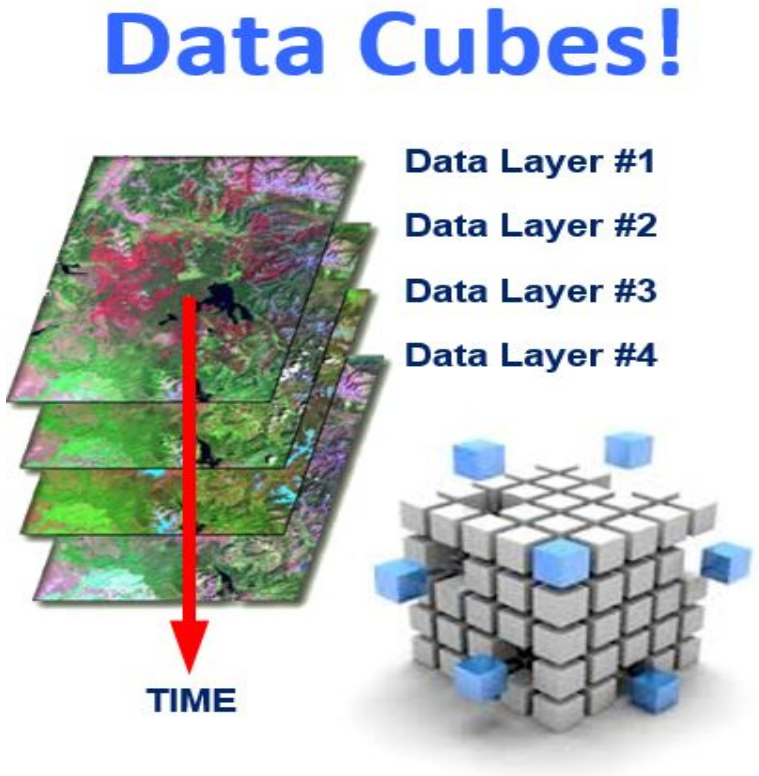
Data Cube Overview

- What is a Data Cube?
- Need for a Data Cube
- Implementing the Data Cube

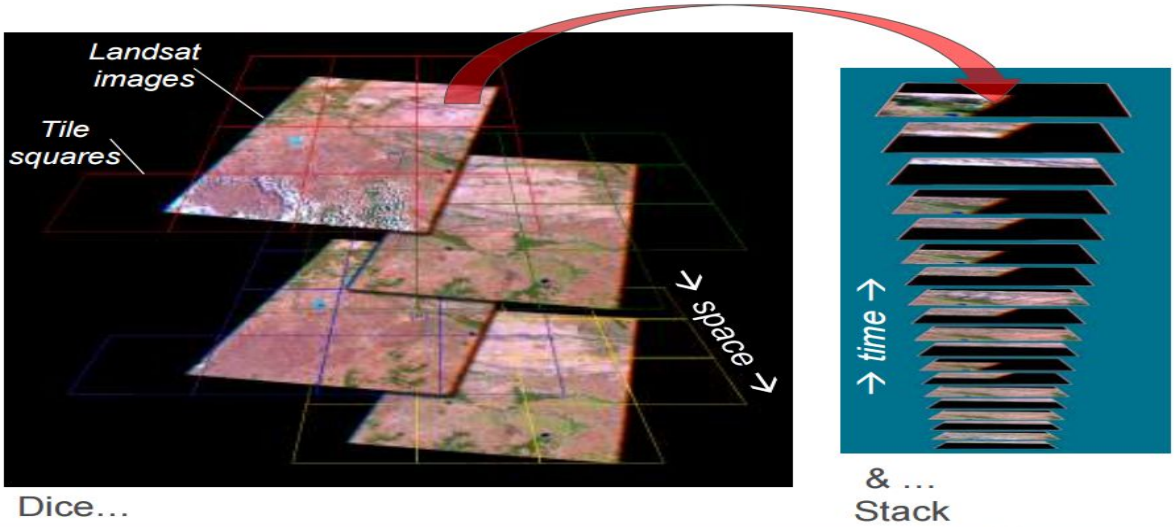
What is a Data Cube?

- Concept is to “cube” Earth Observation datasets by stacking satellite image tiles in time sequences covering the same area of ground.
- The Data Cube arranges 2D(spatial) data temporally and spatially to allow flexible and efficient large scale analysis.



What is a Data Cube?

- The "Dice and Stack" method is used to subdivide the data into spatially-regular (nested grid), time-stamped, band-aggregated layers which can be managed as dense temporal stacks.



Features of Data Cube

- **Key Characteristics:**

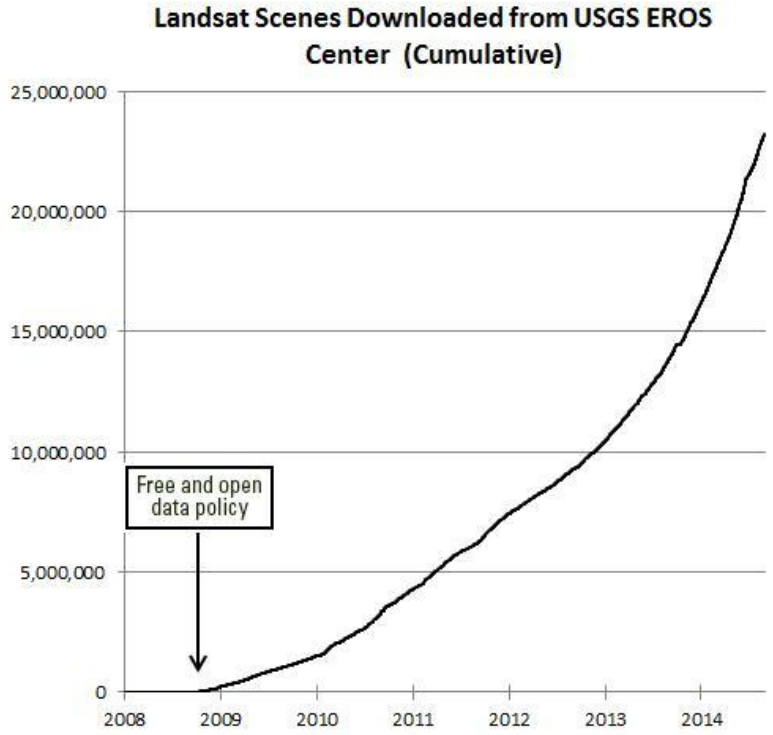
  In contrast to other methods to handle gridded data collection, every unique observation is kept

- **Calibration and Standardisation of the data:**

  Increases the value which can be derived from earth observation, and other sources of large gridded datasets

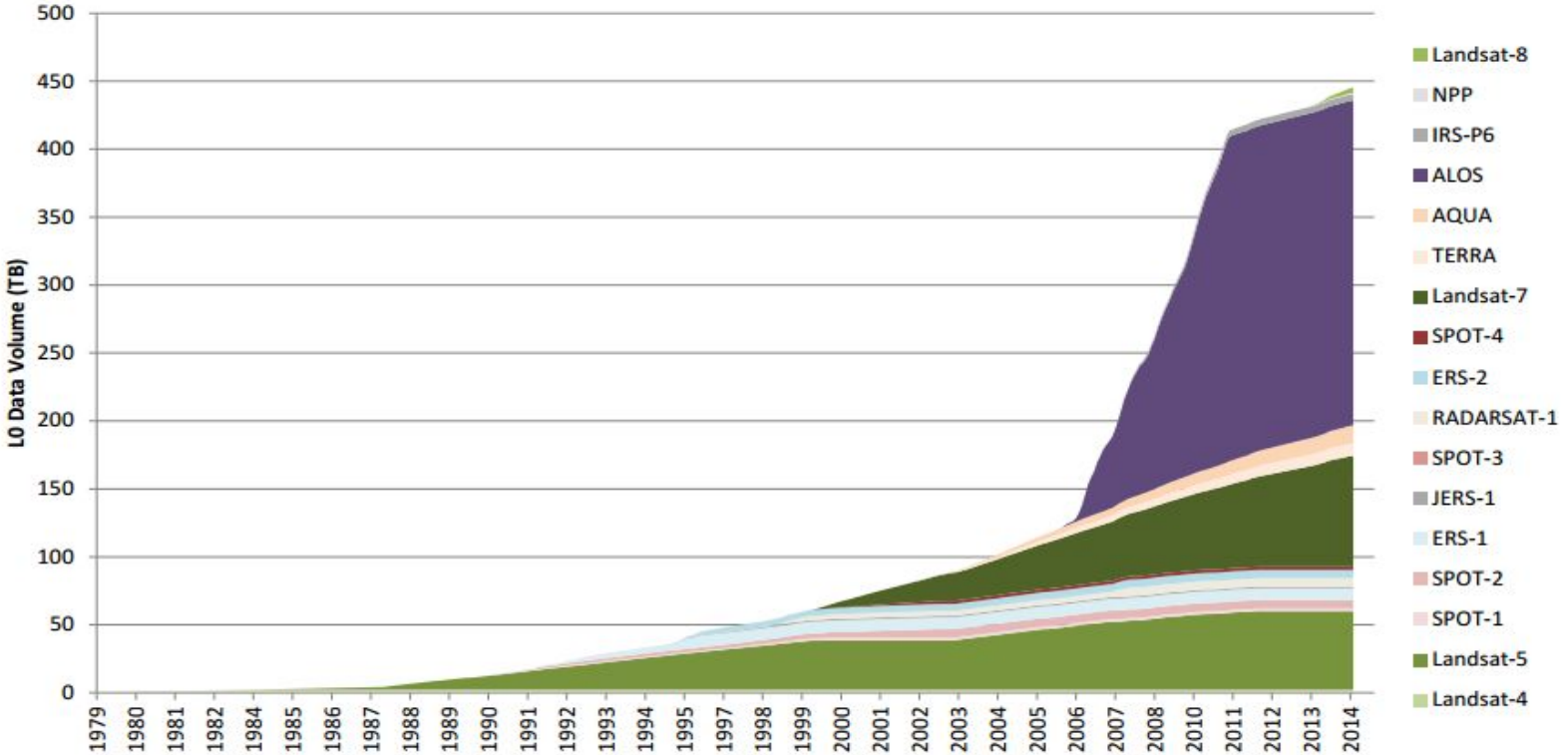
Need for a Data Cube

- Growing volume of Geoscience Data
- Offers great potential for analysis and understanding of the physical environment, reduce risk from natural hazards and assist in securing new resources.
- However, it has proven difficult to interact with such large volumes of data and extract the useful information.

