y,9.6
2009-05-23T01:00:00Z,1,wmo_41001_sensor1,1,2,2,345.0,y,10.4
2009-05-23T02:00:00Z,1,wmo_41001_sensor1,4,0,3,332.0,z,10.5,1,2,334.0,x,10.3,2,3,336.0,z,10.1,3,1,3
2009-05-23T00:00:00Z,3,wmo_41001_sensor2,3,0,13.7,1,16.8,2,19.2
2009-05-23T01:00:00Z,1,wmo_41001_sensor2,3,0,13.5,1,16.4,2,19.3
2009-05-23T02:00:00Z,4,wmo_41001_sensor2,3,0,13.4,1,16.5,2,18.8
```

# CF Support

CF 1.6 Supports the notion of a primary QC flag with the standard_name attribute set to "s

For CBIBS we've omitted the standard_name attribute for the variables that represent a si

It would also be good to identify some set of attributes that pertain to QARTOD that identif

```
seawater_temperature_range_check:qartod_test = "range_check" ;
```

# Client Support

The QARTOD Primary QC flags follow the CF convention for status flags and any client th

Clients that wish to interface with the individual QARTOD tests will need to be customized

# Challenges

- Metadata Support
  - Tried out bit string encoding
  - CF and standard_name support
- Archival
  - Do we archive individual tests or just the primary flag?
- Software Architecture for running QC in a timely manner.
- Flags with complex dimensions

# Questions?

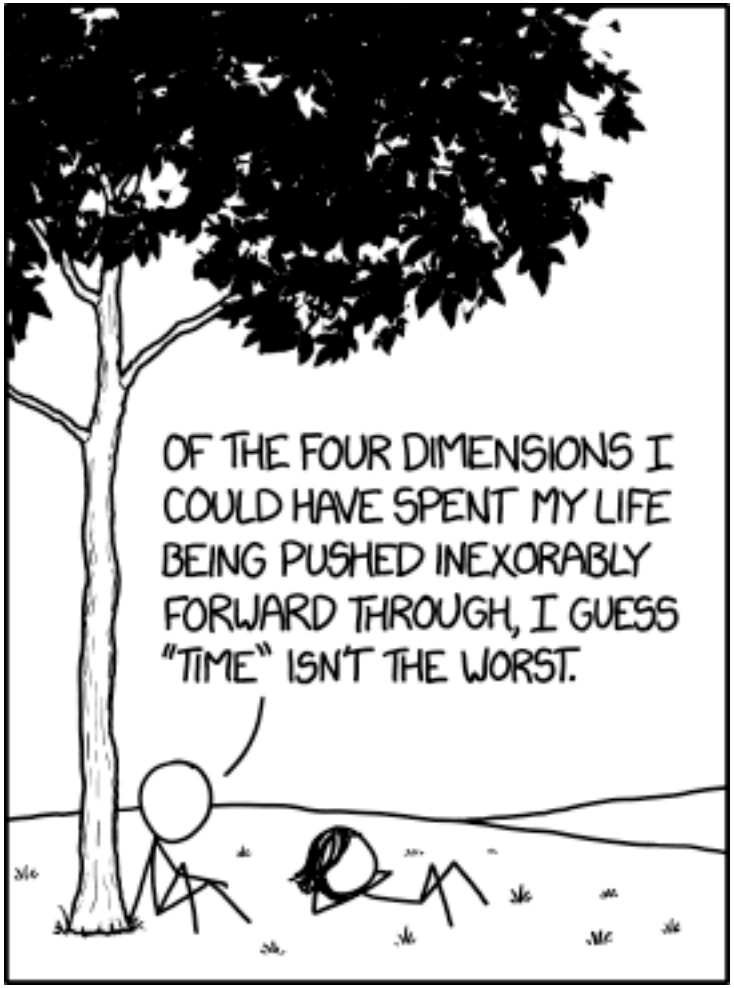



Image courtesy of XKCD (http://xkcd.com)

# Discussion