Following up on I/O Performance On Large Data Files

Outline

Last Time:

- We considered various read routines and different file storage schemes.
- We found that the “file per disk” read did increase net read speeds by approximately 5 times vs hadoop based reads in conservative but loose estimates

This Time:

- We run the disk per read test on 8 different machines.
- Additionally, we run one “longrun” which repeated the test 100 times on another machine.
- No attempt was made to control the computing environment, these are live machines which are running analysis for the OSG.

Read Specifications

Reader Program Psuedocode:

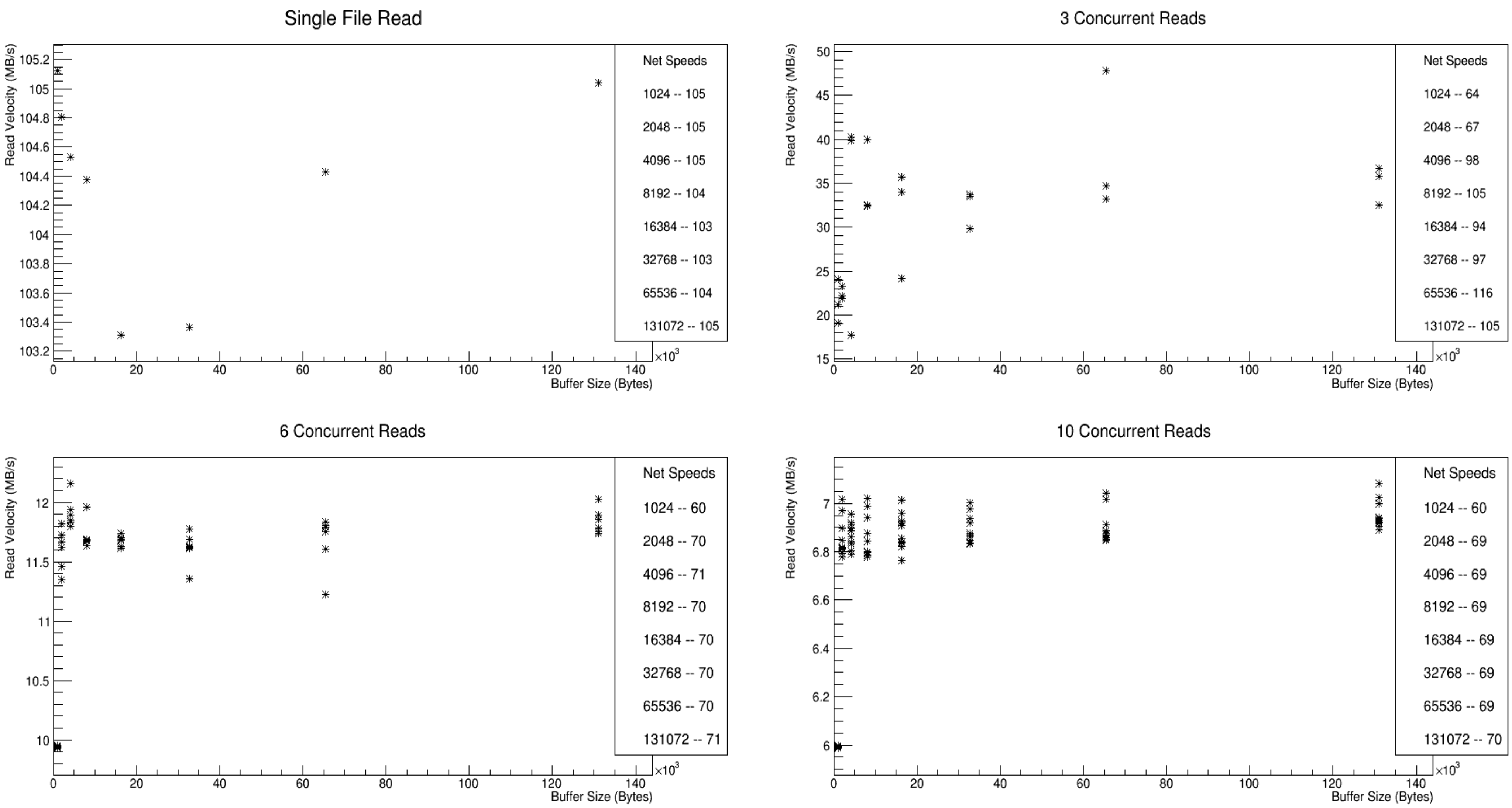
```
Start Timer
    Open File (C “fopen” call)
    While file has unread data
        Read BLOCK_SIZE of data (C “read” call)
End Timer


Get File Statistics (namely filesize)
Compute read velocity
```

Read Specifications

Test Psuedocode:

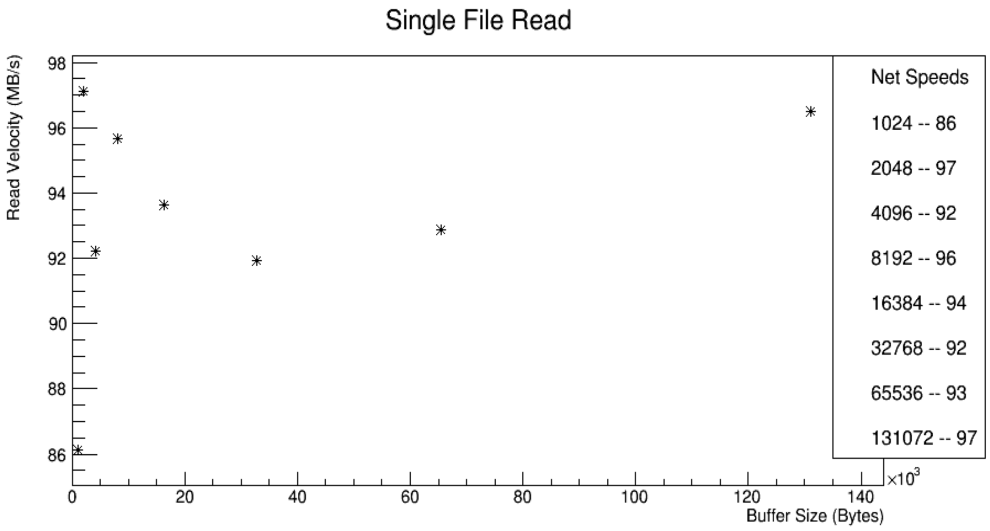
```
For (Buffer Size, Concurrent Read Number) Pairs:
    Drop Ram Cache
    Instantiate the proper number of read programs on proper files
    Wait for programs to finish
```