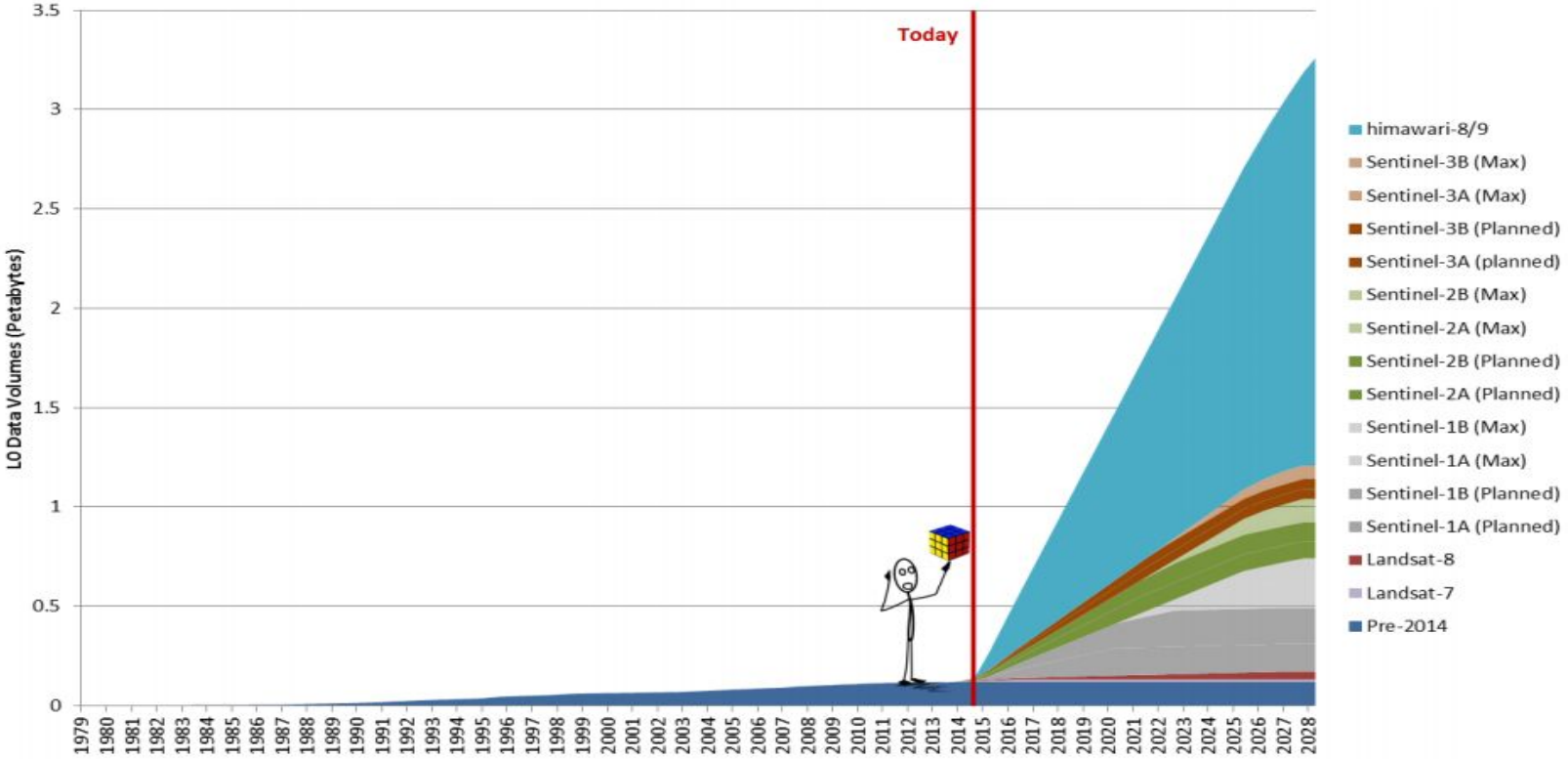


Increasing Satellite Data

Satellite EO Data Holdings at Geoscience Australia



In the next decade



Data Cube is the Solution!!

- A multidimensional data cube is an effective solution to handle such large amounts of data.
- In 2014, Geoscience Australia, CSIRO and the NCI established the AGDC, building on the work of the GA and expanding it to include additional EO and other gridded data collections (e.g. MODIS, DEM).
- Utilising high performance data structures and high performance computing infrastructure, the AGDC provides an integrated EO analysis environment for decades of analysis ready EO and related data from multiple satellite

innovate achieve lead

BITS Pilani
Pilani Campus

Preparation of Dataset

Input Data

- Landsat data of 6 years for Uttrakhand region

Year	Landsat	Sensor
1990	Landsat 4-5	TM
1995	Landsat 4-5	TM
2000	Landsat 7	ETM+
2003	Landsat 7	ETM+
2010	Landsat 4-5	TM
2015	Landsat 8	OLI

- Digital Number (DN)

Our Task

- Top of Atmosphere Reflectance (TOAR)
  - Band 1 (Blue)
  - Band 2 (Green)
  - Band 3 (Red)
  - Band 4 (Near IR) *
- Normalized Difference Vegetation Index (NDVI)
- Land Surface Temperature (LST)

* Band 2,3,4 & 5 respectively in Landsat 8- OLI

Methodology

- DN to Radiance
- Radiance to Reflectance
- NDVI
- DN to Brightness Temperature
- LST using NDVI and BT

Software used

- ENVI 5.1
- ERDAS Imagine

Digital Number (DN)

- Generic term for Pixel value
- Un-calibrated
- Converted to meaningful physical quantities like radiance, reflectance etc.

Radiance

- The amount of radiation coming from an area
- Derived using gain and offset method

$$L_\lambda = \text{Gain*DN} + \text{Offset}$$

- Affected by clouds, atmospheric aerosols and gases
- It also depends on
    - Illumination (both intensity and direction)
    - Orientation and position of the target
    - Path of the light through atmosphere
- Typically, corrected to reflectance: a more usable quantity for quantitative analysis

Reflectance

- Proportion of radiation striking a surface to the radiation reflected off of it
- Many features like vegetation, water and soil have unique surface reflectance spectra

    - desired physical quantity

Top of Atmosphere Reflectance (ToAR)

- Reflectance measured by space based sensor flying higher than earth's atmosphere

$$R = \frac{\pi L d^2}{E_{sun\lambda} \sin\theta}$$

Where: L= radiance (μW/cm^2*str*μm)

d=Earth-Sun distance (AU)

$E_{sun\ \lambda}$ = Mean solar exoatmospheric irradiance

θ= solar elevation angle (90-solar zenith angle)

- Includes contribution from cloud, aerosol etc.

Normalised Difference Vegetation Index (NDVI)

- Gives a measure of vegetation cover on land surfaces
- NDVI = (NIR – red) / (NIR + red)
- NDVI takes values between -1 and 1.
- We can easily identify areas of dense vegetation, areas with little or no vegetation.
- NDVI also identifies water and ice.

Area category	NDVI range
Lakes, Rivers and oceans	Negative values
Barren Areas, Sand or Snow	-0.1 – 0.1.
Clouds	0 -0.075.
Sparse Vegetation	0.2 – 0.5.
Dense Vegetation	> 0.5.

Land Surface Temperature (LST)

- The amount of radiation coming from an area
- Derived using gain and offset method

$$L_\lambda = \text{Gain*DN} + \text{Offset}$$

- Affected by clouds, atmospheric aerosols and gases
- It also depends on
    - Illumination (both intensity and direction)
    - Orientation and position of the target
    - Path of the light through atmosphere
- Typically, corrected to reflectance: a more usable quantity for quantitative analysis

Land Surface Temperature

(2) Calculating Emissivity using NDVI generated :

$$\varepsilon = \varepsilon_s (1 - P_V) + \varepsilon_v P_V + d\varepsilon$$

$$P_v = \left[\frac{NDVI - NDVI_{min}}{NDVI_{max} - NDVI_{min}} \right]^2$$

And $d\varepsilon = (1 - \varepsilon_s)(1 - P_V) F \varepsilon_v$

$$\varepsilon = 0.004 P_V + 0.986.$$

Ref: Estimation of land surface temperature–vegetation abundance relationship, Qihao Wenga,, Dengsheng Lub,1, Jacquelyn Schubringa

http://www.uv.es/ucg/articulos/2005/Publications_2004_10.pdf

Calculating Land Surface Temperature

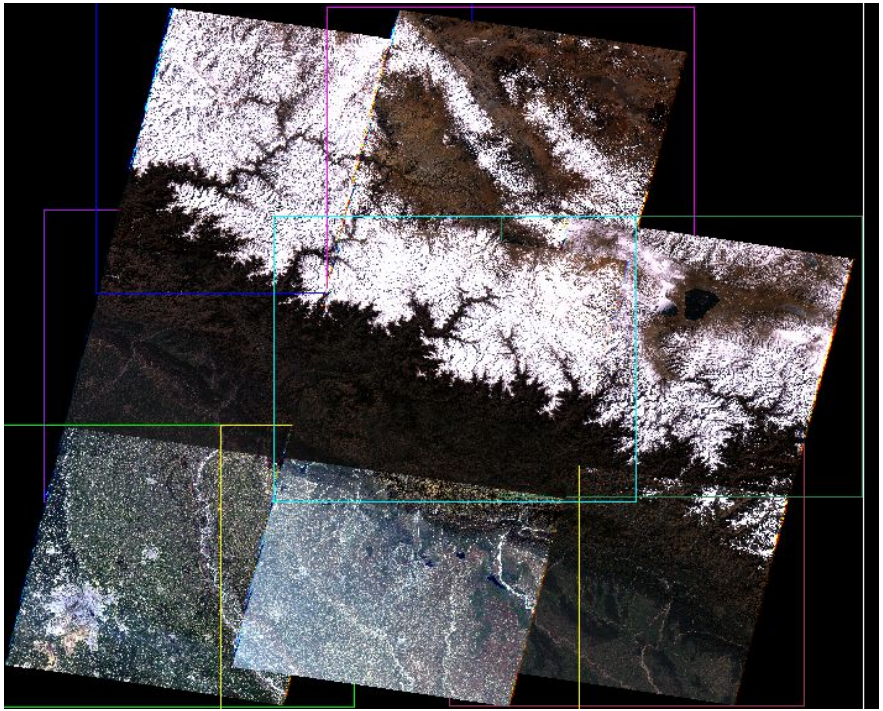
$$R = \frac{\pi L d^2}{E_{sun\lambda} \sin\theta}$$

Applications of LST

- Climate change
- Land cover changes
- Crop management
- Water management
- Fire monitoring
- Geological applications

Mosaicking

Small images of different tiles need to be mosaicked combined into a single image.



