Status and Evolution of Multi-Processing in ROOT

G Ganis

CERN, PH/SFT

17 September 2015
ROOT 2015, Saas-Fee, Switzerland

Content

- PROOF
  - Status, Perspectives
- Beyond PROOF
  - Multiproc
- Summary

PROOF

- Coordination of multiple ROOT sessions to speed-up embarrassingly parallel task
- Master/Workers model, pull architecture

- Designed for TTrees
  - Work scheduling is TTree-driven
- Main goal: increase effective I/O

PROOF usage

- Systems built around of / using it
  - VAF, Virtual Analysis Facility (see CHEP 2013)
  - PAF (-> next talk)
- LHC
  - ALICE: CAF moved to VCAF, analysis framework supporting PROOF
  - ATLAS: PROOF-Lite, PoD on LSF/lxbatch, PanDA
  - CMS: Spontaneous users (PROOF-Lite, local clusters)
- Italian National Project (PRIN)
  - LHC data analysis on clouds using PROOF/PoD
  - ALICE / ATLAS / CMS
- Overall PROOF appears in 7 contributions to CHEP2015

PROOF since ROOT 2013

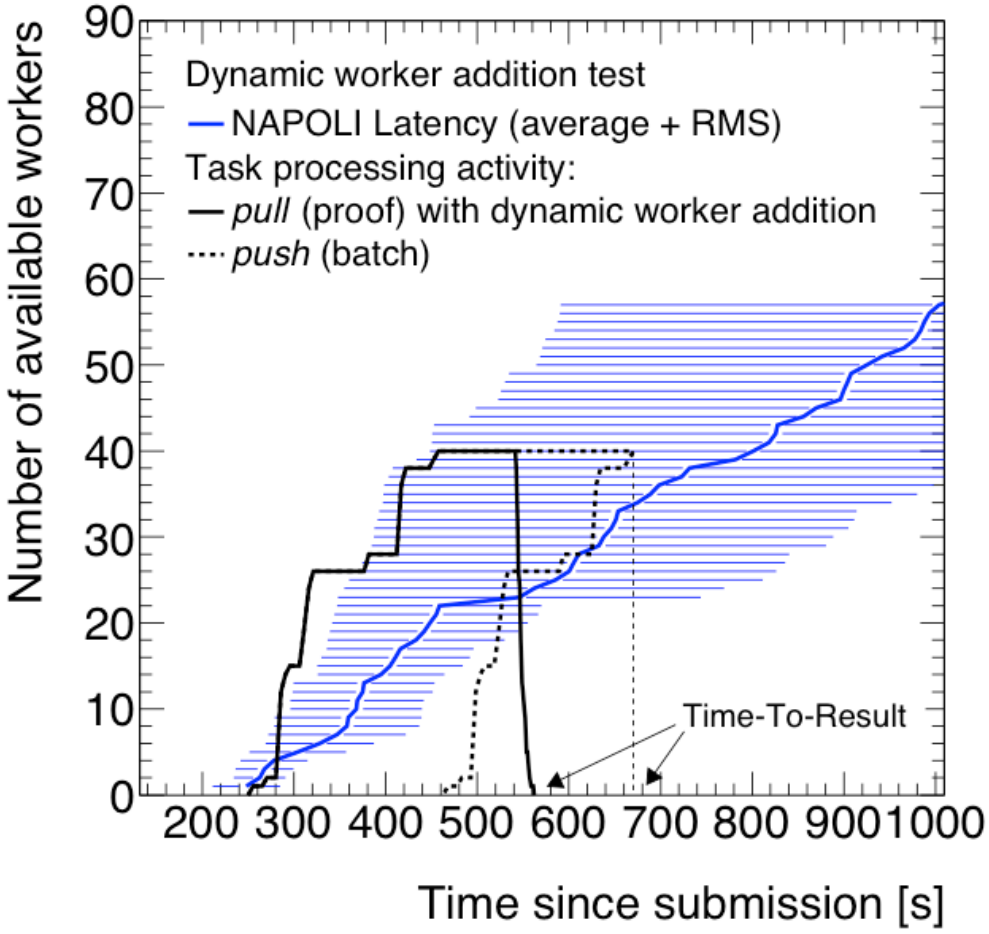
- Consolidation
- Established PoD as recommended RMS
  - Relevant code on CernVM-FS
- Virtual Analysis Facility based on CernVM
- Dynamic addition of workers
  - Includes adaptation of main packetizers
- Merging optimization
  - Memory footprint, improved lookups
- Interface with igProf
- Prototype of ROOT/PROOF Draw unification

