Consolidation and Performance measurements of ROOT Multiproc Core

Student: Anda-Catalina Chelba
Supervisor: Gerardo GANIS (EP-SFT)

August 29, 2016

Data Processing in Root

Serial || Parallel

Aug 29, 2016

Anda-Catalina Chelba

Parallel Data Processing in Root

Aug 29, 2016

Anda-Catalina Chelba

Parallel Data Processing in Root

Proof	Proof-Lite	MultiProc

Parallel Data Processing in Root

Proof	Proof-Lite	MultiProc
Multi-node parallelism		
Drawbacks • environment settings • limited scope		

Parallel Data Processing in Root

Proof

Multi-node parallelism

Drawbacks

- environment settings
- limited scope

Proof-Lite

A re-adaptation of PROOF

1 machine, multiple processes

Successful

Drawback

- inherits the setup technology from PROOF

MultiProc

Parallel Data Processing in Root

Proof

Multi-node parallelism

Drawbacks

- environment settings
- limited scope

Proof-Lite

A re-adaptation of PROOF

1 machine, multiple processes

Successful

Drawback

- inherits the setup technology from PROOF

MultiProc

Introduced last year

Goal :
- Fix the problems listed above
- More powerful & user-friendly

1 machine, multiple processes

Tree processing is a vital functionality, but was never benched => existent bugs, not efficient

MultiProc module : My mission

Consolidate the existent version

- Complete the tree processing interface & bug fix

Develop a Bench Tool

Work on a new version of packetizing

The bench tool

Different technologies can be tested :

Serial (Tchain)

ProofLite

MultiProc

Multi-Thread

The bench tool

Different technologies can be tested :

Serial (Tchain)

MultiProc

ProofLite

Multi-Thread

Features :

- Nb workers

The bench tool

Different technologies can be tested :

Serial (Tchain)

MultiProc

ProofLite

Multi-Thread

Features :

- Nb workers
- Local/remote root files

The bench tool

Different technologies can be tested :

Serial (Tchain)

MultiProc

ProofLite

Multi-Thread

Features :

- Nb workers
- Local/remote root files
- With/without cache

The bench tool

Different technologies can be tested :

Serial (Tchain)

MultiProc

ProofLite

Multi-Thread

Features :

- Nb workers
- Local/remote root files
- With/without cache
- Processing function to be used

The bench tool

Different technologies can be tested :

Serial (Tchain)

MultiProc

ProofLite

Multi-Thread

Features :

- Nb workers
- Local/remote root files
- With/without cache
- Processing function to be used
- Path to the output file

The bench tool

Different technologies can be tested :

Serial (Tchain)

