EOSDIS
NASA'S EARTH OBSERVING SYSTEM
DATA AND INFORMATION SYSTEM

Utilizing HDF4 File Content Maps for the Cloud

Hyokyung Joe Lee
The HDF Group

This work was supported by NASA/GSFC under
Raytheon Co. contract number NNG15HZ39C

HDF File Format is for Data.

- PDF for Document, HDF for Data

- Why PDF over MS Word DOC?
  - Free, Portable, Sharing & Archiving
- Why HDF over MS Excel XLS(X)?
  - Free, Portable, Sharing & Archiving
- HDF: HDF4 & HDF5

EOSDIS

HDF4 is “old” format.

- Old = Large volume over long time
- Old = Limitation (32-bit)
- Old = Less and less support

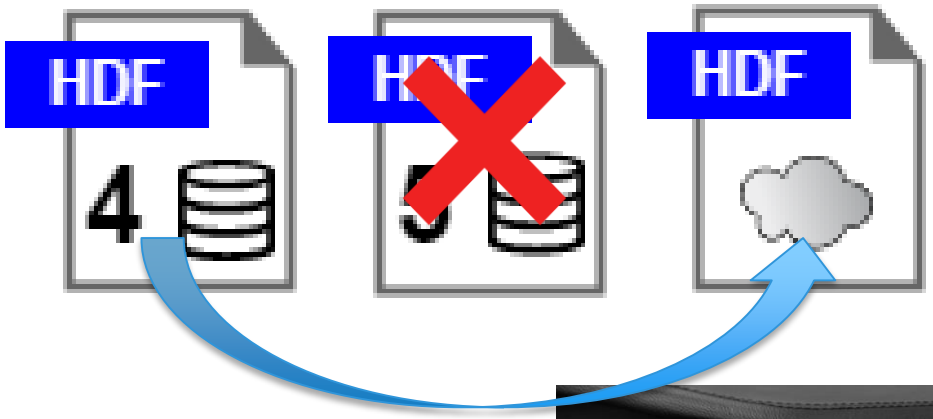
EOSDIS

HDF4 is old. So What?

- Convert it to HDF5.

EOSDIS

Any alternative? Cloudification!

