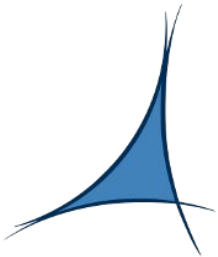
Digi - Reco Profiling

Alejandro Alonso [PIC / UAB]

Tutor:
Josep Flix

Related to: This presentation



PIC
port d'informació
científica

UAB
Universitat Autònoma
de Barcelona

IFAE
25
Institut de Física
d'Altes Energies

EXCELENCIA
SEVERO
OCHOA

BIST
Barcelona Institute of
Science and Technology

GOBIERNO
DE ESPAÑA
MINISTERIO
DE CIENCIA
E INNOVACIÓN

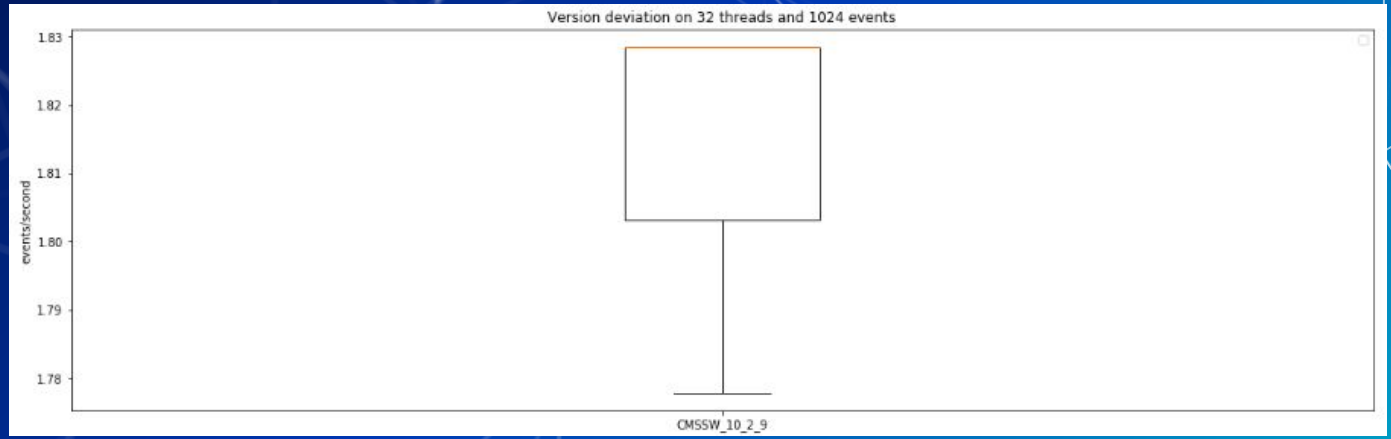
Ciemat
Centro de Investigaciones
Energéticas, Medioambientales
y Tecnológicas

Environment and tests

- Same environment as cms - sim profiling
- Test are done with a local file
- There are also test done with remote files

Check execution variance (5 executions)

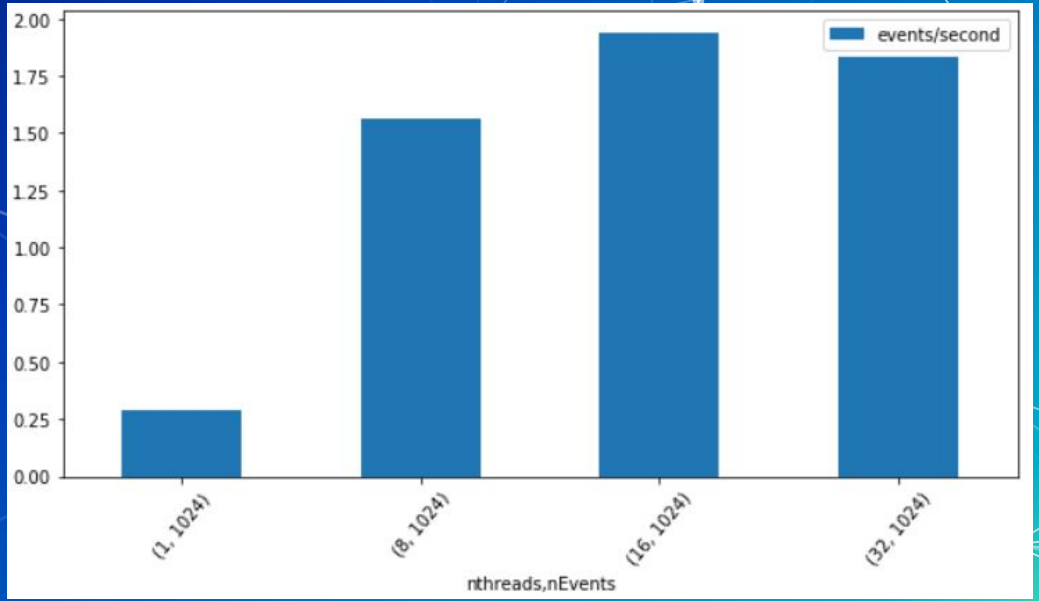
 # Almost no variance between same executions 0,05 events/second
 # Safe to use one execution to profile



