Abstract

The present document deals with the ongoing construction of ALICE Tier-2 computing centre at GSI. To examine utilization of computing resources by high energy physics data analysis jobs various tests are carried out. The stress is put on the analysis speed which directly depends on data throughput rates. The first set of tests is aimed at illustrating results delivered by unoptimized analysis. The second set of tests points to various ways to optimize analysis scheme by either tuning hardware configurations, modifying analysis code, or changing the way analysis data is stored. This document is meant to help efficiently configure GSI data analysis environment for ALICE.

Table of Contents

Notice:

- tests are accounted in the order of their execution

The purpose of research

To set up an optimal computing environment to run analysis jobs at ALICE Tier-2 centre in GSI.

First set of research tests

The purpose of the first set of tests:

To find out the speed (events per second) and data throughput rate (MB per second) of ALICE analysis job processing local data, data from a remote fileserver, and the effects on speed caused by running jobs in parallel. To find out how to configure analysis to minimize its running time.
In order to disentangle implications on data reading speed of ALICE analysis framework features from the ROOT and hardware ones, tests were in parallel carried out on generated Pythia data with ROOT's TSelector class used for analysis.

The initial conditions of the first set of tests:

Test server. Local test server lxb281.gsi.de is a representative example of a GSI server used for ALICE data analysis. This is where analysis jobs are run, and where the data is stored for local disk tests.

Properties:

Xeon X5355 @ 2.66 GHz
2 Quad-core processors => 8 cores
x86_64 => 64-bit
16 GB RAM
OS: Debian 4.0
Disk Controller: 3Ware 9650 SE-4LPML
4 Disks: Western Digital WD 5000 4S-01MPB1, configured as RAID5
Default readahead value 0,125 MB

Remote fileserver. Fileserver lxfs51.gsi.de is a representative example of a GSI fileserver used for ALICE data storage. It is used for tests of remote data analysis.

Properties:

Xeon @ 2.8 GHz
1 single-core processor => 1 core
i686 => 32-bit
3.2 GB RAM
OS: Debian 3.1
2 Disk Controllers: 3Ware 9550 SX 8LP
8+6 Disks configured as RAID5 (2 of 8 for OS)
Default readahead value 0,125 MB

Network capacity between servers is 1 Gb/s.
ALICE analysis source ESD data (event summary data) :
1 Million pp collisions, v4-05-Rev-03 (PDC07)
10000 of AliESDs.root files
1 file = 100 events ≈ 3,3 MB
10000 files = 1 000 000 events = 33,6 GB
File compression level of AliESDs.root is 1
Tree compression factor is ~3.5
Number of branches is 406
ALICE analysis code:
“Simple task” - a lightweight (simple) AliAnalysisTask example taken from ALICE Offline Bible;
“Train of tasks” - a train of tasks containing 3 AliAnalysisTasks (one aforementioned example, and two “real life” examples). Event by event all branches are read.

ROOT environment:
version 515-06 (compiled in 64-bit mode)
. alilogin v4-05-Release
Pythia analysis source data:
1 file = 190 events ≈ 3,3 MB
10000 files = 1 900 000 events ≈ 33 GB
File compression level is 1
Tree compression factor is ~4.16
Number of branches is 22
Pythia analysis code:
Analysis code for Pythia events boils down to simply reading all events without any additional computation using TSelector class
Lustre cluster:
13 fileservers of the same as architecture as a “remote fileserver”(lxfs51).

How measurement is done:

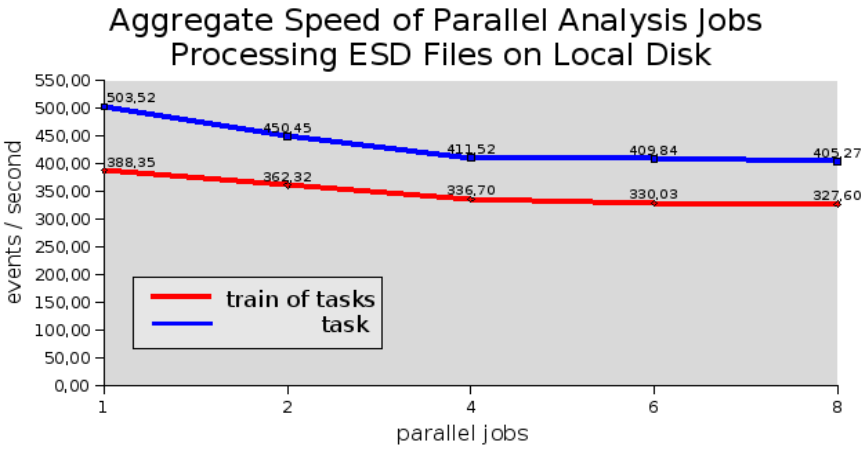
To ensure for each job in the tests that all data files are read from disk, memory cache is flushed between test runs. Every job processes its own data set.

scripts in appendix

Test 1.1 Initial unoptimized analysis of source data

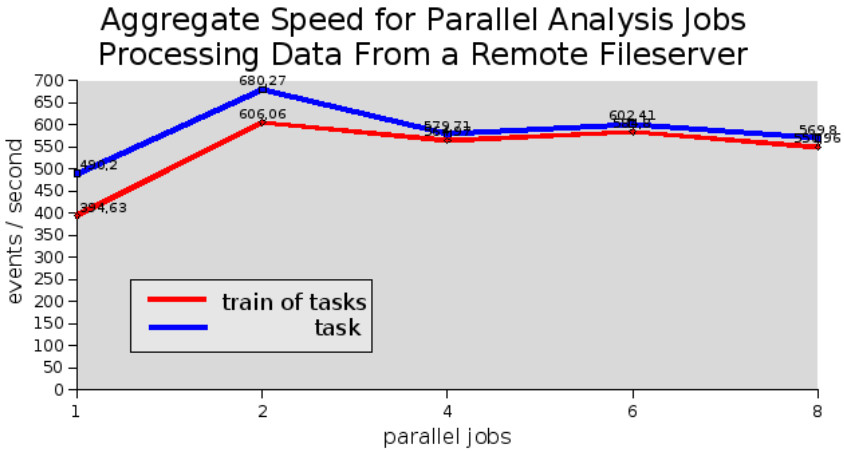
file simple_vs_train_lxb281_local.ods
Object1.1.1: Speed of an analysis job processing local ESD data. Analysis job processes 10000 files of 100 events each.
Result1.1.1: ~535 ev/s for simple task; ~414 ev/s for train of tasks

Object1.1.2: Aggregate speed of parallel analysis jobs processing local data. One analysis job processes 10000 files of 100 events each.
Result1.1.2:



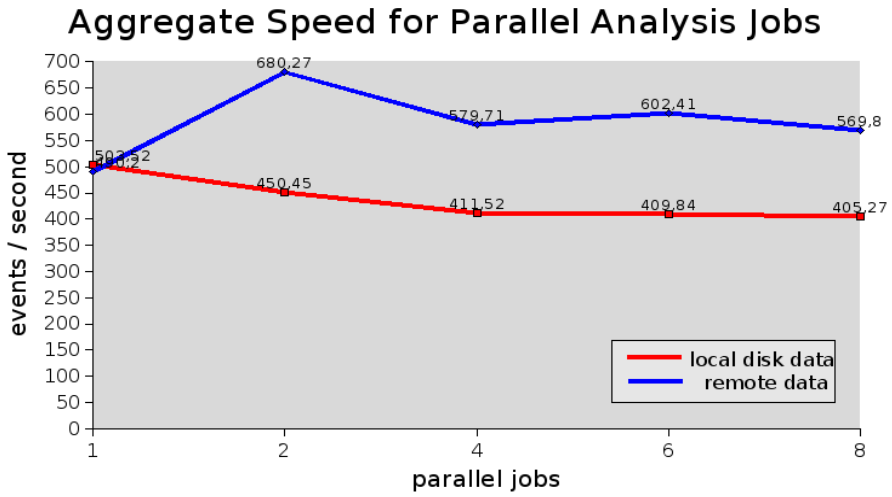
file simple_vs_train_lxfs.ods
Object1.1.3: Speed of an analysis job processing data from a remote fileserver. Analysis job processes 10000 files of 100 events each.
Result1.1.3: ~495 ev/s for simple task; ~383 ev/s for train of tasks