Dynamic addition of workers

- Exploit cloud-awareness of the PROOF model
  - Adapts to heterogeneous resources
  - Runs on elastic clouds
- Flexible workflow
  - Start the query as soon as submission
  - Workers added as available dynamically adapting to the size of pool
- Full implementation since v5.34/27 / v6.04
- Included in an ATLAS contribution at CHEP 2015 (next slide)

PROOF@ATLAS, CHEP 2015

- PROOF w/ PoD/PanDA
- Measured average worker ramp-up rate for the site (blue line, hashed area)
- 40 workers asked, PROOF launched right away
- Active workers during query time (full black line) w/ dynamic worker addition
- Estimated execution time inn push (batch) mode (dashed black line)
- Time-to-result reduced by 20% in this example



Di Nardo et al., "PROOF-based analysis on ATLAS Italians Tiers with ProdSys2 and Rucio", CHEP 2015, #179

PROOF: result of survey

Used by 121/364 of survey participants	33%
PROOF-Lite	32%
Local clusters	50%
PoD	18%
Start from TChain	66%
Put all code in the selector file	47%
Use improved output handling	80%
Write own Merge	12%
Use for non-tree driven tasks	13%
Features less used (or known)	
Processing by-object (10%), feedback (5%), log viewer (10%)	

PROOF plans

- "Frozen" functionality
  - But remove gap with plain TTree::Process
- Consolidation
  - Remove duplications, unused parts
  - Full move to XRootD-4
- Documentation
  - Dedicated primer

Beyond PROOF

What works

- Exploitation of multi-cores w/ PROOF-Lite
- "Automatic" merging is nice
  - Single file with the results
  - Multiple level of merging, submasters
- *Interactive* exploitation of batch resources w/ PoD allow to fully use the assigned slot
  - Use and reuse several times
- Pull model may effectively reduce time-to-result
- Easy-to-setup tools (PROOF-Lite, PoD)

What works less

- Requires modification of the code
  - From BIG macro to TSelector
    - Requires a class (TSelector) even for very simple things
- TSelector requires hacks to work on PROOF
  - Steered by different internal code
- Lack of automatic environment consistency
  - Loaded libs, envs settings, …
- Stability issues
  - Non-PoD deployments not very stable
  - PROOF-blamed even when not faulty
  - Difficult Error Handling/Recovery
- Lack of practical examples

Future of Multi Processing

- Needs more flexibility
  - Non TTree driven
    - Generic task list
  - Support for generic interface, macros, lambdas, …
- Transparent environment setting
- Minimal - and extendible – protocol

MP: three cases

- Multi-core machines
- Local clusters
  - Group or department farm
  - Sharing file system
- Geographically distributed resources
  - Clouds