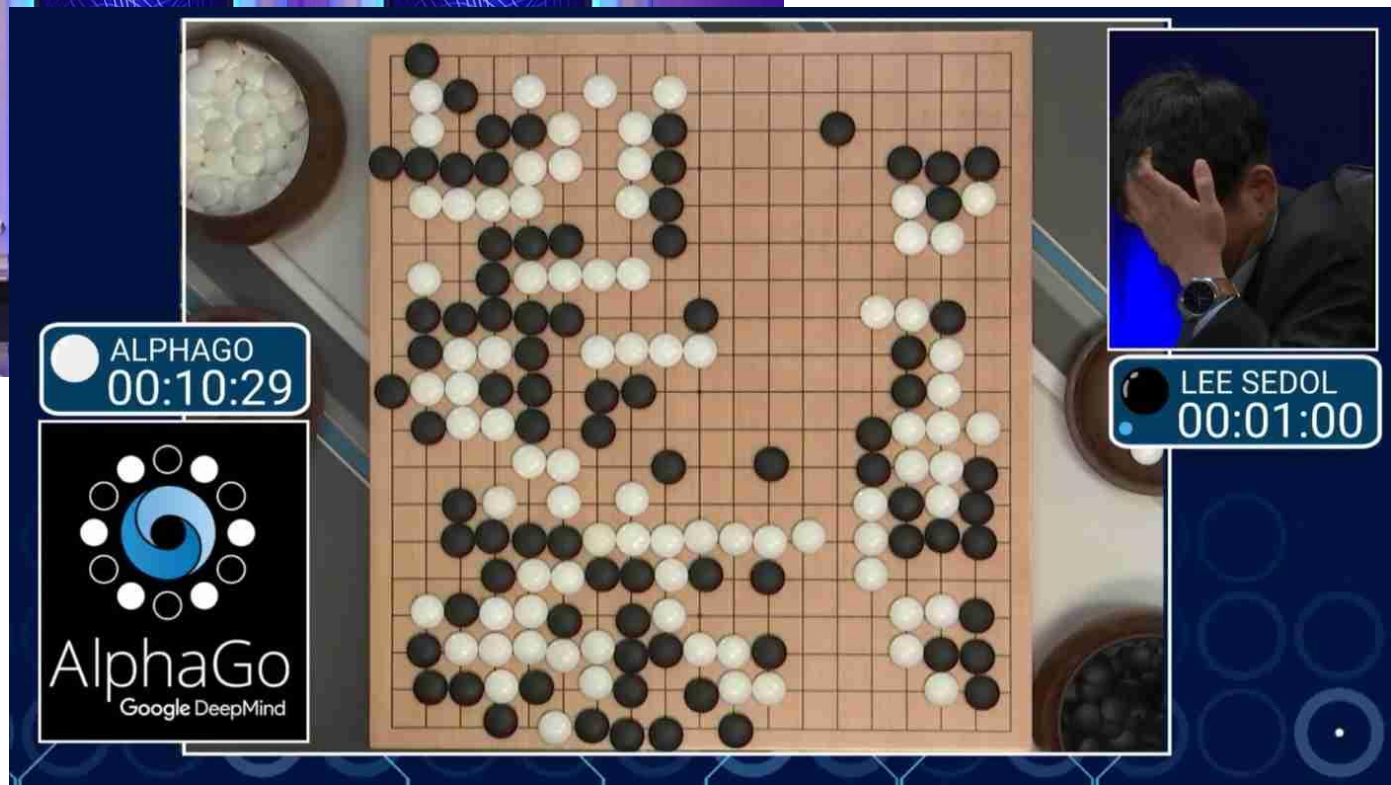


EOSDIS

Cloudification - Wiktionary

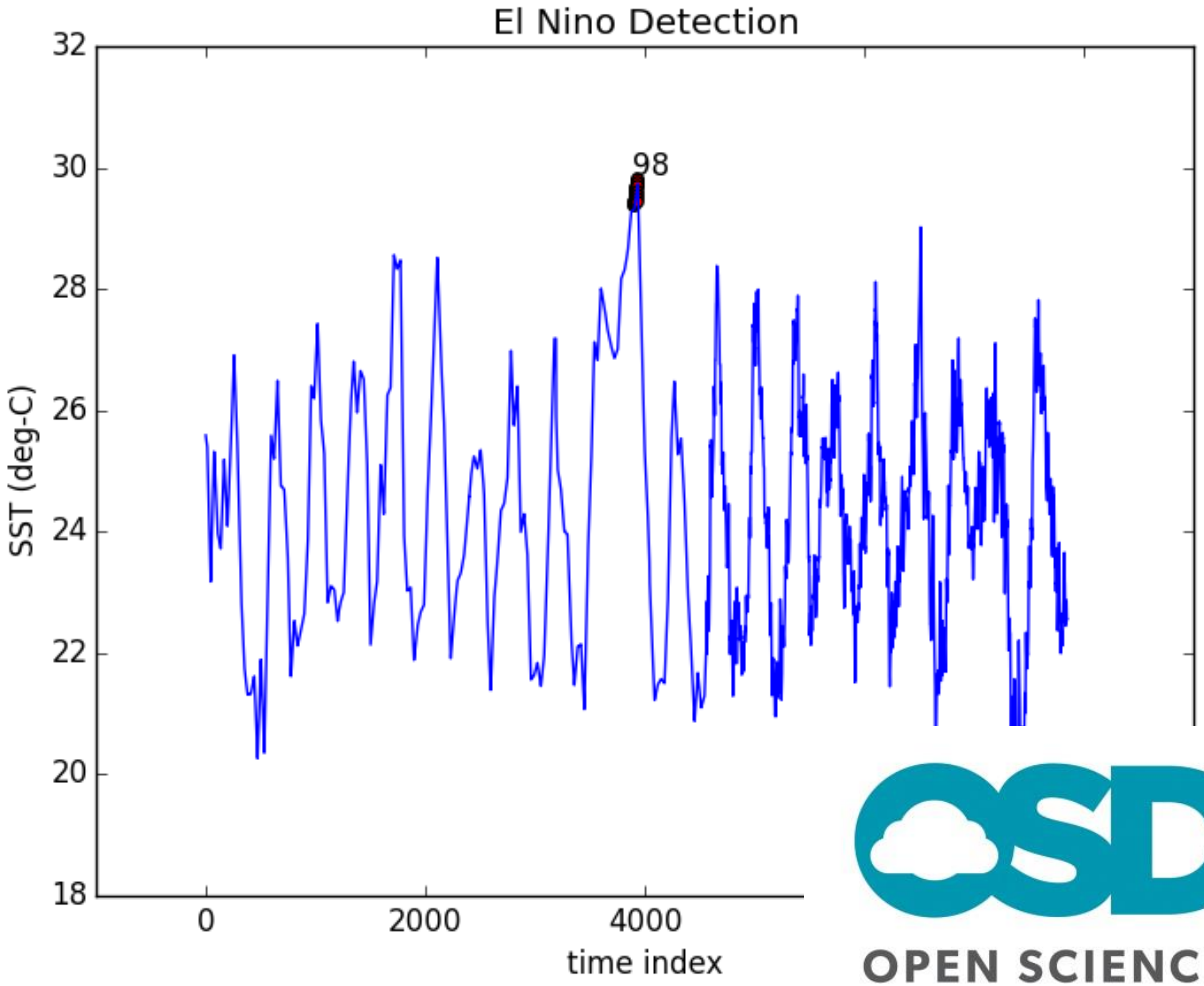
The conversion and/or migration of data and application programs in order to make use of cloud computing