Number of threads analysis

Speedup	1024 events
8 threads	5.41
16 threads	6.73
32 threads	6.34

 # Performance = events/second
 # 1024 events on each experiment → gen - sim was stable with 1024
 # Multithreading is totally ineficient
 # hyperthreading is worst than simple threading (overhead)



Resource usage analysis

Mainly CPU usage analysis and it’s different parts

CPU efficiency

 # We can see 4 phases

Searching files → 2 min

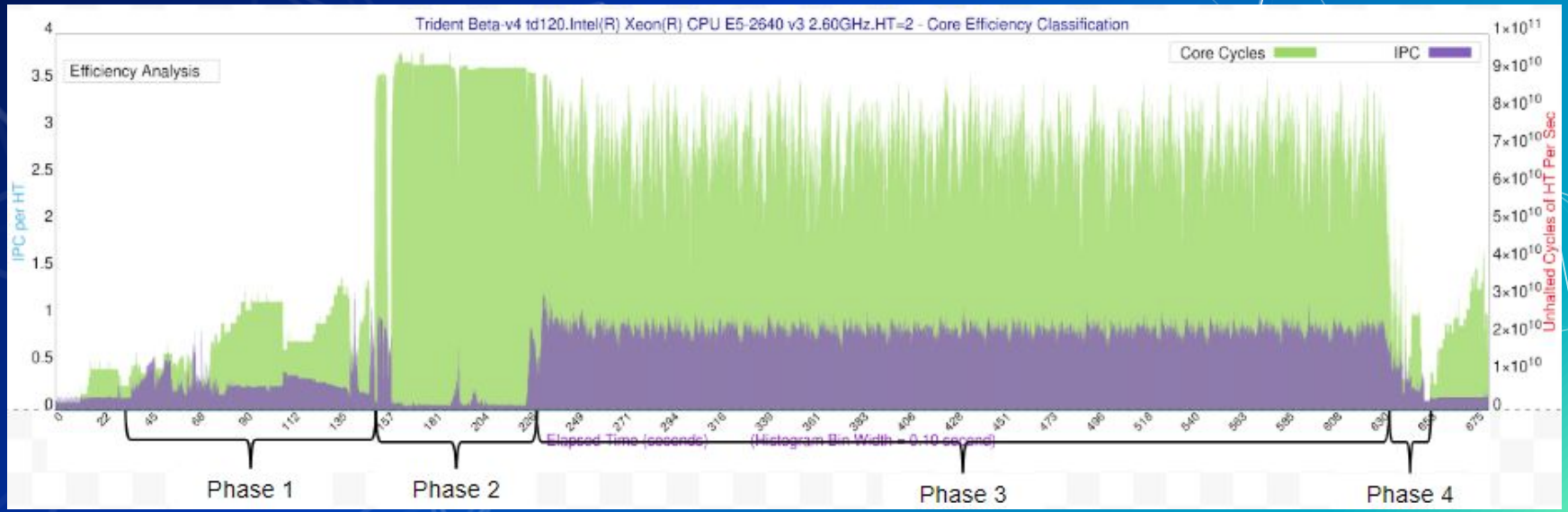
Reading files and loading .root into memory → 1 min 30 sec

