Studying ROOT I/O performance with PROOF-Lite

C. Aguado-Sanchez, J. Blomer, P. Buncic, I. Charalampidis,
G. Ganis, M. Nabozny, F. Rademakers
CERN

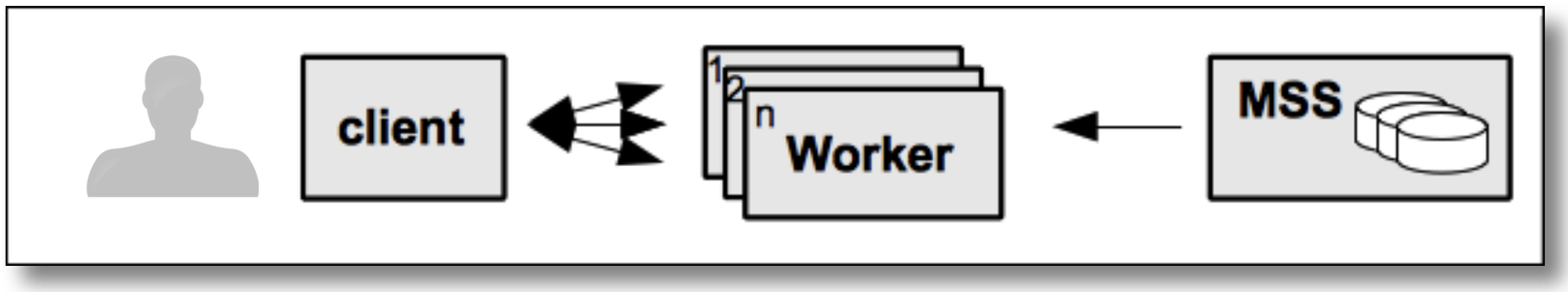
CHEP2010, Taipei, Taiwan, October 2010

Data Analysis needs I/O

- HEP End-User analysis profits from the large increase in CPU (simulations, fits, ...) but for data mining I/O is important
  - 10 TB in ~1 day implies ~200 MB/s
- Multi-core exploitation requires to run in parallel
  - Multi-process or multi-thread
- Needs to evaluate I/O performance in concurrent access scenarios

PROOF-Lite

- PROOF: multi-process parallelism in ROOT
- PROOF-Lite: 0-config version optimized for multi-cores



- Multi-process represents also the case of many concurrent batch jobs on a given worker node

I/O Rate Scaling

- Rate R for N_p processes

$$R = \frac{R_1 \cdot N_p}{1 + \frac{R_1}{R_{IO}} \cdot (N_p - 1)}$$

- where

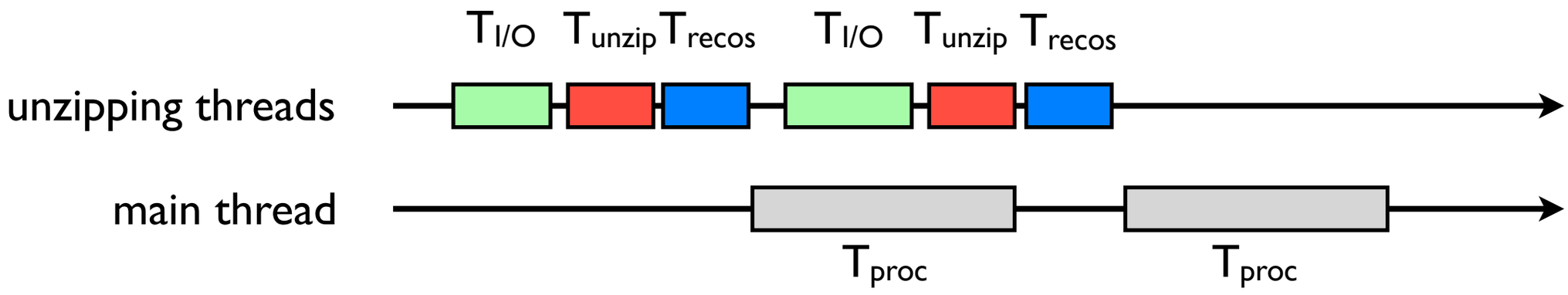
$$\frac{1}{R_1} = \frac{1}{R_{IO}} + \frac{1}{R_{dec}} + \frac{1}{R_{proc}}$$

$R_{I/O}$ = pure I/O rate
R_{dec} = effective decompression rate
R_{proc} = processing rate

- Saturation term small for
  - Large I/O bandwidth ($R_1 << R_{I/O}$)
  - Small I/O weight ($R_{proc} << R_{I/O}$)
- $R \rightarrow R_{I/O}$ for large N_p

Decompression

- Decompression requires a fair amount of CPU (~20÷30 MB/s effective rate)
- Network bandwidths and amount of data still justifies compression (factors ≥3 in LHC experiments)
- Pure CPU task: ideally use free cores to help



- $T_{I/O}$ limited by I/O bandwidth

$T_{I/O}$ = time to read
T_{unzip} = time to unzip
T_{recos} = time to rebuild entry structures
T_{dec} = T_{unzip} + T_{recos}
T_{proc} = time to process the entry

