TACC
TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

IO Workload Throttling on Supercomputers

Si Liu

Analyzing Parallel I/O

Nov 13, 2018

Team Members

Lei Huang
huang@tacc.utexas.edu
Texas Advanced
Computing Center

Si Liu
siliu@tacc.utexas.edu
Texas Advanced
Computing Center

Issues of Parallel Shared Filesystem

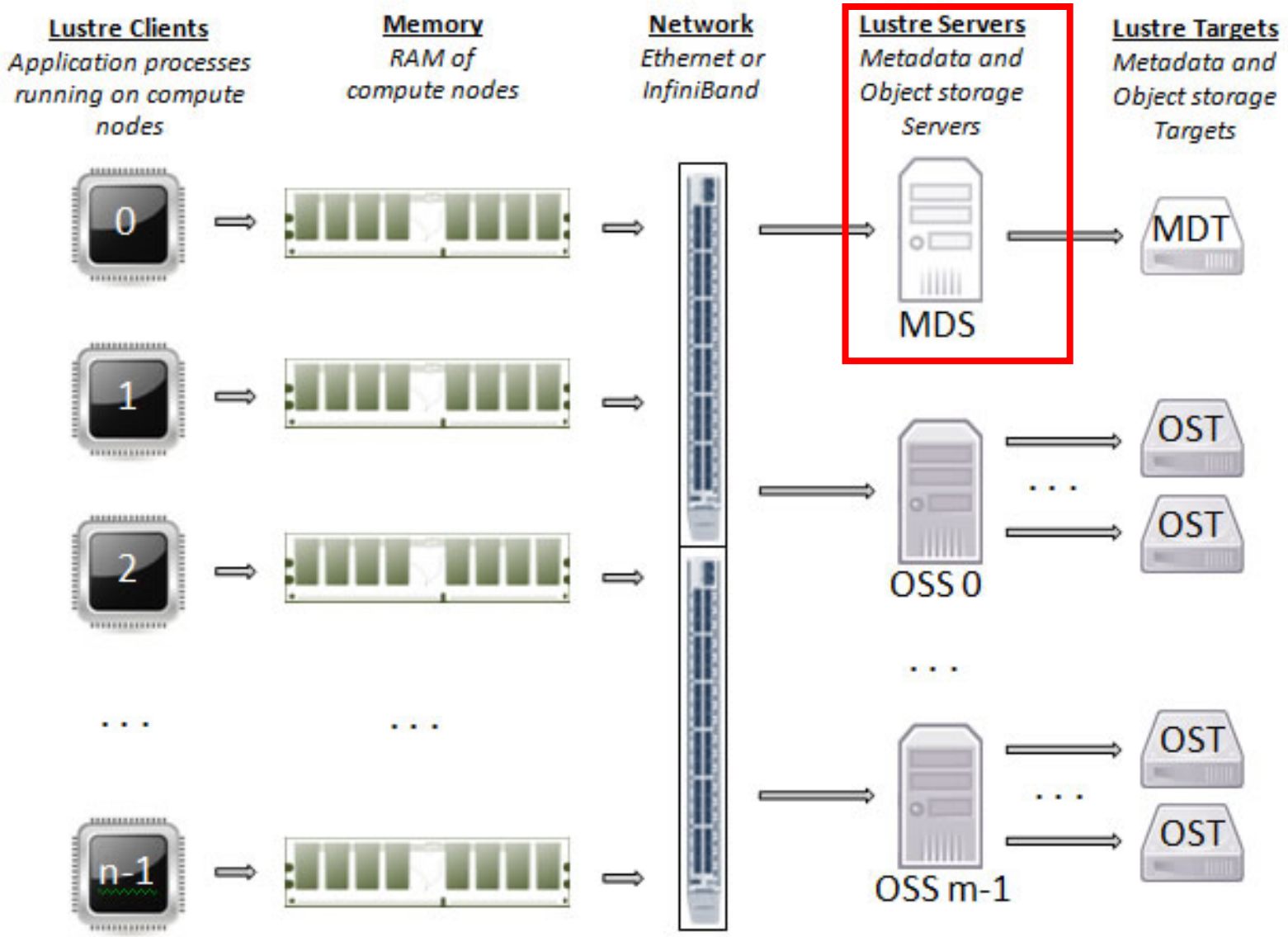
- Achilles' heel of HPC: filesystem is shared by all users on all nodes (even crossing multiple clusters). It is a weak point of modern HPC.
- Overloading metadata server results in global filesystem performance degradation and even unresponsiveness.
- Many practical applications (in computational fluid dynamics, quantum chemistry, machine learning, etc.) raise a huge amount of IO requests in a very short time.
- There is no strict enforced IO resource provisioning in production (e.g. metadata sever throughput, bandwidth) on user level or node level.