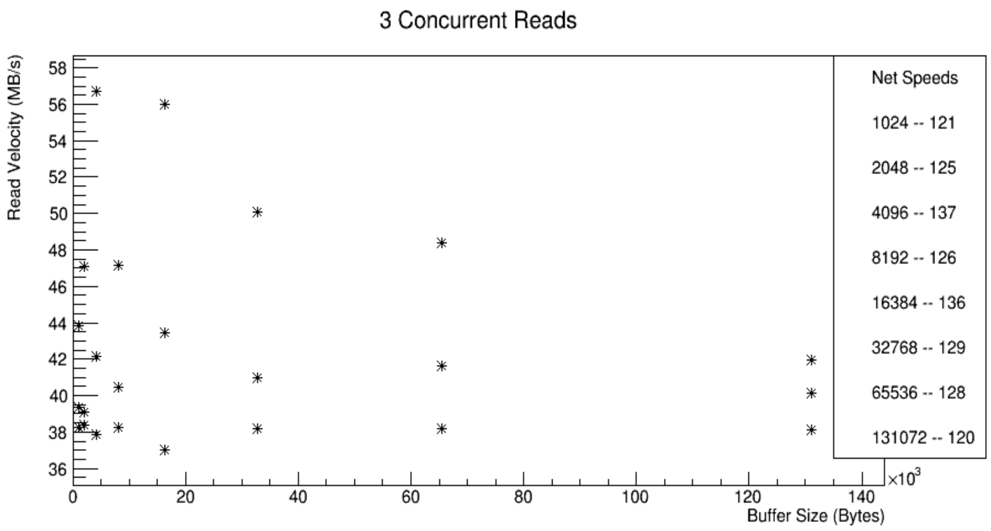
Single Disk Read on Cabninet-8-8-0

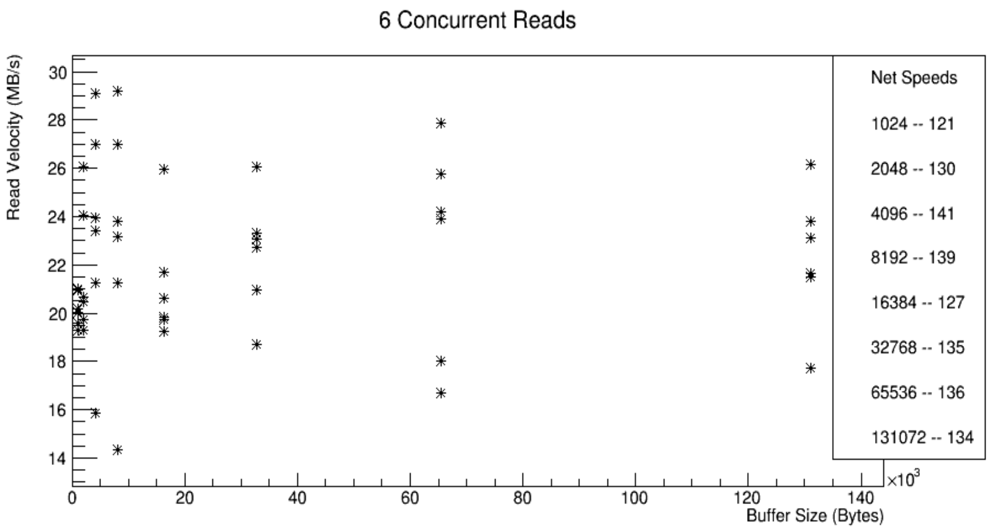


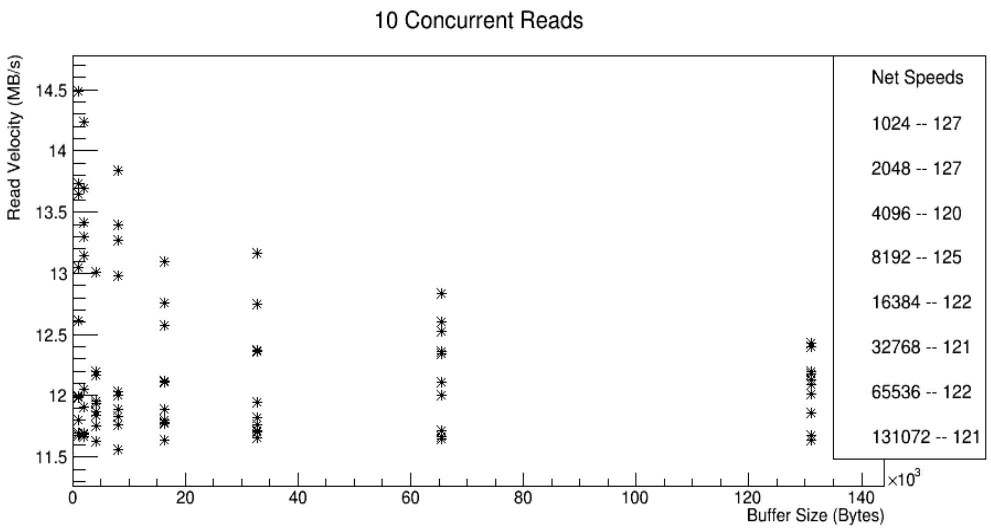
- Recall this data from first set of slides
- This is sort of a benchmark for our hard drives
- All files are stored on the same disk
- Heavy performance drop with multiple-file reads

Hadoop Read (C API)