Why Cloud? AI+Bigdata+Cloud =



EOSDIS

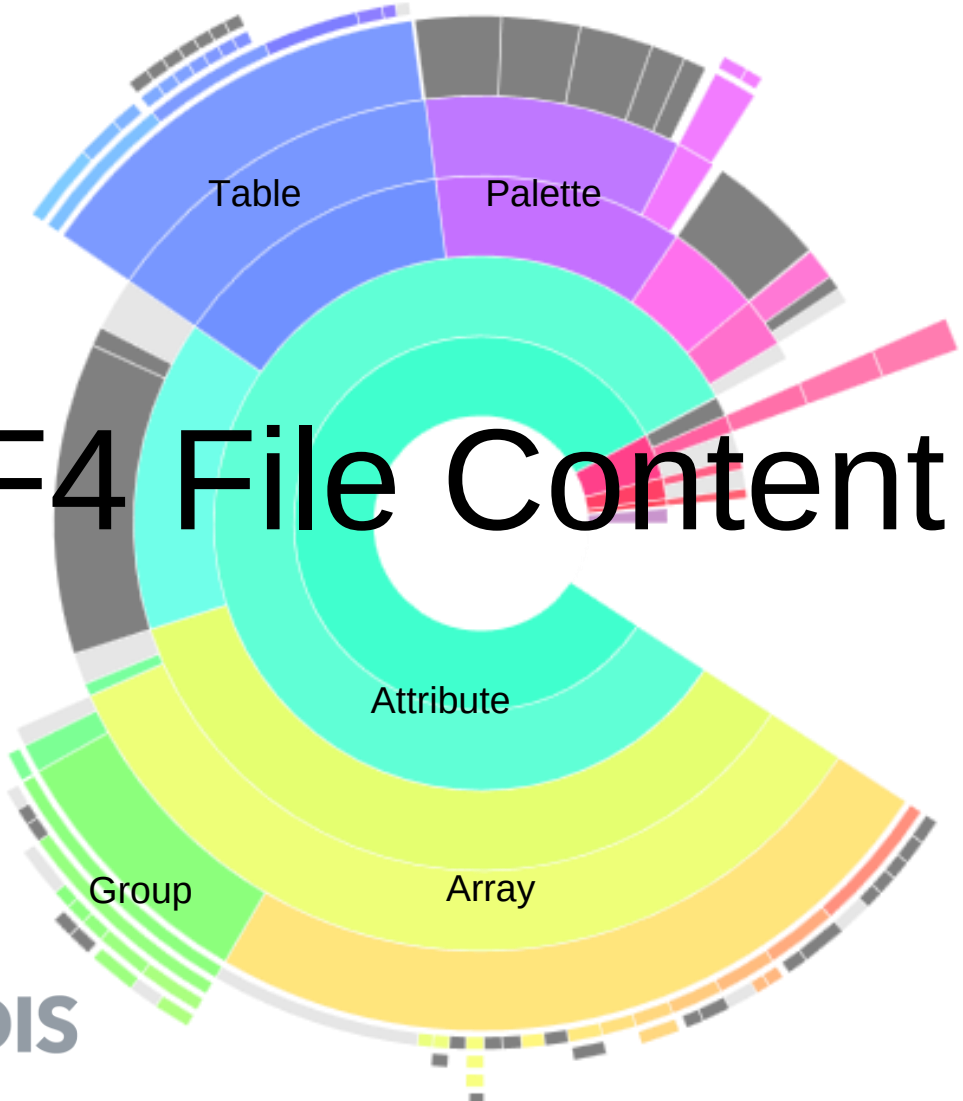
ABC Example: El Nino Detection



OPEN SCIENCE DATA CLOUD

EOSDIS

Cloudification is cool but how?



Use

HDF4 File Content Map.

EOSDIS

What is HDF4 Map?