Object1.1.4: Aggregate speed of parallel analysis jobs processing data from a remote fileserver. One analysis job processes 10000 files of 100 events each.
Result1.1.4:



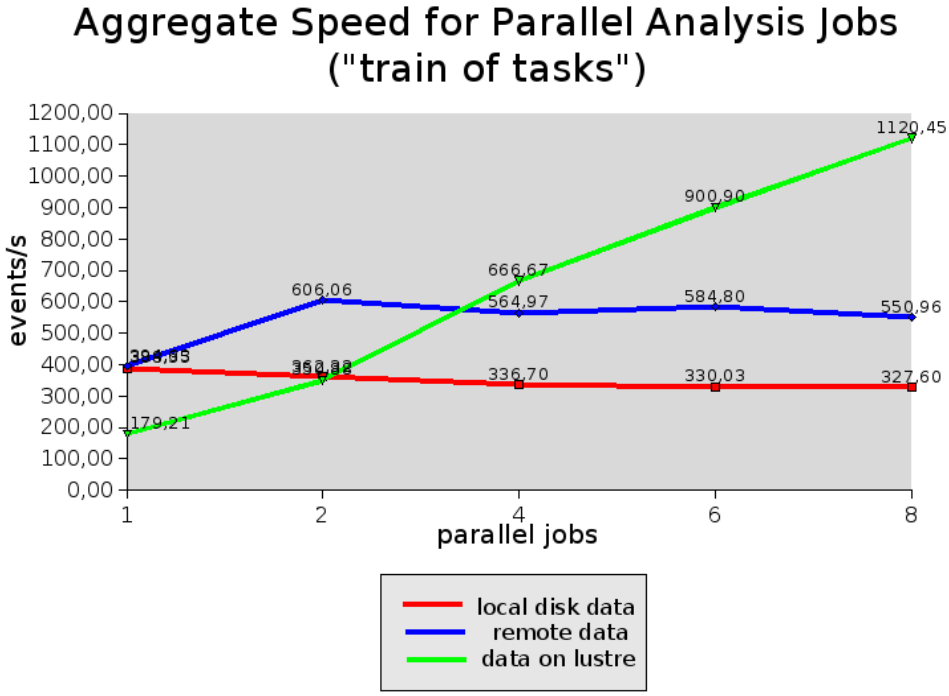
file local_vs_lxfs_simple.ods
Object1.1.5: Difference in speed between a job processing local data, and a job processing data from a remote fileserver. Analysis job (“simple task”) processes 10000 files of 100 events each.
Result1.1.5: 535 ev/s for local, 495 ev/s for remote

Object1.1.6: Difference in speed between parallel jobs processing local data, and parallel jobs processing data from a remote fileserver. One analysis job (“simple task”) processes 10000 files of 100 events each.
Result1.1.6:



file local_vs_lxfs_train.ods
Object1.1.7: Difference in speed between a job processing local data, job processing data from remote fileserver, and a job processing data from Lustre cluster. Analysis job (“train of tasks”) processes 10000 files of 100 events each.
Result1.1.7: 414 ev/s for local, 383 ev/s for remote, 183 ev/s for Lustre

Object1.1.8: Difference in speed between parallel jobs (train of tasks) processing local data, parallel jobs processing data from remote fileserver, and parallel jobs processing data from Lustre cluster. One analysis job (“train of tasks”) processes 10000 files of 100 events each.
Result1.1.8:



Summary A:

1. The data throughput for one ESD analysis job (“simple task”) processing data files from a local disk storage is around 16 MB/s, which brings analysis speed to around 535 ev/s.
2. Aggregate data throughput for parallel jobs is at the same level as for one job, which means that throughput per job shrinks linearly.
3. Analysis of data from a remote fileserver is faster than of the local data. Possible reasons: link is free so the network in between is transparent, and fast xrootd protocol is used.
4. Analysis speed of data stored on Lustre cluster scales perfectly for parallel jobs, though the speed for one job is surprisingly low.

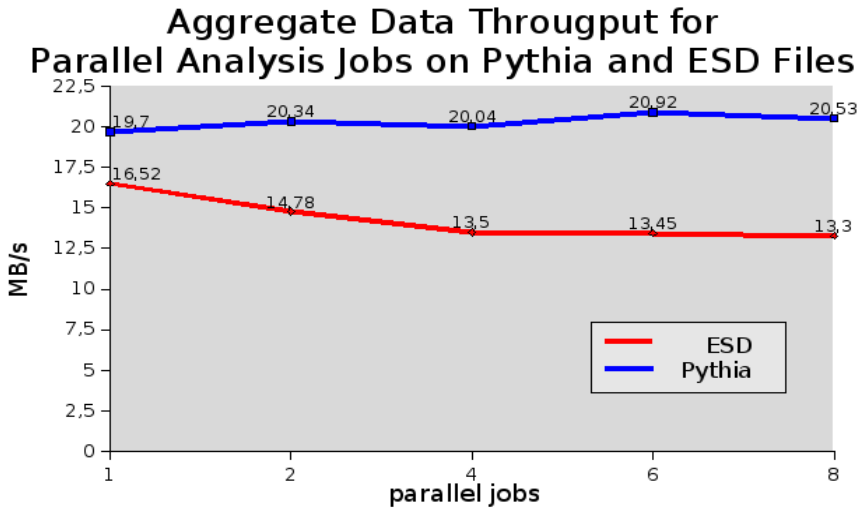
Test 1.2 ESD analysis vs. Pythia analysis

file pythia_vs_ESD.ods

Object1.2.1: Difference in data throughput rates between parallel jobs (simple task) processing local ESD data, and parallel jobs processing (just reading event by event) local Pythia data. ESD analysis job (“simple task”) processes 10000 files of 100 events each. Pythia analysis job processes 10000 files of 190 events each.