Event compute → 8 min

Termination → 15 sec

 # A lot of spikes on event compute → interruption of CPU time

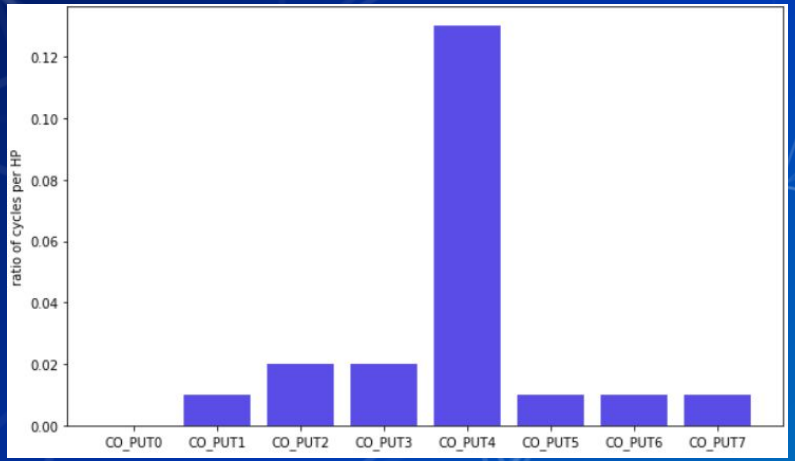
 # Due to slow file lectures or memory access ?



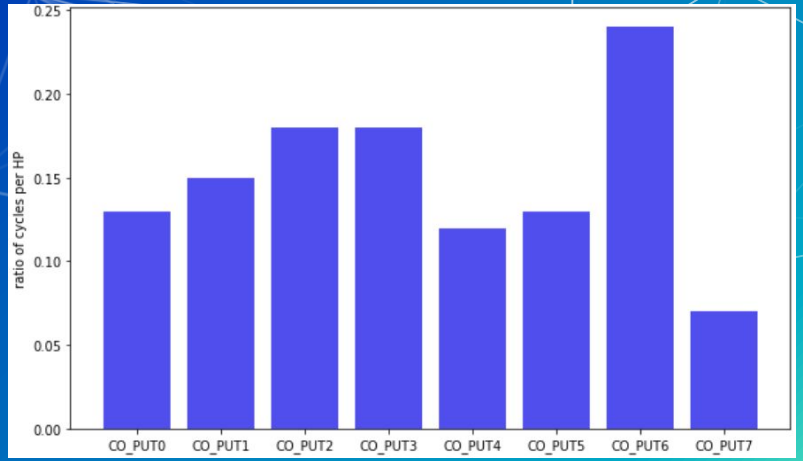
Processor pipeline port usage

 # Haswell pipeline architecture
 # Ports 0,1,5,6 compute, 2,3,4,7 memory
*# Maximum usage on port 6 (integer port) with 25% * 2 = 50% → no saturated resources*

