

Optimizing I/O Performance for ESD

Analysis

Misha Zynovyev, GSI (Darmstadt)



Preview

- Research subjects and goals
- General conditions
- Logical layout of ALICE ESD tree
- Physical layout of ALICE ESD file
- Evaluation of disk access performance
- Ways to optimize disk access performance
- Summary

Research Subjects and Goals

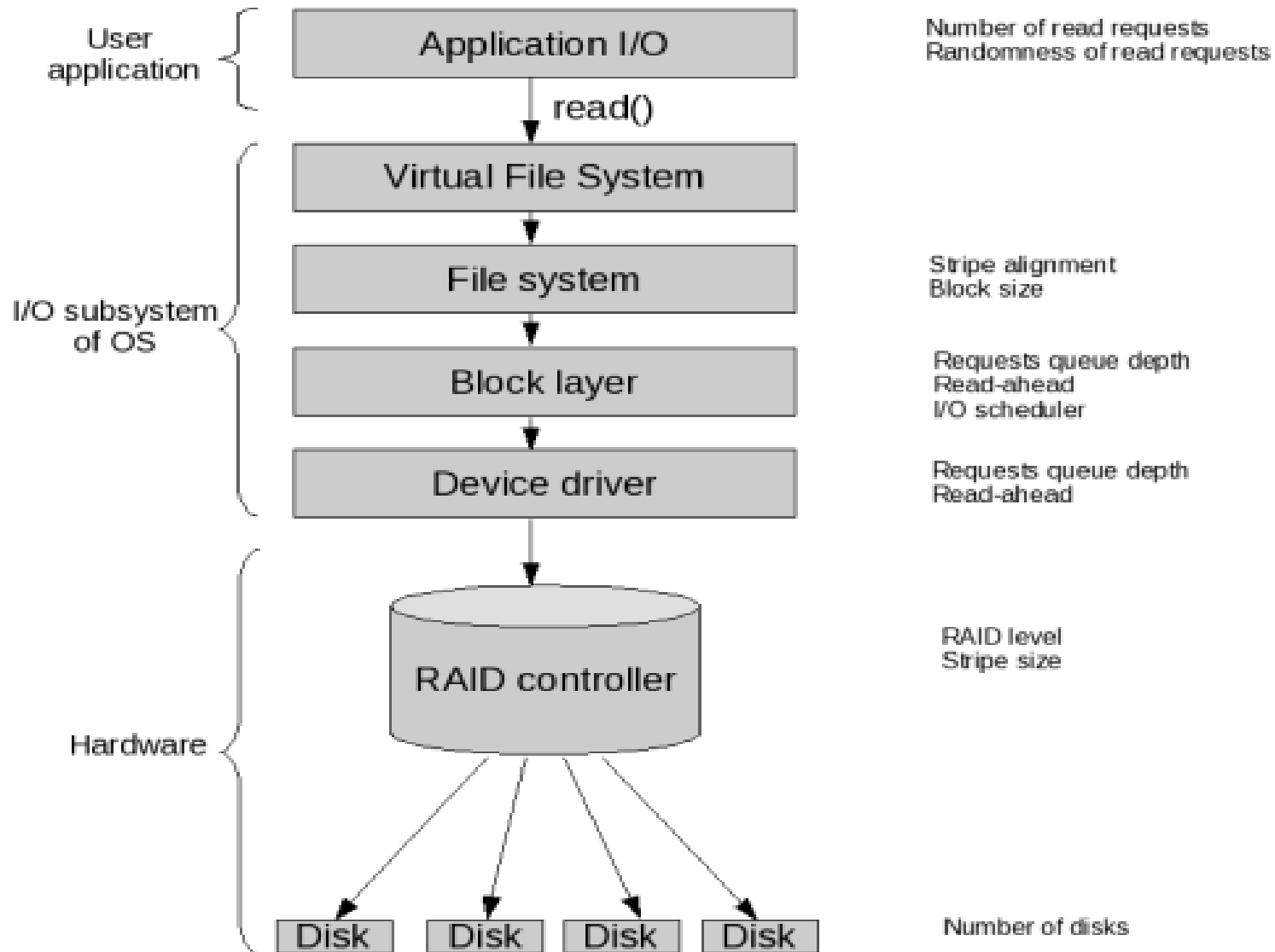
Subjects:

- Read performance of AliROOT analysis tasks and trains
- Read performance for ALICE ESD data-files

Goals:

- To understand what is limiting the performance
- To find ways to improve the performance

The Path of I/O Request



...trying to figure out what happens on every level

General Conditions

ROOT 523-02

AliROOT v4-16-Rev-08

pp collisions at 10 TeV

Number of files in data set: 100 000

Events per file: 100

File size: from 2.3 MB up to 8 MB

Each analysis job runs on dedicated 1000 files

General Conditions

8 core Intel Xeon E5430 2.66 GHz

RAID controller: *HP Smart Array P400*

- **array A**: RAID 1+0 (2 disks) - Debian 4.0
- **array B**: RAID 5 (5 disks x 250 GB + 1 spare disk) – xfs with analysis data
 - 64 KB stripe size
 - block device readahead: 4 MB

Logical Layout of ALICE ESD Tree

499 branches in a tree:

- 484 branches are resident in the AliESDs.root
- 6 branches in AliESDfriends.root
- 9 are resident in memory

Physical Layout of AliESDs.root

- 675 baskets
 - 70% hold 100 events per basket
 - 15 % hold 1 event per basket
 - 7% hold 2 events per basket
 - 8% - else

the physical layout of tree contents

Train and Task Properties

23 tasks in the train

- 8 require MC data (otherwise crash)