Potential Solutions

- System level
  - A strong parallel filesystem that can handle any kind of IO requests from all users without losing efficiency, e.g., upgrade hardware of MDS to achieve better IO throughput
    - Impractical, expensive or limited improvement
  - Burst buffer
    - Needs extra hardware and software, even changes in user code
- Application level
  - A well-designed workflow with reasonable IO workload
    - Recommended way
    - Expertise required
- User level
  - Users give up planned IO work to avoid heavy IO requests or decrease the number of jobs
    - A compromise rather than a solution

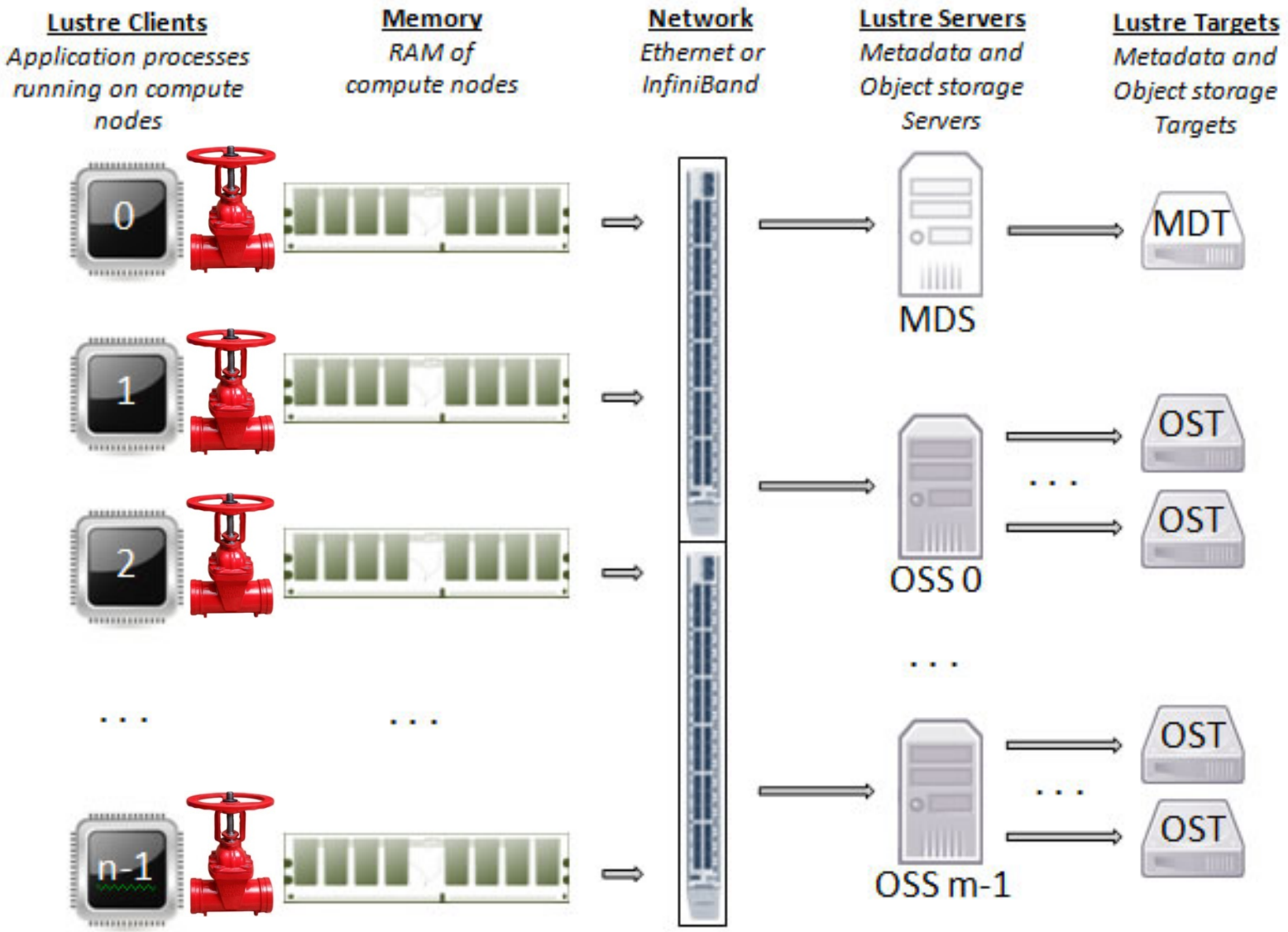
Potential Solutions

- System level
  - A strong parallel filesystem that can handle any kind of IO requests from all users without losing efficiency, e.g., upgrade hardware of MDS to achieve better IO throughput
    - Impractical, expensive or limited improvement
  - Burst buffer
    - Needs extra hardware and software, even changes in user code
- Application level
  - A well-designed workflow with reasonable IO workload
    - Recommended way
    - Expertise required
- User level
  - Users give up planned IO work to avoid heavy IO requests or decrease the number of jobs
    - A compromise rather than a solution
  - An optimal system that makes heavy IO work under control
    - Without rewriting users'code 



Lustre Architecture (NICS website)

https://www.nics.tennessee.edu/computing-resources/file-systems/lustre-architecture



Lustre Architecture (NICS website)

https://www.nics.tennessee.edu/computing-resources/file-systems/lustre-architecture

Our Proposed User-side Solution

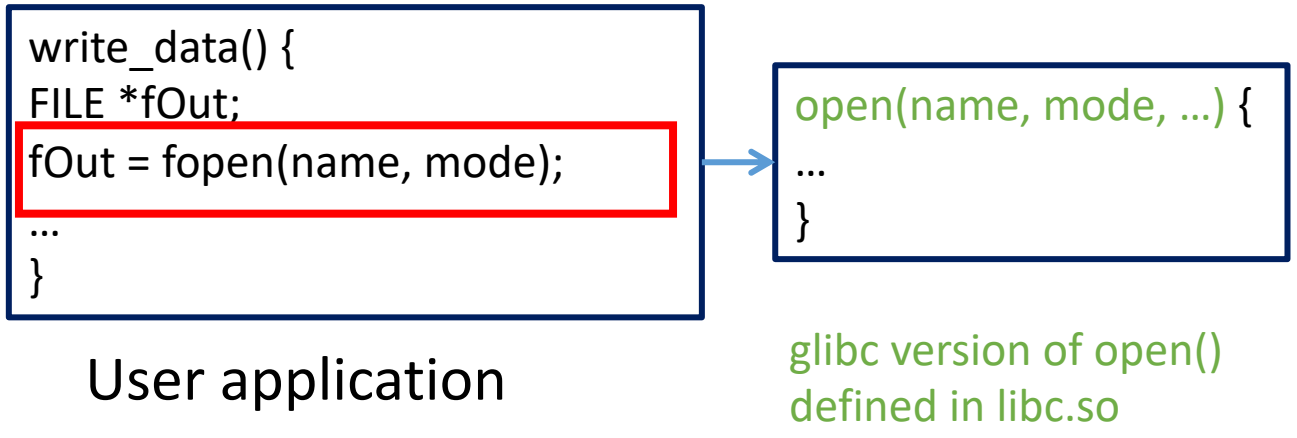
- Intercept IO related functions (open(), stat(), etc.) within applications and keep a record of
  - IO operation time (response time)
  - IO operation frequency (calculated from saved time stamp of recent function calls)
- Evaluate filesystem status (busy/modest used/free)
  - Responding time per operation
- Evaluate IO workloads (recent IO request frequency)
  - Node based and user based
- Insert proper delays when necessary

Optimal Overloaded IO Protection System (OOOPS)