Phase 2



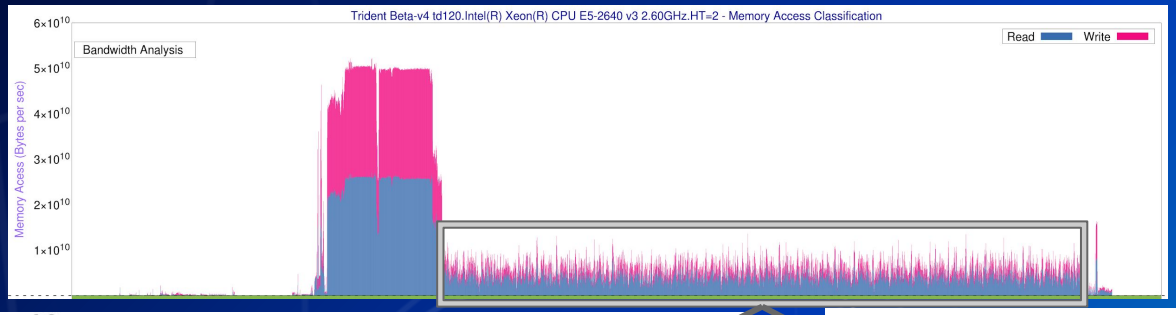
Phase 3



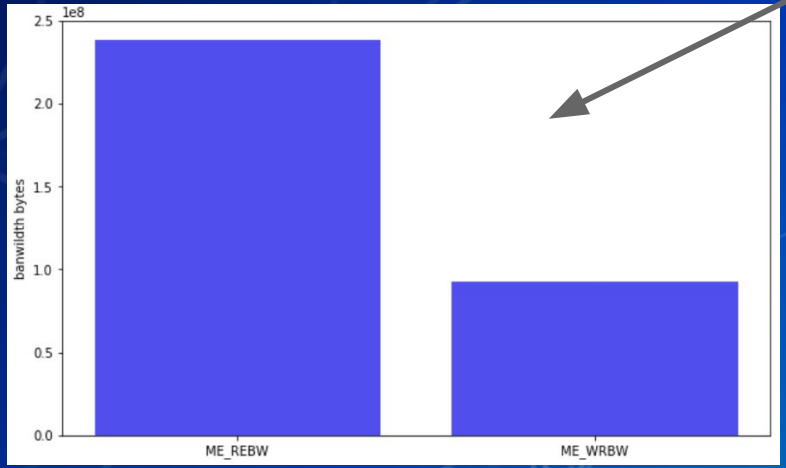
Memory hierarchy and disk analysis

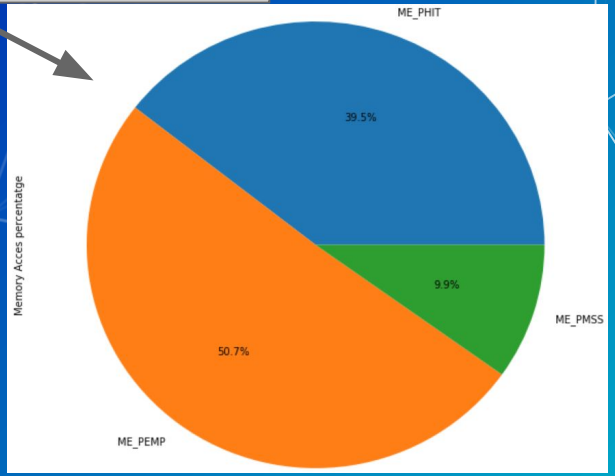
Different levels of memory cache, DRAM, disk

DRAM Bytes transferred and memory access type



 # Memory access still very low compared to machine bandwidth
 # Memory access type is better than cms - sim
 # Bandwidth isn't saturated during phase 2 where there is more memory usage
 # Memory affects performance but isn't that big of a problem