Steps

Select the input files

Select the data values to ignore (background pixels)

Set the Scene order

Feathering

Feathering

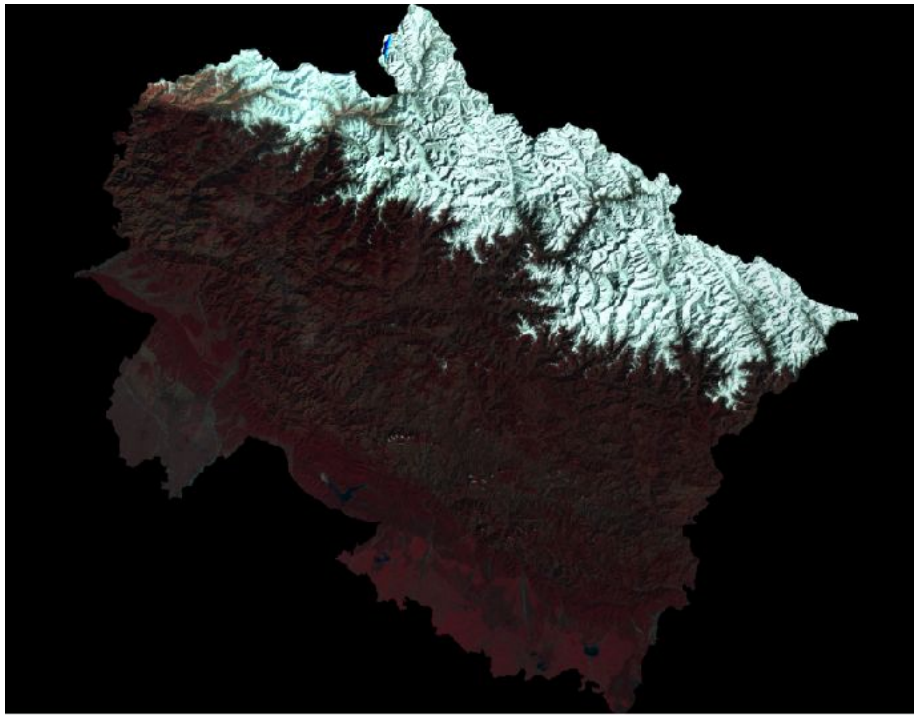
Edge Feathering:

This blends the pixels inside the edge of each scene's footprint boundary with underlying scenes that are within the feathering distance.

Seamline Feathering:

Seamlines are auto-generated and the effective mosaic polygons are displayed. It blends the pixels on both sides of the effective mosaic polygons with the underlying scenes.

Clipping to Uttrakhand shape file



DATA STORAGE and OPTIMIZATION

DATA STORAGE FORMATS

Problems –

- Data is LARGE!
- Data is COMPLEX!
- Data is heterogeneous.
- Storage needs parallel I/O.
- Storage needs random access.

Chosen Formats –

- NetCDF (Network Common Data Form)
- HDF5 (Hierarchical Data Form 5)
- ECW (Enhanced Compression Wavelet)

NetCDF

NetCDF –

- Supports efficient sub-setting and array oriented data access.
- Has self-describing header.
- Supports concurrent readers.

NetCDF4 (added advantages)-

- Built on HDF5.
- Large files possible.
- Multiple unlimited dimensions. Has classification flexibility.
- Data Compression through HDF5 libraries.

Parallel NetCDF –

- Can’t be used with NetCDF4!!! (specifically HDF5).

HDF5

HDF5 -

- Great for large data sets.
- Users can provide headers.
- Can represent wide variety of metadata.
- No limit on size.
- Supports chunking of data. (No need to load all in memory).
- Access time and storage space is optimal.
- Has web Browser plugin to display hdf5 files on the web.
- Supports parallel I/O, compression and more such advanced features.
- Hierarchical structure advantage.

ECW

ECW -

- Proprietary.
- Lossy compression format. Not lossless. (as per Wikipedia and other sources).
- Easy to transfer tera-byte sized images over the internet.
- Very fast and very efficient data compression.
- Very fast read-only SDK available for free.

CONCLUSION-

- NetCDF4
- HDF5 for parallelizability.
- ECW for efficient transfer over the internet.

DATA CONVERSION

<u>Objective</u> – Convert GeoTIFF to each of the three formats.
Configure GDAL to use these three.

To NetCDF4

- Convert TIF to NetCDF3, then convert NC3 to NC4.

To HDF5

- Convert NC4 to HDF5 (Just dump). (Problem – naming commands).
- Convert TIF to HDF4, then use h4toh5 to convert to HDF5. (We used this).

To ECW

- ERDAS Imagine since proprietary.
- Problems with GDAL ECW support.

All references at – *http://github.com/rishabhjoshi/DataCube/tree/master/DataConversion*

DATA COMBINATION

Objective – To convert multiple TIF files to one output file.

- Merge then Convert. (easier)
- Convert then merge.

Tools -

- gdalwarp (very fast).
- gdal_merge.py

Performance analysis of compression methods

Performance Parameters:

Compression time, Percentage Compressed, Decompression time

Compression techniques:

Single threaded - Gzip, bzip2, lzma, xz, lzop, lz4, snappy, 7z, zip

Multi threaded – Lbzip2, pigz, pbzip2

.tif file – run on laptop

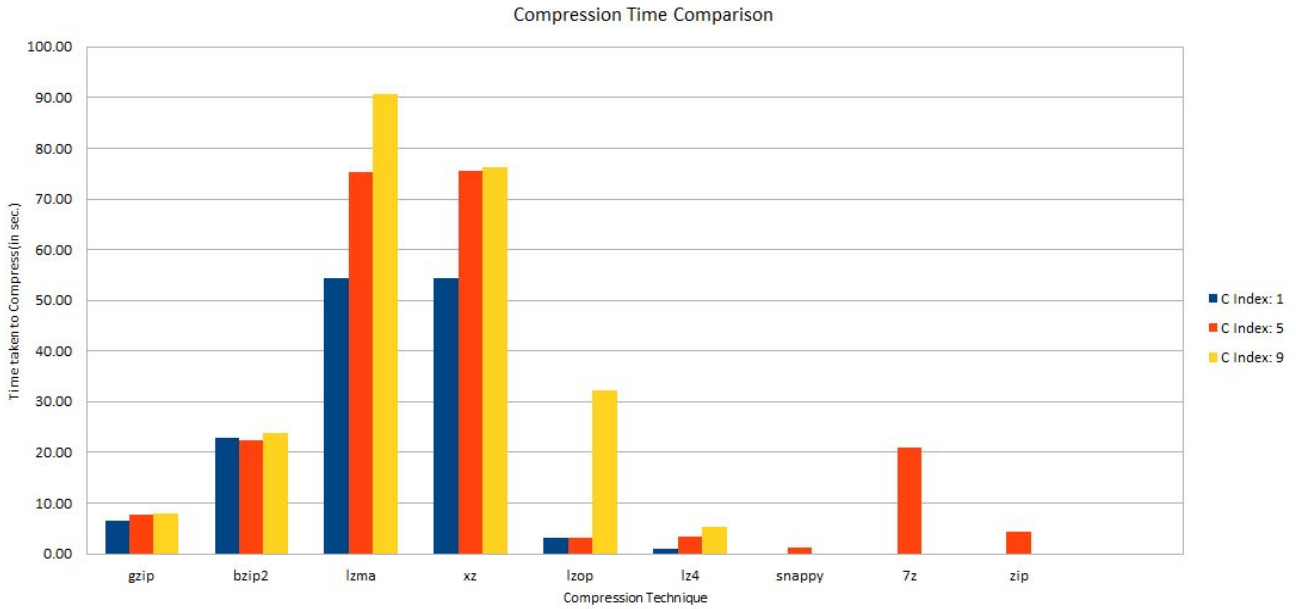
Compression Time comparison for different techniques at different compression index

File type: .tif		Compression time per run (in sec.)		
Runs = 5		C Index: 1	C Index: 5	C Index: 9
	gzip	6.39	7.68	8.04
	bzip2	22.77	22.34	23.78
	lzma	54.33	75.26	90.71
	xz	54.47	75.61	76.37
	lzop	3.16	3.21	32.14
	lz4	0.86	3.31	5.33
	snappy		1.12	
	7z		20.94	
	zip		4.29	

C Index stands for Compression Index

Compression time analysis (Less is better)

Least time - Snappy



.tif file – run on laptop

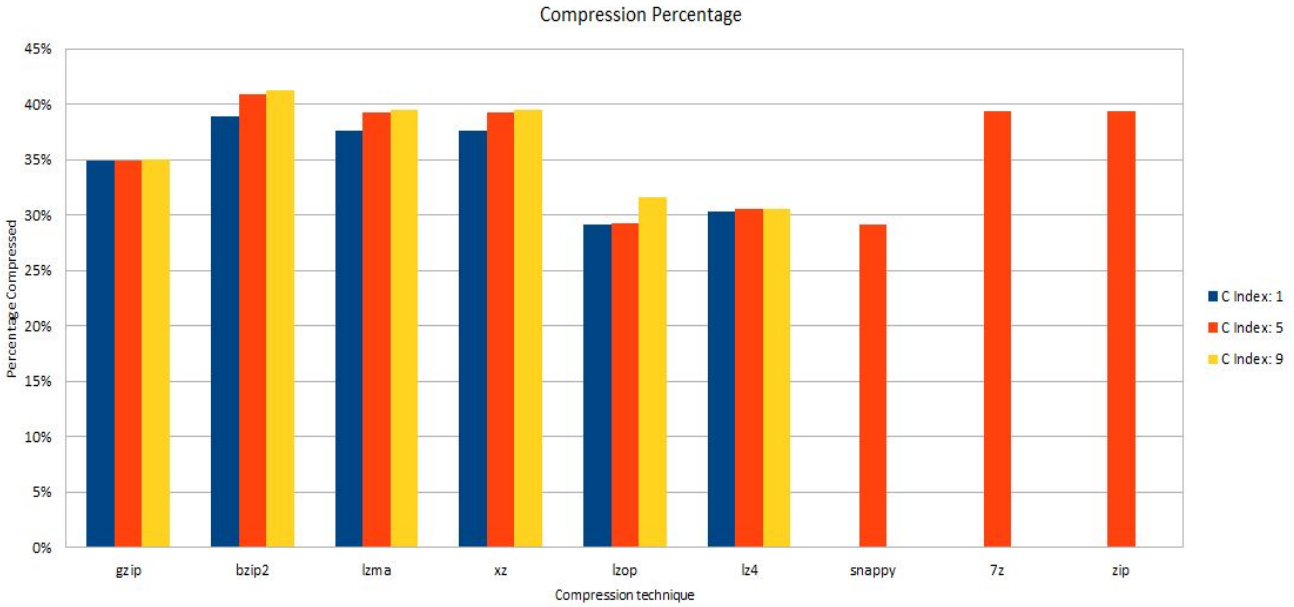
Compression Percentage comparison for different techniques at different compression index