ESD	Pythia
1 file = 100 events ≈ 3,3 MB 10000 files = 1 000 000 events = 33,6 GB File compression level is 1 Tree compression factor is ~3.5 Number of branches is 406	1 file = 190 events ≈ 3,3 MB 10000 files = 1 900 000 events ≈ 33 GB File compression level is 1 Tree compression factor is ~4.16 Number of branches is 22

Result1.2.1:



Summary B:

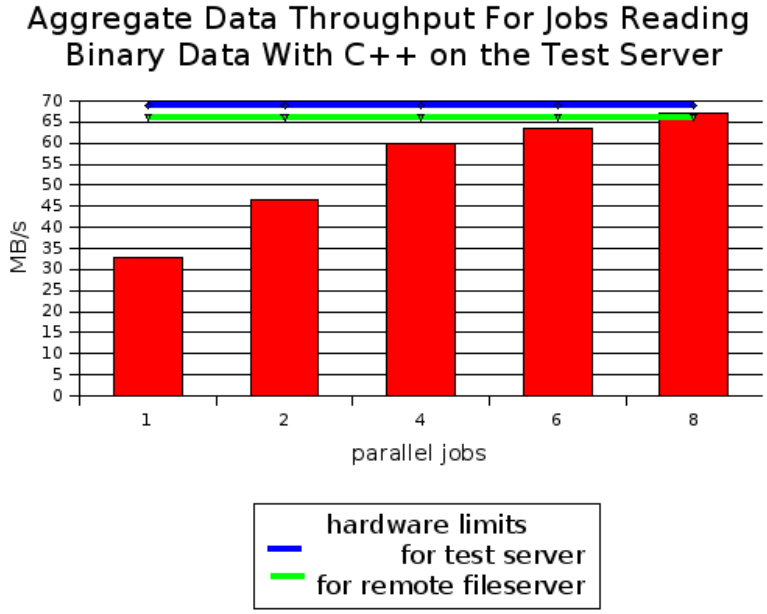
1. Pythia trees are read faster than ESD ones because the amount of branches is smaller by a factor of 20.

Questions so far:

1. Why data throughput is so low for ROOT code?
2. Why data throughput doesn't scale for parallel ROOT jobs?

Test 1.3 Hardware disk throughput limitations

file hardware_limits.ods
Object1.3.1: Highest achievable data throughput limited by hardware for test servers. Aggregate data throughput for parallel C++ jobs. One C++ job processes 100 ESD files of 100 events each. It reads ESD data files as binary data by chunks of 1 KB. Hardware limits to be measured by hdparm.
Result1.3.1:



Summary C:

1. Data throughput for a ROOT job is twice smaller than for a plain C++ job.
2. Data throughput for parallel C++ jobs scales, and for 8 jobs reaches the hardware limit.

Test 1.4

file pythia_compress.ods
Object1.4.1: The influence of data compression level on data throughput rate for a ROOT job. To be measured on Pythia data. Analysis job processes 100 Pythia files of 190 events each.
Result1.4.1:

Reading 100 Pythia data files (190 events per file) with ROOT analysis code

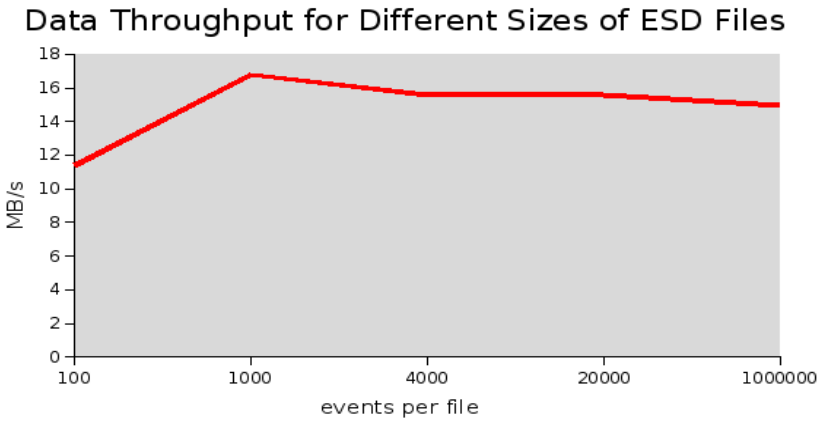
compression level	file size, MB	total data read, MB	total events	real time, s	user time, s	MB/s	events/s
1	~3,4	340	19000	17	09	20	1117,65
0	~14	1385,3	19000	26	04	53,27	730,77

Summary D:

1. Throughput for uncompressed Pythia data is higher by a factor of 2.65, but the amount of binary data to read grows by a factor of 4. This leads to the conclusion that analysis speed of compressed Pythia data is faster by a factor of 1.5.

Test 1.5 Influence of data file size on data throughput

file dif_file_sizes.ods
Object1.5.1: The influence of ESD file size on data throughput for ROOT analysis job. Source ESD files to be merged into larger files of different sizes. Besides event tree files contain some additional data. Only event tree is to be merged. Analysis job (“train of tasks”) processes varied number of files of varied size as shown in the table.
Result1.5.1:



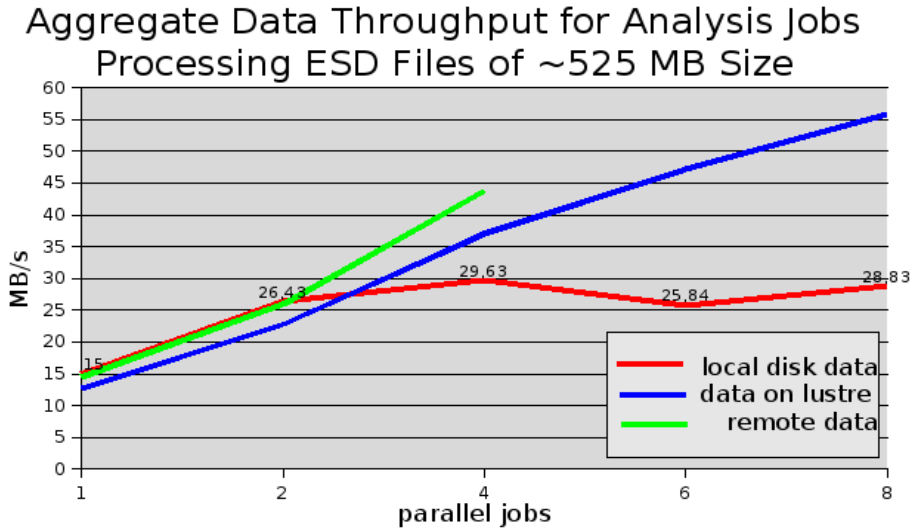
	number of files	average file size, MB	events per file
original	10000	3,28	100
only merged esdTree	10000	2,68	100
	1000	26,11	1000
	250	105,68	4000
	50	522,24	20000
	1	26042	1000000

Summary E:

1. In comparison to analysis of 10000 source ESD files of 3,3 MB size data throughput for analysis of 1000 files of 26 MB size grows by 1/3. At this level of 1000 events per file data throughput rate starts to saturate. Possible reason: For each file an initialization procedure has to be carried out which cannot be done in parallel. While the number of files increases the times for initialization sum up.

Test 1.6 Throughput for larger data files

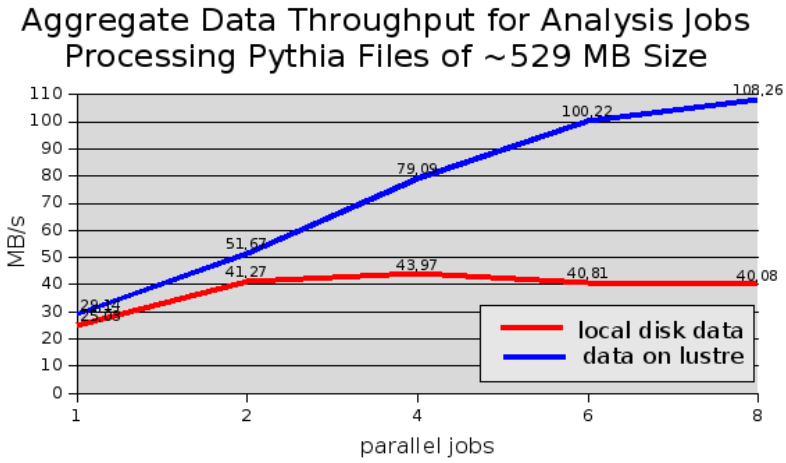
file big_50esd_scale.ods
Object1.6.1: Data throughput comparison for analysis of ESD files located on local disk, on a remote fileserver, and on a Lustre cluster. One analysis job processes 50 ESD files of ~525 MB size (20000 events per file).
Result1.6.1:



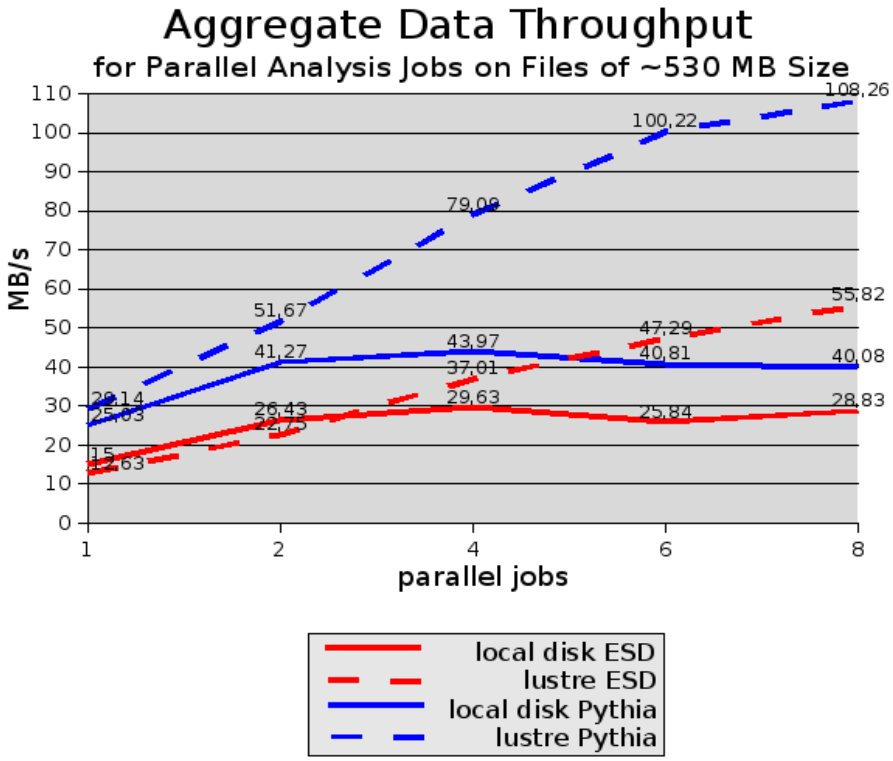
Summary F:

1. First scaling in data throughput rate for parallel ROOT analysis jobs on local disk data is observed. Aggregate data throughput rate for 2 parallel jobs is higher than for one job by a factor of 1.75. With addition of more parallel jobs the level of data throughput rate saturates.
2. A subset of parallel jobs running analysis on data stored on a remote fileserver runs into a ROOT segmentation violation error. It may be 4 jobs or 1 job, but it happened at all test reruns. Possible explanation: .

Object1.6.2: Data throughput comparison for analysis of Pythia files located on local disk, and on a Lustre cluster. One analysis job processes 50 Pythia files of ~530 MB size (30000 events per file).
Result1.6.2:



Object1.6.3: Data throughput comparison for analysis of Pythia and ESD files located on local disk, and on a Lustre cluster. One ESD analysis job processes 50 ESD files of ~525 MB size (20000 events per file). One Pythia analysis job processes 50 Pythia files of ~530 MB size (30000 events per file).
Result1.6.3:



Summary G:

1. Aggregate data throughput rate behaviour is similar for parallel jobs processing ESD files, and Pythia files. The gap in rate values can be explained with difference in tree structures. First, as mentioned before, the number of branches is significantly higher for ESD tree than for Pythia tree (by a factor of 20). Second, ESD tree compression factor is 5 times higher than compression factor for Pythia tree. For ESD tree compression factor has risen from 3.5 for files of ~3MB size to 20 for files of ~525 MB size. For Pythia tree compression factor remains on the level of 4.16.
2. Aggregate data throughput rate for 8 parallel jobs processing Pythia files from Lustre cluster almost reaches network link capacity which is 1 Gb/s.

Overall summary:

1. The data throughput for one ESD analysis job (“simple task”) processing data files from a local disk storage is around 16 MB/s, which brings analysis speed to around 535 ev/s.
2. Analysis of data from a remote fileserver is faster than of the local data. Possible reasons: link is free so the network in between is transparent, and fast xrootd protocol is used.
3. Data throughput for a ROOT job is twice smaller than for a plain C++ job that reads into memory.
4. Data throughput for parallel C++ jobs scales, and for 8 jobs reaches the hardware limit.
5. In comparison to analysis of 10000 source ESD files of 3,3 MB size data throughput for analysis of 1000 files of 26 MB size grows by 1/3. At this level of 1000 events per file data throughput rate starts to saturate. Possible reason: For each file an initialization procedure has to be carried out which cannot be done in parallel. While the number of files increases the times for initialization sum up.
6. First scaling in data throughput rate for parallel ROOT analysis jobs on local disk data is observed when files are merged into much larger ones (from 3.3 MB to ~525 MB).
7. A subset of parallel jobs running analysis on data stored on a remote fileserver runs into a ROOT segmentation violation error. It may be 4 jobs or 1 job, but it happened at all test reruns. Possible explanation: .
8. Aggregate data throughput rate behaviour is similar for parallel jobs processing ESD files, and Pythia files. The gap in rate values can be explained with difference in tree structures. First, as mentioned before, the number of branches is significantly higher for ESD tree than for Pythia tree (by a factor of 20). Second, ESD tree compression factor is 5 times higher than compression factor for Pythia tree.
9. Aggregate data throughput rate for 8 parallel jobs processing Pythia files from Lustre cluster almost reaches network link capacity which is 1 Gb/s.

Second set of research tests

The purpose of the second set of tests:

To find out how various hard disks configurations affect the data throughput for analysis jobs. To increase data throughput rate for analysis jobs.

The initial conditions of the second set of tests:

Test server A. Server grid33.gsi.de was set up with RAID 5 comprising 4 disks as storage.

Properties:

Xeon X5355 @ 2.66 GHz
2 Quad-core processors => 8 cores
x86_64 => 64-bit
32 GB RAM
OS: Debian 4.0
Disk Controller: 3Ware 9650 SE-4LPML
4 Disks configured as RAID5
Default readahead value 0,125 MB (hardware maximum 8MB)

Test server B. Server grid34.gsi.de was set up with 4 disks mounted separately as storage.