If MC data is queried additional read calls are issued to:

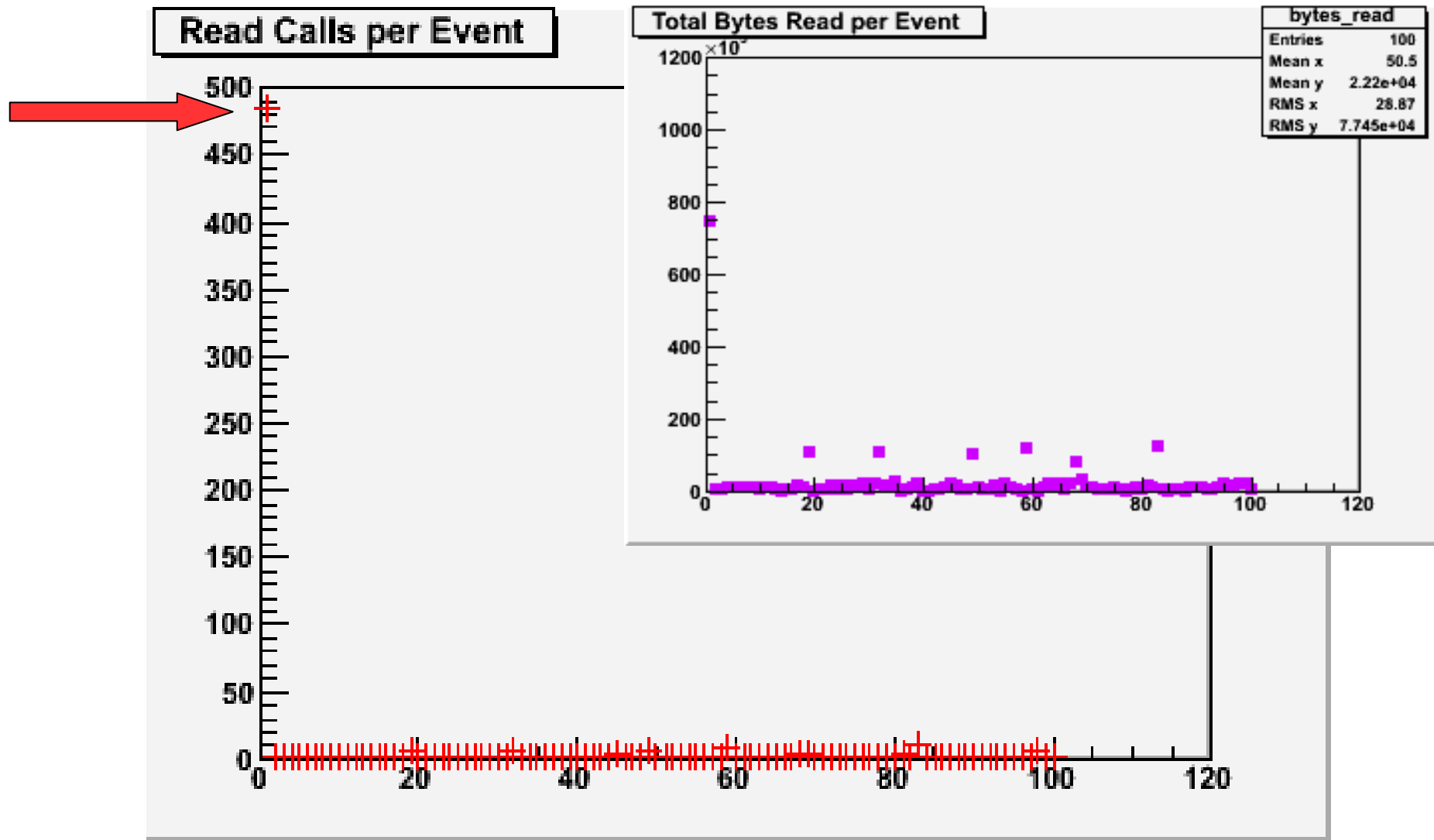
- galice.root,
- Kinematics.root,
- TrackRefs.root

AliESDfriends.root is not used => 484 branches are read;