Single File Read

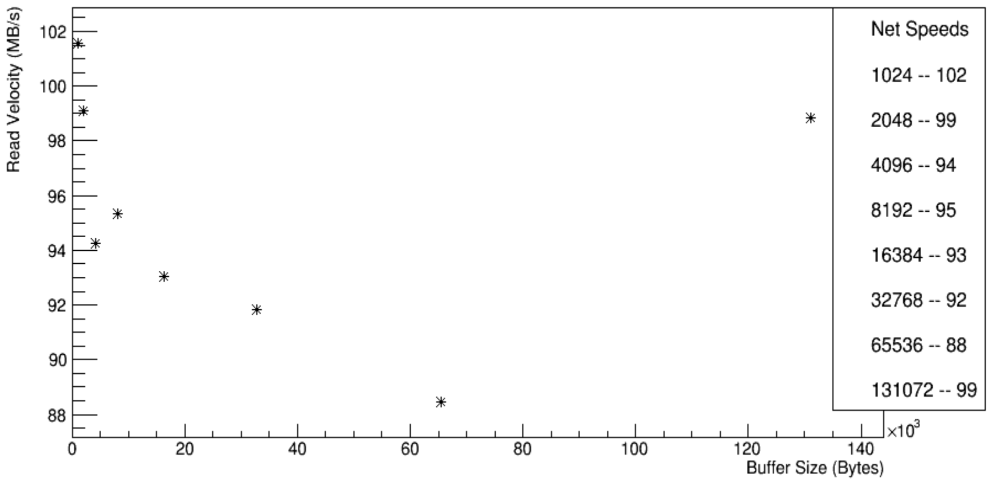

3 Concurrent Reads


6 Concurrent Reads

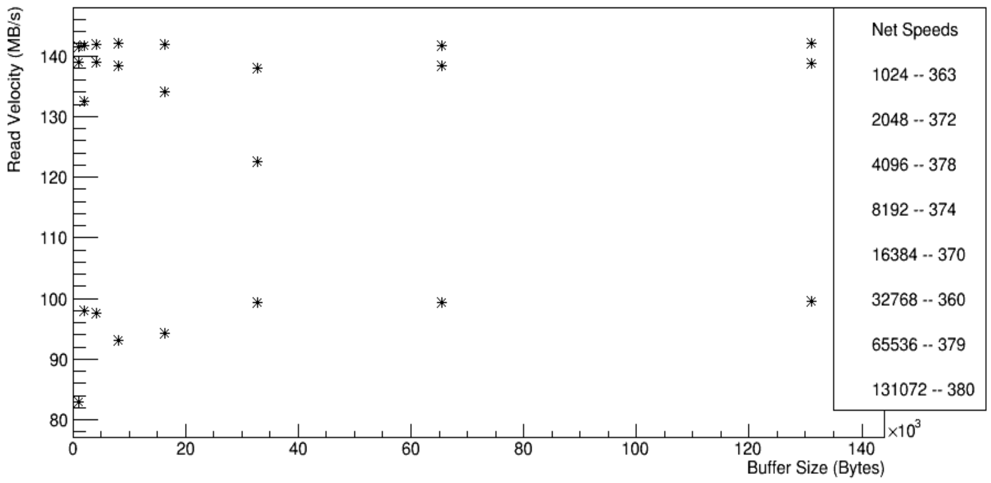

10 Concurrent Reads

- Also Data from first set of slides
- Takes a slight hit on single file read
- Almost twice as fast on many file reads