Properties:

Xeon X5355 @ 2.66 GHz
2 Quad-core processors => 8 cores
x86_64 => 64-bit
32 GB RAM
OS: Debian 4.0
Disk Controller: 3Ware 9650 SE-4LPML
4 Disks mounted separately

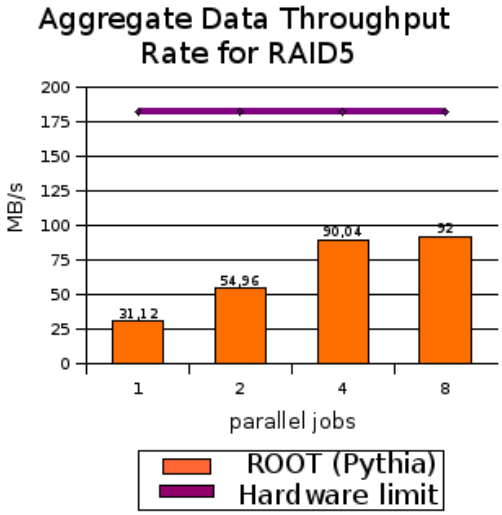
ALICE analysis source ESD data (event summary data) :
1 Million pp collisions, v4-05-Rev-03 (PDC07)
10000 of AliESDs.root files
1 file = 100 events ≈ 3,3 MB
10000 files = 1 000 000 events = 33,6 GB
File compression level of AliESDs.root is 1
Tree compression factor is ~3.5
Number of branches is 406
ALICE analysis code:
“Train of tasks” - a train of tasks containing 3 AliAnalysisTasks (one aforementioned example, and two “real life” examples)
ROOT environment:
version 515-06 (compiled in 64-bit mode)
Pythia analysis source data:
1 file = 190 events ≈ 3,3 MB
10000 files = 1 900 000 events ≈ 33 GB
File compression level is 1
Tree compression factor is ~4.16
Number of branches is 22
Pythia analysis code:
Analysis code for Pythia events boils down to simply reading all events without any additional computation using TSelector class

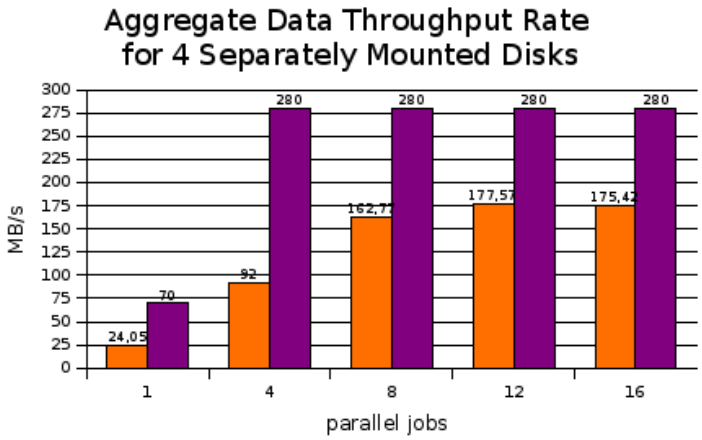
How measurement is done:

Measurement scheme is the same as for the first set of tests.

Test 2.1 Influence of disk storage configuration

file tuning_tests_py.ods
Object2.1.1: Comparison of aggregate data throughput rates for parallel analysis jobs processing Pythia files stored on local RAID5 storage, and on 4 local disks that are mounted separately. RAID5 readahead is increased to hardware maximum 8MB. Hardware limits are measured by hdparm. One analysis job processes 4 Pythia files of 30000 events each (~530 MB file size). For the test of 4 separately mounted disks: 1 job analyzes data from 1 disk, 4 jobs – one job per disk, 8 jobs – 2 jobs per disk, 12 jobs - 3 jobs per disk.
Result2.1.1:



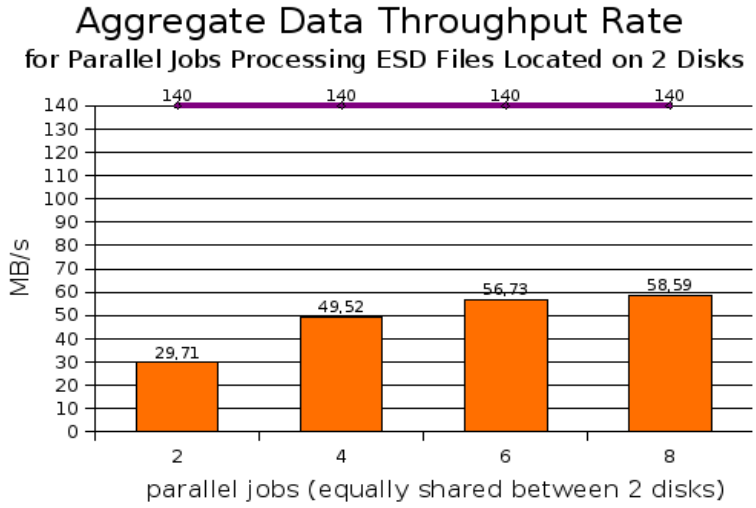


Summary H:

1. The usage of the maximum possible RAID5 readahead increases maximum possible data throughput rate for hardware from 70 MB/s (default readahead) to 180 MB/s. Aggregate data throughput rate for parallel analysis jobs on Pythia files scales up to 90 MB/s for 4 jobs. At this level throughput rate starts to saturate.
2. When using 4 separately mounted SATA disks the aggregate data throughput for all disks is the sum of each disk's throughput capability. In this case, 70 MB/s – for one disk, 280 – for 4 disks together. Aggregate data throughput rate for parallel analysis jobs on Pythia files peaks at 178 MB/s for 12 jobs (3 jobs per disk), saturation starts at the level of 2 jobs per disk.

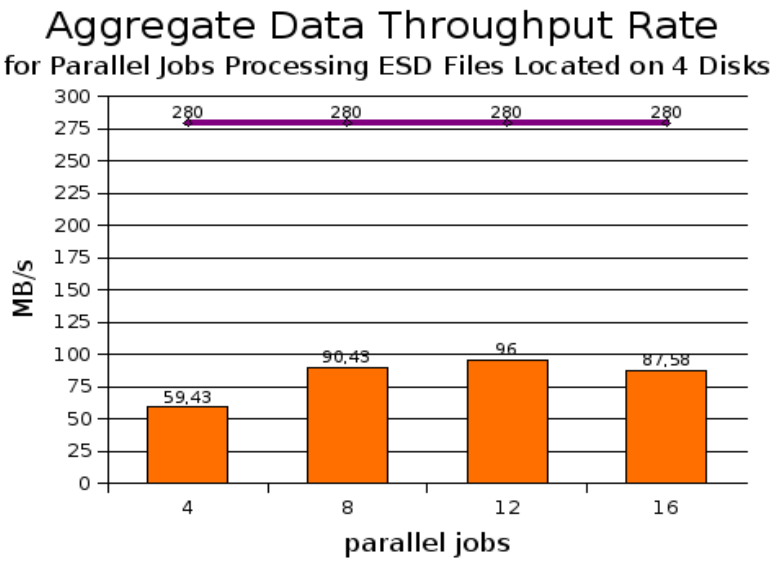
file disk_tests.ods
Object2.1.2: Aggregate data throughput rate for parallel analysis jobs processing ESD files located on a single disk. One analysis job (“train of tasks”) processes one ESD file of 20000 events each (~520MB file size).
Result2.1.2:

Object2.1.3: Aggregate data throughput rate for parallel analysis jobs processing ESD files located on two separately mounted disks. One analysis job (“train of tasks”) processes one ESD file of 20000 events each (~520 MB file size). Parallel jobs are equally shared between 2 disks: for 2 jobs – 1 job per disk, for 4 jobs – 2 jobs per disk, for 6 jobs – 3 jobs per disk, for 8 jobs – 4 jobs per disk.
Result2.1.3:



Object2.1.4: Aggregate data throughput rate for parallel analysis jobs processing ESD files located on four separately mounted disks. One analysis job (“train of tasks”) processes one ESD file of 20000 events each (~520 MB file size). Parallel jobs are equally shared between 4 disks: for 4 jobs – 1 job per disk, for 8 jobs – 2 jobs per disk, for 12 jobs – 3 jobs per disk, for 16 jobs – 4 jobs per disk.

