What was tried as data mining with spark on Datarmor

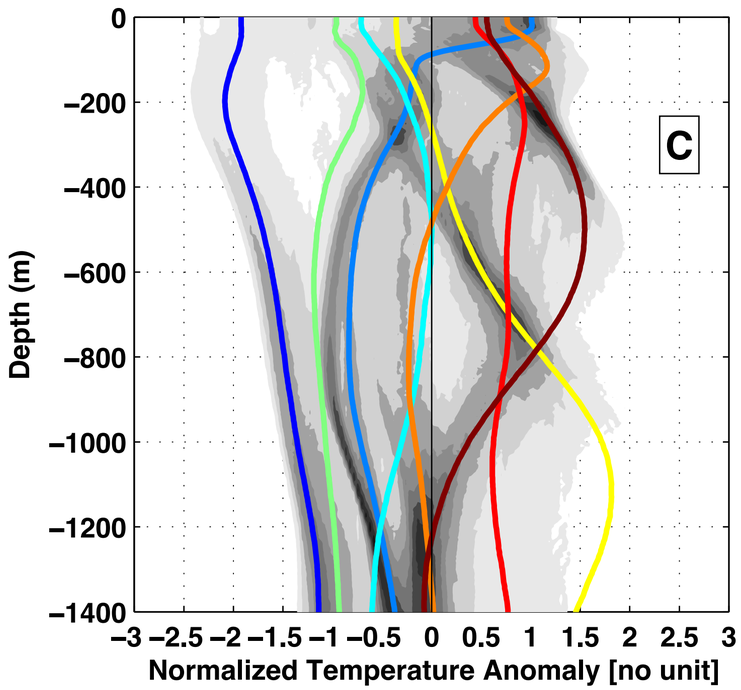
G. Maze, K. Balem (ODE-LOPS)
T. Odaka, A. Queric (IRSI-RIC)

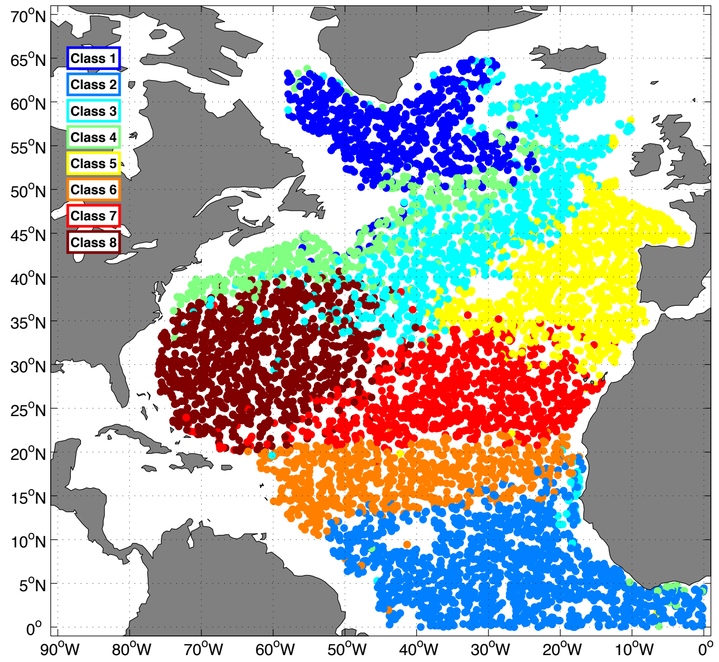
17th PCIM meeting, 2018/05/17

Unsupervised classification

Example: Profile Classification Modelling

- interpolate profiles on standard depth levels
- extract 2D plain matrix from 4D products (time series of 3D fields)
- center/standardise
- compute eigenvectors and singular values
- train Classification Model (computation and inversion of multiple covariance matrices)
- compute weighted statistics of profiles





Maze et al, 2017

Unsupervised classification

Example: Profile Classification Modelling

Machinery: A script to launch an interactive Hadoop cluster was developed

```
> qsub -I -q mpi_4 -l walltime=01:00:00
> bash
> module load rdma-hadoop/1.3.5
> source /appli/hibd/rdma-hadoop-2.x-1.3.5-x86/sbin/quick-hadoop-get-env.sh -
conda-env rdma-hadoop-1.3.5
> pyspark
Python 2.7.12 | packaged by conda-forge | (default, Sep  8 2016, 14:22:31)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-15)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.1.0
      /_/

Using Python version 2.7.14 (default, Nov  4 2017 10:15:28)
SparkSession available as 'spark'.
>>>
```

Unsupervised classification

Example: Profile Classification Modelling

Machinery: Data mining scripts developed with interactive pyspark shell and jupyter notebook

```
DATA CONNECTION/LOADING (RDD)
>>> flist = glob.glob(data_root + '*/ISAS13_20*15_fld_TEMP.nc') # 132 files
>>> reader = IsasProfileReader(dpt=slice(0,1000)) # reader based on xarray
>>> shape = reader.shape(flist[0])
    {'depth': 102, 'n_samples_per_file': 215,173, 'n_samples': 28,402,836}
>>> rdd_data = sc.parallelize(flist).flatMap(reader('TEMP')) # Spark RDD
>>> print rdd_data.first() # 1 row = 1 temperature profile
[-1.425 -1.432 -1.436 -1.445 -1.416 -1.383 -1.385 -1.387 -1.416 -1.446
 -1.475 -1.504 -1.551 -1.597 -1.644 -1.691 -1.737 -1.749 -1.76  -1.772
 -1.592 -1.542 -1.442 …

DATA NORMALISATION
>>> from pyspark.mllib.feature import StandardScaler, StandardScalerModel
>>> scaler = StandardScaler(withMean=True, withStd=True).fit(rdd_data)
>>> rdd_norm = scaler.transform(rdd_data)
>>> sample_mean = scaler.call('mean')
>>> sample_mean
DenseVector([11.2451, 11.2374, 11.2301, 11.2002, 11.165, 11.1189, 11.0488,
9.9632, 9.8308, 9.7071, 9.5862, 9.4682, …
```