MP: three cases

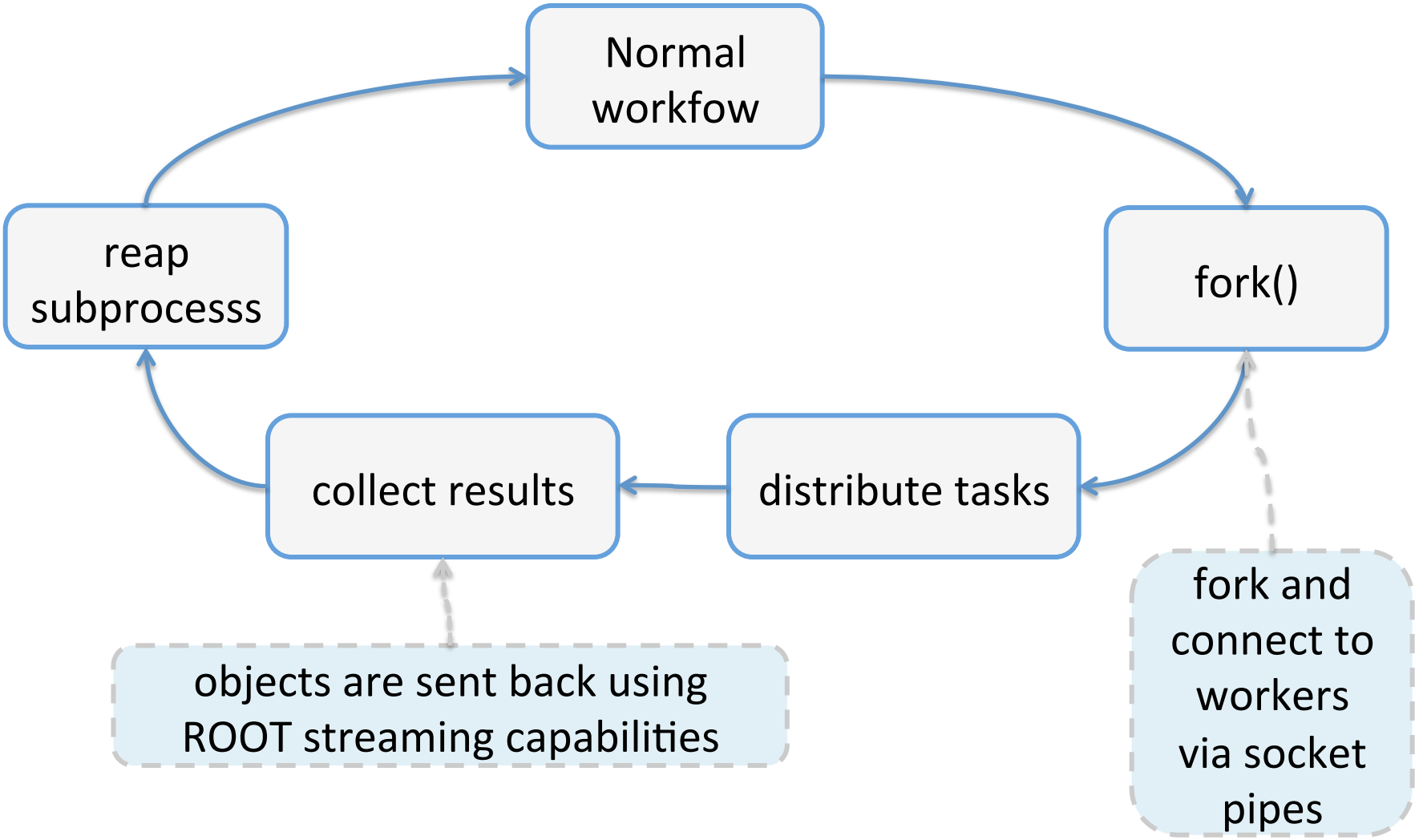
- Multi-core machines
- Local clusters
  - Group or department farm
  - Sharing file system
- Geographically distributed resources
  - Clouds

