OTB Users Days 2016

Processing your data with 500+ cores: yes wen can!

OTB Team

07/06/2016

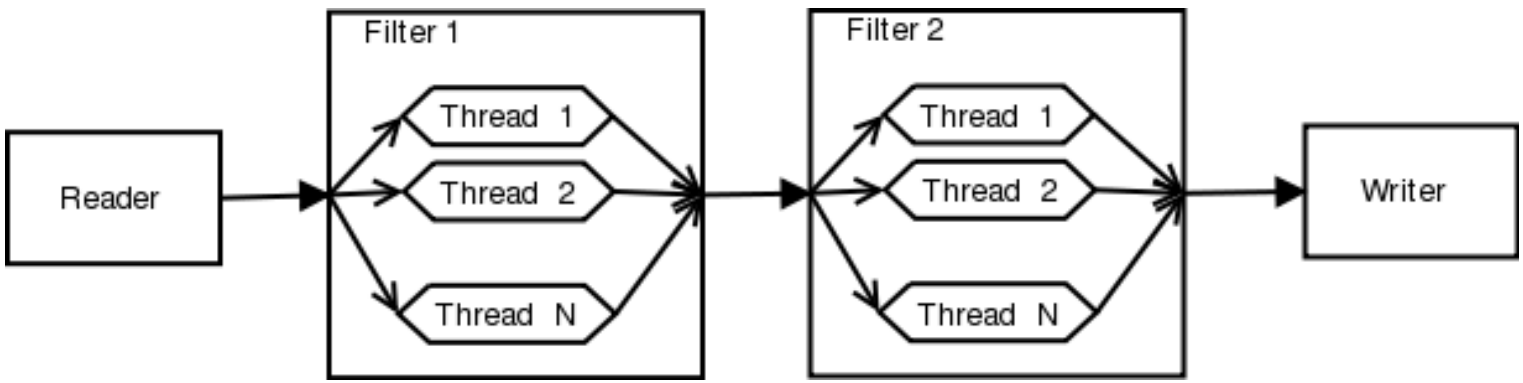
What is RFC-26

- Enable MPI-based parallel writing in OTB, for cluster with GPFS infrastructures
- Follow up of Remi Cresson talk at OTB Users Days 2015
- Additional work done by Emmanuelle Sarrazin on CNES cluster
- Everything integrated in a feature branch + seamless app mechanism

Why MPI ?

Actual OTB parallelism

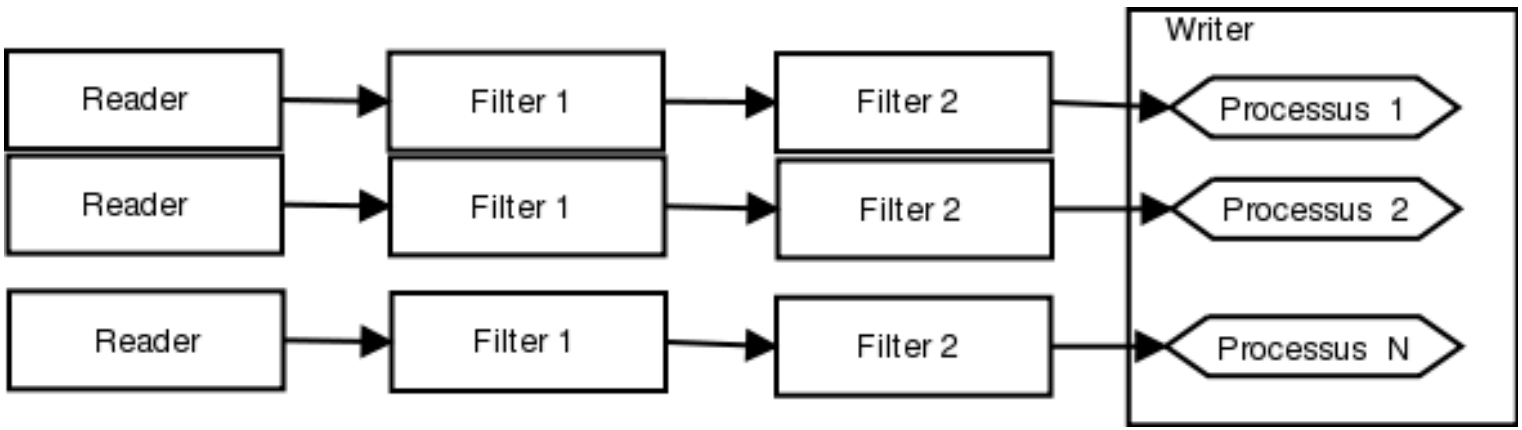
- Based on multithreading provided by ITK
- Limitations :
  - not optimized (thread creation and destruction in each filter),
  - limited scalability
  - limited to one compute node



Why MPI ?