XML (ASCII) file that explains the content of binary file.

```
 <h4:Array name="c" path="/a/b/" nDimensions="4">
<h4:dataDimensionSizes>180 8 32 4</h4:dataDimensionSizes>
<h4:chunks>
 <h4:chunkDimensionSizes>1 8 32 4</h4:chunkDimensionSizes>
<h4:fillValues value="-9999.000000" chunkPositionInArray="[0,0,0,0]"/>
…
<h4:byteStream offset="70798703" nBytes="2468"
chunkPositionInArray="[114,0,0,0]"/>

<h4:byteStream offset="89101024" nBytes="32"
chunkPositionInArray="[172,0,0,0]"/>

<h4:byteStream offset="89127527" nBytes="32"
chunkPositionInArray="[173,0,0,0]"/>
```

```
0000   FF D8 FF E1   1D FE 45 78   69 66 00 00   49 49 2A 00
0010   08 00 00 00   09 00 0F 01   02 00 06 00   00 00 7A 00
0020   00 00 10 01   02 00 14 00   00 00 80 00   00 00 12 01
0030   03 00 01 00   00 00 01 00   00 00 1A 01   05 00 01 00
0040   00 00 A0 00   00 00 1B 01   05 00 01 00   00 00 A8 00
0050   00 00 28 01   03 00 01 00   00 00 02 00   00 00 32 01
0060   02 00 14 00   00 00 B0 00   00 00 13 02   03 00 01 00
0070   00 00 01 00   00 00 69 87   04 00 01 00   00 00 C4 00
0080   00 00 3A 06   00 00 43 61   6E 6F 6E 00   43 61 6E 6F
0090   6E 20 50 6F   77 65 72 53   68 6F 74 20   41 36 30 00
00A0   00 00 00 00   00 00 00 00   00 00 00 00   B4 00 00 00
00B0   01 00 00 00   B4 00 00 00   01 00 00 00   32 30 30 34
00C0   3A 30 36 3A   32 35 20 31   32 3A 33 30   3A 32 35 00
00D0   1F 00 9A 82   05 00 01 00   00 00 86 03   00 00 9D 82
00E0   05 00 01 00   00 00 8E 03   00 00 00 90   07 00 04 00
```

It is a map with address.

XML file that x-rays the content of binary file

<h4:Array name="c" path="/a/b/" nDimensions="4">
<h4:dataDimensionSizes>180 8 32 4</h4:dataDimensionSizes>
<h4:chunks>
<h4:chunkDimensionSizes>1 8 32 4</h4:chunkDimensionSizes>
<h4:fillValues value="-9999.000000" chunkPositionInArray="[0,0,0,0]"/>
…

<h4:byteStream offset="70798703"
nBytes="2468" chunkPositionInArray="[114,0,0,0]"/>

<h4:byteStream offset="89101024" nBytes="32"
chunkPositionInArray="[172,0,0,0]"/>

Byte size in map is quite useful.

Bigger the size, it may have more information.

```
 <h4:Array name="c" path="/a/b/" nDimensions="4">
<h4:dataDimensionSizes>180 8 32 4</h4:dataDimensionSizes>
<h4:chunks>
 <h4:chunkDimensionSizes>1 8 32 4</h4:chunkDimensionSizes>
<h4:fillValues value="-9999.000000" chunkPositionInArray="[0,0,0,0]"/>
…
<h4:byteStream offset="70798703" nBytes="2468"
chunkPositionInArray="[114,0,0,0]"/>

<h4:byteStream offset="89101024" nBytes="32"
chunkPositionInArray="[172,0,0,0]"/>

<h4:byteStream offset="89127527" nBytes="32"
chunkPositionInArray="[173,0,0,0]"/>
```

Nothing interesting

This chunk may have useful information.

Some chunks are repeated.

Clear sky in cloud data, ocean in land data, etc.

…

<h4:byteStream offset="70798703" nBytes="2468"
chunkPositionInArray="[114,0,0,0]"/>

<h4:byteStream offset="89101024" nBytes="32"
chunkPositionInArray="[172,0,0,0]"/>

<h4:byteStream offset="89127527" nBytes="32"
chunkPositionInArray="[173,0,0,0]"/>

These chunks may have same information.

Some chunk sizes are repeated.