MultiProc

ProofLite

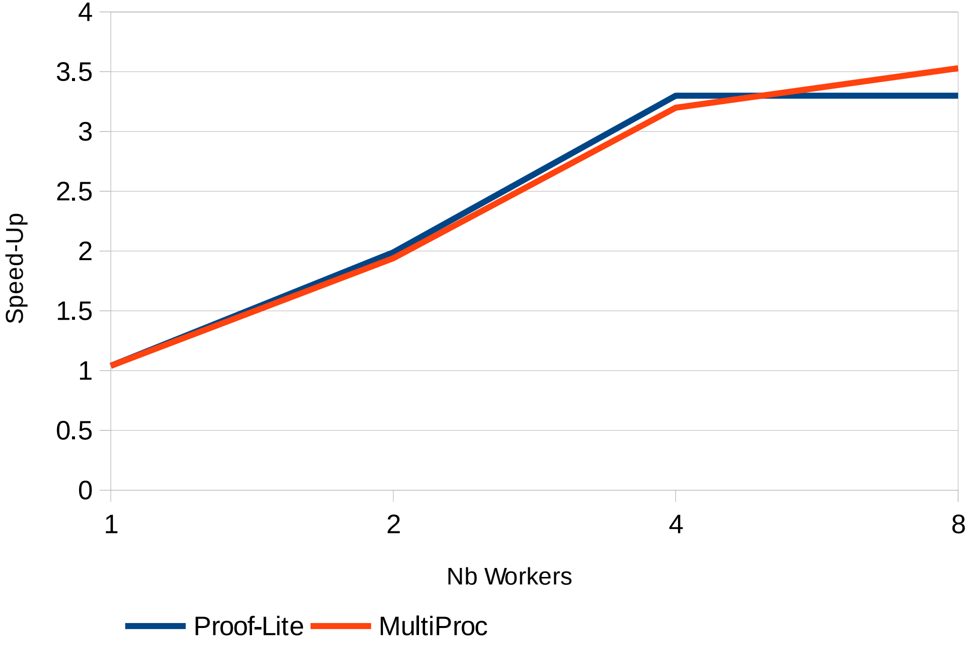
Multi-Thread

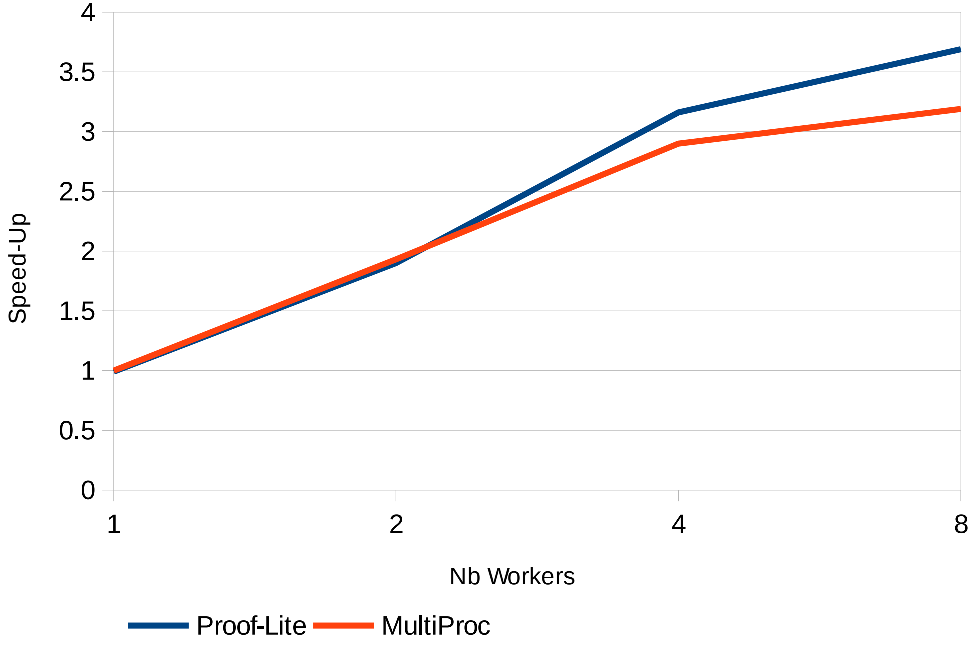
Features :

- Nb workers
- Local/remote root files
- With/without cache
- Processing function to be used
- Path to the output file
- Visualization tools

The first results

Test conditions
* 24 core Ubuntu machine
* local harddisk
* no ROOT tree cache
* no file system cache


4 files processed


10 files processed

MultiProc module V1

- How it works ?
  - 2 workers, 3 files (same size)

MultiProc module V1

- How it works ?
  - 2 workers, 3 files (same size)
    - Step 1:
      - W1 gets 1 file
      - W2 gets 1 files

MultiProc module V1

- How it works ?
  - 2 workers, 3 files (same size)
    - Step 1:
      - W1 gets 1 file
      - W2 gets 1 files
    - Step 2:
      - W1 gets 1 file
      - W2 idle

MultiProc module V1

- How it works ?
  - 2 workers, 3 files (same size)
    - Step 1:
      - W1 gets 1 file
      - W2 gets 1 files
    - Step 2:
      - W1 gets 1 file
      - W2 idle
  - Problem
    - Work is not split equally