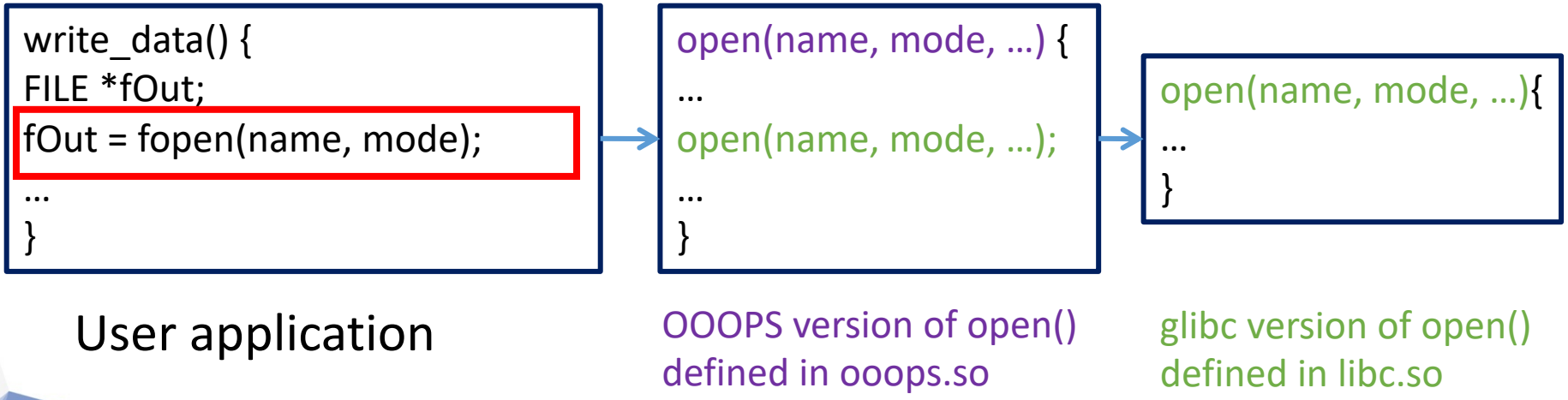
- An innovative IO workload managing system that optimally controls the IO workload from the users' side.
- Automatically detect and throttle excessive IO workload from supercomputer users to protect parallel shared filesystems.