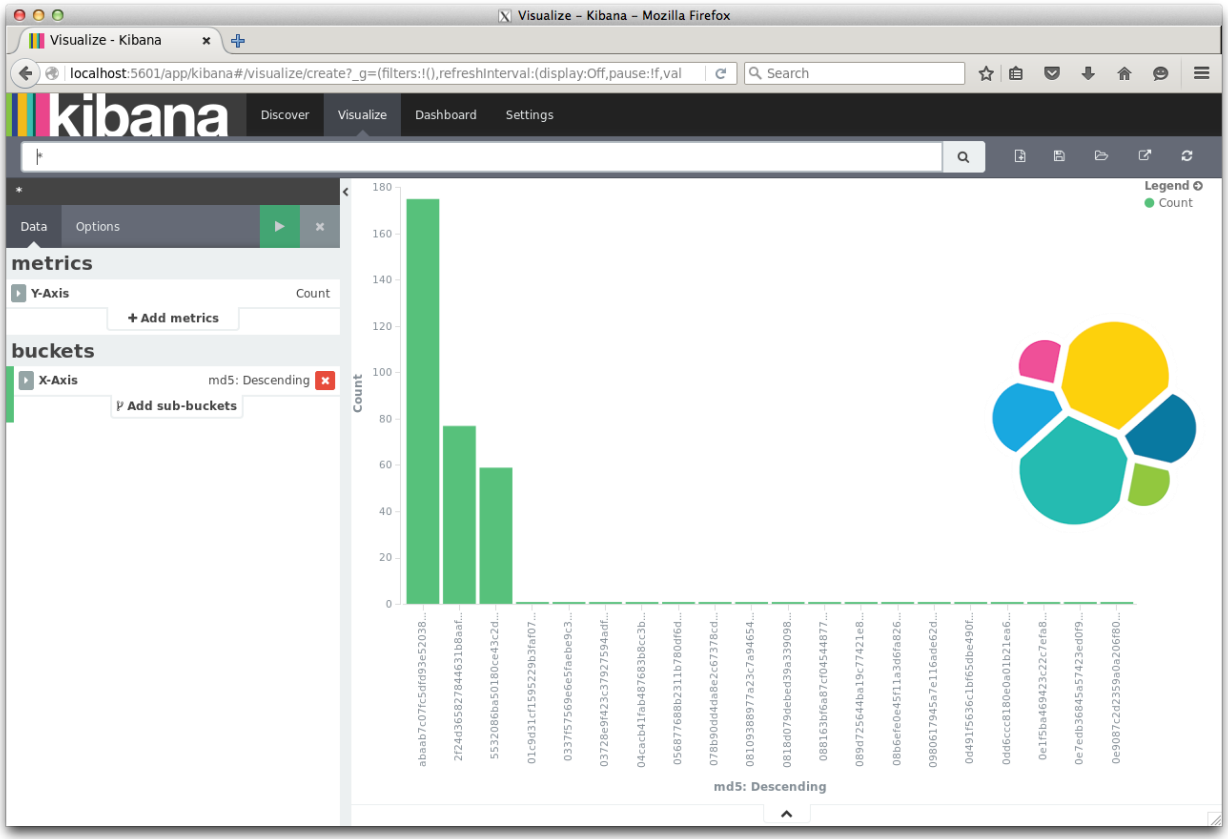
Candidates: Clear sky in cloud data, ocean in land data, etc.

```
…
<h4:byteStream offset="70798703" nBytes="2468"
chunkPositionInArray="[114,0,0,0]"/>

<h4:byteStream offset="89101024" nBytes="32"
chunkPositionInArray="[172,0,0,0]"/>

<h4:byteStream offset="89127527" nBytes="32"
chunkPositionInArray="[173,0,0,0]"/>
```

These chunks may have same information.

Run data analytics on maps.

