A Parallelised ROOT for Future HEP Data Processing

D. Piparo (CERN, EP-SFT) for the ROOT team

ROOT
Data Analysis Framework
https://root.cern
Challenges ahead of us
ROOT’s approach to parallelism
Parallel components
The future

- Distributed analysis
- Heterogeneous platforms

Unless explicitly stated, everything available in ROOT 6.14!
HL-LHC: Challenge for data processing and analysis SW
- In both areas ROOT is a key component

Parallelism: not the solution, a prerequisite

Find and create opportunities for parallelism in ROOT
- Replace components for which evolution is not possible
- Provide programming model which makes scientists productive - cannot require too broad technical skill set from neophytes
ROOT’s Approach to Parallelism
Implicit parallelism: operations run in parallel w/o user’s intervention

- Just invoke ROOT::EnableImplicitMT()
- Task based backed by multithreading, TBB library in the backend
  - Must not overcommit node, can share pool with other libraries
- Data parallelism: 1st class citizen (VecOps) with vectorisation support

Explicit parallelism: user expresses parallelism, ROOT provides low level tools to do that efficiently

- Map, MapReduce helpers (T{Process,Thread}Executor)
- Forking based multiprocessing backend in ROOT
- Mutexes, async launcher
Get a dataset in a file
  - Columns: px, py, E (collections)
Fill a histogram with square sum of px and py for entries where E > 100

Imperative way, explicit double loop
Ergonomic Interfaces

Declarative, type safe, jit to simplify, task parallelism, vectorised operations on collections

ROOT::EnableImplicitMT();
RDataFrame f(treename, filename);
f.Define("good_pt", "sqrt(px*px + py*py)[E>100]")
  .Histo1D({"pt", "pt", 16, −.5, 3.5}, "good_pt")−>Draw();

See E. Guiraud RDFrame: Easy Parallel ROOT Analysis at 100 Threads, Track 6, Hall 9, 10th July 11:45

D. Piparo - Parallelised ROOT for Future HEP Data Processing - CHEP18
Parallelised Components
What is Implicitly Parallelised?

ROOT::EnableImplicitMT() activates parallelised:

- **RDataFrame** event-loop
- **TTree::GetEntry** read of multiple branches
- **TTree::FlushBaskets** write of baskets
- **TTreeCacheUnzip** decompression of TTreeCache content
- **TH{1,2,3}::Fit** evaluation of the objective function over the data
- **TMVA::DNN** trains NN in parallel

And more to come!
### IMT Effect On CMS Data Processing

#### 32 thread RECO-AOD-MINIAOD

<table>
<thead>
<tr>
<th>Module</th>
<th>Total Loop Time</th>
<th>Total Loop CPU</th>
<th>CPU Utilization</th>
<th>Events/Second</th>
<th>RSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard w/o IMT</td>
<td>1</td>
<td>52</td>
<td>32</td>
<td>2.94</td>
<td>9454</td>
</tr>
<tr>
<td>Standard w/IMT</td>
<td>1</td>
<td>34</td>
<td>4.21</td>
<td></td>
<td>8981</td>
</tr>
<tr>
<td>Parallel 6x6x3</td>
<td></td>
<td></td>
<td></td>
<td>4.47</td>
<td>13817</td>
</tr>
<tr>
<td>Parallel 1x6x3</td>
<td></td>
<td></td>
<td></td>
<td>4.59</td>
<td>10745</td>
</tr>
<tr>
<td>NoWrite</td>
<td></td>
<td></td>
<td></td>
<td>4.65</td>
<td>12140</td>
</tr>
<tr>
<td>NoFill</td>
<td></td>
<td></td>
<td></td>
<td>5.41</td>
<td>7201</td>
</tr>
</tbody>
</table>

- **Activate Implicit MT:**
  - +43% Throughput

- **Add Parallel Writing:**
  - +13% Throughput

18% away from asymptotic value (not filling nor writing output datasets)

---

D. S. Riley, **CMS And ROOT IO**, ROOT IO Workshop, 20 June 2018, CERN
See G. Amadio **Writing ROOT Data in Parallel with TBufferMerger** Track 5, Hall 3, 10th July 12:15

D. Piparo - Parallelised ROOT for Future HEP Data Processing - CHEP18
No IO, KNL, 64 Physical cores

Monte Carlo QCD low-pt events generation+analysis on the fly

Ad-hoc implementation (patched ROOT5 & POSIX threads) Vs RDataFrame

Original code by experienced developer (R. Brun), intentionally not thread safe (*RDF always is*)

**ROOT Can Scale Well on Extreme Architectures**
BDT’s: implicit parallelism

- Specific operations in tree construction process
- x 1.6 speedup for 4 threads

Cross Validation: multiprocessing based

- Evaluate each fold independently
- Almost linear scaling!

And of course, Cuda based implementation of DNNs in TMVA
Evaluation of objective functions is parallelised and vectorised

- Adapt TF1, TFormula, fitting internal classes
- Leverage ROOT::Double_v SIMD type - based on VecCore
- AVX2, 2x2 cores: factor 10x not uncommon!

Introduced \texttt{ROOT::RVec<T>}: vectorised operations made easy

- \texttt{std::vector} like interface, ergonomic support of analysis operations
- Can adopt memory or own it
- Vectorised arithmetic operations, math functions

See also L. Moneta \textit{Vectorisation of ROOT Mathematical Libraries}, Track 5, Hall 3, 9th July 15:45
RVec<double> mus_pt {15., 12., 10.6, 2.3, 4., 3.};
RVec<double> mus_eta {1.2, -0.2, 4.2, -5.3, 0.4, -2.};
RVec<double> good_mus_pt = mus_pt[mus_pt > 10 && abs(mus_eta) < 2.1];

RVec<float> vals = {2.f, 5.5f, -2.f};
RVec<float> sin_vals = sin(vals);

ROOT::EnableImplicitMT();
RDataFrame f(treename, filename);
f.Define("good_pt", "sqrt(px*px + py*py)[E>100]"
  .Histo1D({"pt", "pt", 16, -.5, 3.5}, "good_pt") -> Draw();
The Future
Distributed Analysis

Investigate and prototype a complement to PROOF

- Parallelism on many nodes
- Transparent distribution
- Support several different backends

```
d = RDataFrame("t", dataset)
f = d.Define(...) .Define(...) .Filter(...)
h1 = f.Histo1D(...) h2 = f.Histo1D(...) h3 = f.Histo1D(...)```

Minimal/No change in analysis code

Not in 6.14 Working prototype available!
Support for Heterogeneity

Key element of future HEP software and computing

- TMVA already takes advantage of CUDA (DNNs)

Work ongoing to access NVidia devices from ROOT’s interpreter:

- Allow to interpret CUDA code
  - \texttt{gKernel1\lll1,1\rrr}(deviceOutput1);
  - \textbf{More than plans:} pieces already in \texttt{ROOT master branch}!
- Supports templates, runtime shared memory

Thanks to \textbf{Simeon Ehrig} for diving into the Cling-CUDA integration work!
ROOT: getting ready for the HL-LHC, also through parallelisation

- Emphasis on programming model, runtime and scaling

Substantial parallelism delivered in ROOT 6.14

- Scaling at the level of ad-hoc solutions written by experts
- Boost CMS amount of evts/s processed
- Parallelism in TMVA and fitting: factors can be achieved

Many opportunities ahead of us

- Provide a distributed system to further scale
- Embrace heterogeneity
- Drive renovation of ROOT with natively parallel components only