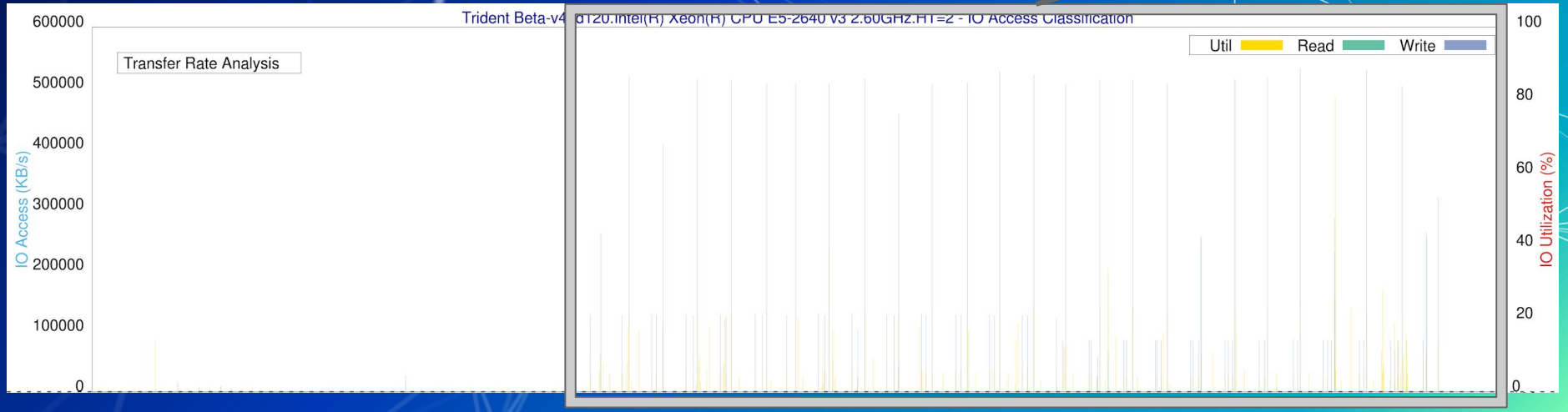




Disk access

 # Quite number of accesses during event compute
 # Almost all accesses give a 90% of usage
 # 90% of usage is enough to say that a resource is saturated, even though accesses seem so distant
 # This explains the spikes of interruption on cpu time

Phase 3

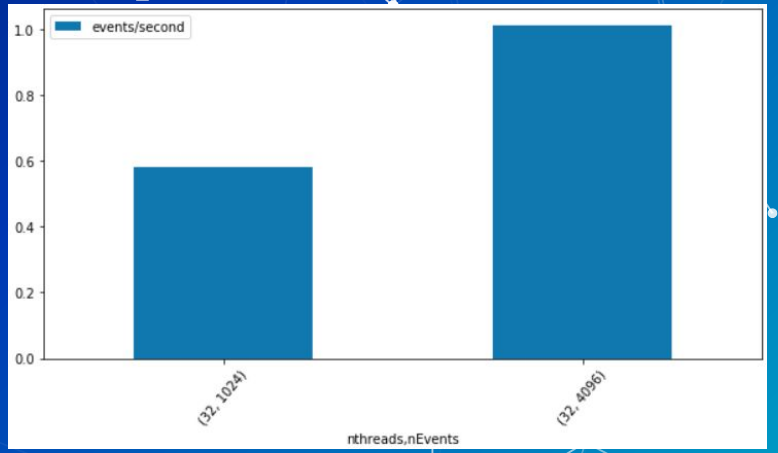


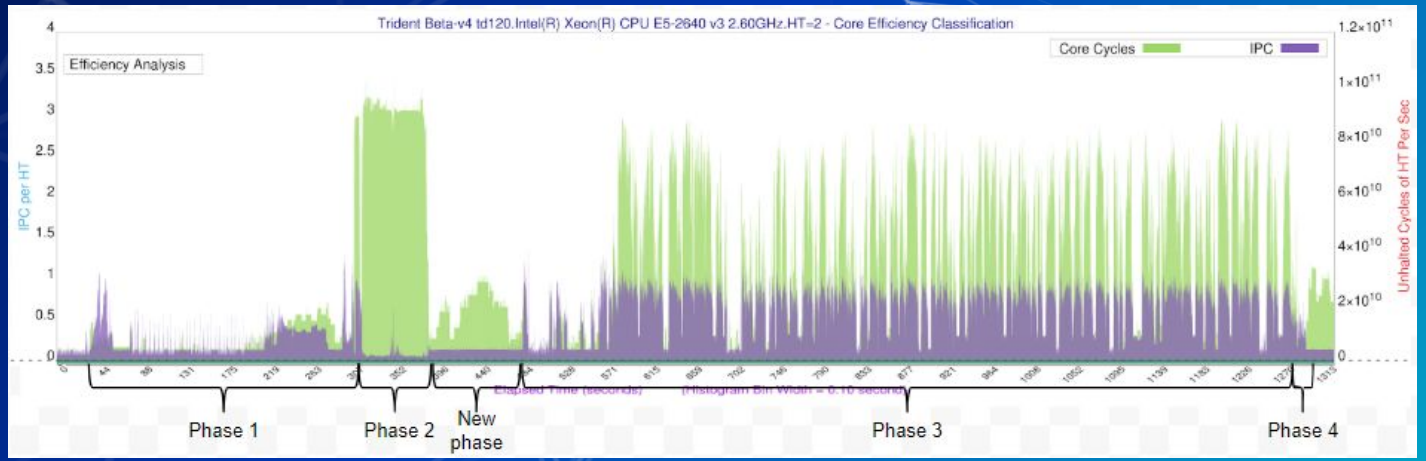
Comparison between remote and local files