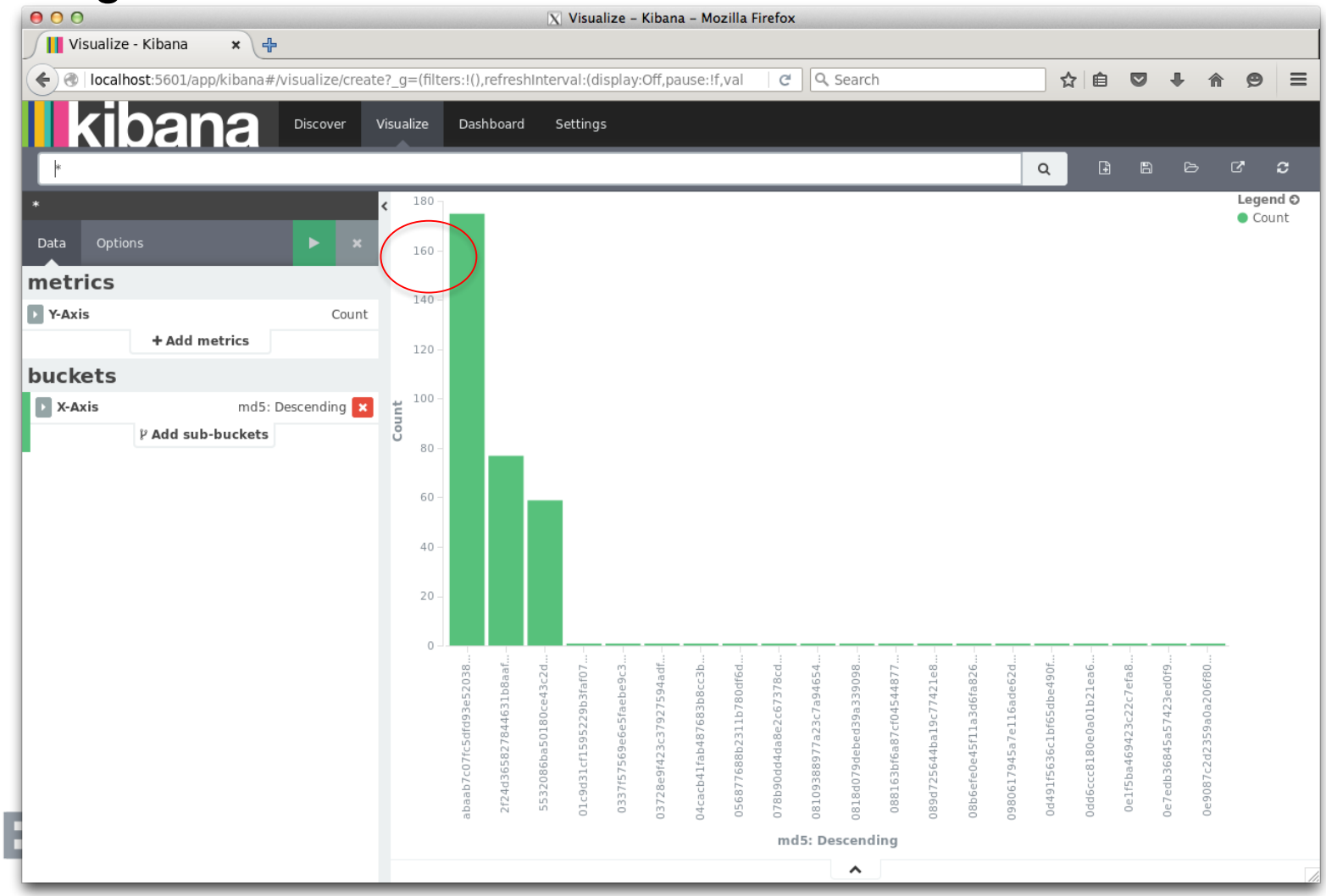
Compute checksum and use Elastic Search & Kibana.