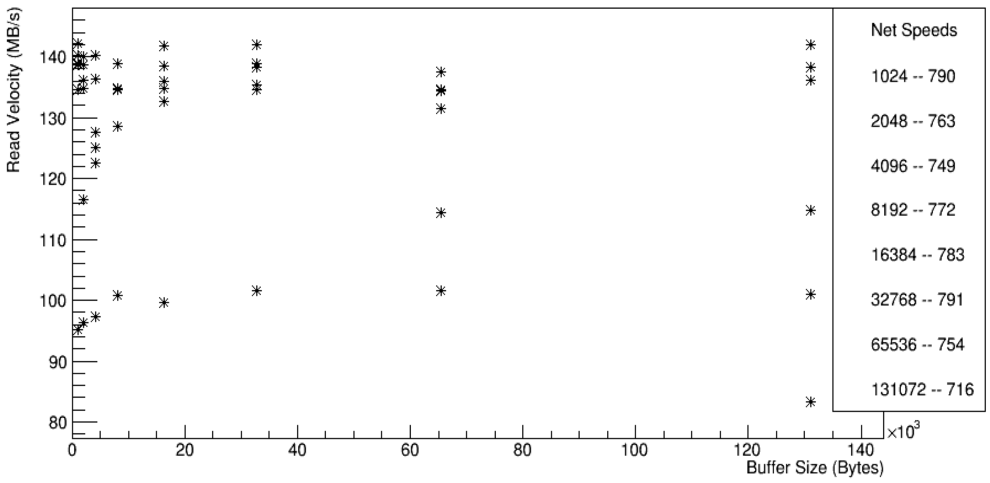
cabinet-7-7-0 Through Condor



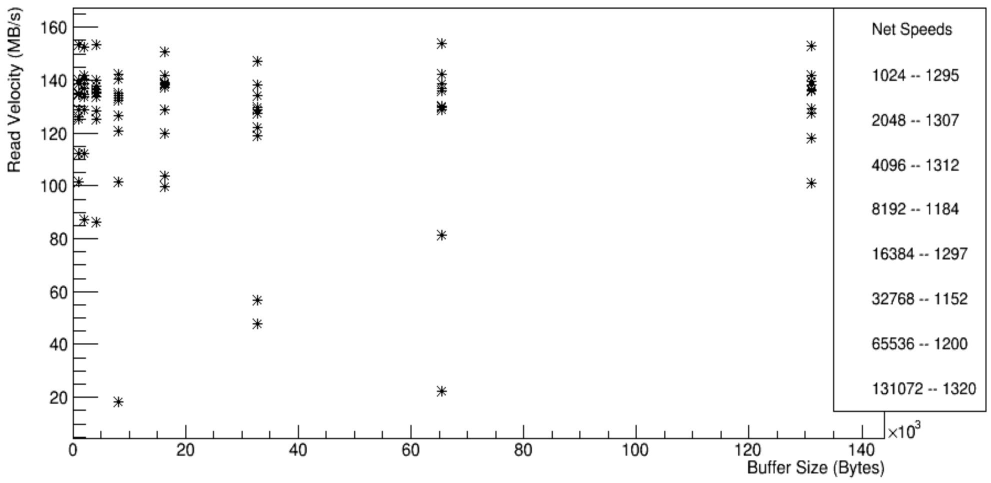
Single File Read



3 Concurrent Reads



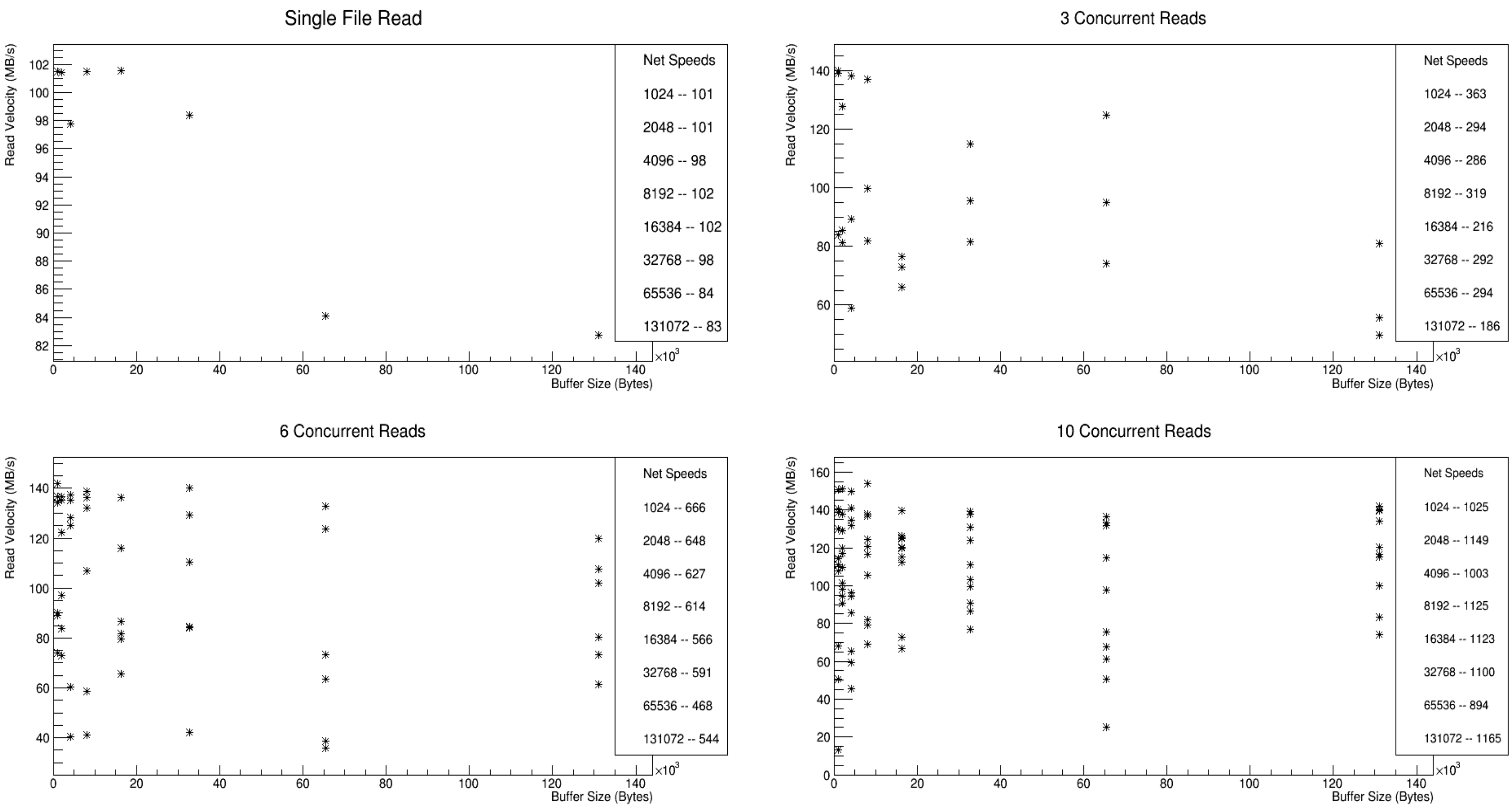
6 Concurrent Reads



10 Concurrent Reads

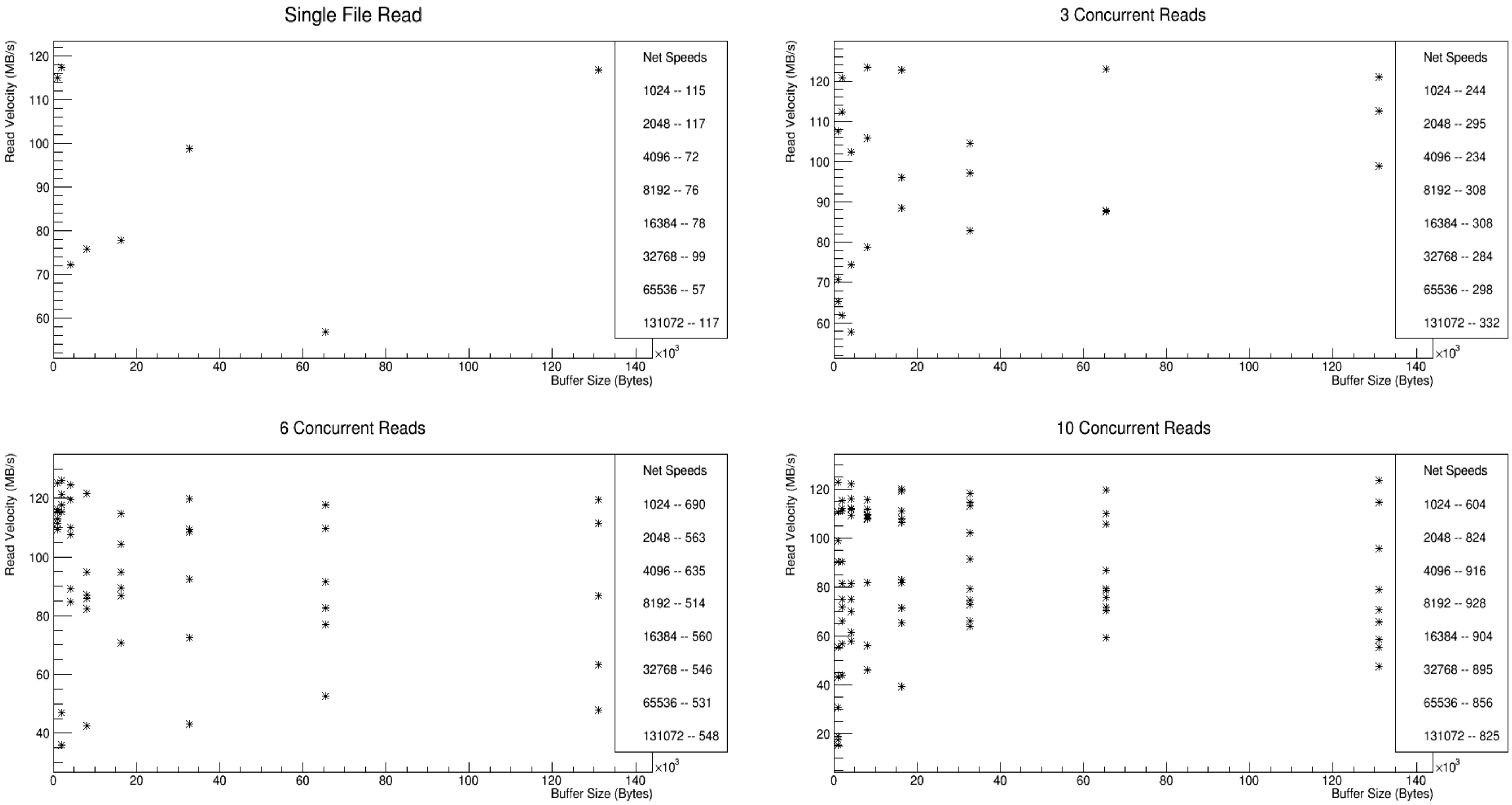
- Impliments a Disk per file read initiated through Condor
- Cabinets were in normal usage conditions
- An order of magnitude speed up compared to the Hadoop Read on 10 files

cabinet-7-7-0 Through Condor (Run 2)