Unsupervised classification

Example: Profile Classification Modelling

Machinery: Data mining scripts developed with interactive pyspark shell and jupyter notebook

DATA REDUCTION

```
>>> from pyspark.mllib.feature import PCA as PCAmllib
>>> Neof = 20
>>> reducer = PCAmllib(Neof).fit(rdd_norm)
>>> rdd_reduced = reducer.transform(rdd_norm)
```

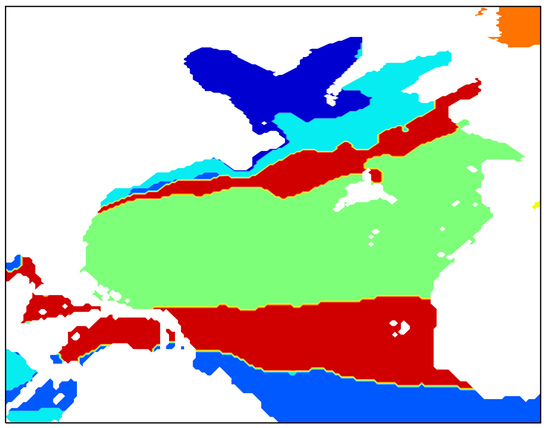
DATA CLASSIFICATION

```
>>> from pyspark.mllib.clustering import KMeans as KMeansmllib
>>> NBCLUSTERS=8
>>> INITMODE='kmean||' # or: 'random'
>>> clusters_kmean = KMeansmllib.train(
                rdd_reduced,
                NBCLUSTERS, maxIterations=200, runs=20,
                initializationMode=INITMODE)
>>> res_classif_kmean = clusters_kmean.predict(rdd_reduced)
```

Unsupervised classification

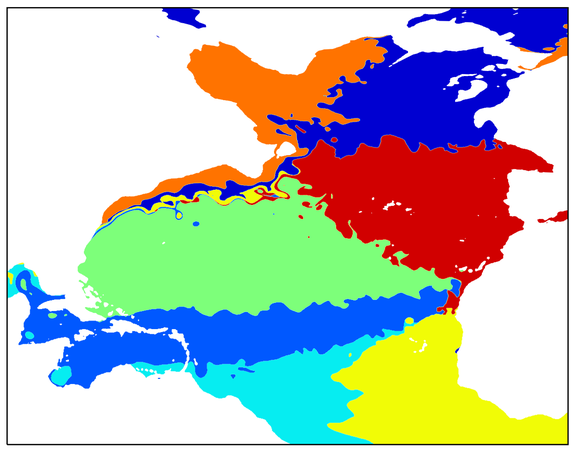
Example: Profile Classification Modelling

Machinery: Data mining scripts developed with interactive pyspark shell and jupyter notebook



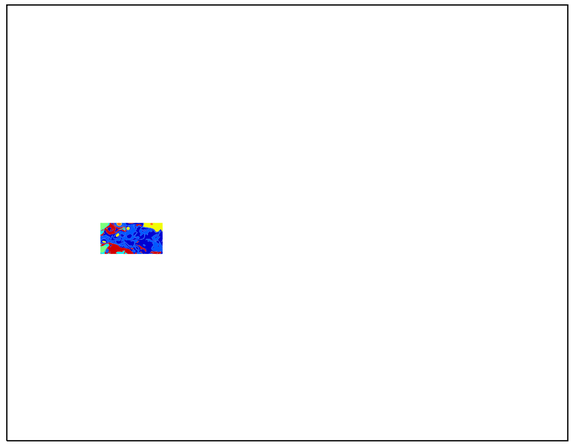
26 Millions of profiles (ISAS13)
34Gb, 11 years of mm

faster !



200M profiles (NATL12)
290.4 GB, 20 years of mm

issues !



3500M profiles (NATL60)
2.5To, 1 year of dm

Spark/datamining

Example: Profile Classification Modelling

here: 15 years, N.Atl.: 0.1×10^6 profiles

x10

All Argo: 15 years, global: 1.5×10^6

x10

ORA-S4: 50 years, monthly, global 1/1 gridded: 26×10^6

ISAS13+nrt: 13 years, monthly, global 1/2 gridded: 43×10^6

x10

HadGEM: 140 years, monthly, global 1/1 gridded: 92×10^6

x10

ORCA025: 40 years, weekly, global 1/4 gridded: $1\,400 \times 10^6$

CMIP5: 50 years, monthly, global 1/1 gridded, 50 runs: $1\,500 \times 10^6$

DRAKKAR12: 20 years, weekly, global 1/12 gridded: $6\,400 \times 10^6$

x10

OCCIPUT: 50 years, weekly, global 1/4 gridded, 50 runs: $8\,900 \times 10^6$

Our goal is not reached yet, but the data mining workflow is working !
Serious challenge in accessing large datasets:
we need a database with the appropriate primary index key (here: depth)
We need a persistant storage with optimised access to data samples

From a general 5-dimensional dataset, we need fast access to any possible set of sampling/feature dimensions.
Example: 5-d temperature: [X,Y,Z,T,M]
> Sampling: [X,Y,T,M] (to be stacked)
> Features: Z
Most of the time, the dataset is: TxM files, with [X,Y] features and Z sampling