	Compression Percentage		
File type: .tif			
Runs = 5	C Index: 1	C Index: 5	C Index: 9
gzip	35%	35%	35%
bzip2	39%	41%	41%
lzma	38%	39%	39%
xz	38%	39%	39%
lzop	29%	29%	32%
lz4	30%	31%	31%
snappy		29%	
7z		39%	
zip		39%	

C Index stands for Compression Index

Size of .tif file = 117.4 mb

Percentage Compressed analysis (More is better)
Max compressed – bzip2



.tif file – run on laptop

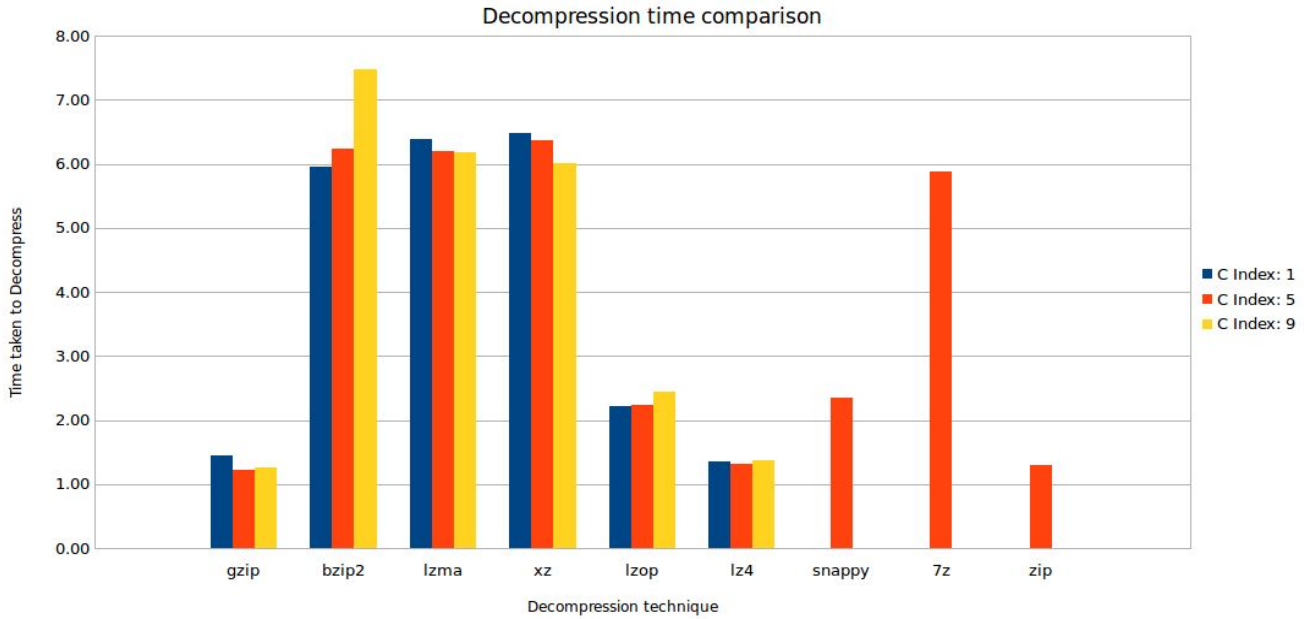
Decompression Time comparison for different techniques at different compression index

		Decompression time per run (in sec.)		
File type: .tif		C Index: 1	C Index: 5	C Index: 9
Runs = 5	gzip	1.44	1.22	1.26
	bzip2	5.96	6.24	7.48
	lzma	6.38	6.19	6.18
	xz	6.48	6.37	6.01
	lzop	2.23	2.23	2.45
	lz4	1.35	1.33	1.38
	snappy		2.35	
	7z		5.88	
	zip		1.29	

C Index stands for Compression Index

Decompression time analysis (Less is better)

Least time - gzip



.tif file – run on laptop

<u>On laptop</u>: (Processor: Intel® Core™ i5-4210U CPU @ 1.70GHz; RAM: 4 GB)

- Fastest compression – snappy
- Best compression - bzip2 (index 9)
- Fastest decompression - gzip
- <u>Best overall</u> - gzip

.nc file – run on HPC (server)

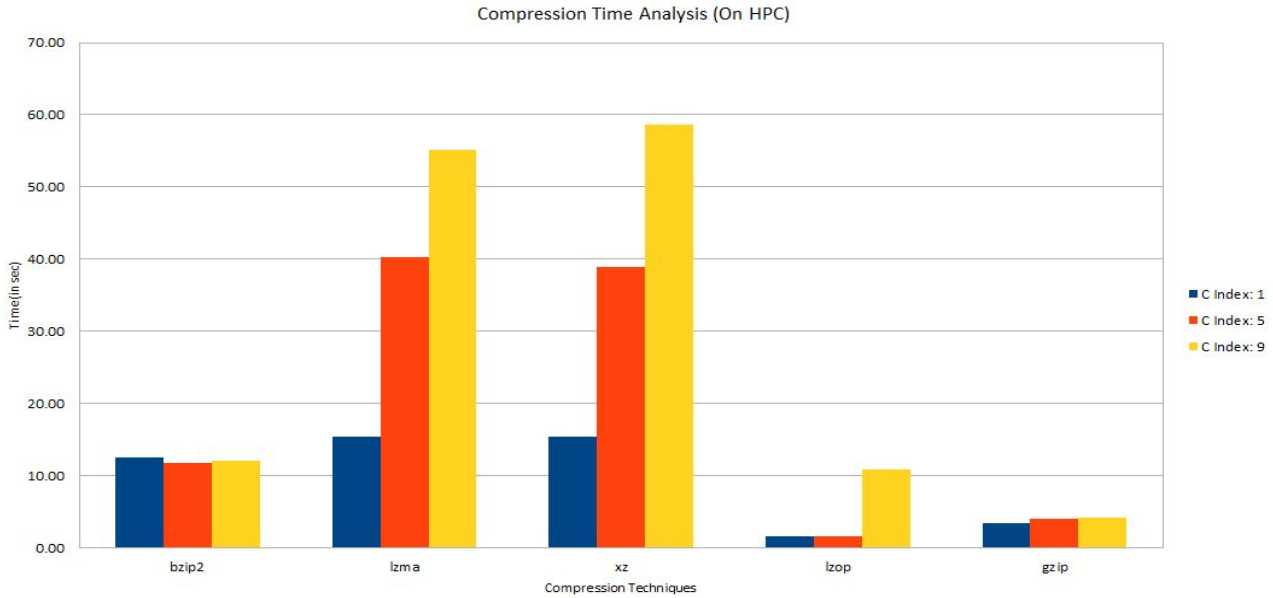
Compression Time comparison for different techniques at different compression index

File type: .nc		Compression time per run (in sec.)		
Runs = 5		C Index: 1	C Index: 5	C Index: 9
	bzip2	12.51	11.83	12.10
	lzma	15.37	40.28	55.16
	xz	15.40	38.97	58.64
	lzop	1.66	1.60	10.86
	gzip	3.46	4.07	4.20

C Index stands for Compression Index

Compression time analysis (Less is better)

Least time - Lzop



.nc file – run on HPC (server)

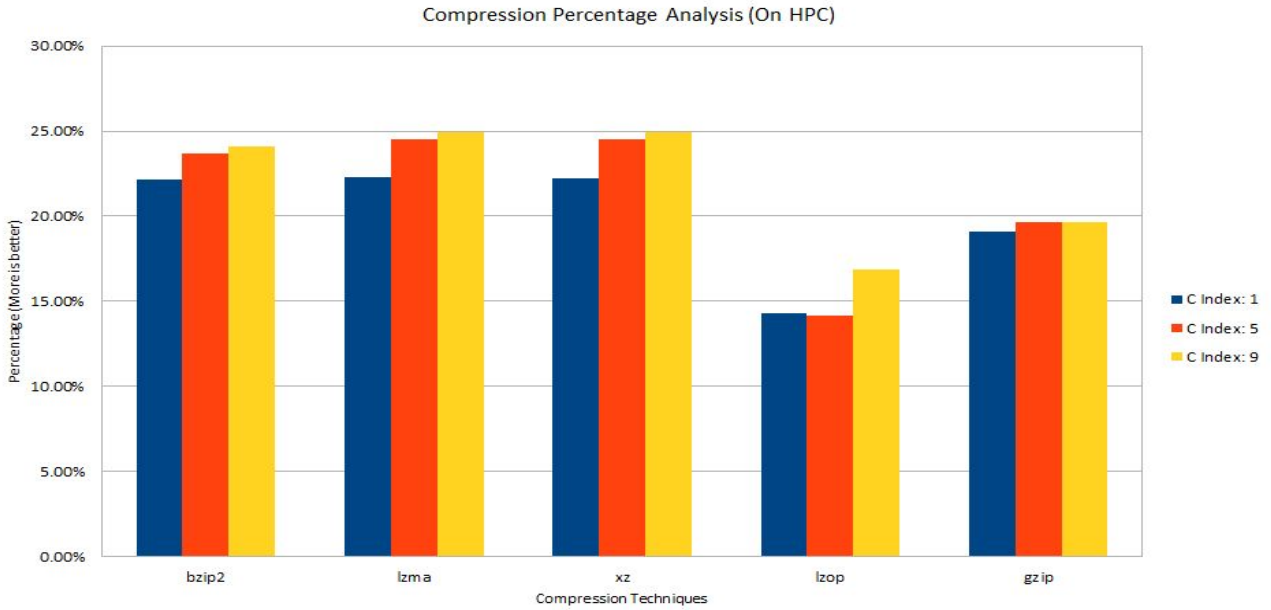
Compression Percentage comparison for different techniques at different compression index

		Compression Percentage		
File type: .nc		C Index: 1	C Index: 5	C Index: 9
Runs = 5				
	bzip2	22.12%	23.69%	24.12%
	lzma	22.26%	24.55%	24.92%
	xz	22.24%	24.54%	24.91%
	lzop	14.32%	14.12%	16.89%
	gzip	19.10%	19.61%	19.62%
	C Index stands for Compression Index			

Size of .nc file = 106.3 mb

Percentage compressed analysis (More is better)

Max compressed - Lzma



.nc file – run on HPC (server)