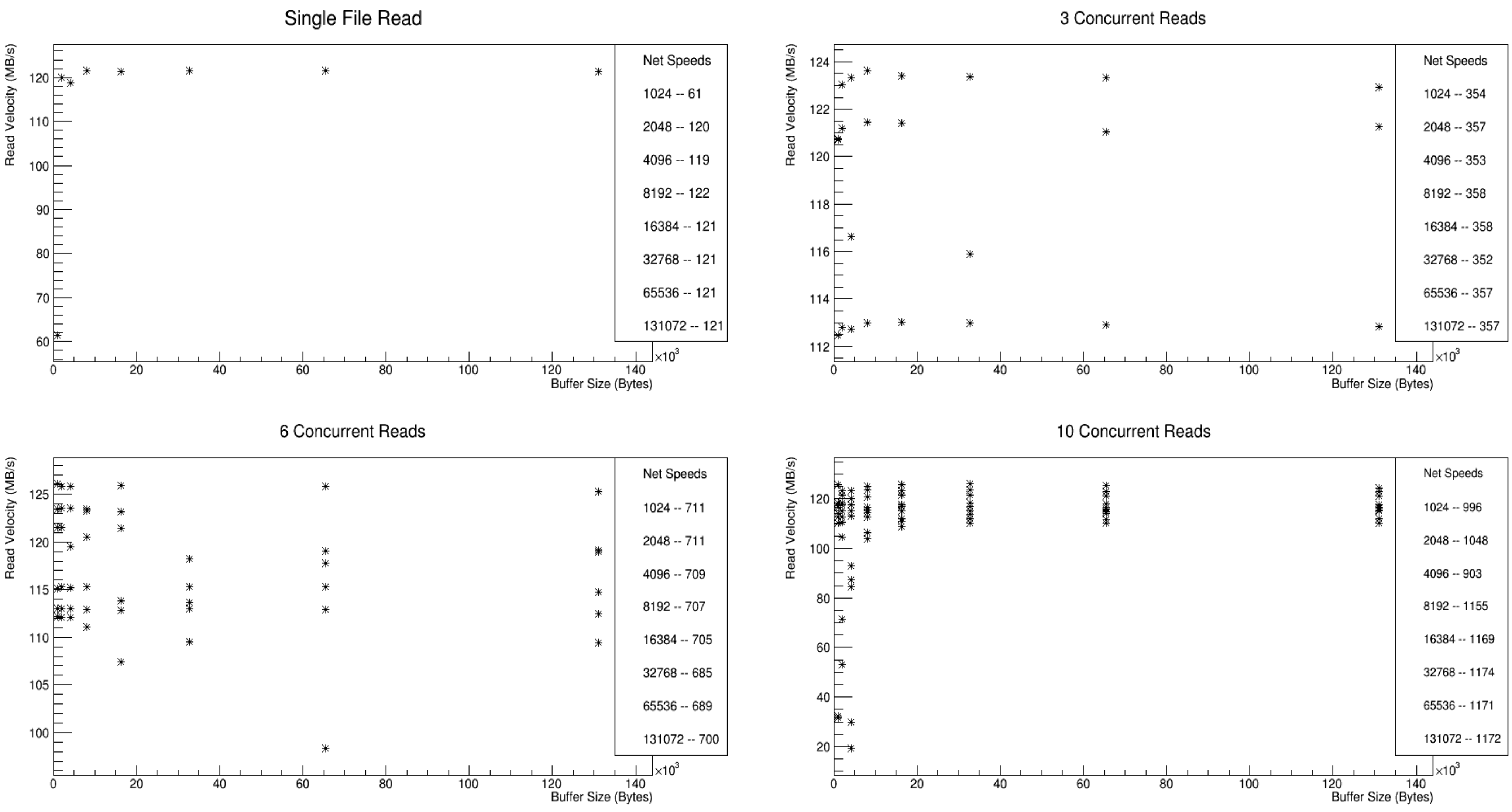
- Another trial of the exact same test as previous slide
- Run approximately 2 day apart
- Cabinet was under normal usage conditions
- Notice that despite the variance in the 10 file growing sizably, the increased performance still persists