Test machines

- 24 core HP DL580, 48 GB RAM, 1 GB/s NIC
  - 4 SAS disks 74 GB 10k RPMs, RAID0
  - 2 SSD disks 160 GB Intel X25-M, RAID0
  - 1 SSD disk 160 GB Intel X25-M

Courtesy of MultiCore CERN R&D

- 8 core, 16 GB RAM, 1 GB/s NIC
  - 24 HDD, different RAID partitions

Courtesy of ATLAS Univ. Wisconsin

- SLC5 on both machines

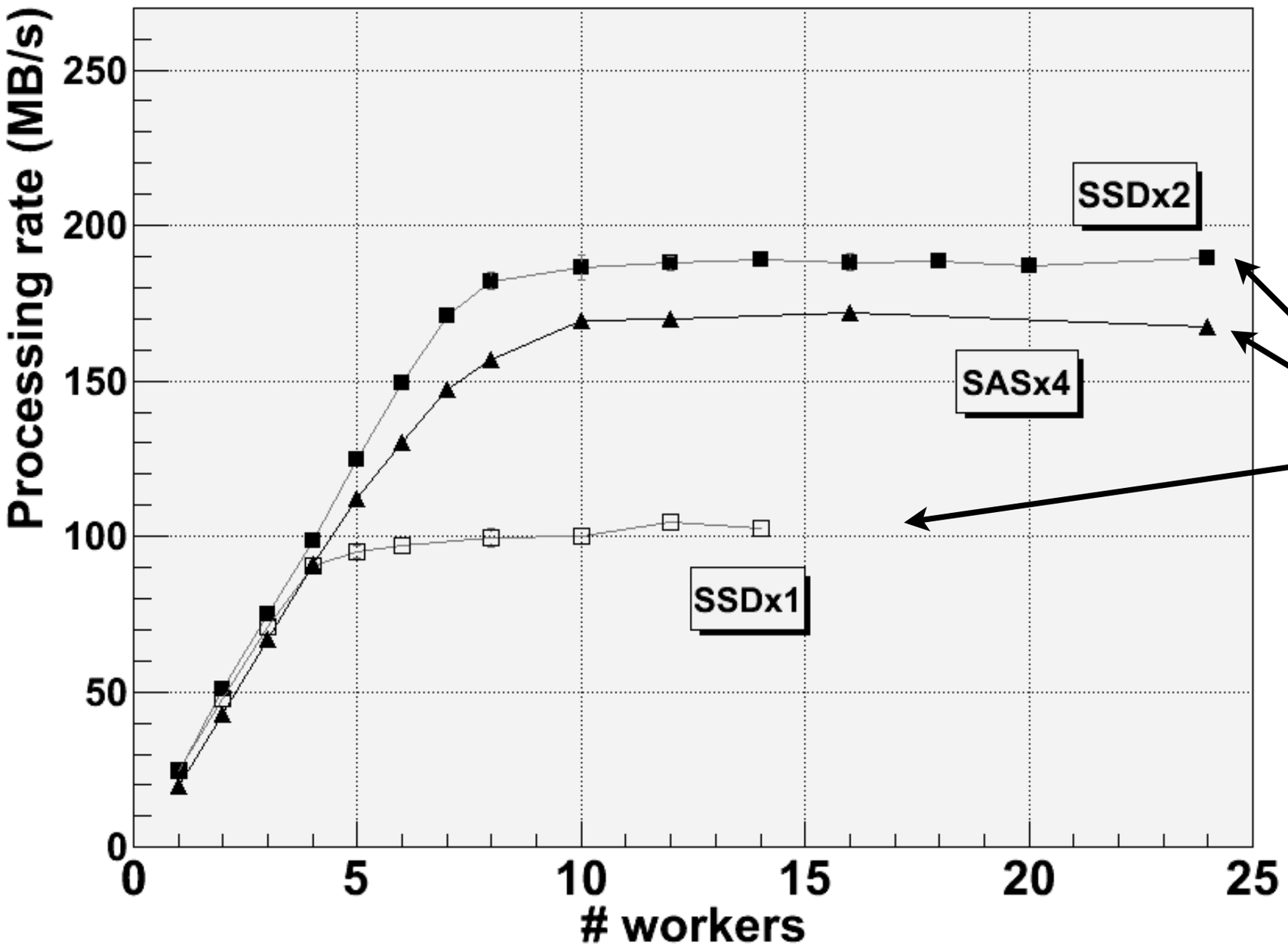
ProofBench tests

- Based on $ROOTSYS/test/Event.h
- 150 files, 11 GB in total, basket size 32K
- Simple analysis reading ~75% of the event and making some distribution (tutorials/proof/ProofEventProc.C)
- 4 runs per point
- System cache cleared after each run with

```
posix_fadvise(fd, 0, 0, POSIX_FADV_DONTNEED)
```