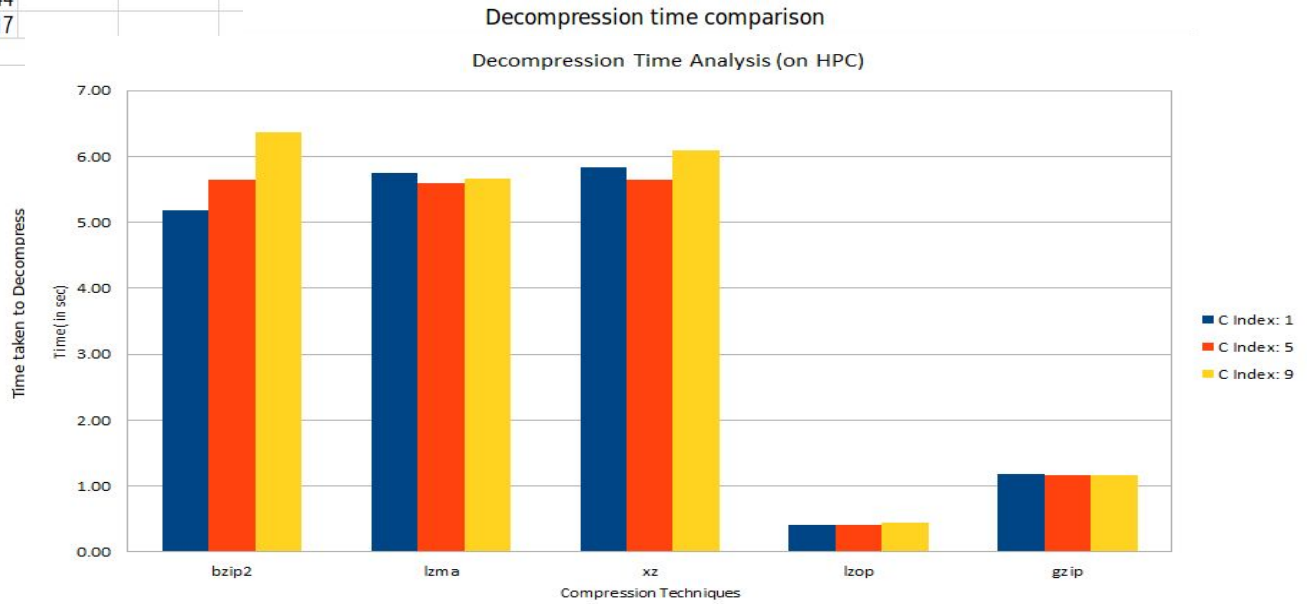
Decompression Time comparison for different techniques at different compression index

File type: .nc		Decompression time per run (in sec.)		
Runs = 5		C Index: 1	C Index: 5	C Index: 9
	bzip2	5.18	5.65	6.36
	lzma	5.75	5.59	5.67
	xz	5.83	5.64	6.10
	lzop	0.42	0.41	0.44
	gzip	1.19	1.16	1.17

C Index stands for Compression Index

Decompression time analysis (Less is better)

Least time - Lzop



.nc file – run on HPC (server)

On HPC: (Single threaded techniques)

- Best compression - lzma
- Fastest compression - lzop
- Fastest decompression – lzop

Gzip and lzop are the best options considering our needs for fast retrieval time and decent compression ratio.

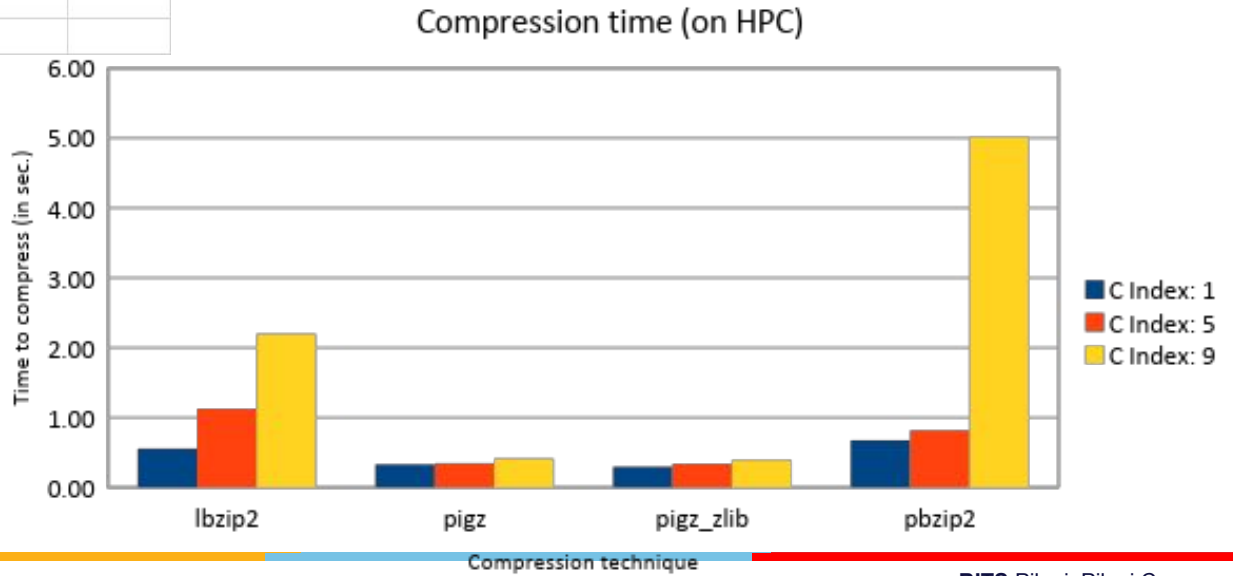
.nc file – run on HPC (Multi threaded techniques)

Compression Time comparison for different techniques at different compression index

File type: .nc		Compression time (in sec.)		
Runs = 5		C Index: 1	C Index: 5	C Index: 9
	lbzip2	0.56	1.12	2.20
	pigz	0.33	0.34	0.41
	pigz_zlib	0.30	0.34	0.39
	pbzip2	0.68	0.82	5.02

Compression time analysis (Less is better)

Least time - Pigz



.nc file – run on HPC (Multi threaded techniques)

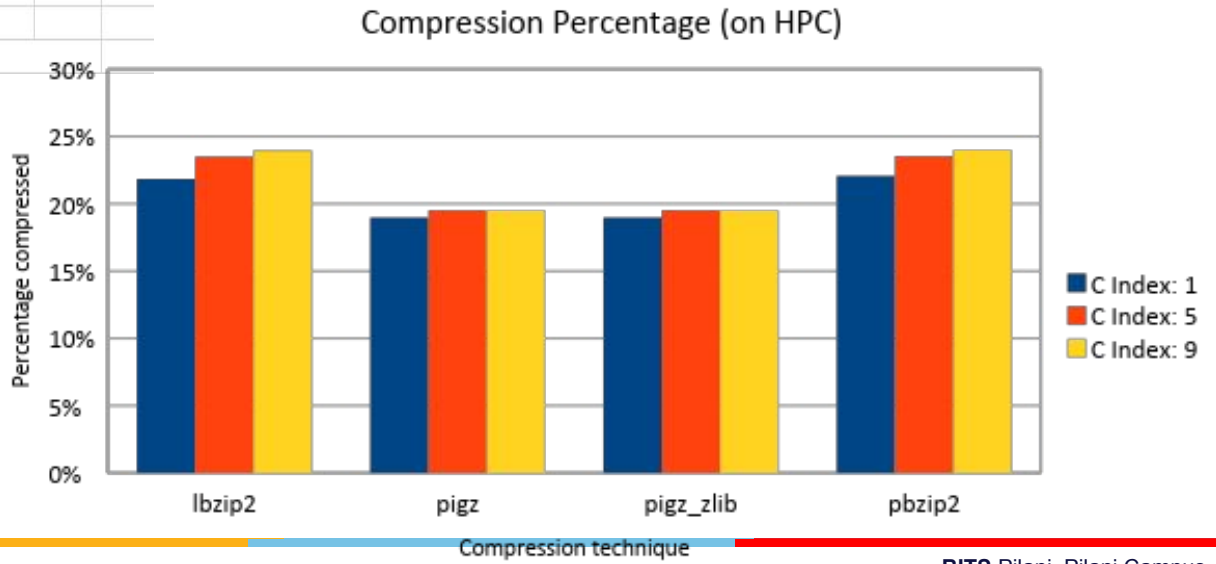
Compression Percentage comparison for different techniques at different compression index

File type: .nc		Compression Percentage		
Runs = 5		C Index: 1	C Index: 5	C Index: 9
	lbzip2	22%	24%	24%
	pigz	19%	20%	20%
	pigz_zlib	19%	20%	20%
	pbzip2	22%	24%	24%

C Index stands for Compression Index

Size of .nc file = 106.3 mb

Percentage compressed analysis (More is better)
Max compressed – Pbzip2



.nc file – run on HPC (Multi threaded techniques)

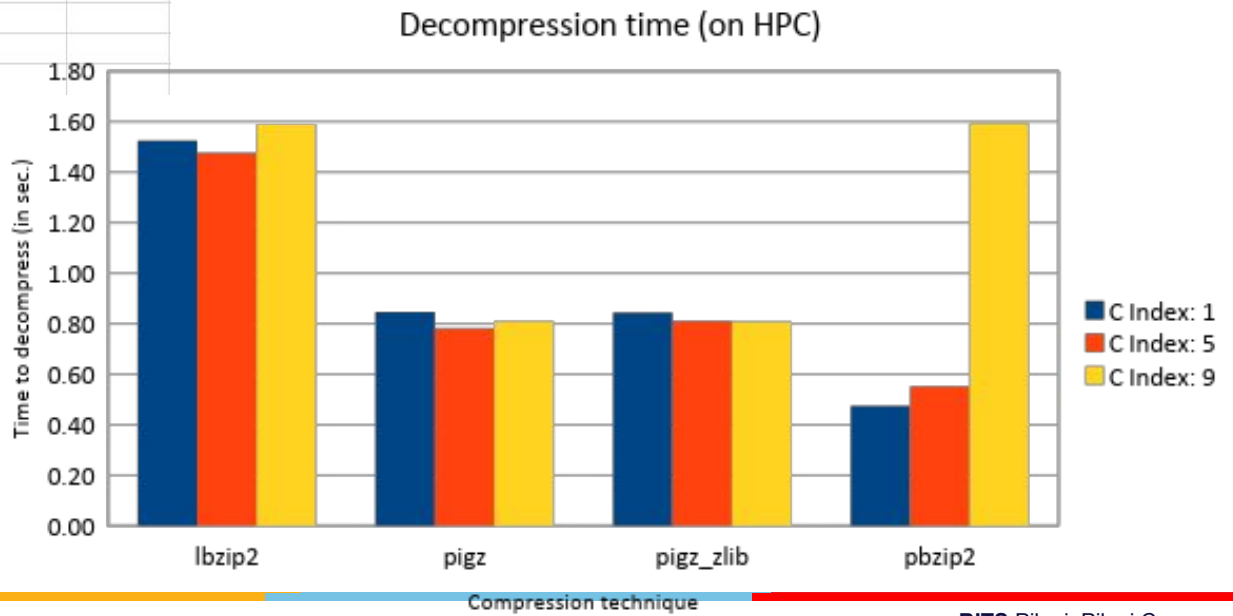
Decompression Time comparison for different techniques at different compression index