Rest of the talk

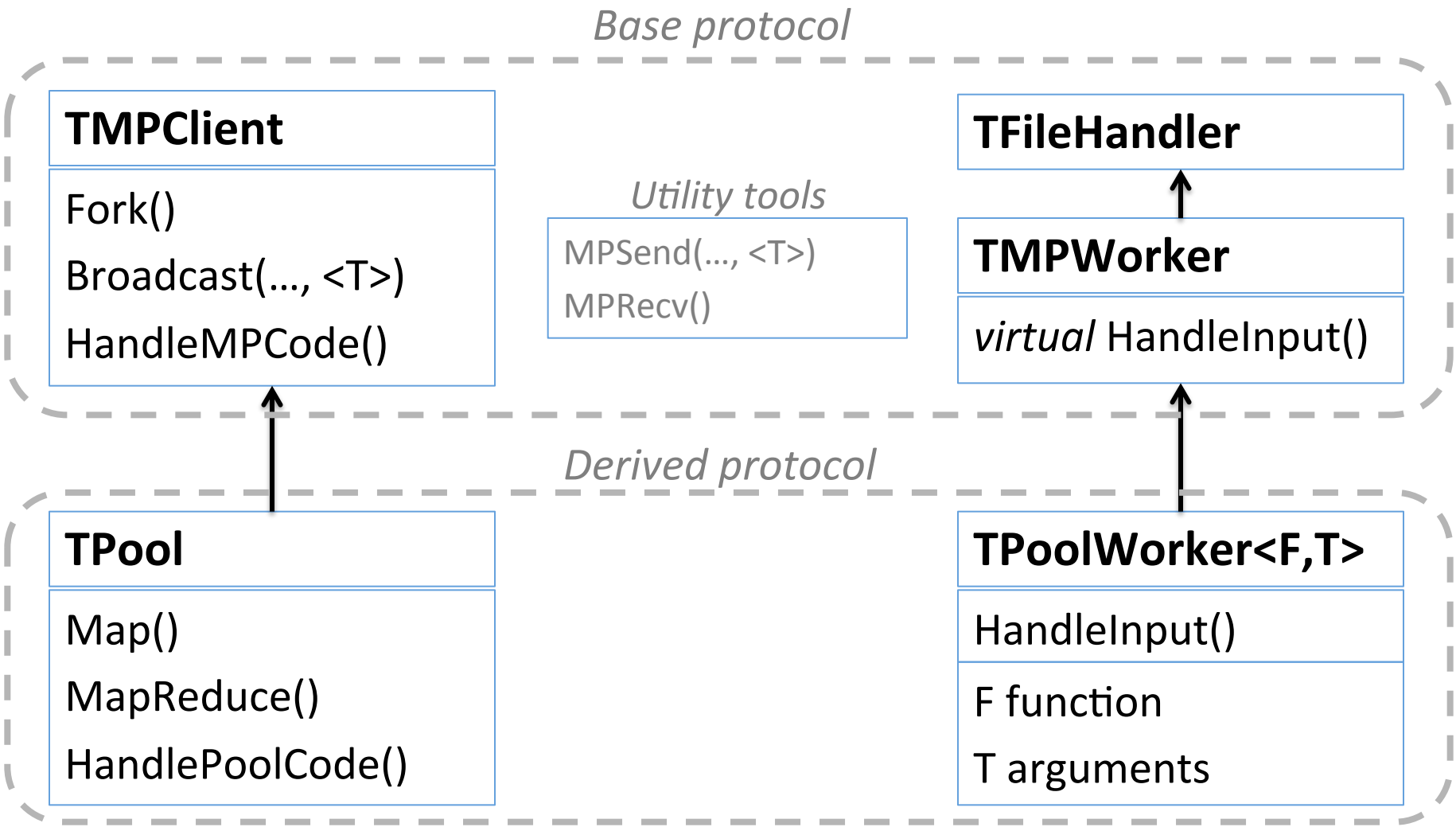
Multi-core

- Idea: use copy-on-write techniques to address environment setup issues in PROOF-Lite workers
- Based on python multiprocessor module
  - Adding reduce, process, …
- E.Guiraud's summer student project
- First version already in v6.05/02
  - Module 'core/multiproc'