Function Interception

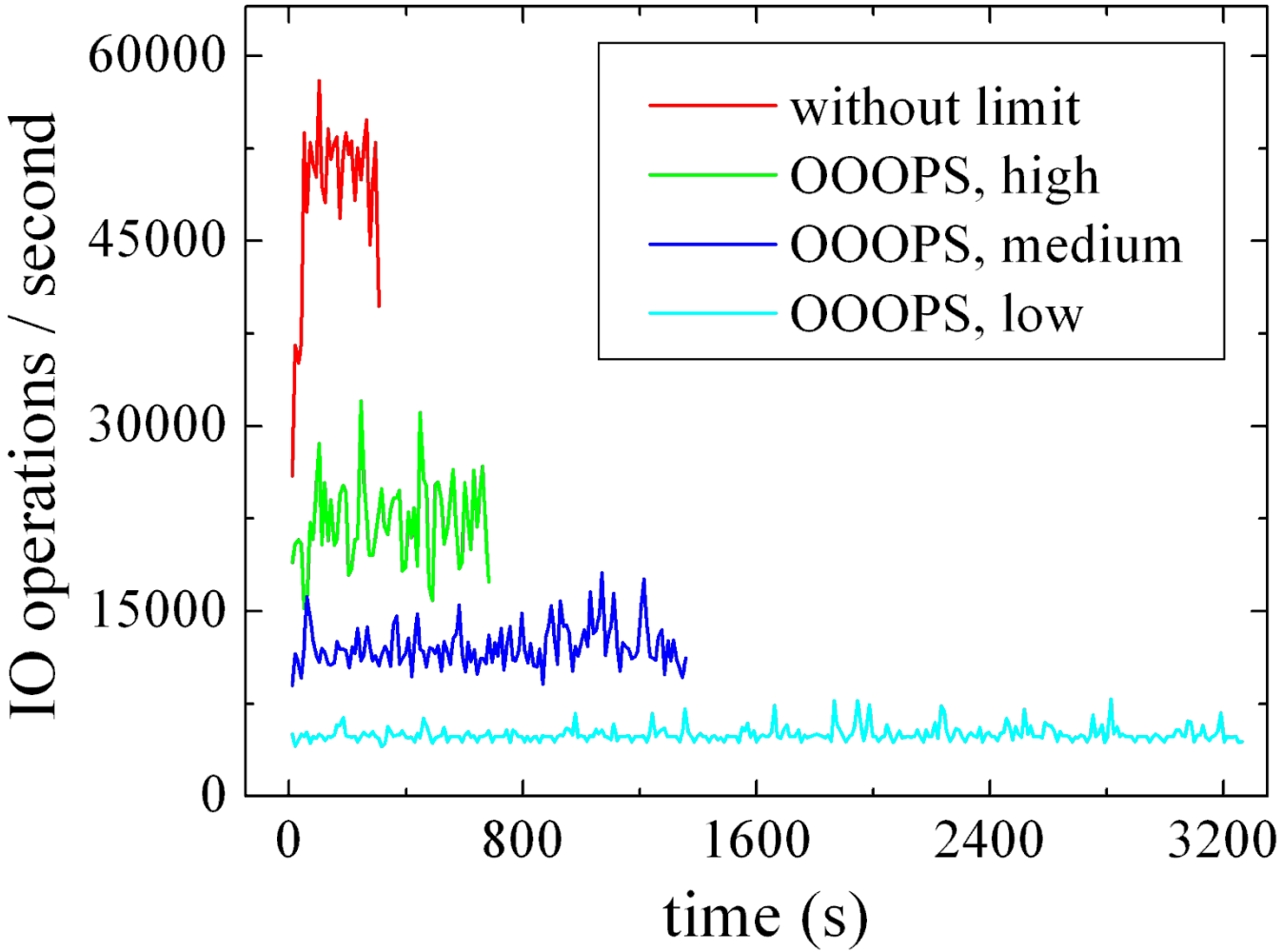
Without OOOPS loaded