File type: .nc		Decompression time (in sec.)		
Runs = 5		C Index: 1	C Index: 5	C Index: 9
	lbzip2	1.52	1.48	1.59
	pigz	0.85	0.78	0.81
	pigz_zlib	0.84	0.81	0.81
	pbzip2	0.47	0.55	1.59

C Index stands for Compression Index

Decompression time analysis (Less is better)

Least time – Pbzip2 (index 1 & 5)

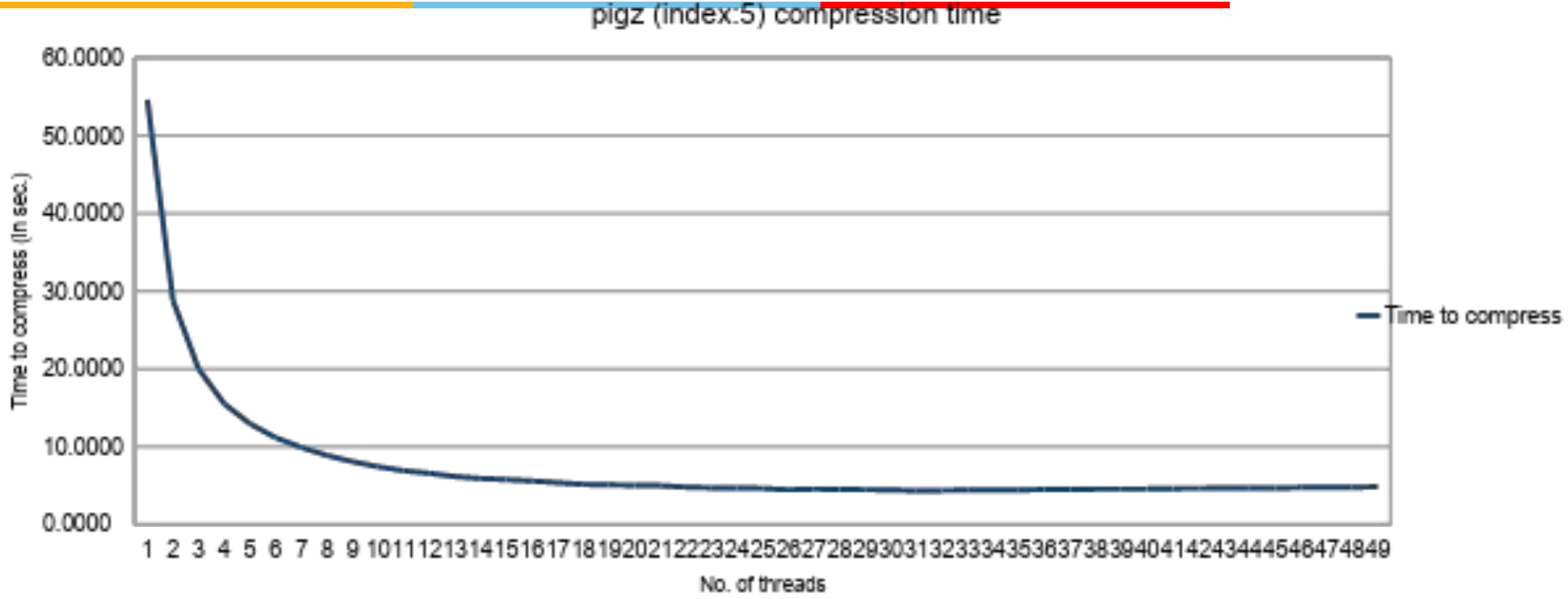


.nc file – run on HPC (Multi threaded techniques)

On HPC:

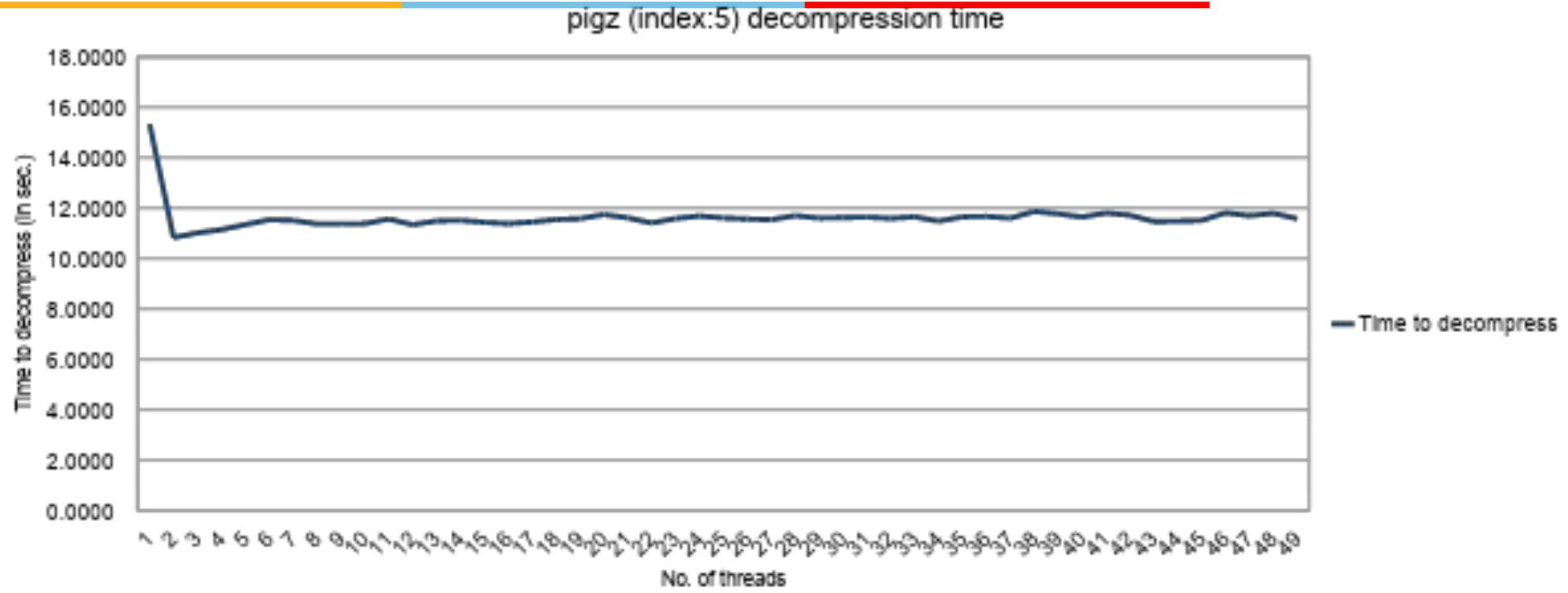
- Fastest compression – pigz
- Best compression - pbzip2 and lbzip2
- Fastest decompression - pbzip2 (index 1 and 5)