- Local cache (TTreeCache) enabled

Typical disk result

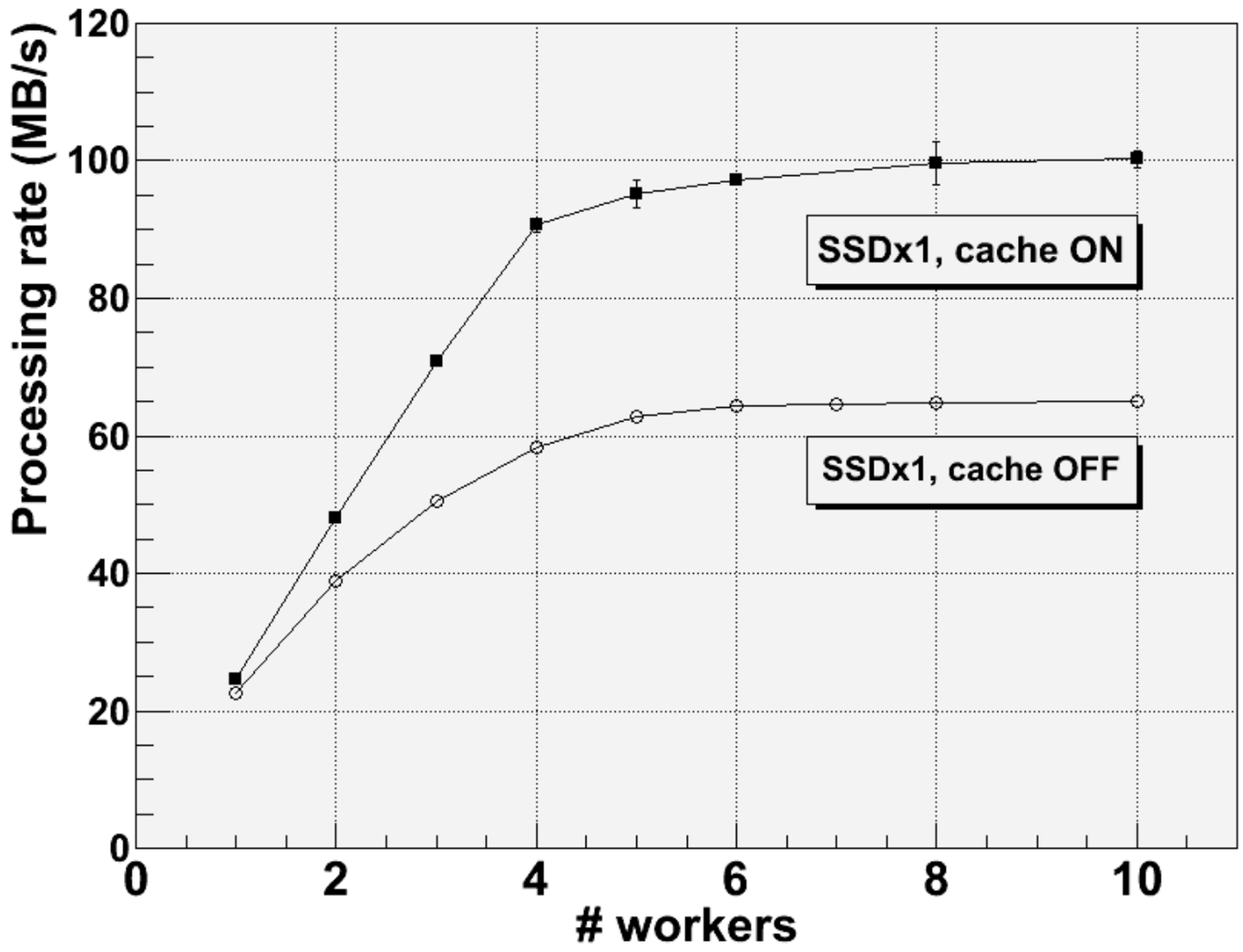


Device bandwidth

About file layout

- File layout optimization has an important role in overall performances
- High-fragmentation may introduce large inefficiencies
- See P. Canal talk in the “Software Engineering, Data Stores and Database” track

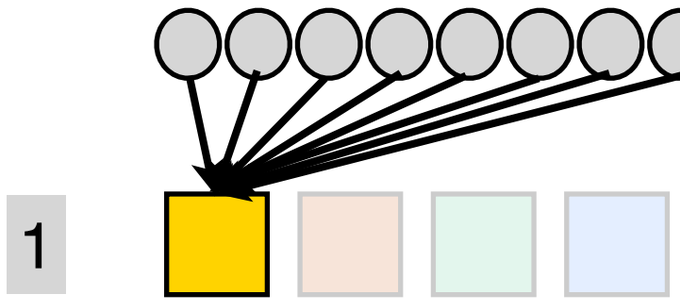
TTreeCache for local files



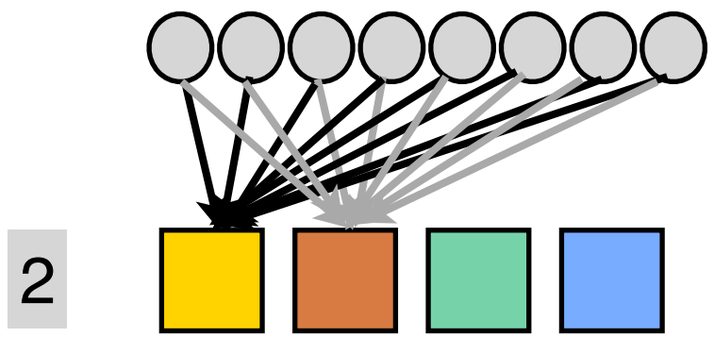
Reduces the number of accesses to disks increasing the efficiency of device sharing