FMD and CaloClusters* branches

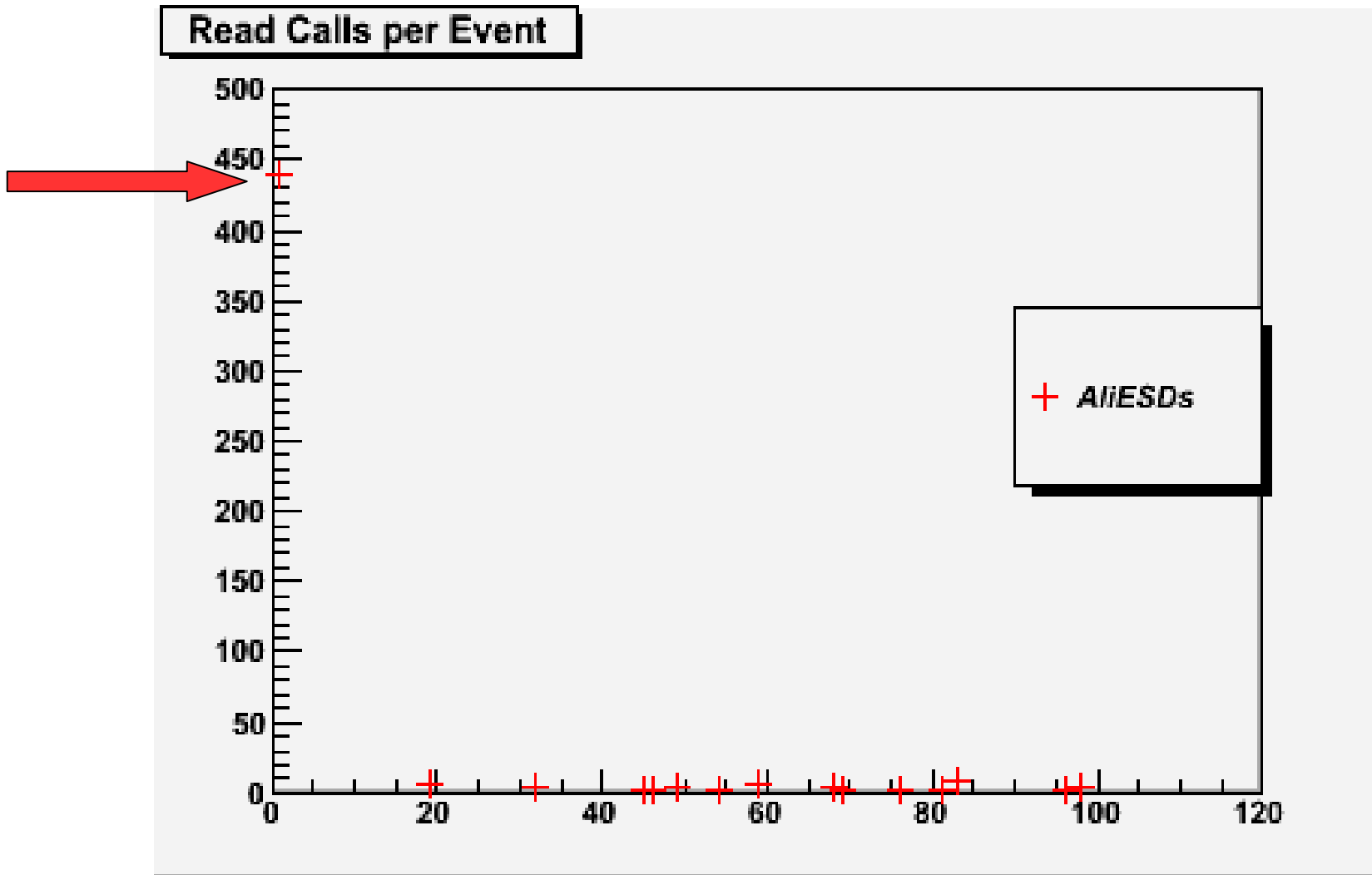
are disabled => $484 - 20 - 24 = 440$ branches are read

I/O Access to AliESDs.root



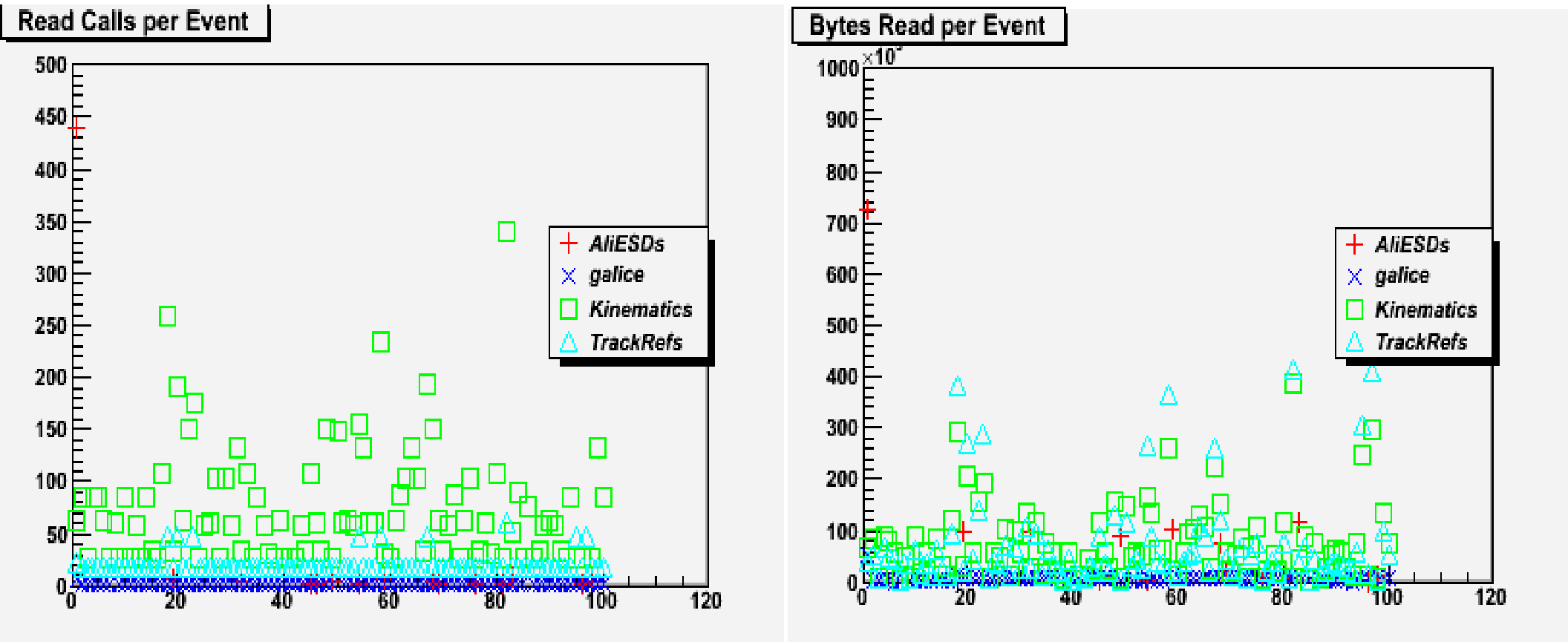
Read calls to AliESDs.root file for each event processed.
Analysis task processing 100 events. All branches are activated.