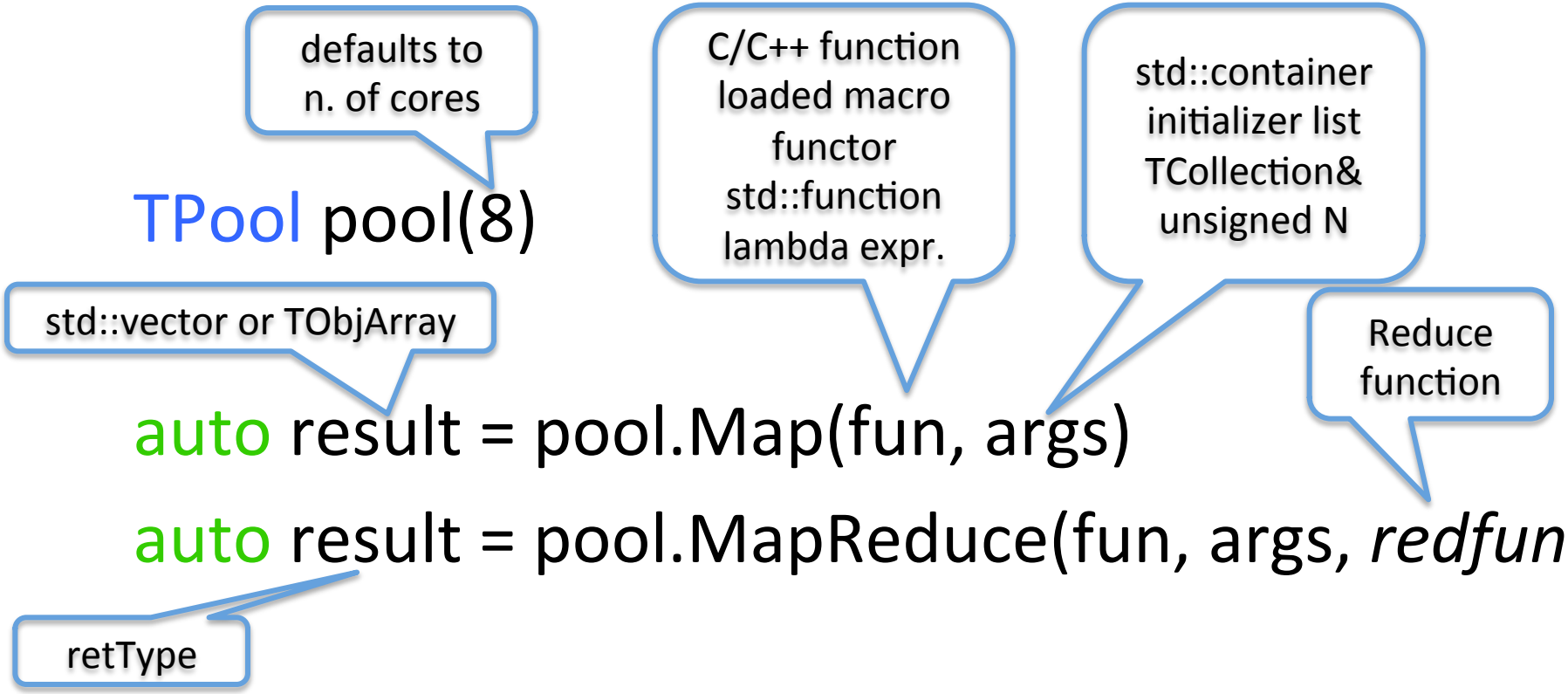
multiproc approach



Implementation



TPool::Map, TPool::MapReduce




```
TPool pool(8)
auto result = pool.Map(fun, args)
auto result = pool.MapReduce(fun, args, redfun)
```

Reduce function: <retType> *redfun*(std::vector<retType>)

Map, MapReduce example

V6.05/02

```
TPool pool()
// define TObject *myMacro(string,int,string)
.L myMacro.cxx
auto res = pool.Map([](string f){
                    return myMacro("opt",12, f);},
                    {"file1", "file2", "file3"})

auto res = pool.MapReduce([](string f){
                    return myMacro("opt", 12, f);},
                    {"file1", "file2", "file3"},
                    PoolUtils::ReduceObjects)
```

Default reducer for TObject's:
namespace PoolUtils { TObject* ReduceObjects(const std::vector<TObject *>& objs); }

TPool::Process

WORK IN PROGRESS

- Dedicated to TTree processing
- Exploit TTreeReader goodnesses
- Flexible range of ‘macros’, functions, …
- Support for TSelector

TPool::Process

fun retType

std::vector
TFileCollection
TChain

C/C++ function
loaded macro
functor, …
TSelector

```
auto result = pool.Process(dset, fun, args)
```

Example w/ generic function

```
<retType> fun(TTreeReader r) {
   // Attached to required leaves
   TTreeReaderValue<Float_t> p0(t, "p0");
   // Loop over the range
   while (r.Next()) {
      // Do something
   }
   return ret;
}
```

Work In Progress

(Final interface can be different!)

TPool AoB

- More TPool::Map signatures available
  - Including possibility to call *fun* N times
- Supports greedy workers implemented
  - Not yet in the repository
- Merging (reduce) is currently serial
  - Plans to implement
    - Parallel (submerger) technology (for histos …)
    - Collector (ParallelMerge) technology (for trees …)
- Some portability issues
  - MacOsX does not like fork()
  - Windows does not have fork()

MP on clusters: ideas

- *Local* (well connected) clusters
  - multiproc interface could be 'kept' if
    - Efficient sharing of working directory
    - Environment setup à la distributed shell
      - Execute commands on all processes concurrently
- Distributed clusters
  - Protocol to loosely couple ROOT sessions
  - Modular functionality to build on top
    - Merger, elastic scheduler, …

Distributed cluster with DDS

- Dynamic Deployment System
  - GSI development built on PoD experience
    - In the context of ALFA (see M. Al-Turany talk)
  - Handles set of processes, exchanging info via key-values pairs
- PROOF on DDS as a first step
  - Requires implementation of DDS message exchanging technology in ROOT
  - Removes need for a connection layer, daemon, …
  - Allows to run the master on the cluster
- Could be the base for a more flexible interconnection protocol

Summary

- PROOF maintenance mode
  - Base for well advanced analysis facility toolkits
- Multi-process framework available for multicores
  - Provides flexible and powerful interface exploiting copy-on-write and C++11 features
- Evaluating / prototyping ideas for the future ROOT MP across machines
- DDS promising 'replacement' of PoD