Using disk partition information for non-RAID

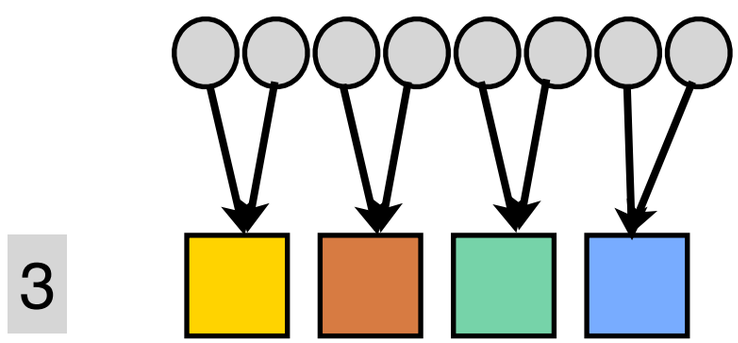
- 8 core UW machine, 4 disks, 8 processes



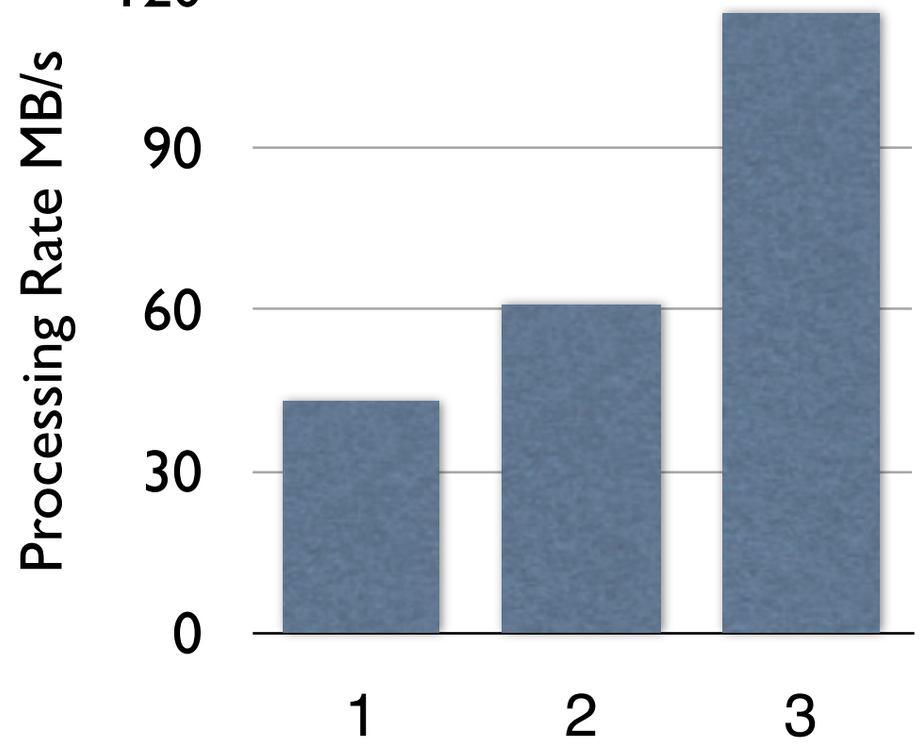
1 disk



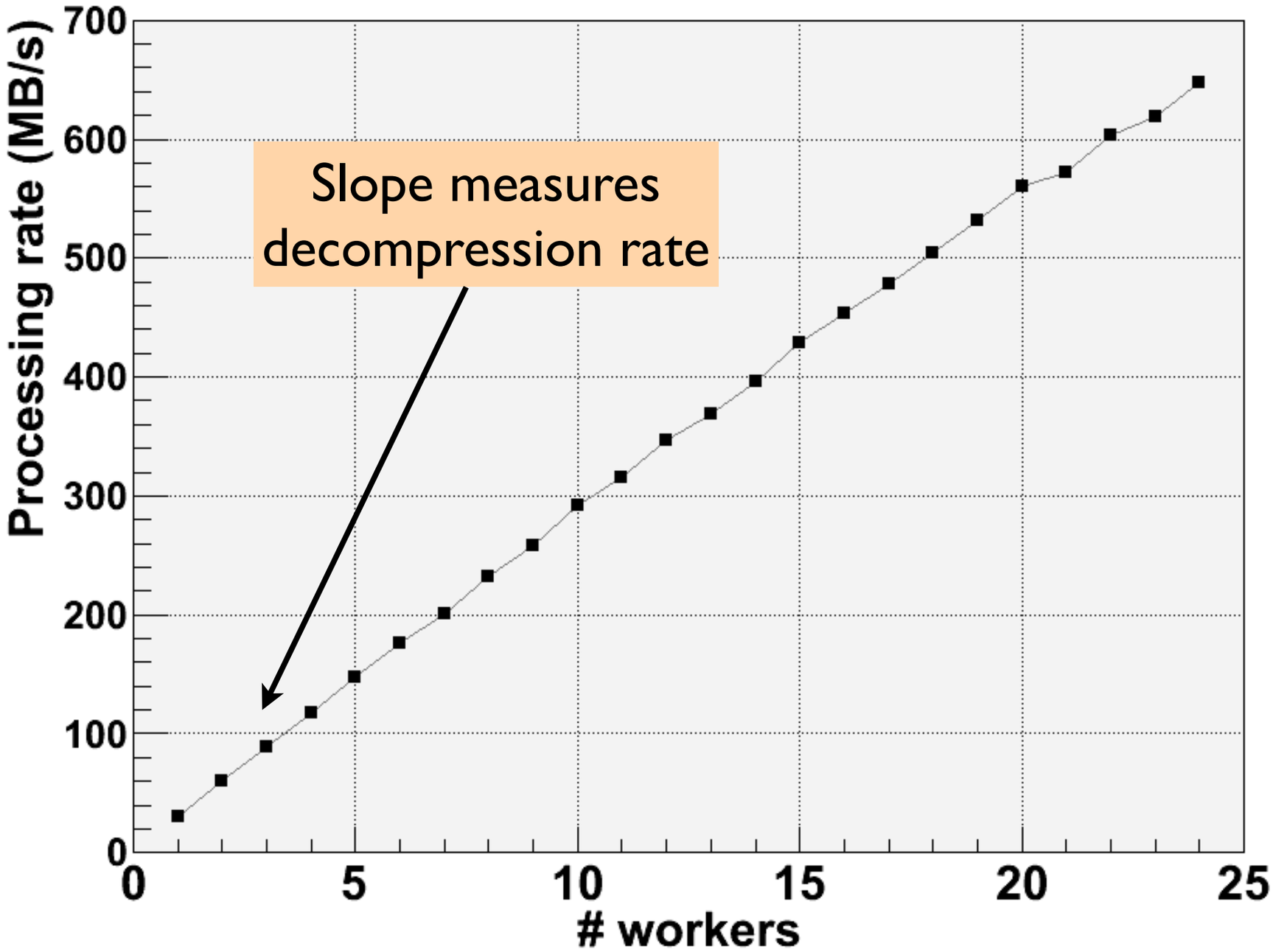
4 disks, sorted
no partition info
(worse case)