MultiProc module V1

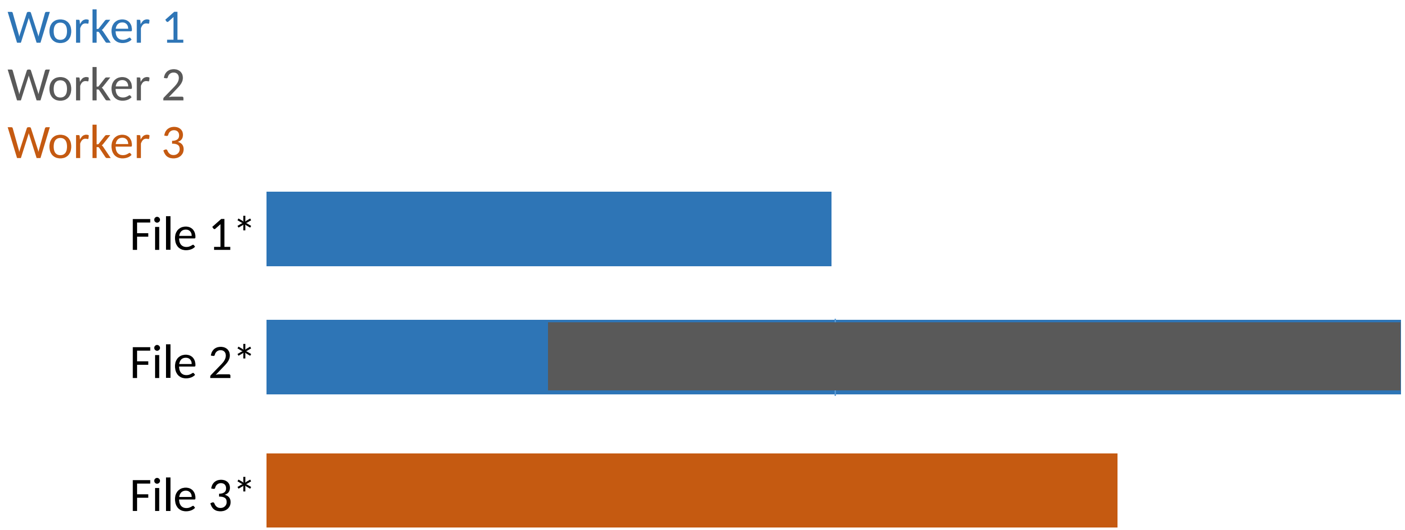
- How it works ?
  - 2 workers, 3 files (same size)
    - Step 1:
      - W1 gets 1 file
      - W2 gets 1 files
    - Step 2:
      - W1 gets 1 file
      - W2 idle
  - Problem
    - Work is not split equally

- Current packetizing technology is not efficient

MultiProc : packetisingV2

Objective: split the work equally

MultiProc : packetisingV2



* Most of the time, files have the same size

MultiProc : packetisingV2

The obvious solution

MultiProc : packetisingV2

What do we need?

- Nb of entries to process

How do we get it?

- Need to open each file

Problems?

- Expensive : time

MultiProc : packetisingV2

The solution

MultiProc : packetisingV2

What we can easily have?