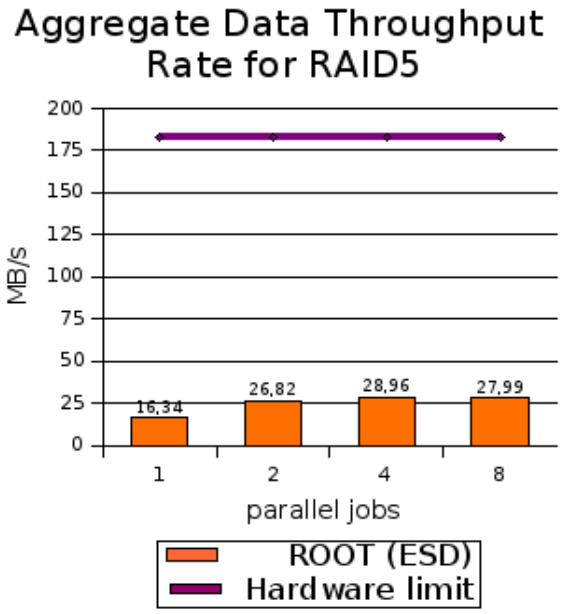
Result2.1.4:

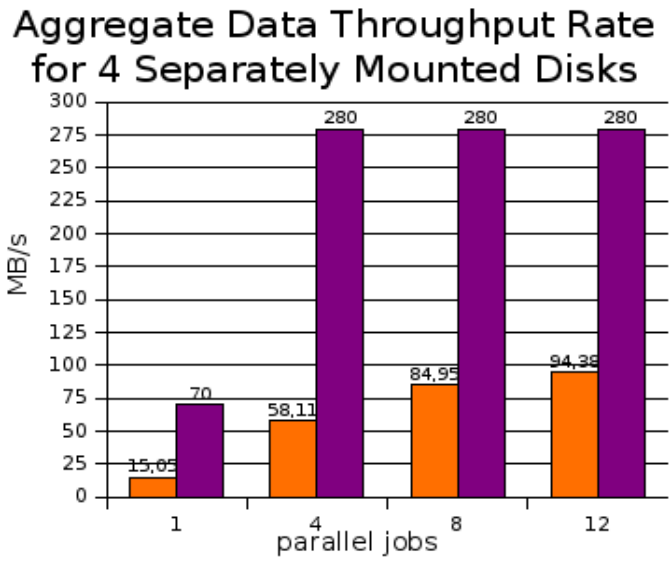


Summary I:

1. In case of using separately mounted disks, saturation of the data throughput rate for parallel ESD jobs starts when 2 jobs are running per disk. Rate value peaks at 96 MB/s for 12 parallel jobs processing ESD files located on 4 separately mounted disks.

file tuning_tests_esd.ods
Object2.1.5: Comparison of aggregate data throughput rates for parallel analysis jobs processing ESD files stored on local RAID5 storage, and on 4 local disks that are mounted separately. RAID5 readahead is increased to hardware maximum 8MB. Hardware limits are measured by hdparm. One analysis job (“train of tasks”) processes 4 ESD files of 20000 events each (~525 MB file size). For the test of 4 separately mounted disks: 1 job analyzes data from 1 disk, 4 jobs – 1 job per disk, 8 jobs – 2 jobs per disk, 12 jobs – 3 jobs per disk.
Result2.1.5:





Summary J:

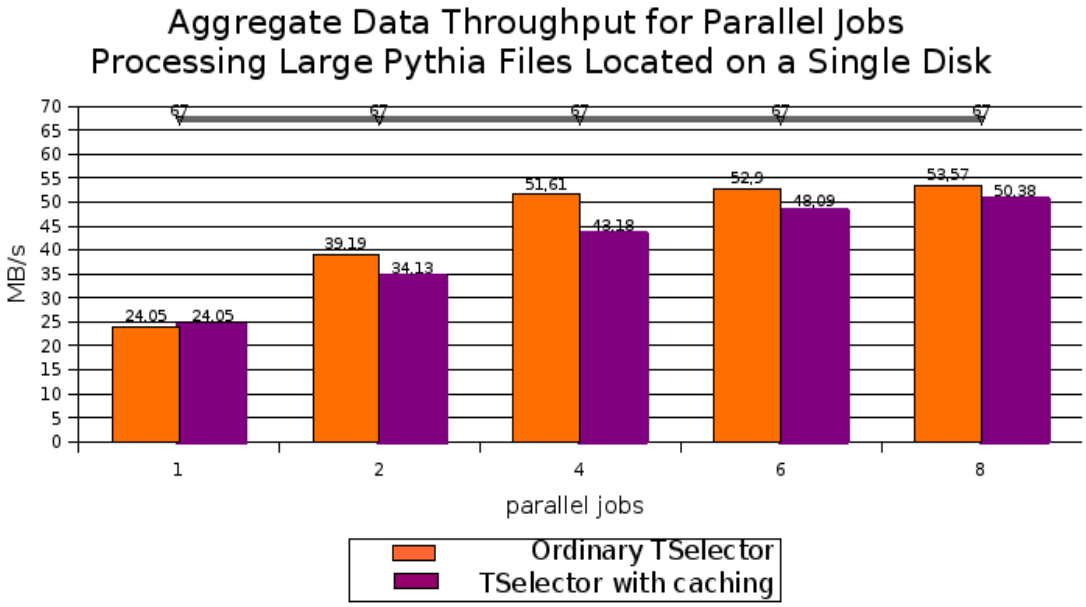
1. The increase of RAID5 readahead to 8MB doesn't affect throughput rate for ESD analysis jobs that read whole events (all branches). The rate values are the same as in 1.6.1
2. Although using 4 separately mounted disks in comparison to using RAID5 improves data throughput rates for parallel analysis jobs, data safety and maintenance efforts for such analysis scheme are called in question.

Test 2.2 Analysis optimization by prior caching of data

file disk_tests.ods

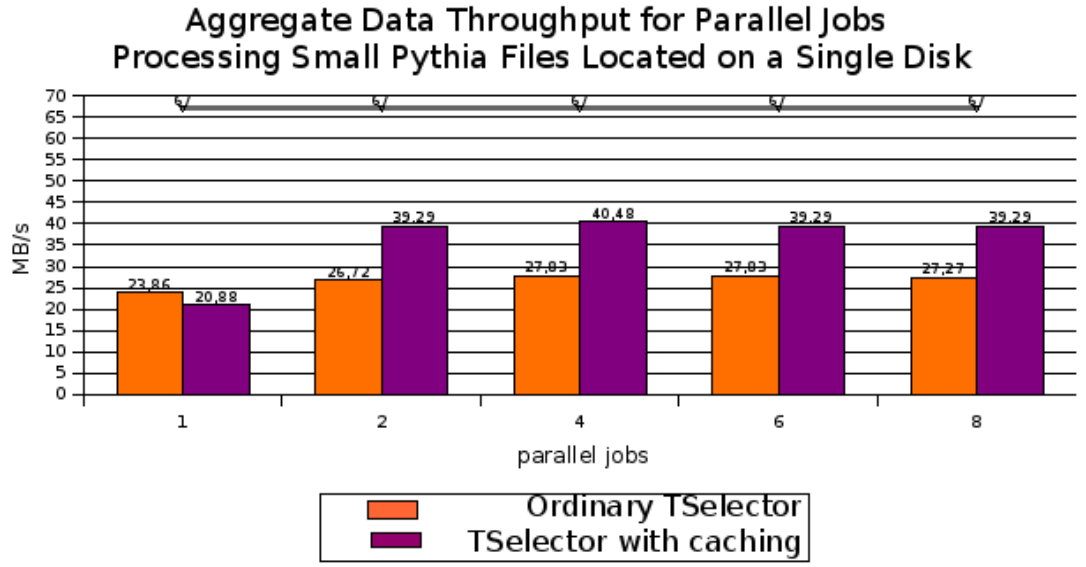
Object2.2.1: Improvement of aggregate data throughput for parallel analysis jobs processing large Pythia files located on a single disk by caching data files beforehand. Data files are cached with C++ function before reading is performed by ROOT inside analysis code. One analysis job processes 1 Pythia file of 30000 events each (~530 MB file size). C++ function reads 1 file with throughput rate of 68 MB/s.