4 disks with
partition info

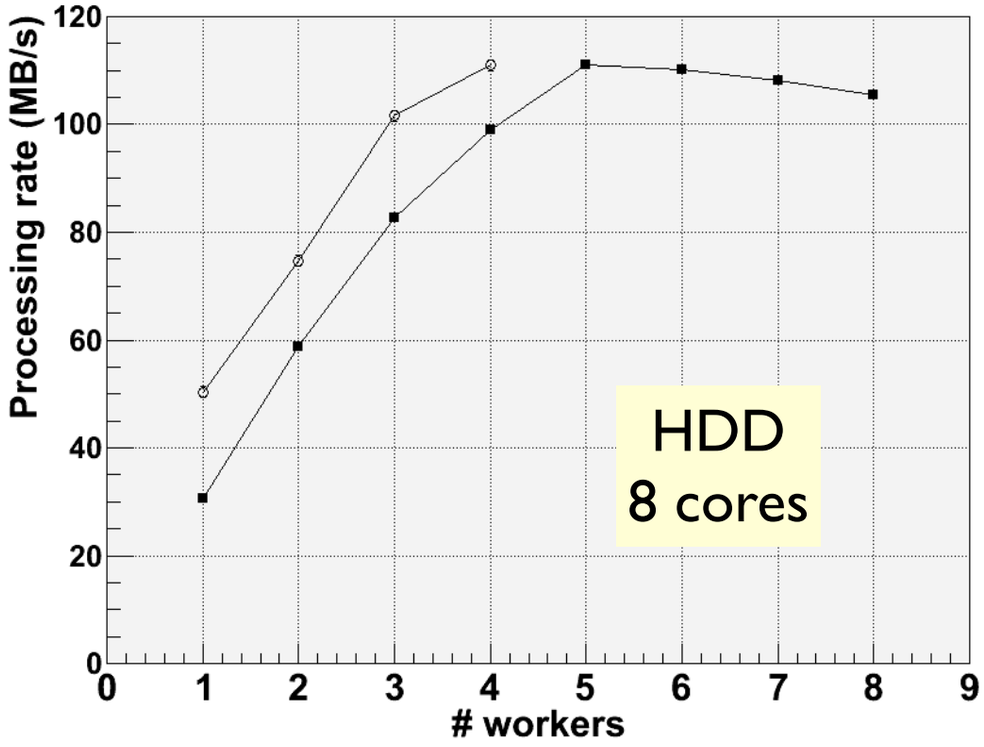


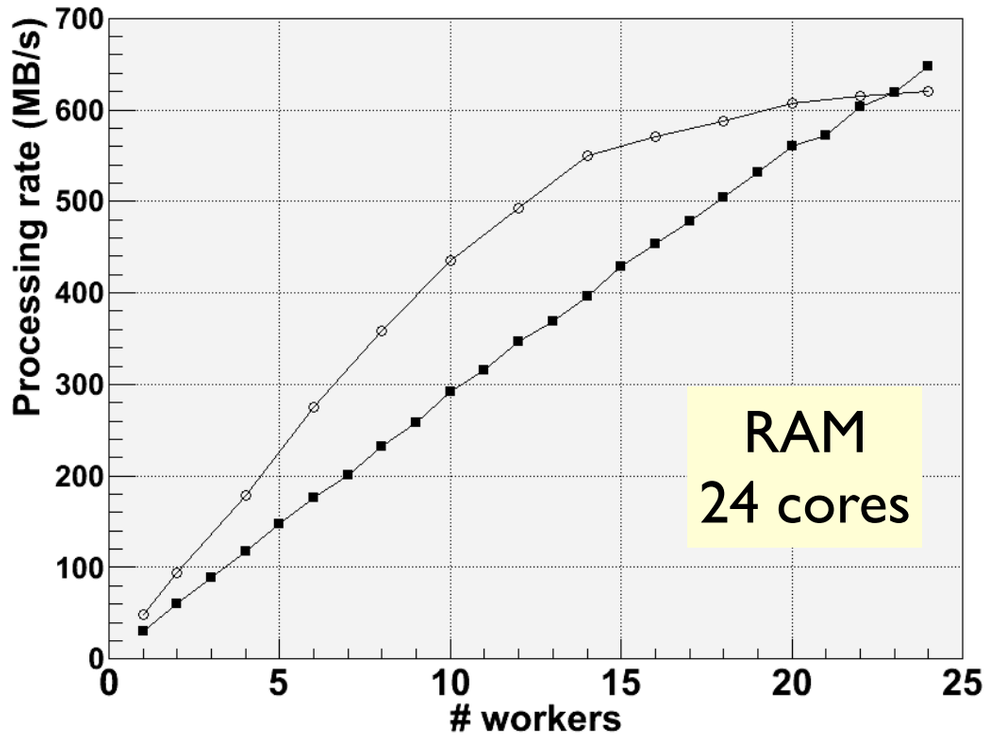
Reading from RAM