New OTB MPI parallelism

- Use several compute nodes
- More scalability
- Streaming and multithreading can still be used in one MPI process



An example on CNES cluster

```
$ mpirun -np $nb_procs --hostfile $PBS_NODEFILE  \
    otbcli_BundleToPerfectSensor \
    -inp $ROOT/IMG_PHR1A_P_001/IMG_PHR1A_P_201605260427149_ORT_1792732101-001_R1C1.JP2 \
    -inxs $ROOT/IMG_PHR1A_MS_002/IMG_PHR1A_MS_201605260427149_ORT_1792732101-002_R1C1.JP2 \
    -out $ROOT/pxs.tif uint16 -ram 1024

------------ JOB INFO 1043196.tu-adm01 -------------

JOBID           : 1043196.tu-adm01
USER            : michelj
GROUP           : ctsiap
JOB NAME        : OTB_mpi
SESSION         : 631249
RES REQSTED     : mem=1575000mb,ncpus=560,place=free,walltime=04:00:00
RES USED        : cpupercent=1553,cput=00:56:12,mem=4784872kb,ncpus=560,vmem=18558416kb,
                  walltime=00:04:35
BILLING         : 42:46:40 (ncpus x walltime)
QUEUE           : t72h
ACCOUNT         : null
JOB EXIT CODE   : 0

------------ END JOB INFO 1043196.tu-adm01 ---------
```

- walltime=00 :04 :35
- ncpus=560

New module : ThirdParty/MPI

- This module is in charge of importing OpenMPI
- Can be turned ON/OFF with $\text{OTB}_{\text{USEMPI}}$ macro
- No dependency

New module : ThirdParty/SPTW

- This module contains sources of Simple Parallel Tiff Writer
- Small program from github allowing for parallel writing of tiff files
- Licence is "Public Domain"
- No packages anywhere ⇒ we embed the sources

New module : MPI/MPIConfig

- This module contains the otb : :MPIConfig class
- Singleton class in charge of :
  - Intializing MPI from argc/argv
  - Finalizing MPI upon destruction (at the end of the program)
  - Returning mpi rank and number of process
  - Logging and error handling
- Completely hides MPI interface (with some limitations)
- Used in both MPIVrtwriter and MPITifWriter

New module : MPI/MPIVrtWriter

How it works

- This module contains a single template function :
  - Use an ImageSplitter to divide image to write into tiles according to MPI nb processes
  - Use ExtractROI + ImageFileWriter to write each tile to a separate tiff file in parallel
  - Generate a VRT file that stitches all tiles into a single dataset (done by process 0)

Pros and cons :

- 40 lines of code and you get parallel writing
- No dependency to SPTW module (and to SPTW)
- Less barrier than the MPITiffWriter : a bit faster
- You do not a get a single file, but multiple tiffs and a vrt

New module : MPI/MPITiffWriter

How it works

This module contains a fork of ImageFileWriter that uses SPTW

Pros and cons :

- Dependency to SPTW
- Code more complex, with code duplication from ImageFileWriter
- Writes a single tiff file, which is really what you want
- Better support of metadata (from duplication of code writing it)

Seamless use in applications

Integration

- When OTB is built with MPI :
- otbApplicationLauncherCommandLine calls MPI init with argc/argv
- If app is called without mpirun, or nbproc = 1 :
  - Use the standard ImageFileWriter class
- Else
  - If file extension is ".tif" and STPW is built : use MPITiffWriter
  - ElseIf file extension is ".vrt" : use MPIVrtWriter
  - Else : report file format not supported error

Benefit

You get both parallel writing methods in all applications, without changing a single line in their code

Small and big things TODO

Small things (for this RFC)

- Enhance code comments and documentation
- Ensure OTB coding style is applied
- Report as many metadata as possible in VRT writer

Bigger things (later)

- Persistent filters and virtual writers (same strategy)
- Handle more complex cases (segmentation codes)