cabinet-7-7-6 Through Condor



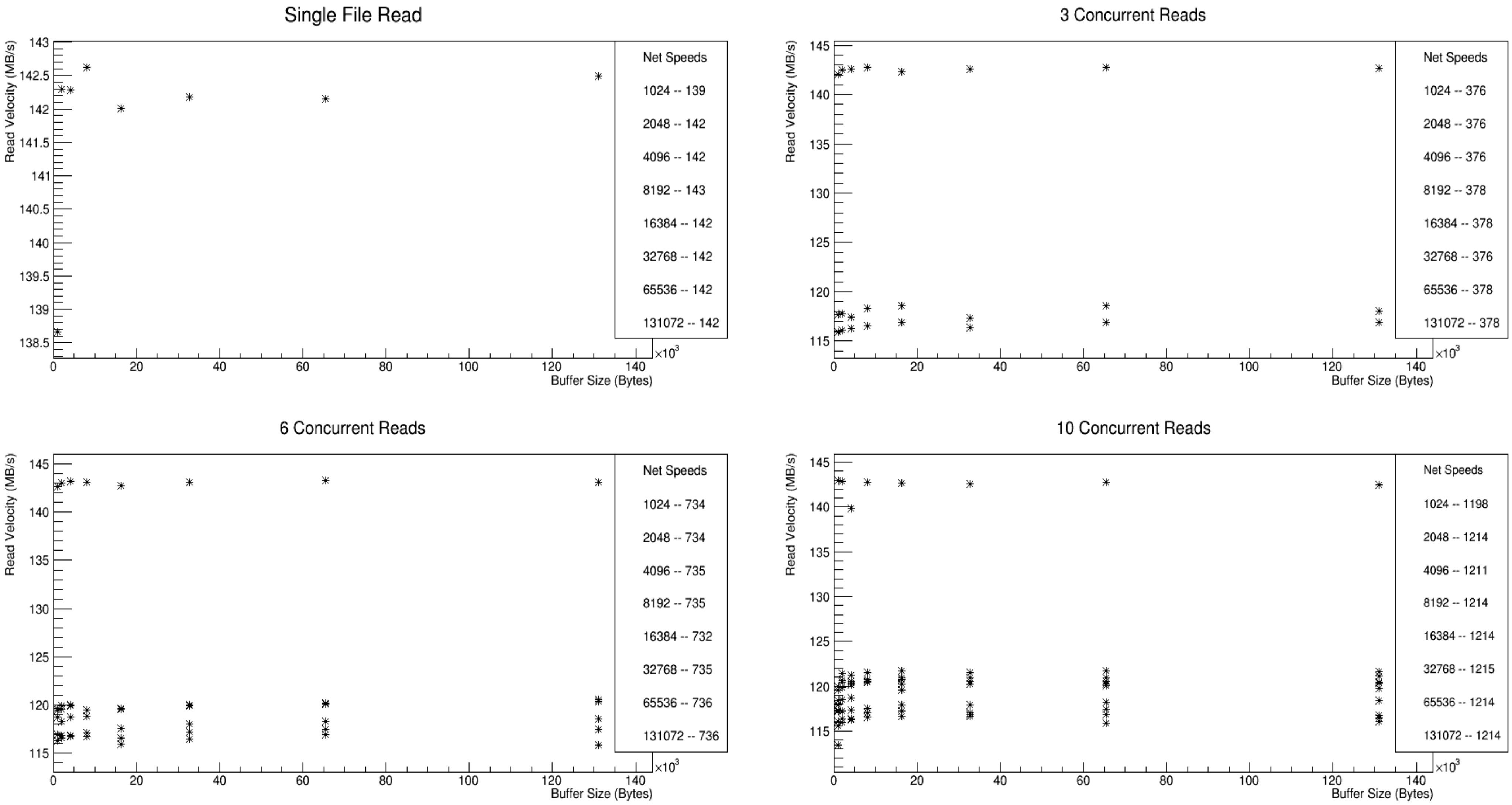
- Same batch of data as from the last two slides on another machine
- Impliments a Disk per file read initiated through Condor
- Cabinet was under normal usage conditions

cabinet-7-7-6 Through Condor (Run 2)



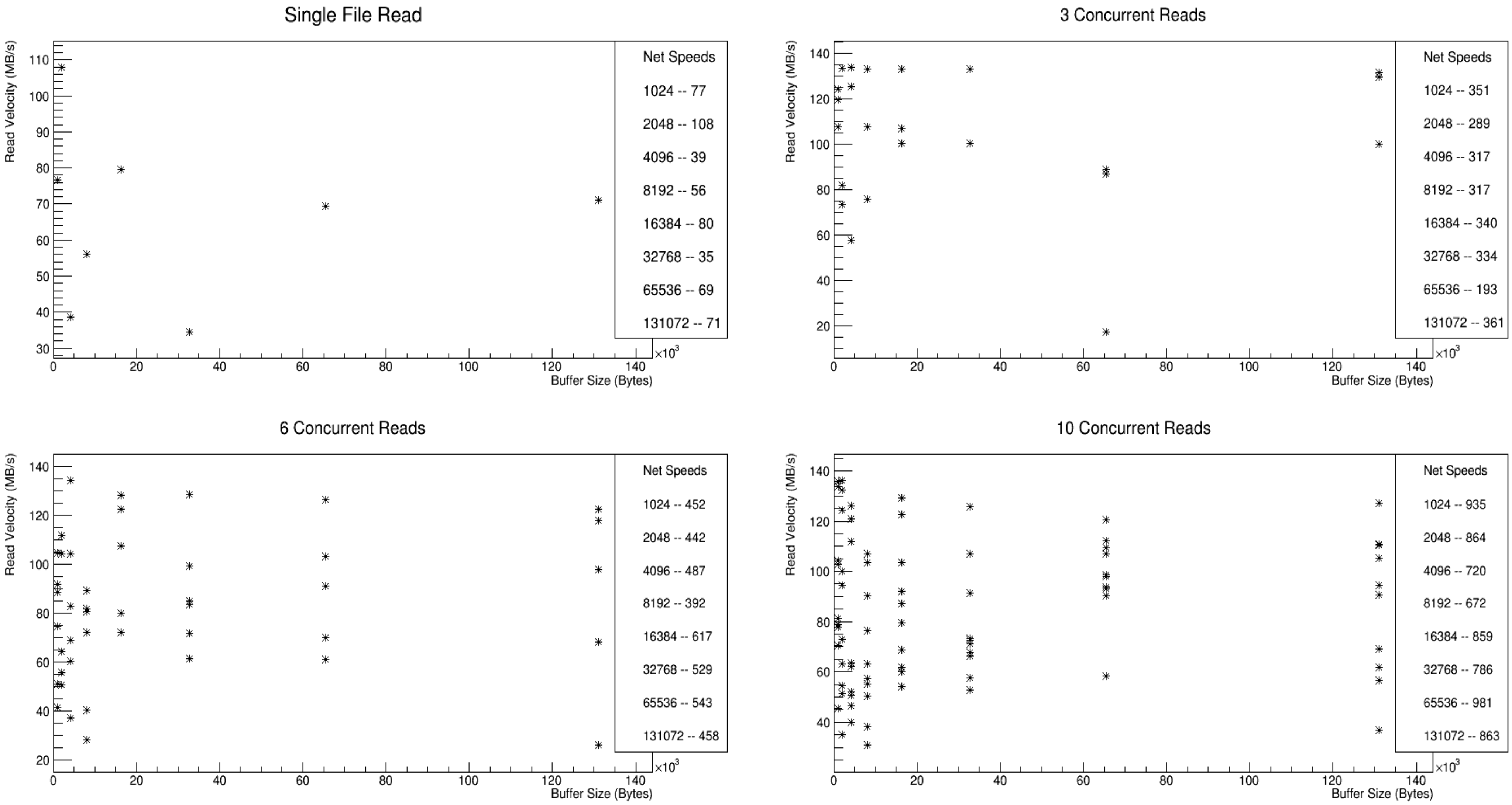
- Another trial on Cab-7-7-6, taken approximately 2 days apart
- Impliments a Disk per file read initiated through Condor
- Cabinet was under normal usage conditions
- Again the read speeds on vary over time