- File size

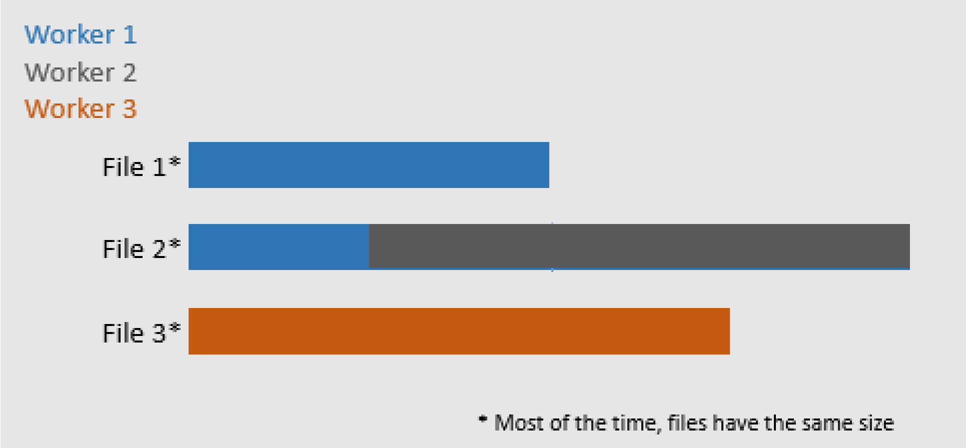
MultiProc : packetisingV2

What we can easily have?

- File size

How to use it?

- File 1 : 1 GB
- File 2 : 2 GB
- File 3 : 1,5 GB
- Each worker needs to process 33% of total work = 1,5GB



* Most of the time, files have the same size

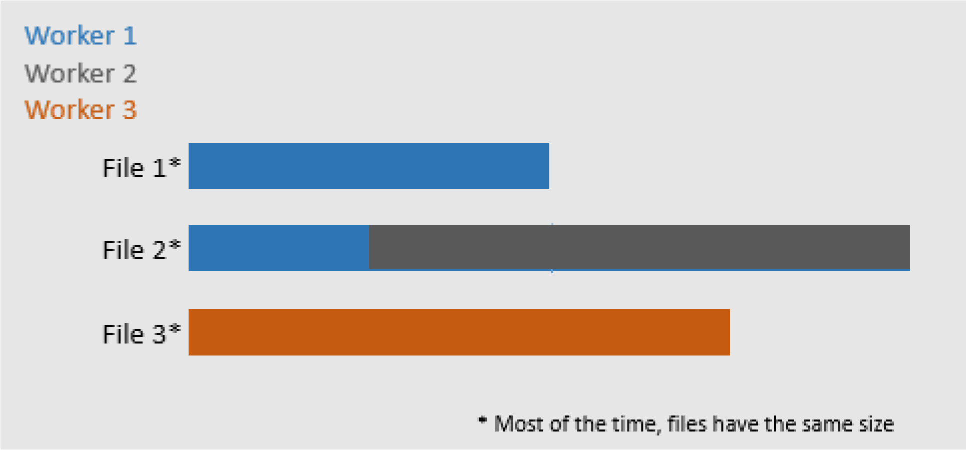
MultiProc : packetisingV2

What we can easily have?

- File size

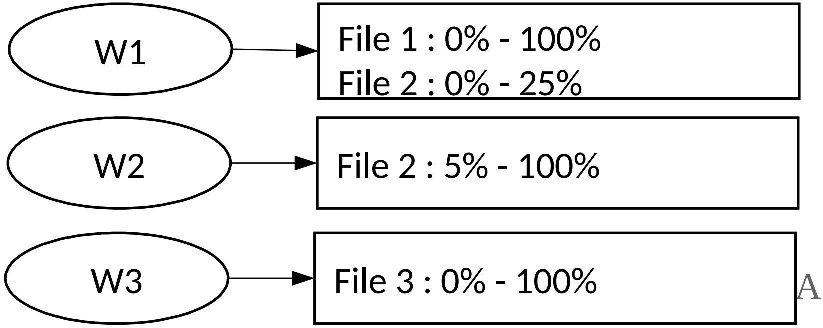
How to use it?

- File 1 : 1 GB
- File 2 : 2 GB
- File 3 : 1,5 GB



* Most of the time, files have the same size

- Each worker needs to process 33% of total work = 1,5GB


W1 → File 1 : 0% - 100%
File 2 : 0% - 25%

W2 → File 2 : 5% - 100%

W3 → File 3 : 0% - 100%

Next steps

Finish the Bench Tool development

Finish the development of the new packetizing technology

Benchmark MultiProc & Multi-Thread technologies

Thank you.