I/O Access to AliESDs.root



Read calls to AliESDs.root file for each event processed.
Analysis task processing 100 events. 44 branches are deactivated.
Among them those with 100 and 50 baskets

I/O Access per each ESD File



Read calls and bytes read statistics for different ROOT files for each event processed in AliESDs.root file. Analysis task processing 100 events. Additional Monte Carlo truth information queried.

I/O Access Share of MC Data Queries

File Name	File Size (B)	Single Task		Train	
		Read Calls	Bytes Read	Read Calls	Bytes Read
<u>AliESDs.root</u>	2 866 469	483	1 377 525	483	1 377 525
<u>galice.root</u>	6 058 811	101	683 172	101	683 172
<u>Kinematics.root</u>	6 150 493	403	320 596	7708	7 407 476
<u>TrackRefs.root</u>	5 728 700	403	199 110	2069	6 722 433

Table 3. Statistics for requested files per single data file AliESDs.root processed with analysis task and with analysis train.

read call = POSIX read() call

Ways to Optimize Disk Access Performance

- Tuning RAID controller and I/O subsystem of OS
- Using TTreeCache (ROOT I/O optimization)
- Prefetching files to memory (avoid ROOT disk access)
- Merging files
 - reduce number of files
 - increase file size
- Changing hardware technology
 - hard disk -> solid state disk

TTreeCache

With the Learning Phase of 1 entry:

- 1) For the first entry all first baskets of every activated branch are requested as usual. **A single request per basket.**
- 2) For a second entry all remaining baskets of every activated branch in the file are requested.

A single request per group of baskets adjacent on disk.

	With <u>TTreeCache</u> enabled	Without <u>TTreeCache</u> enabled
Read Calls to <u>AliESDs.root</u>	2 708	49 006
Bytes Read from <u>AliESDs.root</u>	152 413 985	152 413 985

Table 2. Statistics for analysis task processing a chain of 100 AliESDs files, each with 100 events.

galice.root, Kinematics.root, TrackRefs.root
are not affected

Ways to Optimize Disk Access Performance

- Tuning RAID controller and I/O subsystem of OS
- Using TTreeCache (ROOT I/O optimization)
- Prefetching files to memory (avoid ROOT disk access)
- Merging files
 - reduce number of files
 - increase file size
- Changing hardware technology
 - hard disk -> solid state disk