cabinet-7-7-17 Through Condor

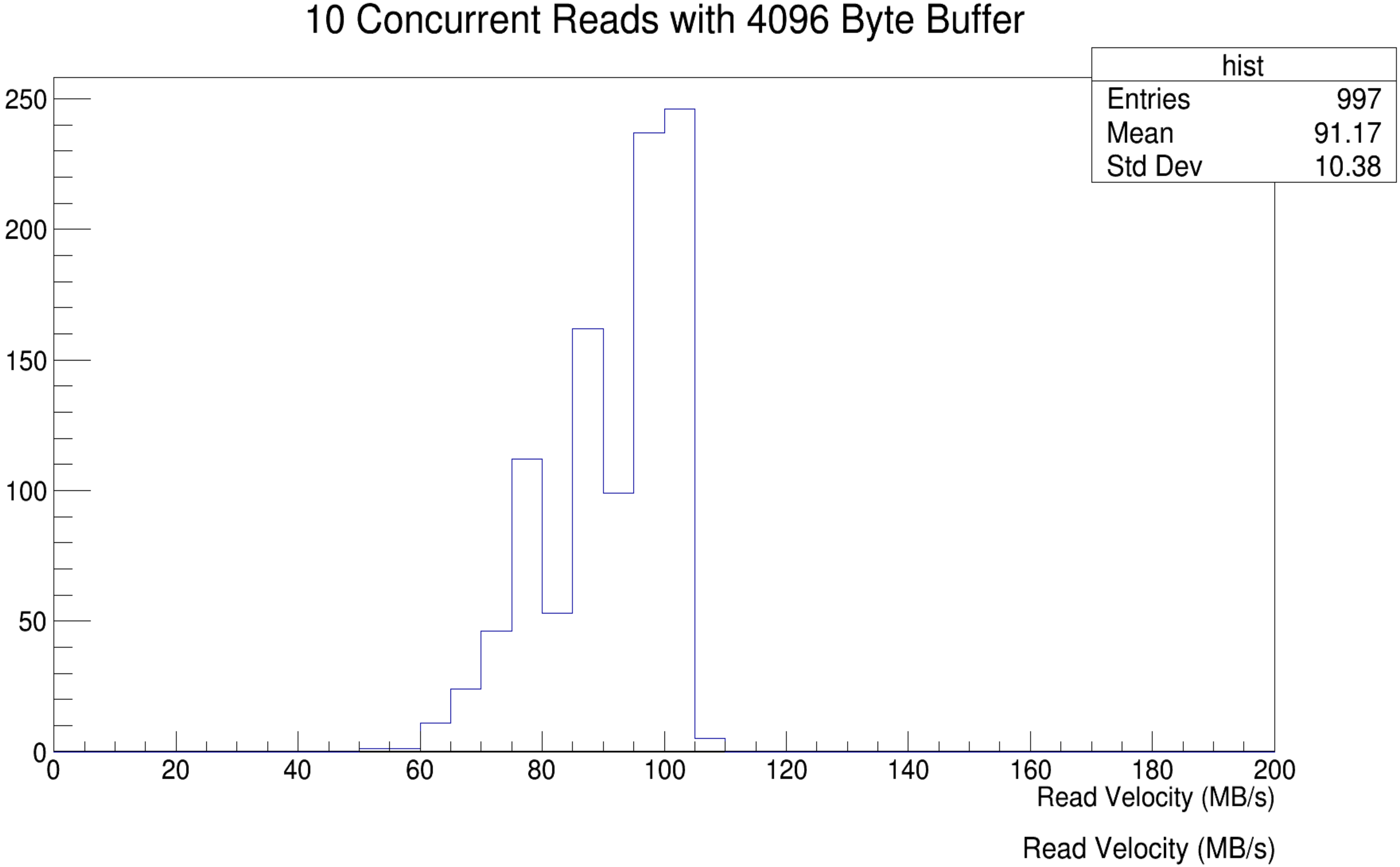


- Same type of data as previous 4 slides
- Included because this machine was generally the best performer. (sort of an upper bound)