Pigz (index:5) compression time analysis at different no. of threads



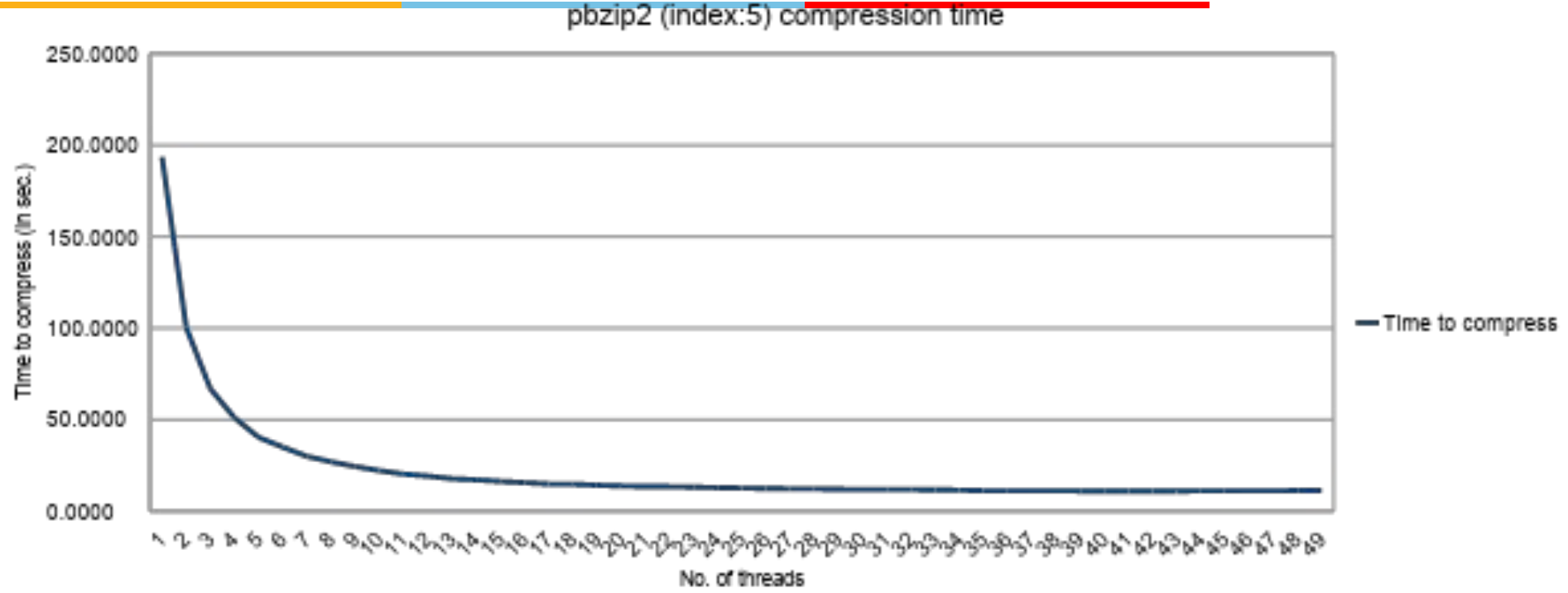
Optimal no. of threads = 26

Pigz (index:5) decompression time analysis at different no. of threads



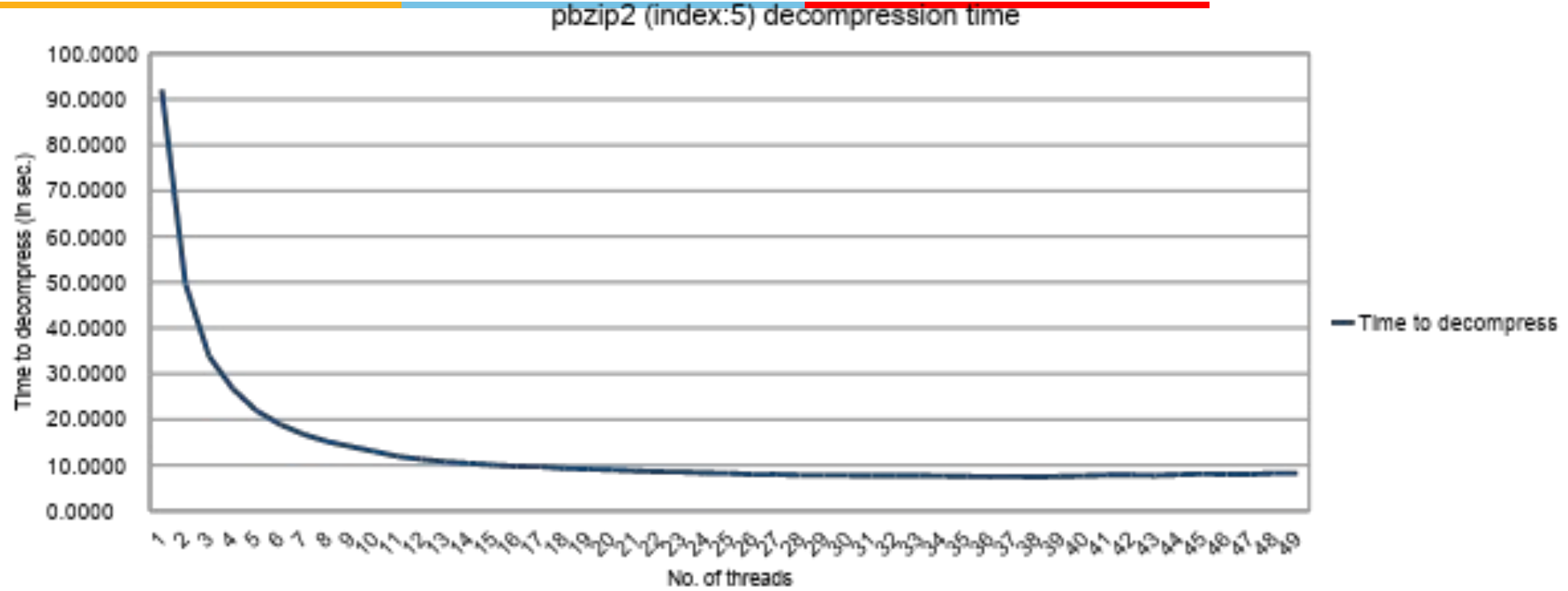
Optimal no. of threads = 3

Pbzip2 (index:5) compression time analysis at different no. of threads



Optimal no. of threads = 37

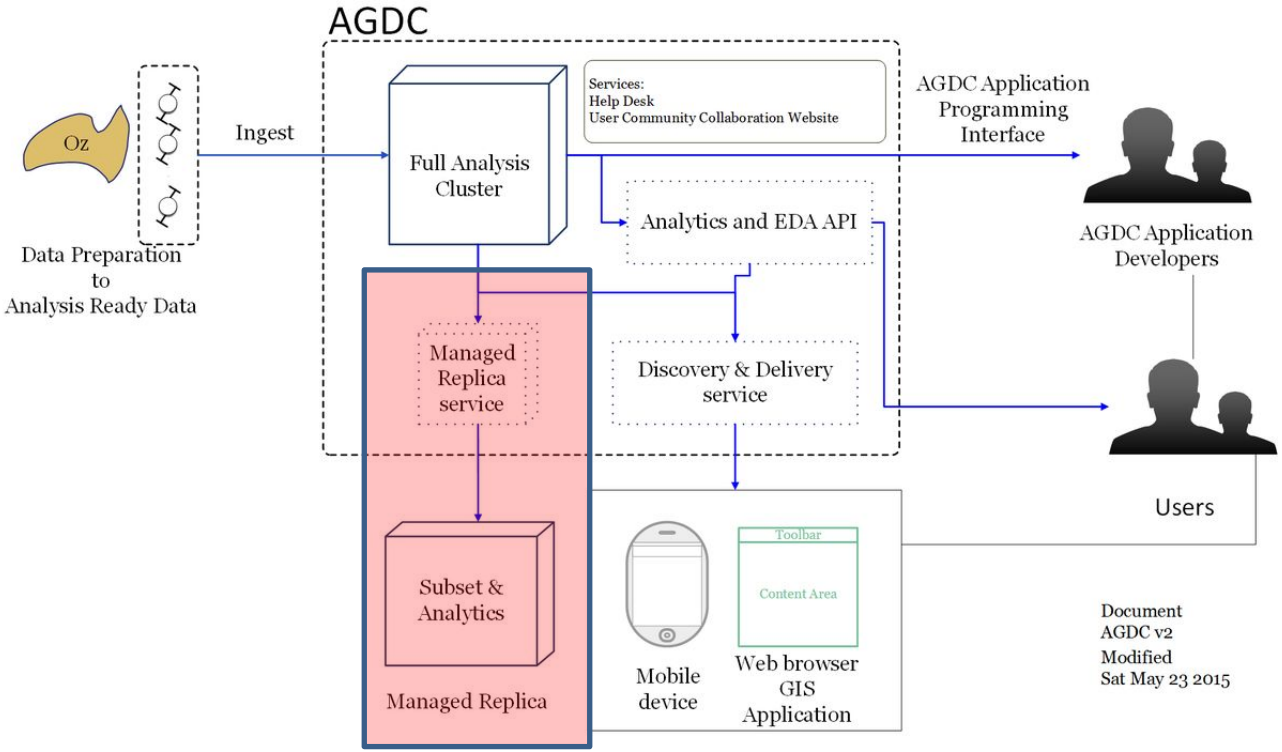
Pbzip2 (index:5) decompression time analysis at different no. of threads