Parallel unzipping
Proof of concept

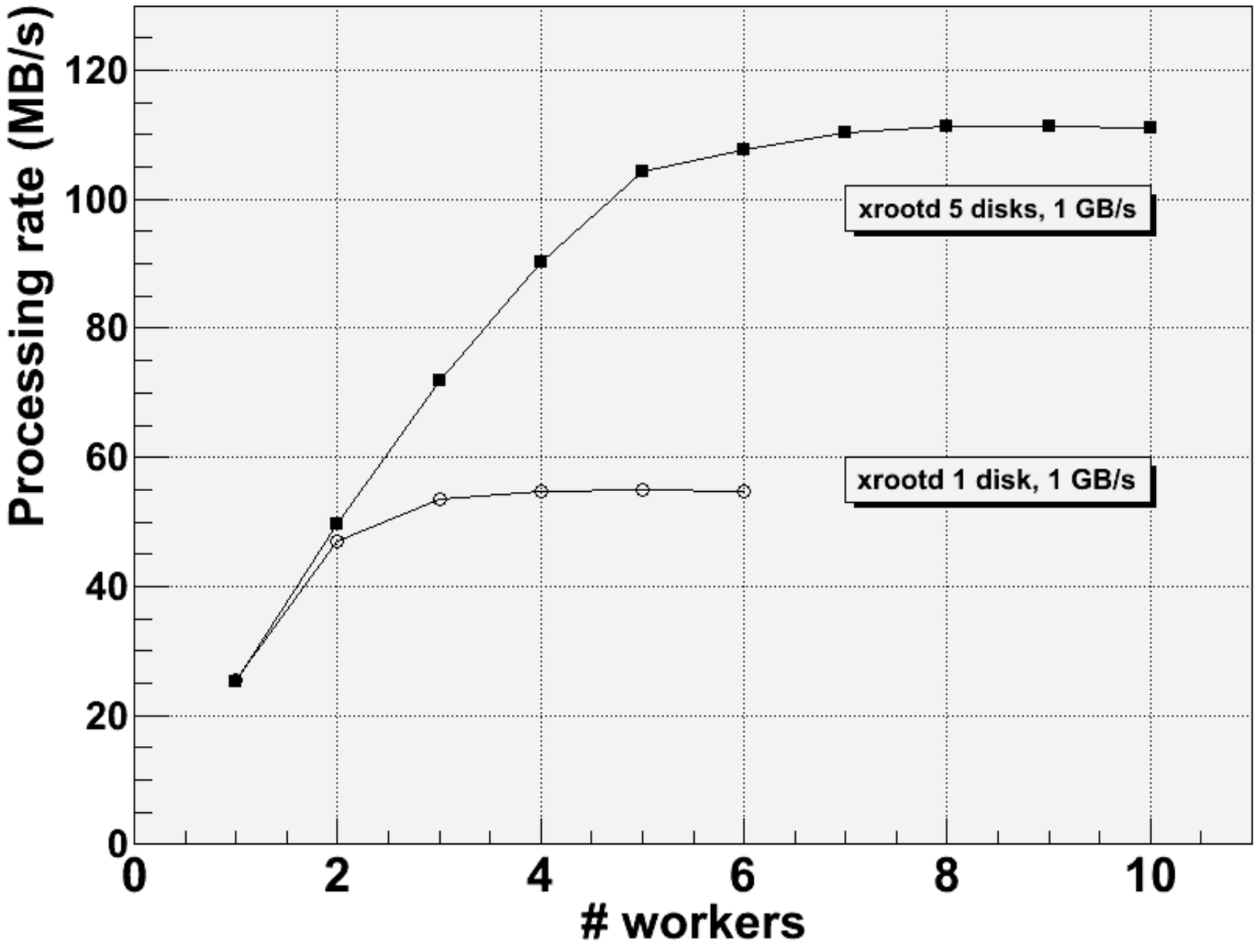
- Prototype implementation with 2 unzipping threads