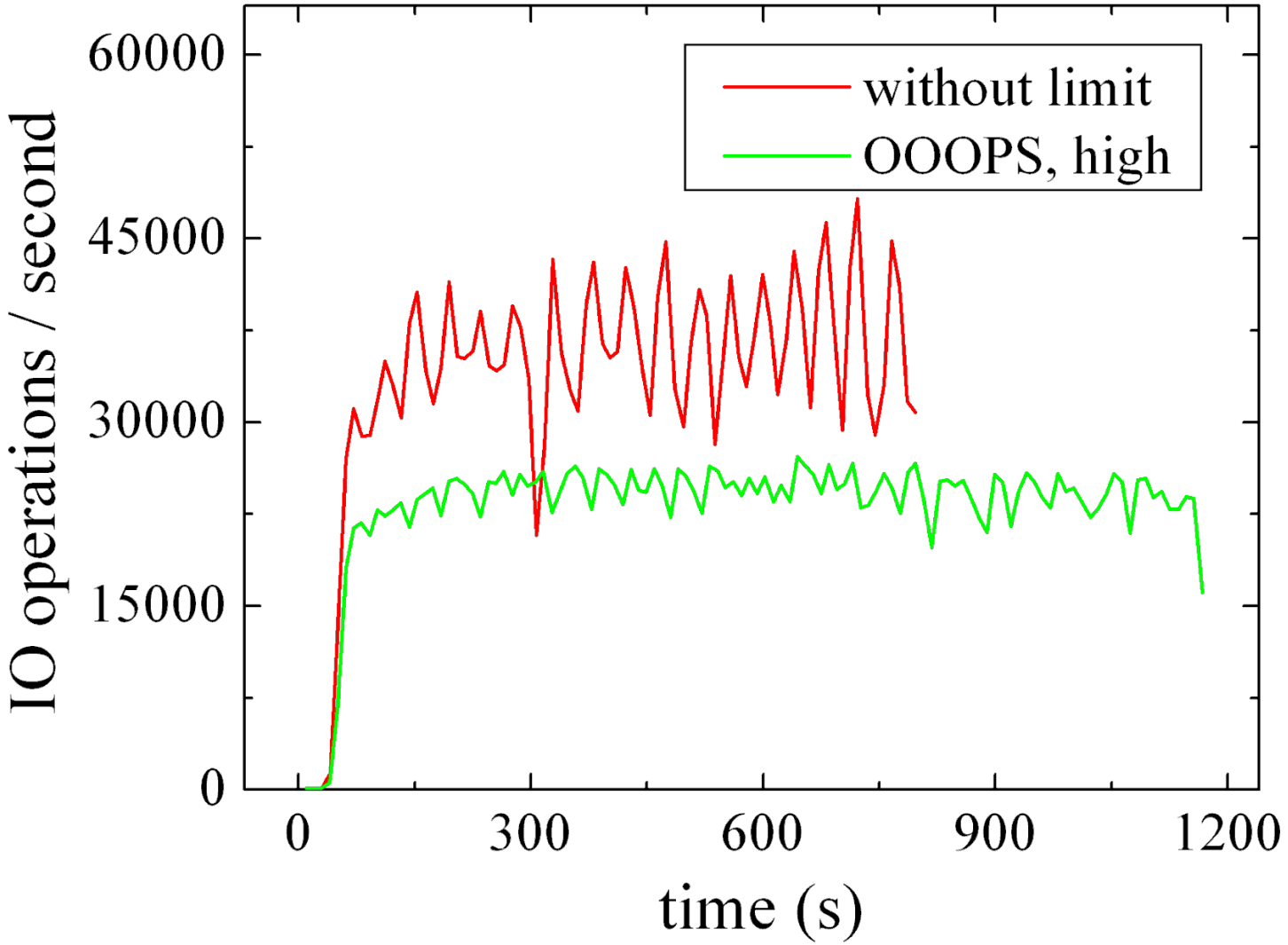
With OOOPS loaded (LD_PRELOAD OOOPS library)