elastic

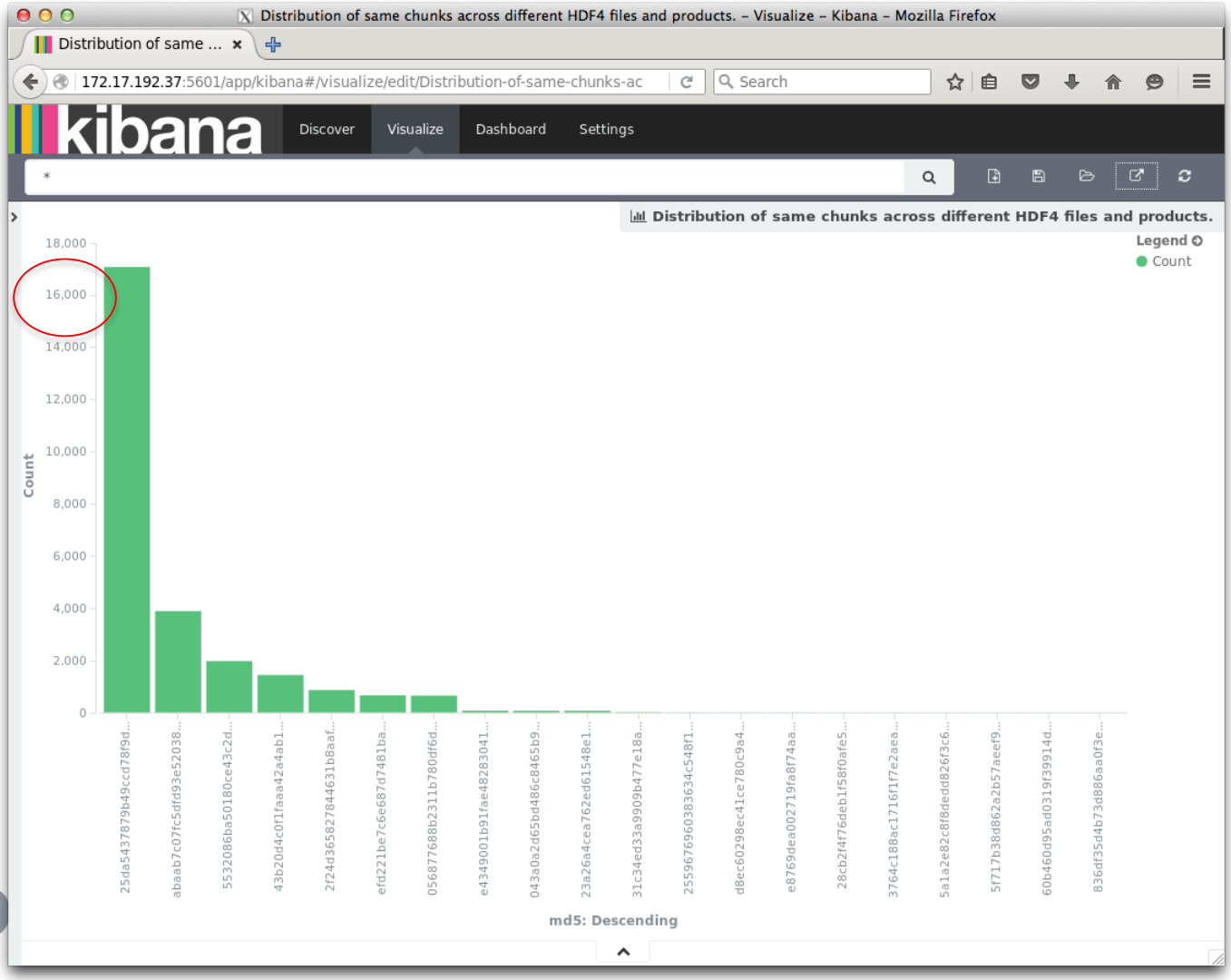
Some chunks are repeated.

A single HDF4 file has 160+ chunks of same data.



At collection level, it scales up.

Hundreds of HDF4 files have the 16K chunks of same data.