- Not effect yet seen on files with many branches and small baskets (e.g. ATLAS/CMS analysis files). More detailed studies and algorithm optimizations needed

Reading from network



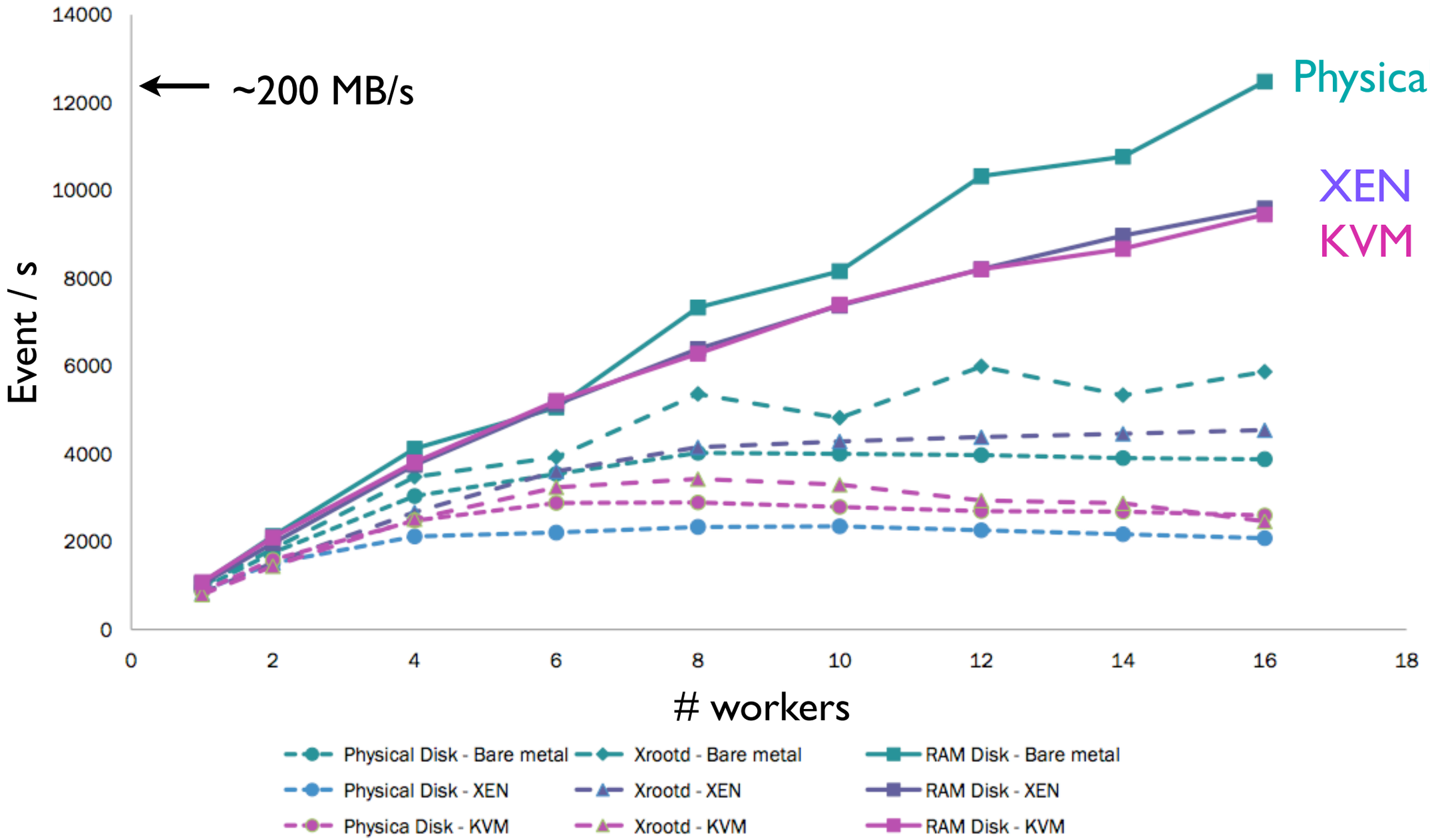
Network limited

Disk limited

Virtual Machine Tests

- VM facilitates the deployment of experiment software on a heterogeneous environment
- The cost to pay is usually a performance penalty for I/O. These tests aimed to quantify the I/O performance penalties on PROOF-Lite due to virtualization in a real use-case
- Test configuration
  - CernVM on KVM, XEN
  - PyROOT-based, I/O bound, ATLAS analysis, with PROOF-Lite
    - ROOT 5.26/00b, Python 2.5.4p2
- I/O devices used: RAM, Physical Disk, Network (Xrootd)