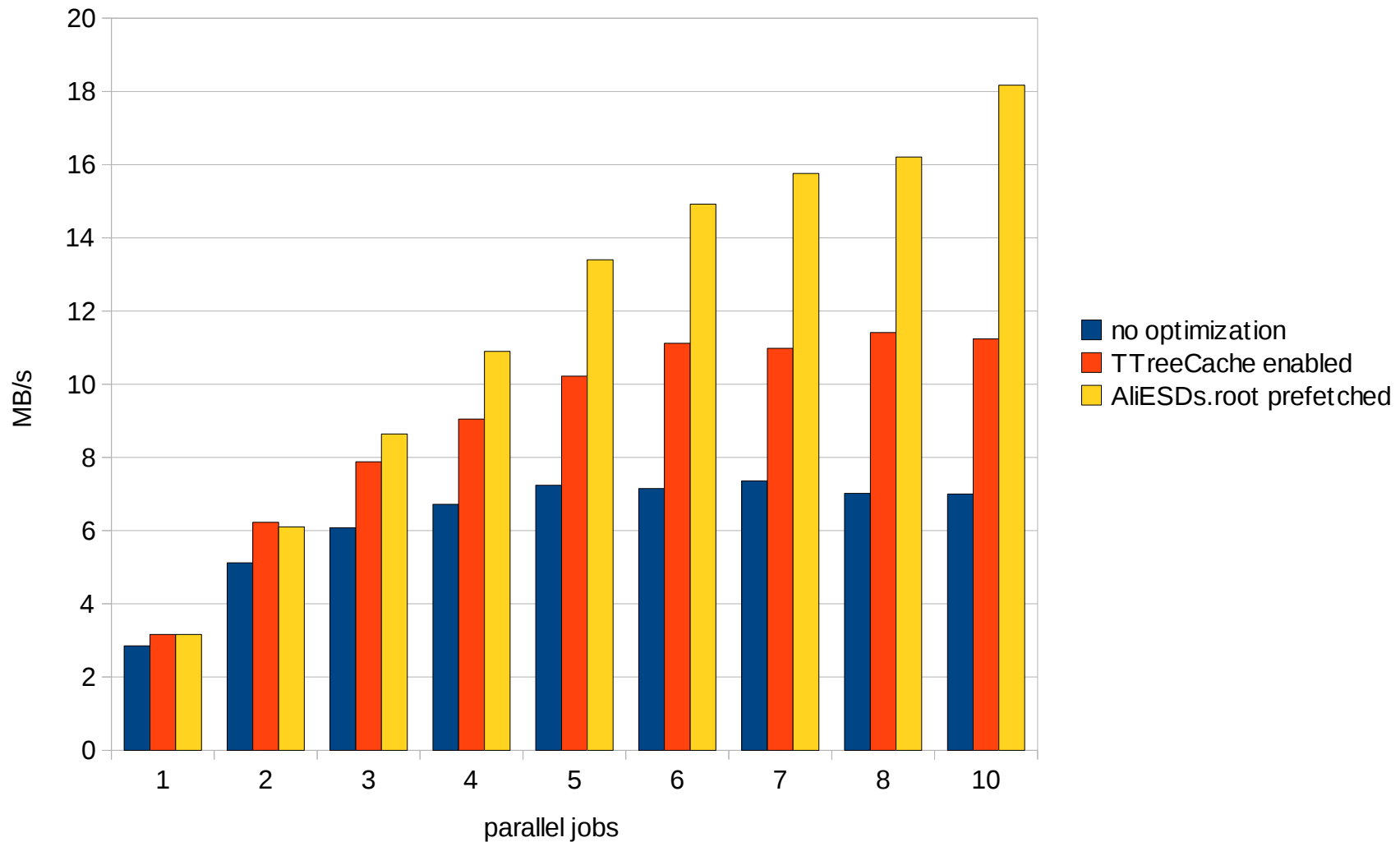
Prefetching

- ESD file is prefetched in Notify() in *ANALYSIS/AliAnalysisManager.cxx*
- Kinematics, galice, TrackRefs files are prefetched in Init() in *STEER/AliMCEventHandler.cxx*

Advantage: sequential disk access

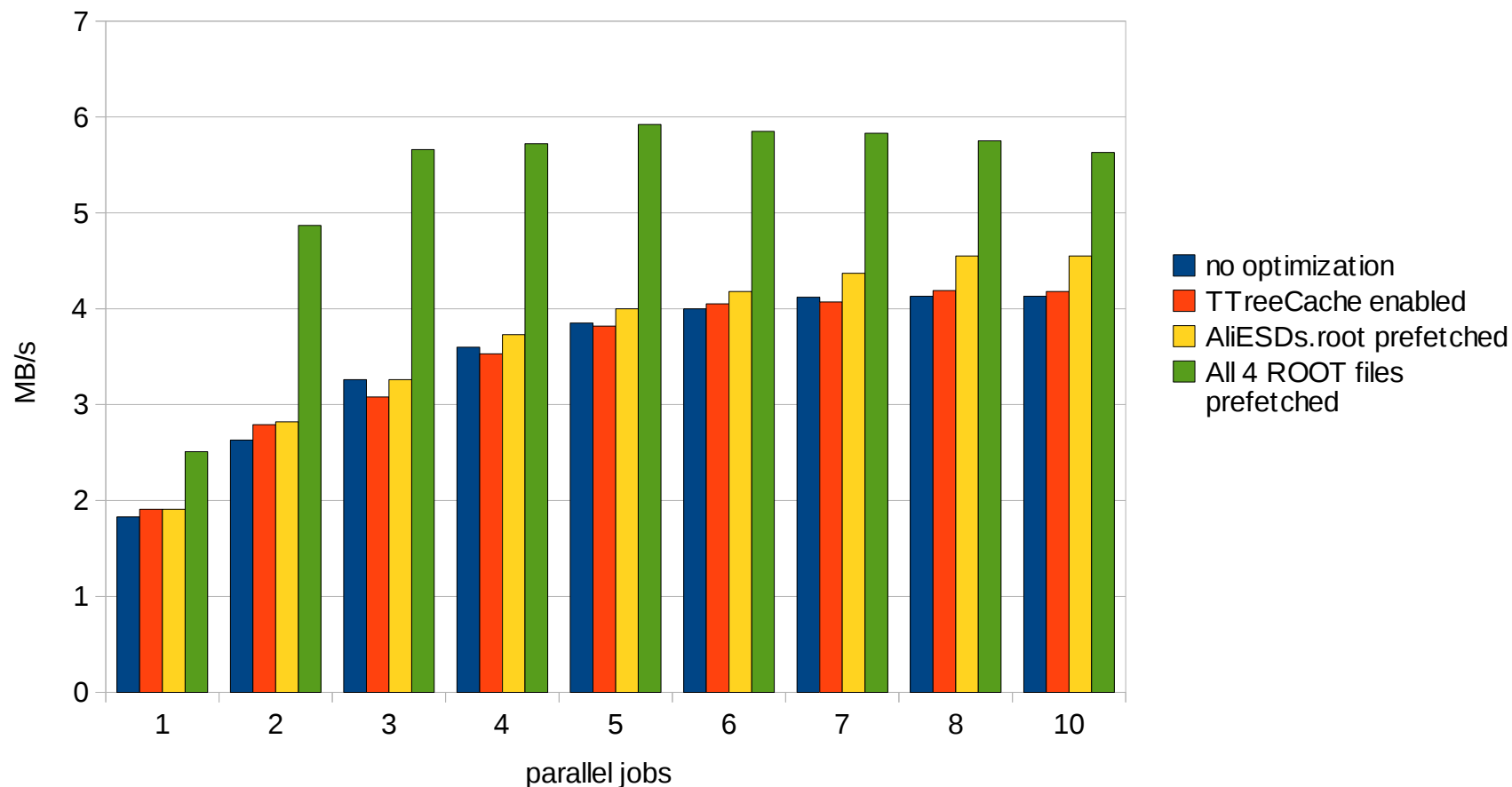
Overall Data Throughput Rate of Jobs Running a Task

without MC queries



Comparison of throughput rates for parallel jobs running a single analysis task. Only AliESDs.root is accessed.