CPU and network comparison

Remote files CPU efficiency

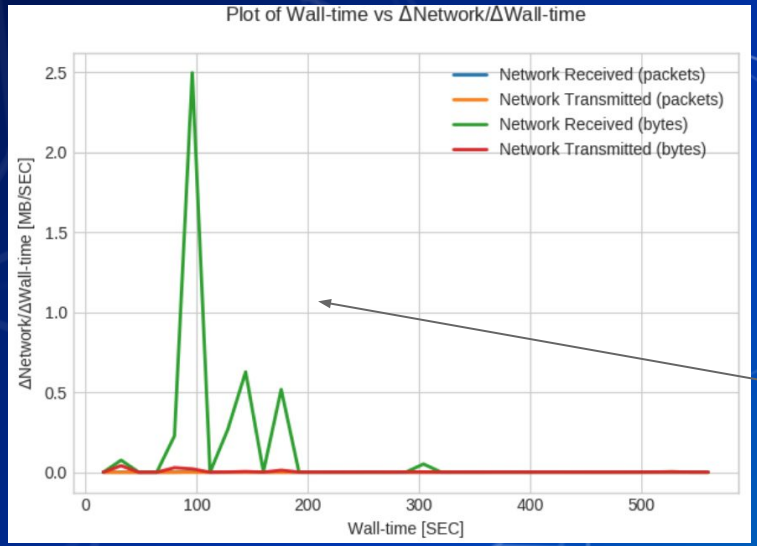
 # New phase between 2 and 3, events must wait for information of the remote files to reach
 # Phase 1 takes longer due to the search of remote files
 # Even more spikes during the event compute, probably due to remote data dependency
 # The amount of events/second needed is higher, this is due to the sequential part being even bigger
 # Multithreading inefficiency must be worst in case of remote files → multithreading is mostly useless
 # Overall performance is 70% worst with remote files



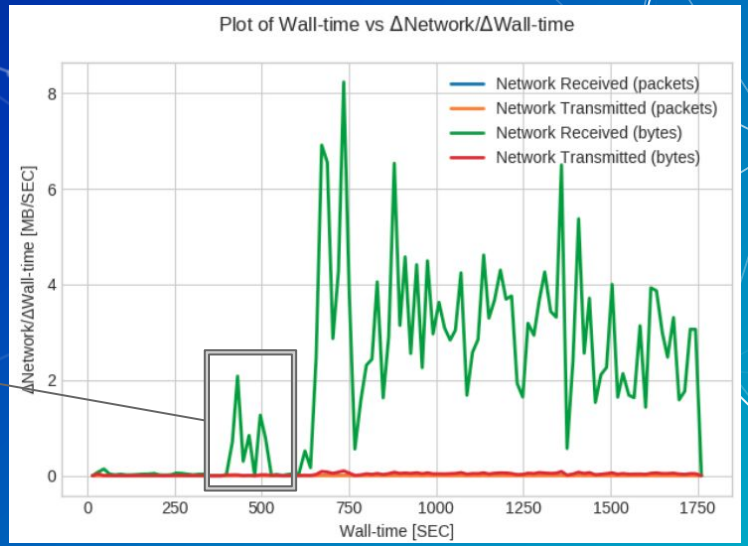


Network comparison

 # Network graphs shows how file streams input data during the event compute
 # This behaviour explains the spikes during the event compute



Local files



Remote files

Conclusions

- Scalability with the number of threads it’s pretty bad
- The number of events needed to make multithreading relevant is enormous due to the sequential part being too long
- Memory accesses are not a big problem
- Disk access is saturated and might clog the disk access while using multithreading
- Remote files create a data dependency that doesn’t allow the program to go faster than the speed of internet data

THANKS!