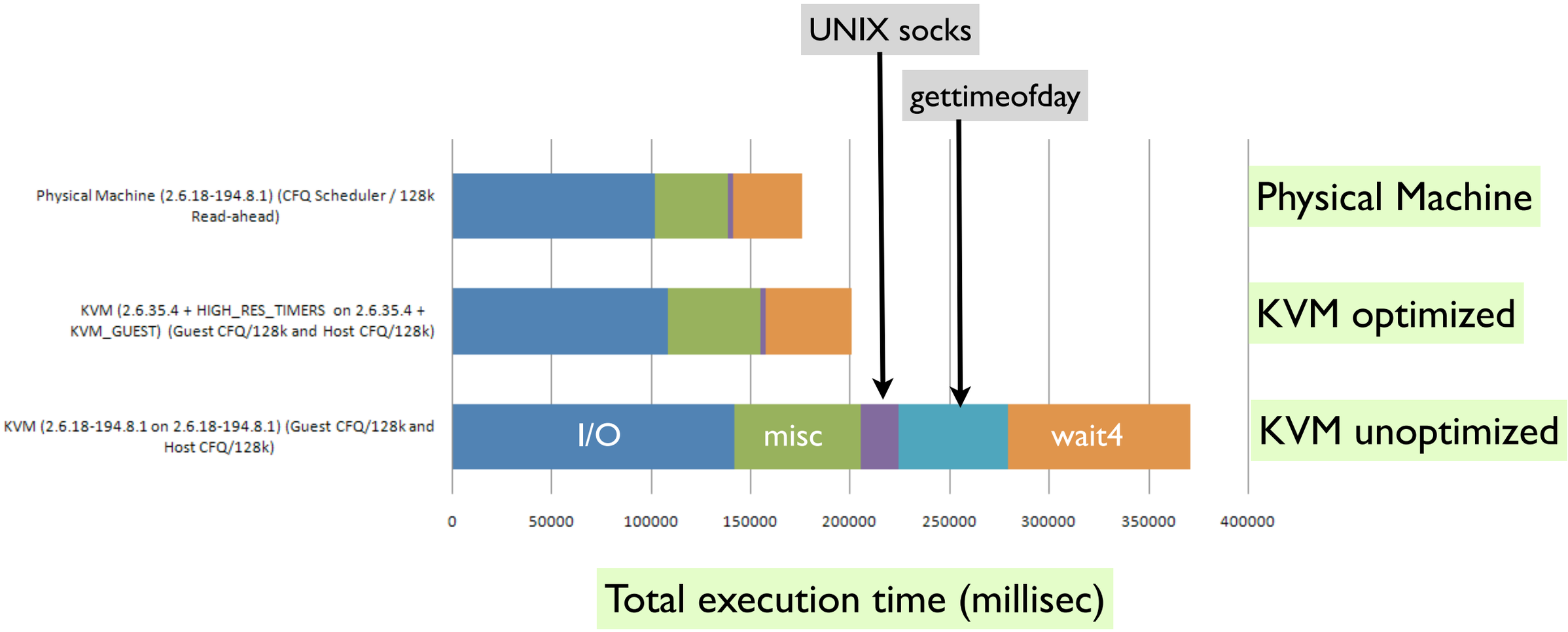
VM preliminary results



VM preliminary results

- Overall penalty of 10%-30%, using vanilla kernels (2.6.18-194.8.1 for host and guest). The penalty is found to increase with N_{CPU}.
- In-depth analysis with Strace and SystemTAP shows that most of the penalty comes from sub-optimal performance of some system calls in a VM environment:
  - UNIX sockets used for communication between PROOF-Lite processes
  - Some time-related system calls (e.g. *gettimeofday* and *wait4*) used mostly by the ROOT event loop
- For KVM, the penalty can be significantly reduced by using the TSC clock source and by optimizing the kernel with CONFIG_KVM_GUEST and CONFIG_HIGH_RES_TIMERS.

KVM In-depth Analysis



CFQ scheduling /128K read-ahead (host + guest) enables a sort of 'memory' effect speeding up I/O in the VM (can be even faster than in the host). Repeated runs on the same dataset could benefit from this.

Similar Studies

- Poster by S.Panitkin et al about a similar study with ATLAS I/O intensive analysis
- Results in fair agreement
- Larger penalty in VM reported there due to a bug in PROOF-Lite startup (affecting only VM runs) which was solved in the study presented here

Summary

- PROOF-Lite is an effective way to exploit multi-cores using multi-processing. It also allows to study ROOT I/O performance in *concurrent access scenarios*
- Ultimate aggregate I/O rate approaches the device I/O bandwidth, provided the file structure is optimized
- Network I/O requires adequate servers behind to saturate the available bandwidth
- Virtualization I/O overhead can be mostly reduced with proper kernel optimizations

Thanks!

Questions?