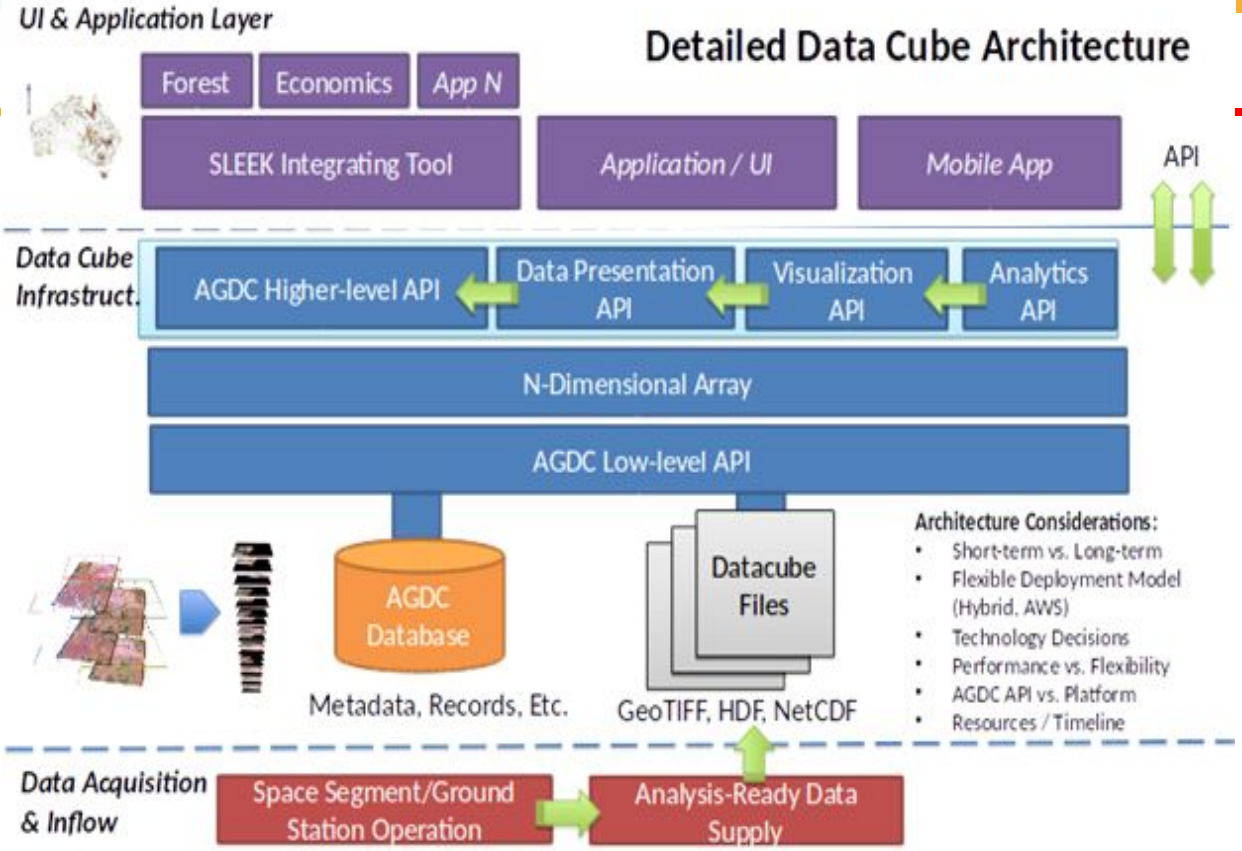
Optimal no. of threads = 32

ARCHITECTURE AND IMPLEMENTATION

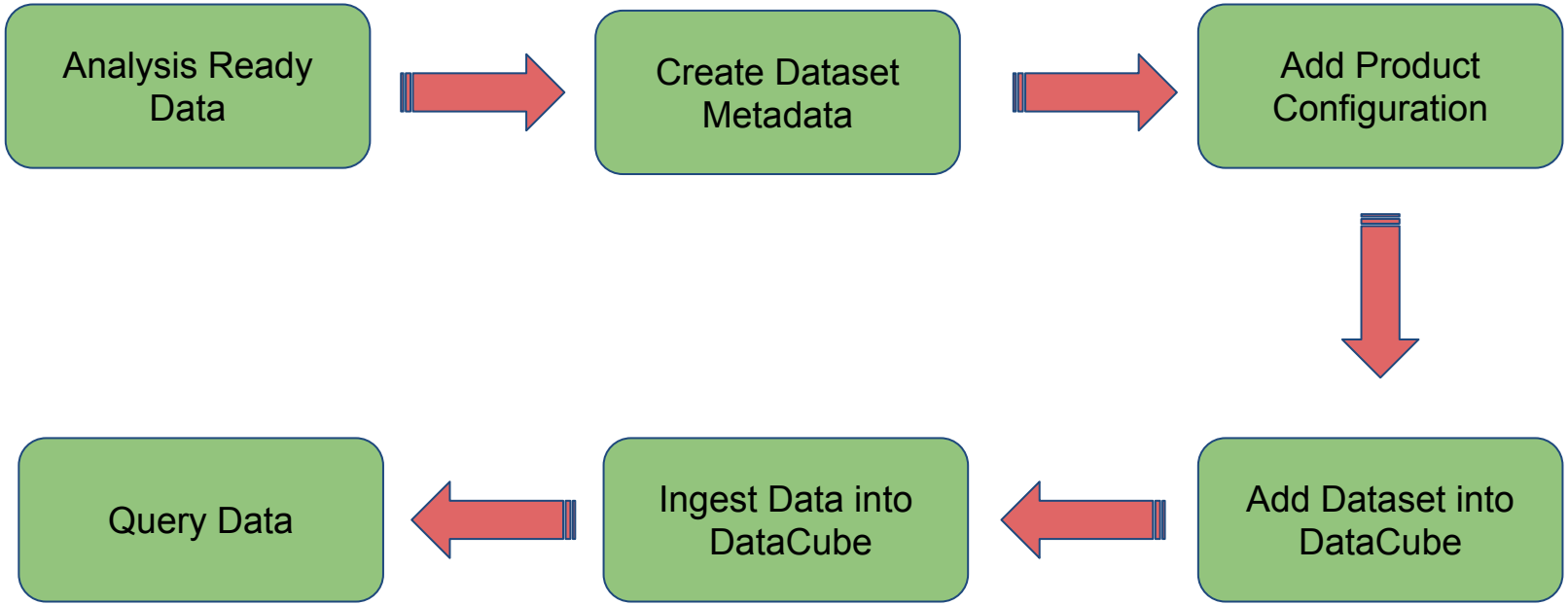
High Level Architecture | AGDC

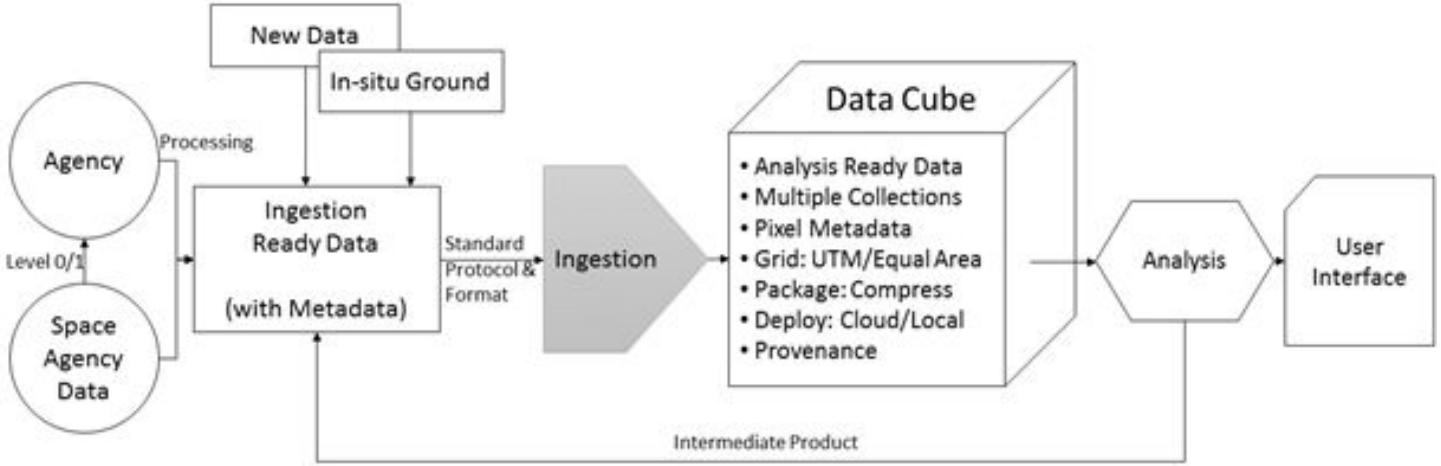


Detailed Data Cube Architecture



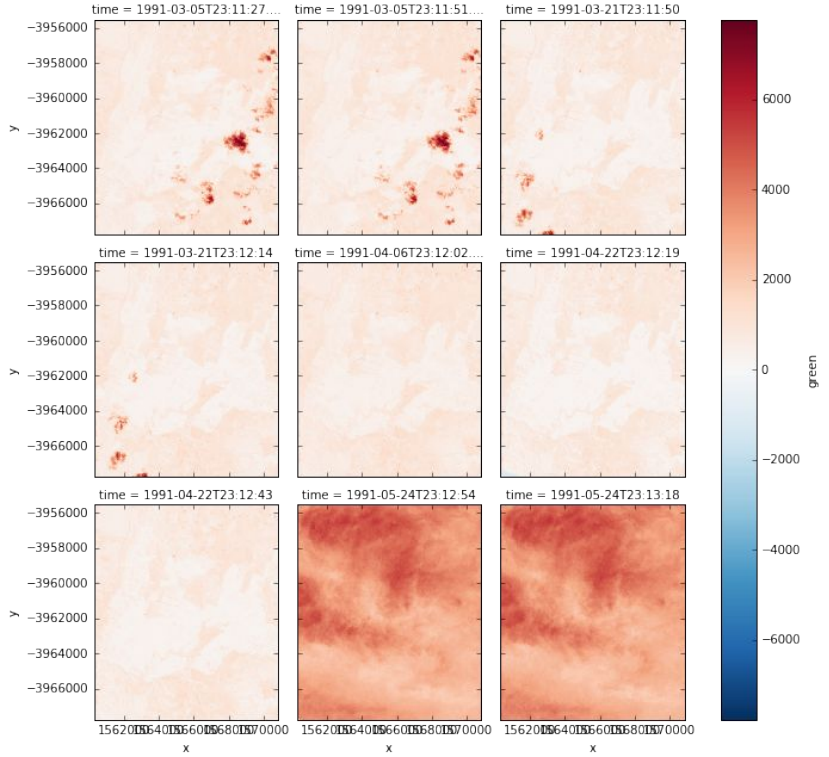
Process Flowchart




New Data
In-situ Ground
Agency
Processing
Level 0/1
Space Agency Data
Ingestion Ready Data
(with Metadata)
Standard Protocol & Format
Ingestion
Data Cube
• Analysis Ready Data
• Multiple Collections
• Pixel Metadata
• Grid: UTM/Equal Area
• Package: Compress
• Deploy: Cloud/Local
• Provenance
Analysis
User Interface
Intermediate Product

OUTPUT

```
>>  la = dc.load(product='ls8_ledaps_albers',
          x=(78.0, 78.05), y=(30.0, 31.05))
>>   a = nbar.green.loc['1991-3':'1991-5']
>>   a.plot(col='time', col_wrap=3)
```



Satellite Data Cube!

Year

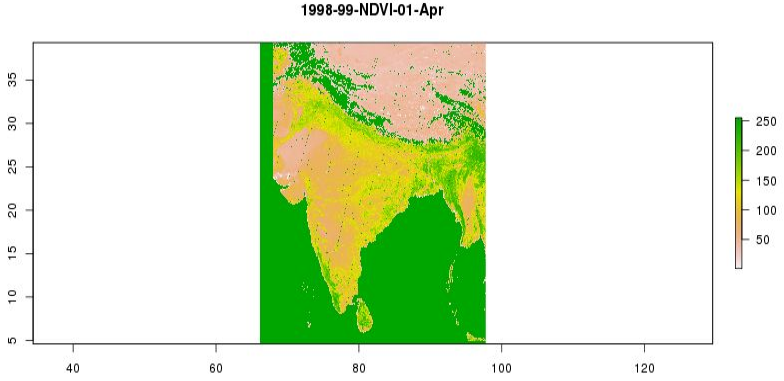
1,998 — 2,013

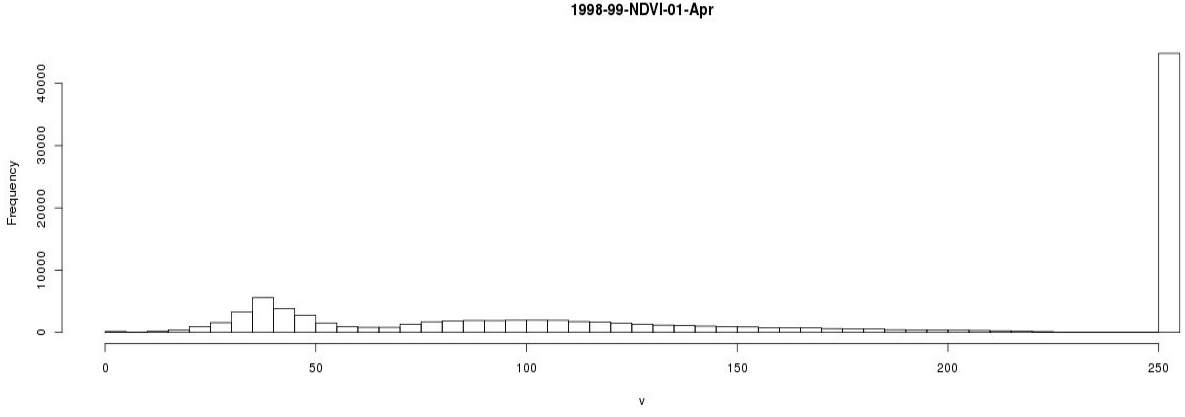
1,998 2,000 2,002 2,004 2,006 2,008 2,010 2,012 2,013

Layer No

1 — 36

1 5 9 13 17 21 25 29 33 36



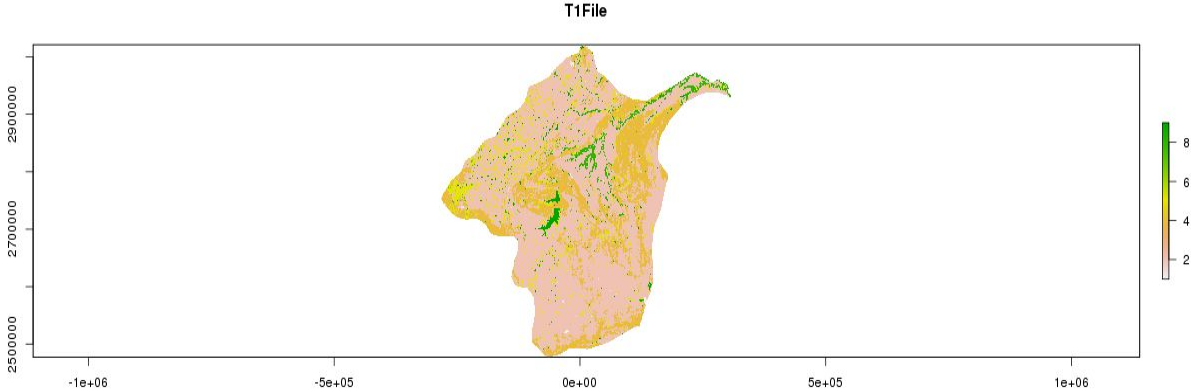


Satellite Data Cube!

Layer To Display

1 54

1 7 13 19 25 31 37 43 49 54



The END!!


THANK YOU
Dziękuję
Děkuji
Terima kasih
감사합니다
Köszönöm
謝謝
Merci
Danke
Gracias
Obrigado
ありがとう
Tesekkür ederim
Kiitos
Takk
Salamat