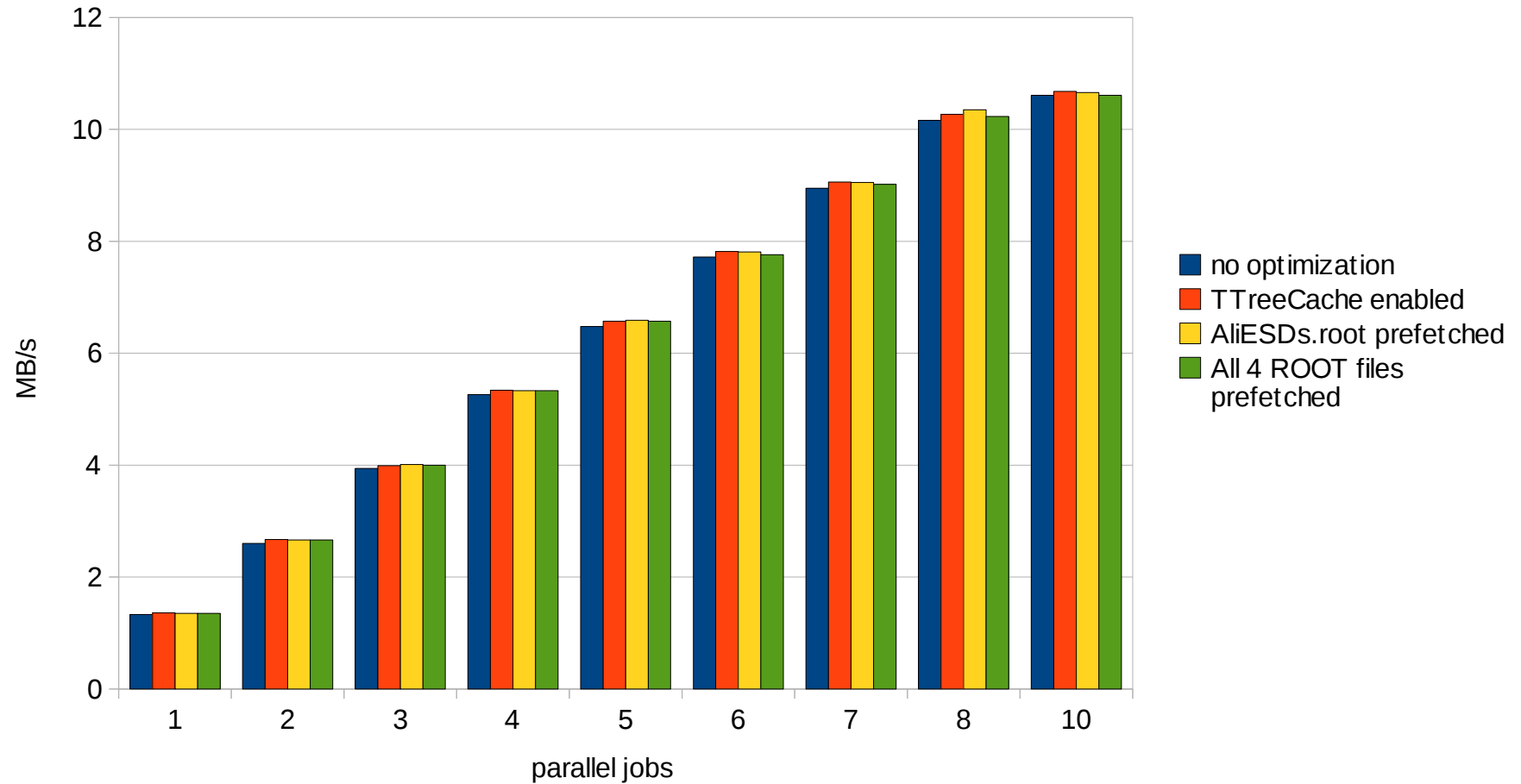
Overall Data Throughput Rate for Jobs Running a Task with MC queries



Comparison of throughput rates for parallel jobs running a single analysis task querying Monte Carlo data. AliESDs.root, Kinematics.root, galice.root, TrackRefs.root are read.

Overall Data Throughput Rate for Jobs Running a Train

with MC queries



Comparison of throughput rates for parallel jobs running an analysis train querying Monte Carlo data. AliESDs.root, Kinematics.root, galice.root, TrackRefs.root are read.

Impact of Parallel Jobs

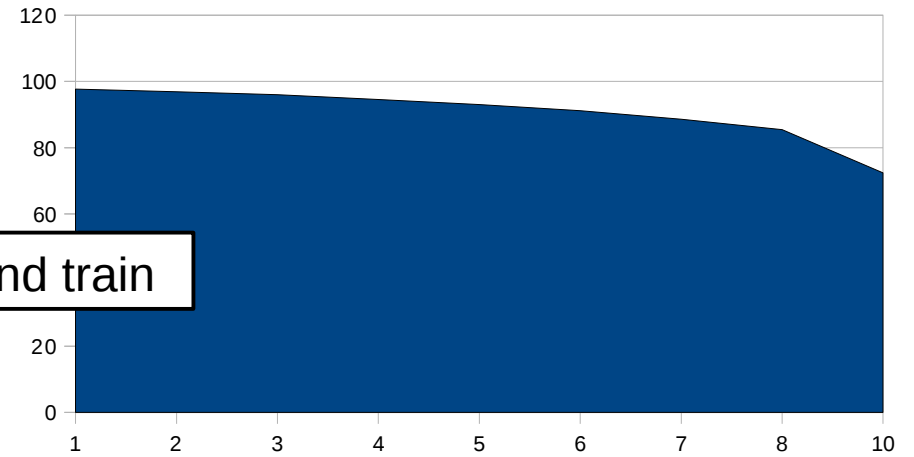
Data Throughput Rate of a Single Analysis Train Job