Result2.2.1:



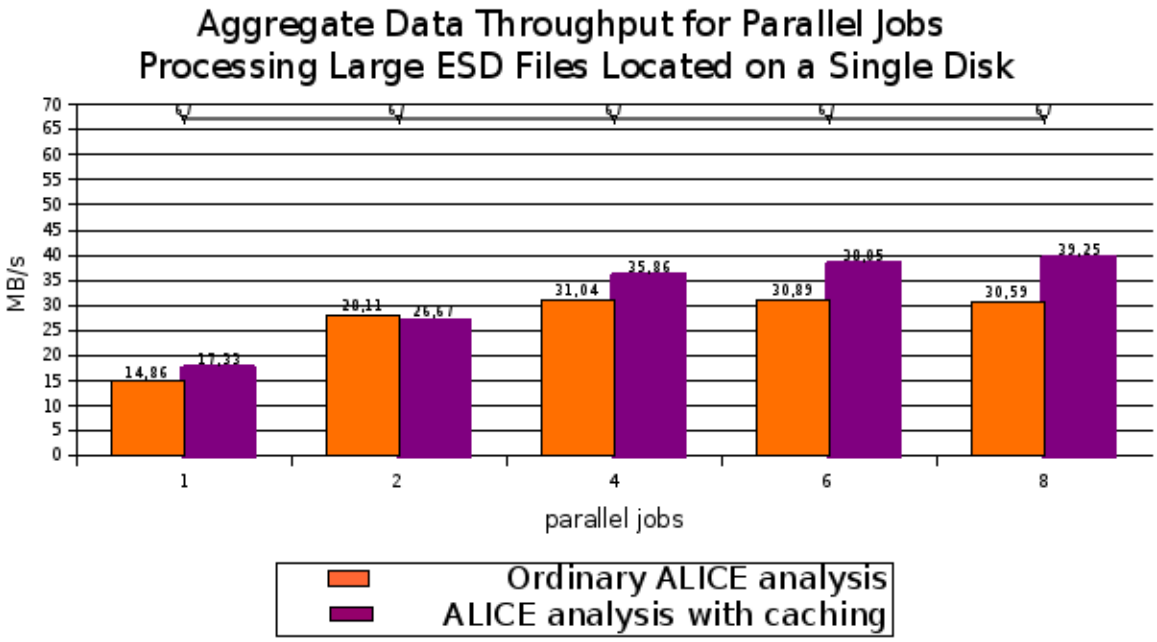
Object2.2.2: Improvement of aggregate data throughput for parallel analysis jobs processing small Pythia files located on a single disk by caching data files beforehand. Data files are cached with C++ function before reading is performed by ROOT inside analysis code. One analysis job processes 100 Pythia files of 190 events each (~3,3 MB file size). To read 1 file with C++ function takes less than 0,1s.

Result2.2.2:



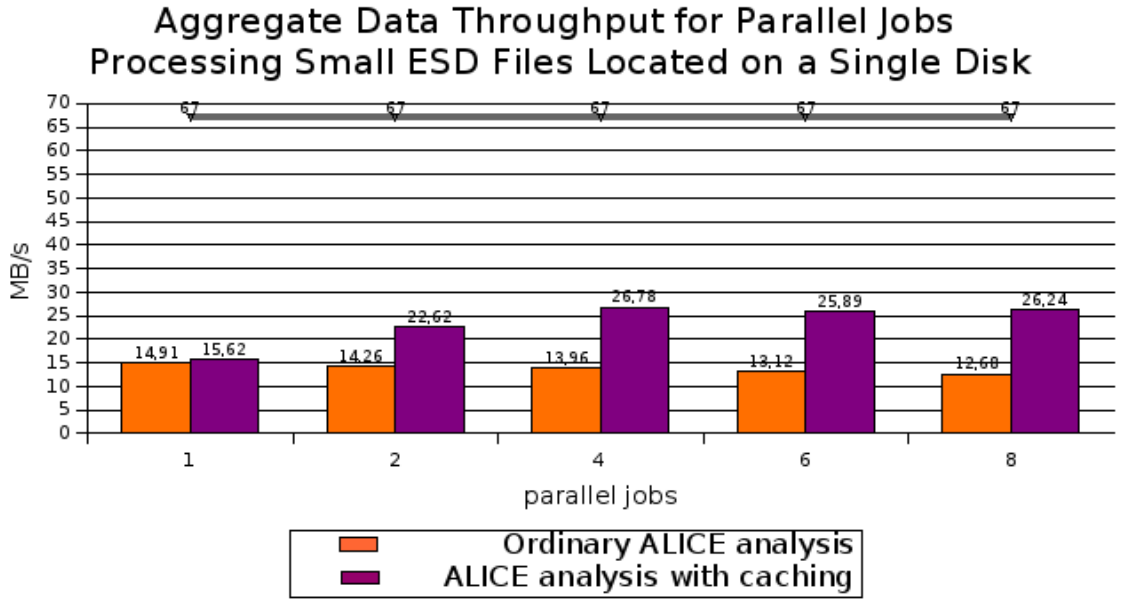
Object2.2.3: Improvement of aggregate data throughput for parallel analysis jobs processing large ESD files located on a single disk by caching data files beforehand. Data files are cached with C++ function before reading is performed by ROOT inside analysis code. One analysis job processes 1 ESD file of 20000 events each (~520 MB file size). C++ function reads 1 file with throughput rate of 61 MB/s.

Result2.2.3:

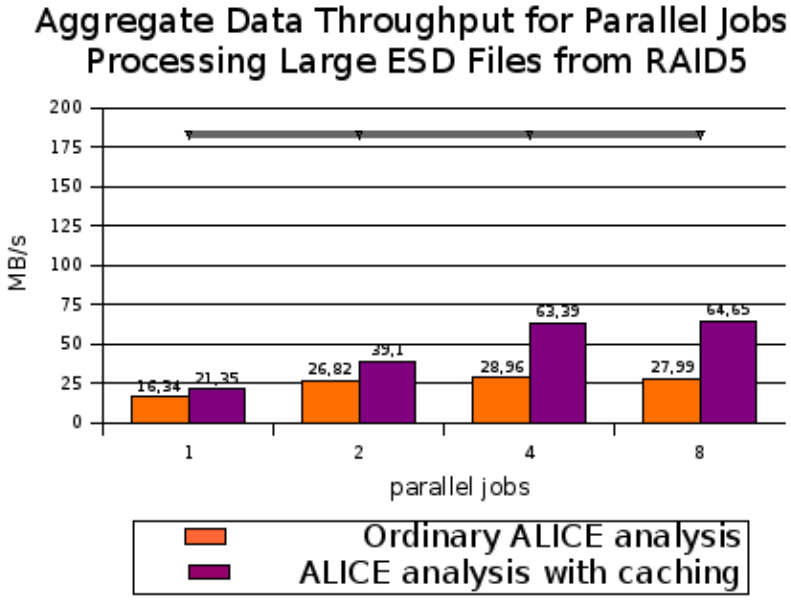


Object2.2.4: Improvement of aggregate data throughput for parallel analysis jobs processing small ESD files located on a single disk by caching data files beforehand. Data files are cached with C++ function before reading is performed by ROOT inside analysis code. One analysis job processes 100 ESD files of 100 events each (~3,3 MB file size). To read 1 file with C++ function takes less than 0,1s.

