CMS RNTuple Workshop Performance Info

Dr Christopher Jones
CCE SOP

20 11 2024

RNTuple Status in CMS

- We have prototype code which can read/write RNTuple files

- Can write any of our standard storage formats
  - RECO, AOD, and MiniAOD

- Only possible because of addition of streamer storage mechanism
  - allows storage of any class that TTree could handle
    - storage of bare pointers
    - storage of polymorphic types
    - storage of classes which directly/indirectly store themselves

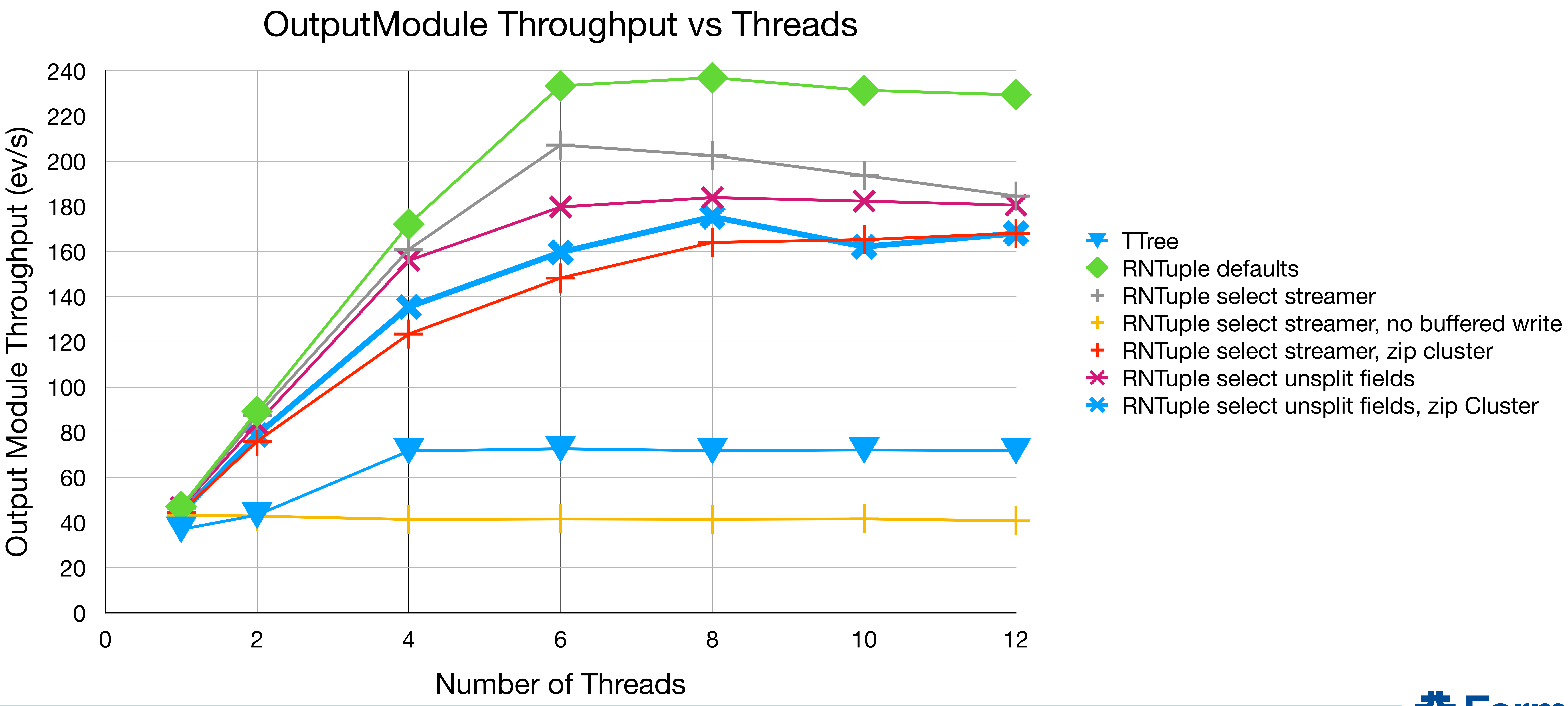
MiniAOD Storage Comparison

- Storage Type
  - **select streamer**: apply streamer storage to data products if it reduces storage size
  - **select unsplit**: do not split floating point fields if it reduces storage size
- Optimization
  - **zip cluster**: `ApproxZippedClusterSize` set to 25M down from 50M
  - **no buf write**: turn off `BufferedWrite`

Storage Type	Optimization	File Size Ratio	Max Allocation	Allocation Diff	Max RSS	RSS Difference
TTree		1.000	2,329,950,216	0	2,640,244,736	0
RNTuple default		**0.981**	2,975,349,944	**645,399,728**	3,188,834,959	**548,590,223**
RNTuple select streamer		**0.937**	3,001,023,440	**671,073,224**	3,285,766,144	**645,521,408**
	zip cluster	**0.933**	2,486,353,160	**156,402,944**	2,807,153,295	**166,908,559**
	no buf write	**0.951**	1,846,704,808	**-483,245,408**	2,240,806,912	**-399,437,824**
RNTuple select unsplit		**0.924**	3,036,038,160	**706,087,944**	3,308,414,566	**668,169,830**
	zip cluster	**0.918**	2,503,737,024	**173,786,808**	2,750,152,704	**109,907,968**
	no buf write	**0.939**	1,853,864,576	**-476,085,640**	2,143,961,108	**-496,283,628**

Thread Scaling for Writing

- No scaling without BufferedWrite


OutputModule Throughput vs Threads
Output Module Throughput (ev/s)
Number of Threads
TTree
RNTuple defaults
RNTuple select streamer
RNTuple select streamer, no buffered write
RNTuple select streamer, zip cluster
RNTuple select unsplit fields
RNTuple select unsplit fields, zip Cluster