23 tasks, with MC data queries



CPU Utilization of a Single Analysis Train Job

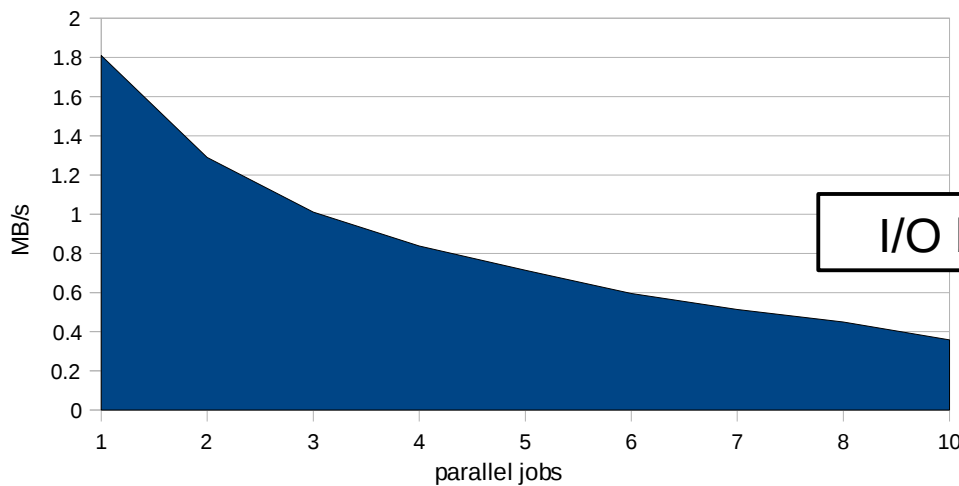
23 tasks, with MC data queries



CPU bound train

Data Throughput Rate of a Single Analysis Task Job

MC data queried



CPU Utilization of a Single Analysis Task Job

MC data queried



I/O bound task

Ways to Optimize Disk Access Performance

- Tuning RAID controller and I/O subsystem of OS
- Using TTreeCache (ROOT I/O optimization)
- Prefetching files to memory (avoid ROOT disk access)
- **Merging files**
 - **reduce number of files**
 - **increase file size**
- Changing hardware technology
 - hard disk -> solid state disk

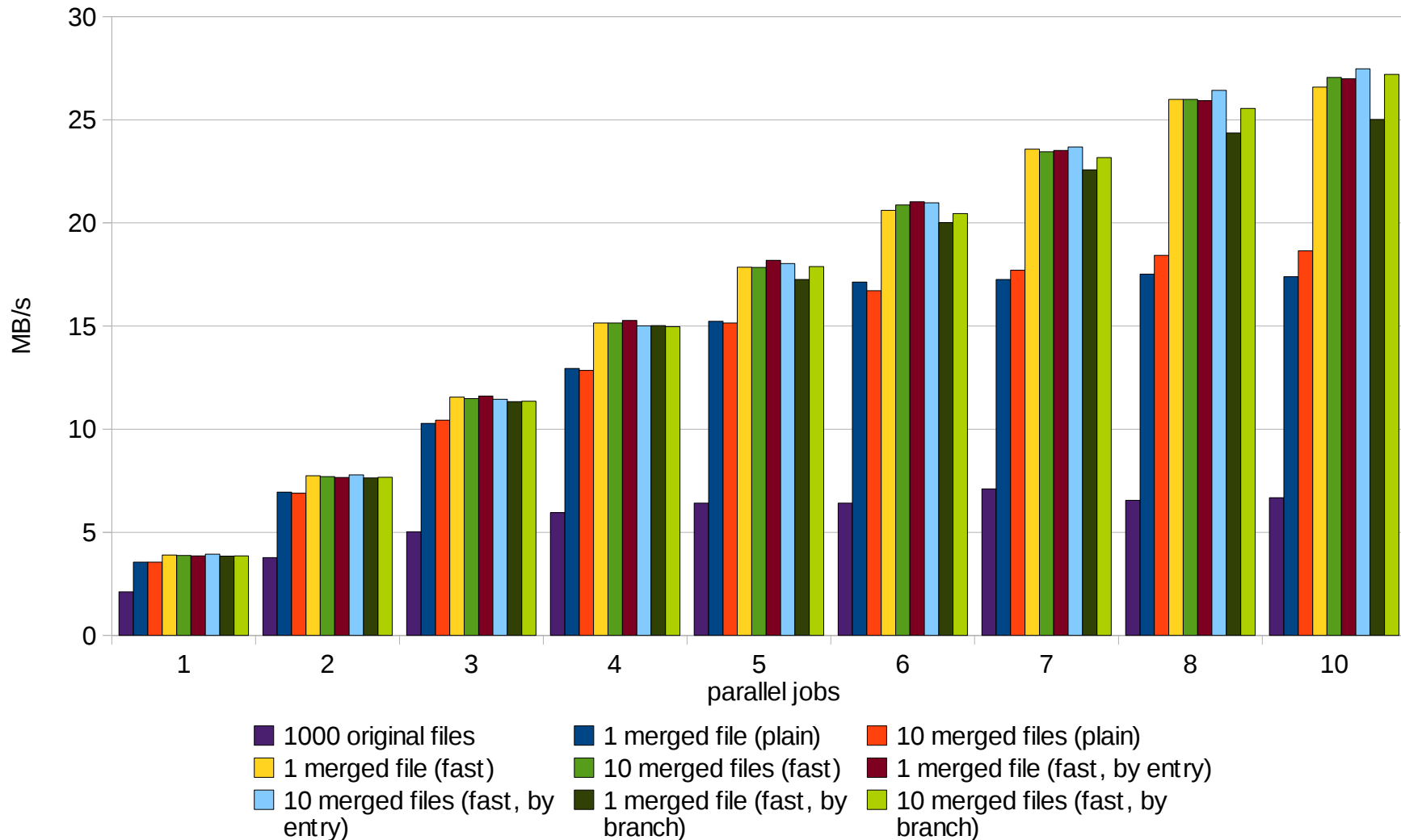
Merging ROOT Files

- Plain merge
- Fast merge
- For unzipped baskets there are currently 3 supported sorting orders:
 - SortBasketsByOffset (the default)
 - SortBasketsByBranch
 - SortBasketsByEntry