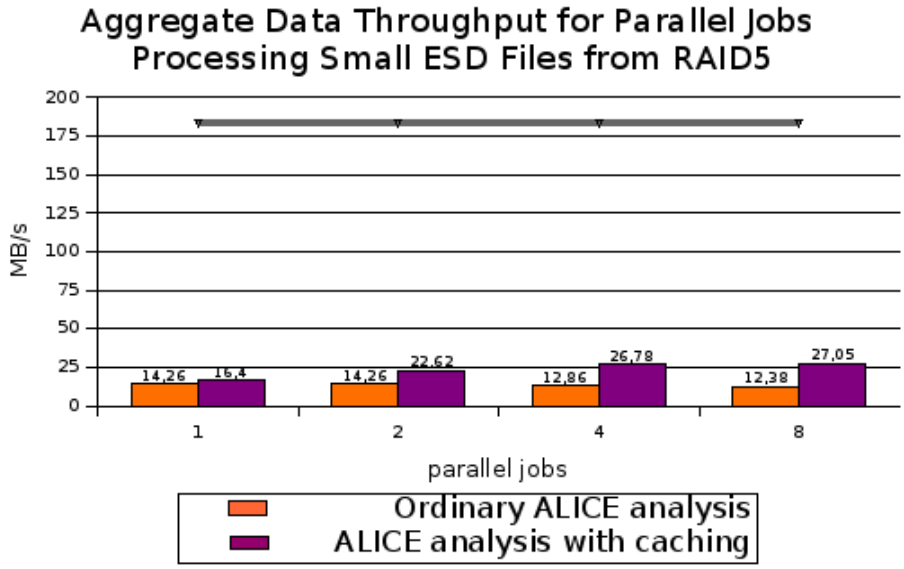
Result2.2.4:



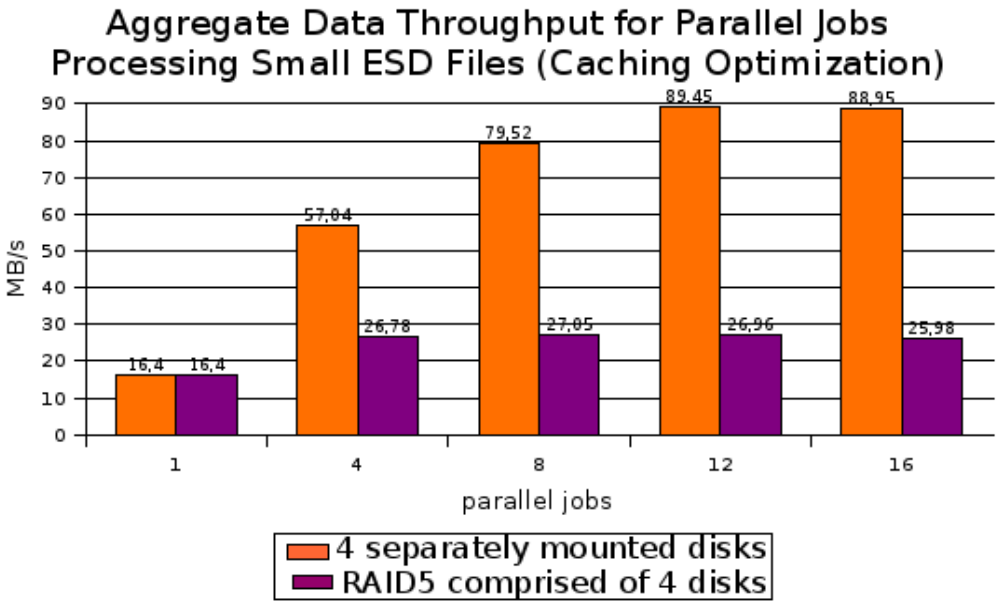
file tuning_tests_esd.ods
Object2.2.5: Improvement of aggregate data throughput for parallel analysis jobs processing large ESD files located on a RAID5 by caching data files beforehand. Data files are cached with C++ function before reading is performed by ROOT inside analysis code. One analysis job processes 4 ESD files of 20000 events each (~523 MB file size). RAID5 readahead is set to hardware maximum 8MB.
Result2.2.5:



file small_esd.ods
Object2.2.6: Improvement of aggregate data throughput for parallel analysis jobs processing small ESD files located on a RAID5 by caching data files beforehand. Data files are cached with C++ function before reading is performed by ROOT inside analysis code. One analysis job processes 100 ESD files of 100 events each (~3.3 MB file size). RAID5 readahead is set to hardware maximum 8MB.
Result2.2.6:



Object2.2.7: Comparison of aggregate data throughput rates for parallel analysis jobs processing ESD files stored on local RAID5 storage, and on 4 local disks that are mounted separately. Data files are cached with C++ function before reading is performed by ROOT inside analysis code. RAID5 readahead is increased to hardware maximum 8MB. One analysis job ("train of tasks") processes 100 ESD files of 100 events each (~3.3 MB file size). For the test of 4 separately mounted disks: 1 job analyzes data from 1 disk, 4 jobs – 1 job per disk, 8 jobs – 2 jobs per disk, 12 jobs – 3 jobs per disk, 16 jobs – 4 jobs per disk.
Result2.2.7:



Summary K:

1. The method of caching data files with C++ function, when ROOT application reads data from memory before analysis starts, improves data throughput rate for parallel jobs processing small ESD and Pythia files on a single local disk because the reason described in summary E is eliminated.
2. The method of caching data files with C++ function, when ROOT application reads data from memory before analysis starts, does not improve data throughput rate for parallel jobs processing large Pythia files, but does for large ESD files on a single local disk.
3. The method of caching data files with C++ function, when ROOT application reads data from memory before analysis starts, significantly improves data throughput rate for parallel jobs processing large and small ESD files from RAID5. Although for large ESD files rate is twice higher.
4. When running parallel analysis jobs optimized with caching method on source small ESD data files, aggregate data throughput rate for 4 separately mounted disks is significantly higher than for RAID5 – by a factor of 3,33 for 12 jobs.

Overall summary:

1. The usage of the maximum possible RAID5 readahead increases maximum possible data throughput rate for hardware from 70 MB/s (default readahead) to 180 MB/s.
2. When using 4 separately mounted SATA disks the aggregate data throughput for all disks is the sum of each disk's throughput capability. In this case, 70 MB/s – for one disk, 280 – for 4 disks together.
3. In case of using separately mounted disks, saturation of the data throughput rate for parallel ESD jobs starts when 2 jobs are running per disk. Rate value peaks at 96 MB/s for 12 parallel jobs processing large ESD files located on 4 separately mounted disks.
4. The increase of RAID5 readahead to 8MB doesn't affect throughput rate for ESD analysis jobs that read whole events (all branches). Possible reason: too many branches.
5. Although using 4 separately mounted disks in comparison to using RAID5 improves data throughput rates for parallel analysis jobs, data safety and maintenance efforts for such analysis scheme are called in question.
6. The method of caching data files with C++ function, when ROOT application reads data from memory before analysis starts, improves data throughput rate for parallel jobs processing small ESD and Pythia files on a single local disk. This method also significantly improves data throughput rate for parallel jobs processing large and small ESD files on RAID5. Although for large ESD files rate is twice higher.