cabinet-7-7-10 Through Condor

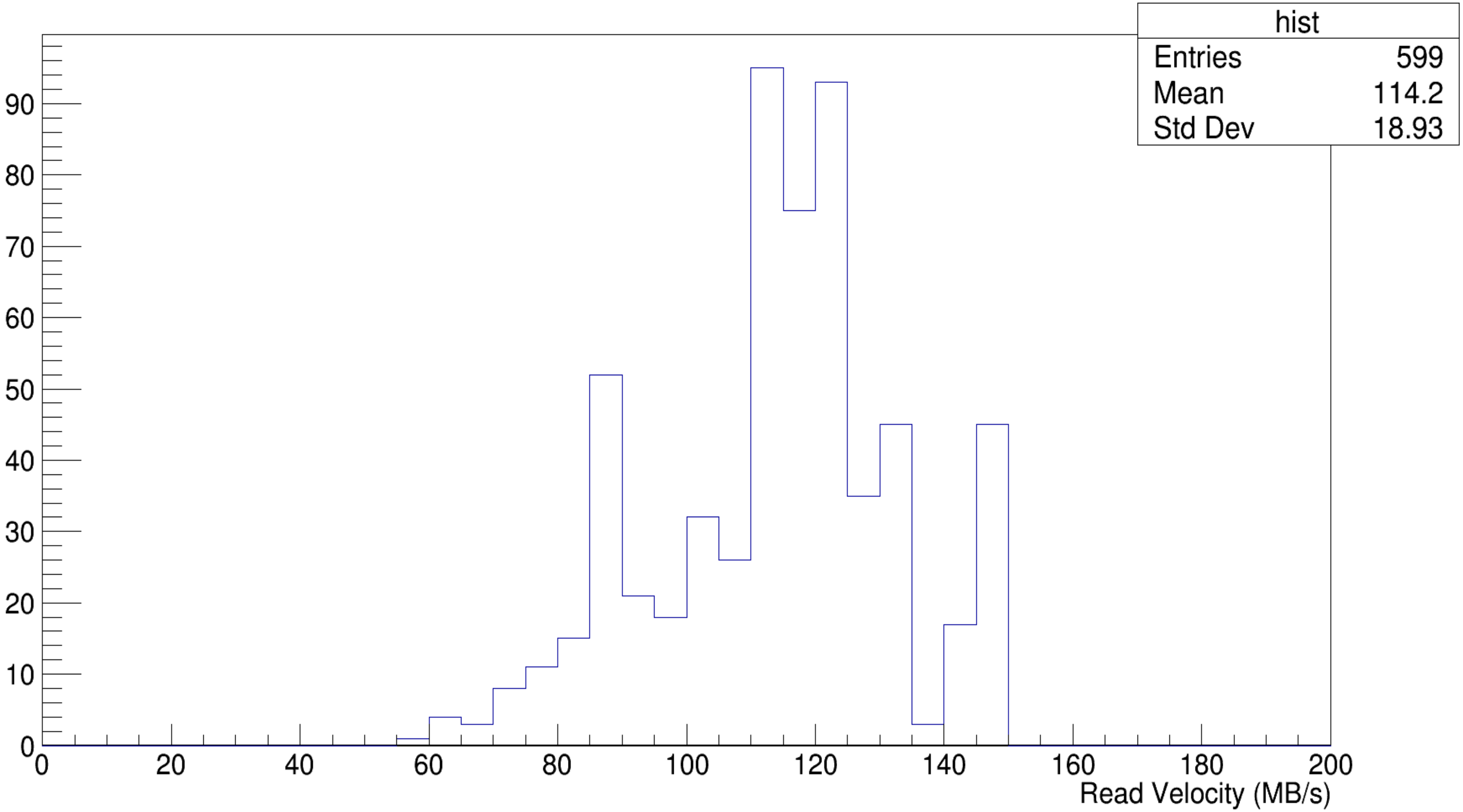


- Same type of data as previous 5 slides
- Included because this machine was generally the worst performer. (sort of a lower bound)



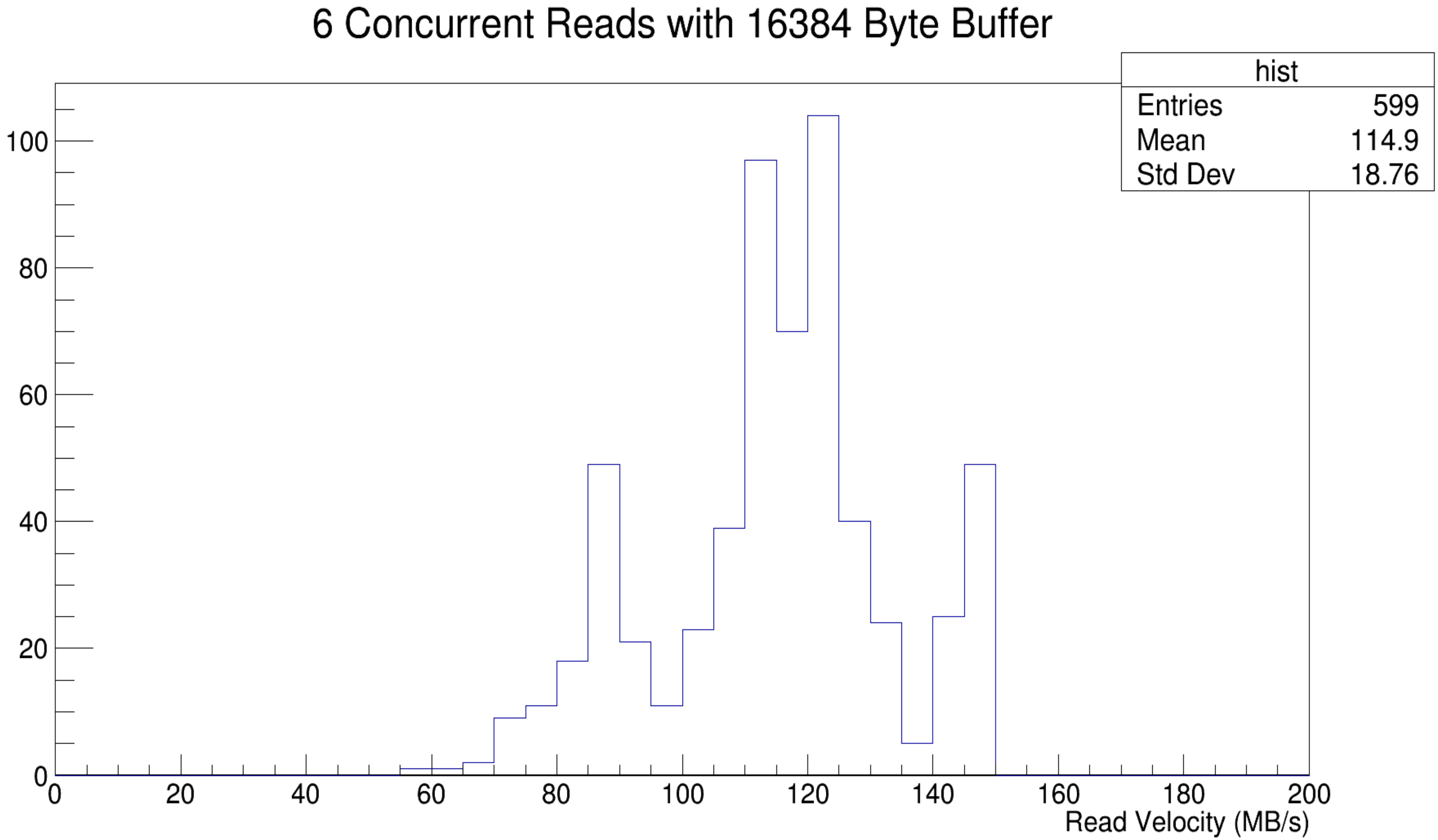
- Data produced from approximately 100 runs of the previous kind
- Gaussian with a cutoff (possibly 3 modes?)

6 Concurrent Reads with 4096 Byte Buffer



Read Velocity (MB/s)

- Data produced from approximately 100 runs of the previous kind
- Again seems to have 3 modes



Read Velocity (MB/s)

- Data produced from approximately 100 runs of the previous kind
- 3 modes persist even through changes in buffer size

Next Steps

1) Investigate 3 modality of read return times?
2) Modify read test to take in ROOT files with selective reading

Conclusions

1) The disk per file method has been again shown to increase read speeds by approximately (Number of Reads at Once)*(Disk Read Speed) for Niave reads

2) Further testing neccesary to confirm that the gains persist when selectively reading ROOT files.