Statistics for Merged Files

Aggregate Data Throughput Rate for Parallel Analysis Jobs

train of 15 tasks, without MC queries, 1 job processes 100 000 events



Ways to Optimize Disk Access Performance

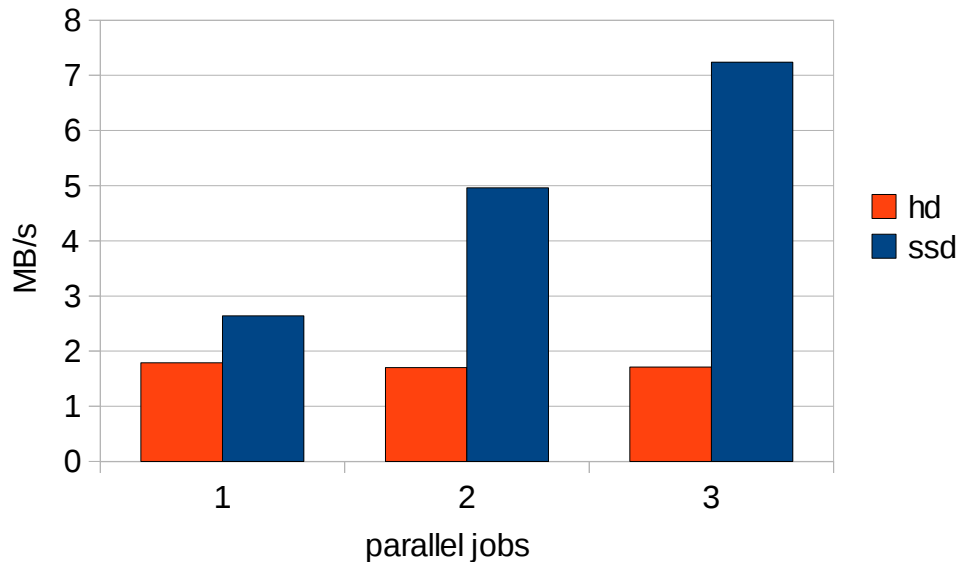
- Tuning RAID controller and I/O subsystem of OS
- Using TTreeCache (ROOT I/O optimization)
- Prefetching files to memory (avoid ROOT disk access)
- Merging files
 - reduce number of files
 - increase file size
- Changing hardware technology
 - hard disk -> solid state disk

Solid State Disk

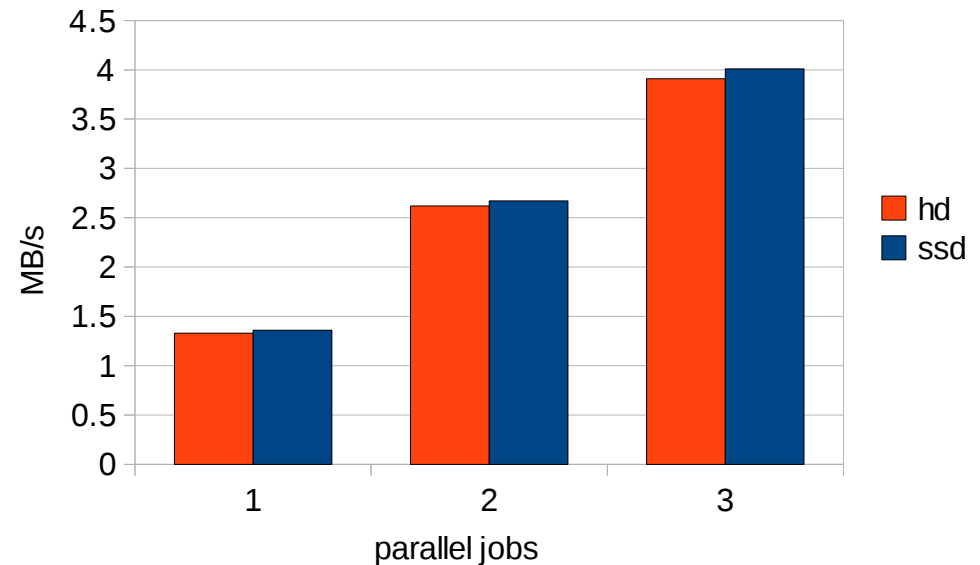
hdparm results for read performance:

- 50 MB/s hard disk
- 130 MB/s solid state disk

Aggregate Throughput Rate of Analysis Tasks
with MC data queries



Aggregate Throughput Rate of Analysis Trains
23 tasks, with MC queries



Summary

- Analysis train is CPU bound
- Analysis task is I/O bound
- Factors contributing to low I/O performance:
 - Large number of relatively small ESD files
 - Large number of branches -> large number of baskets inside ESD files
 - Numerous queries to Kinematics.root, TrackRefs.root

Summary (2)

- I/O subsystem tuning: default values mostly optimal. Block device read-ahead value: assign empirically
- TTreeCache -> reduces number of read calls -> can improve analysis speed
- Merging files: improves data throughput rate (factor of 3 for parallel jobs)

The best option: 'fast SortBasketsByEntry'