Elastic search with maps

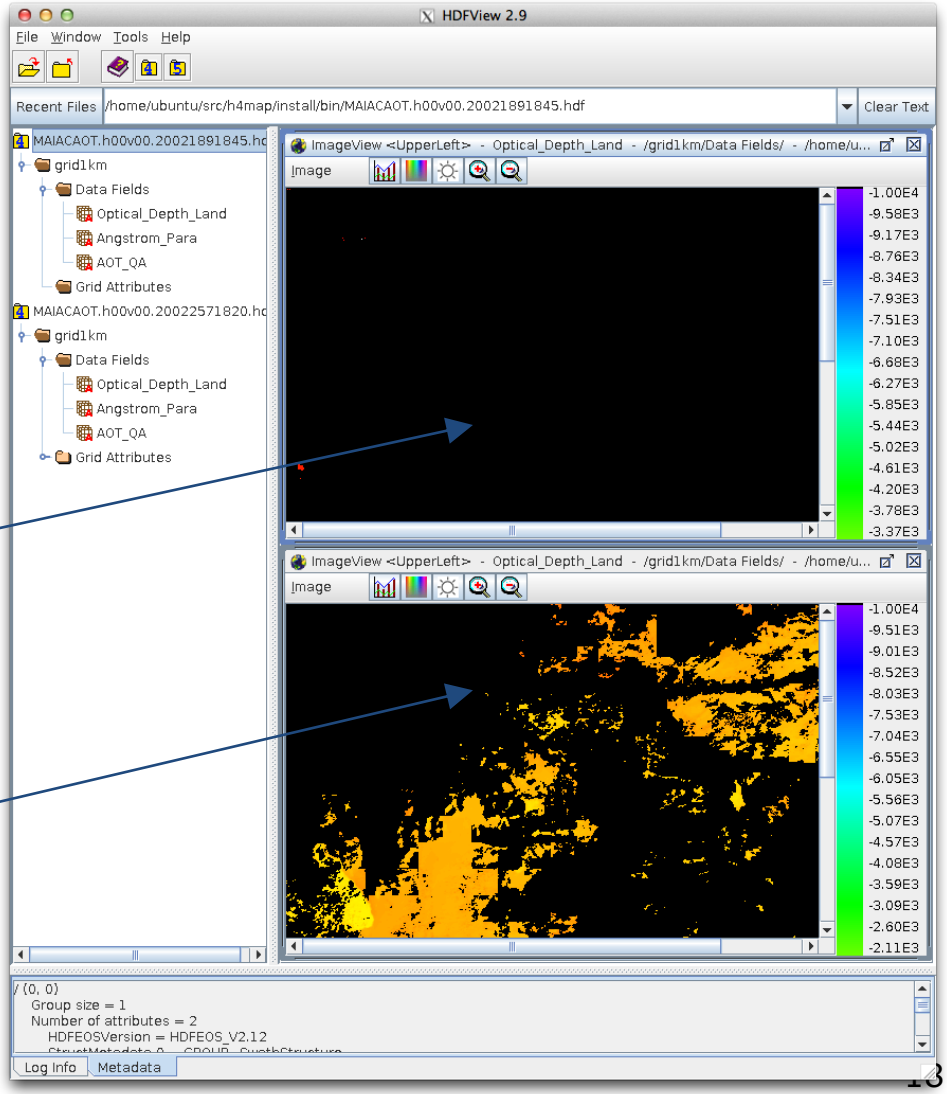
.. can help users to

locate the HDF4 file of

interest.

Nothing interesting

Most interesting



Store chunks as cloud objects

Reduce storage cost (e.g., S3) by avoiding redundancy.

Make each chunk searchable through search engine.

Run cloud computing on chunks of interest.

Shallow Web is not Enough

NASA Earthdata search is too shallow.

Index HDF4 data using maps and make deep web.

Provide search interface for the deep web.

Frequently searched data can be cached as cloud objects.

Users can run cloud computing on cached objects in RT.

Verify results with HDF4 archives from NASA data centers.

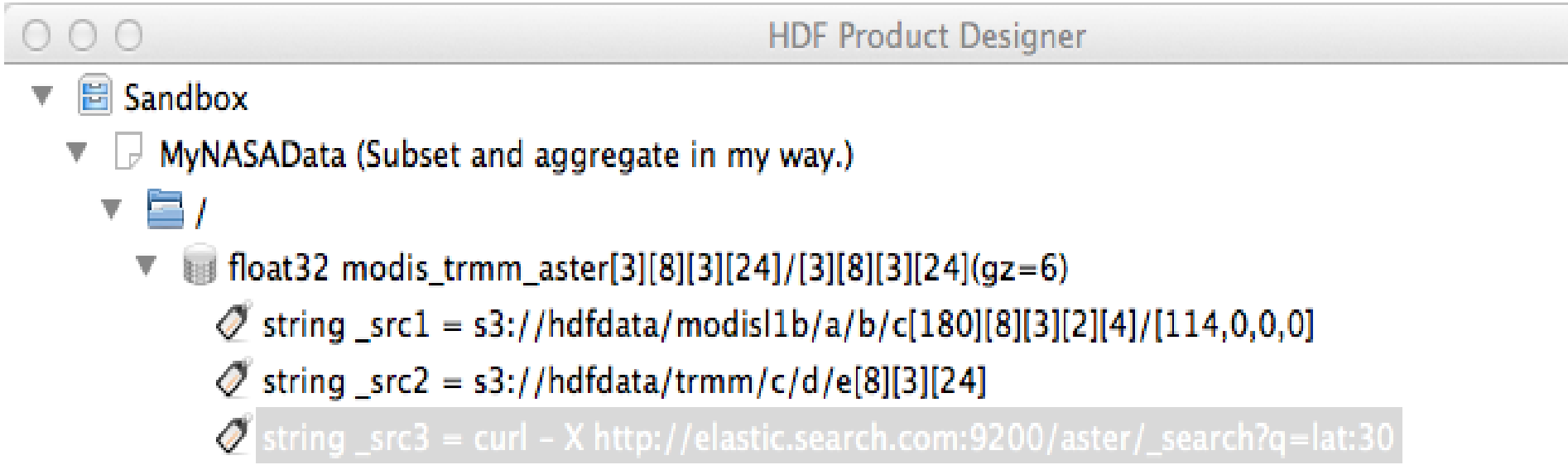
HDF: Antifragile Solution for BACC

(BACC = Bigdata Analytics in Cloud Computing)

1. Use HDF archive as is. Create maps for HDF.
2. Maps can be indexed and searched.
3. ELT (Extract Load Transform) only relevant data into cloud from HDF.
4. Offset/length based file IO is universal - all existing BACC solutions will work. No dependency on HDF APIs.

Future Work

1. HDF5 Mapping Project?
2. Use HDF Product Designer for archiving cloud objects



This work was supported by NASA/GSFC under Raytheon Co. contract number NNG15HZ39C

Raytheon

EOSDIS