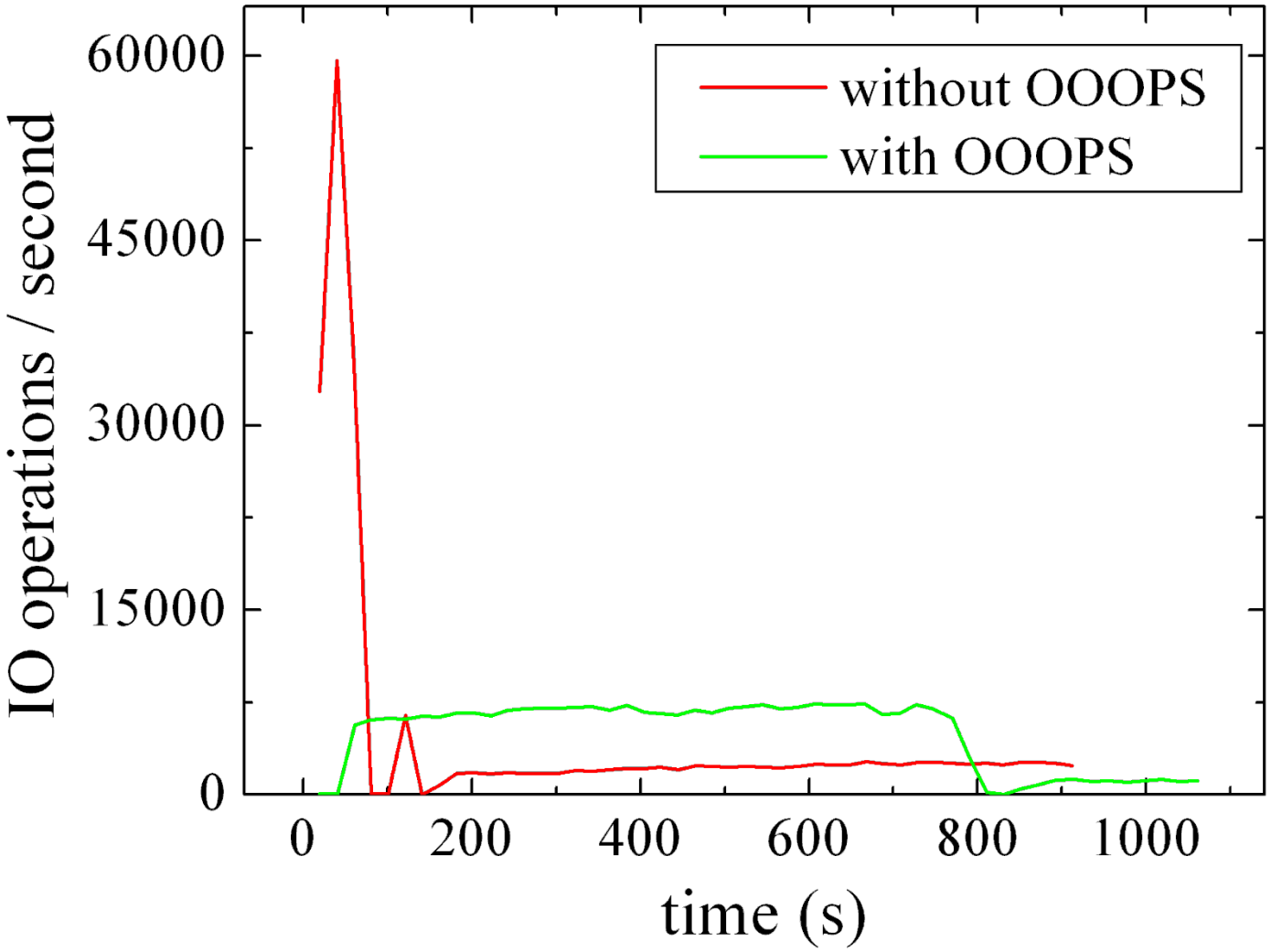
IO Requests with Different Settings



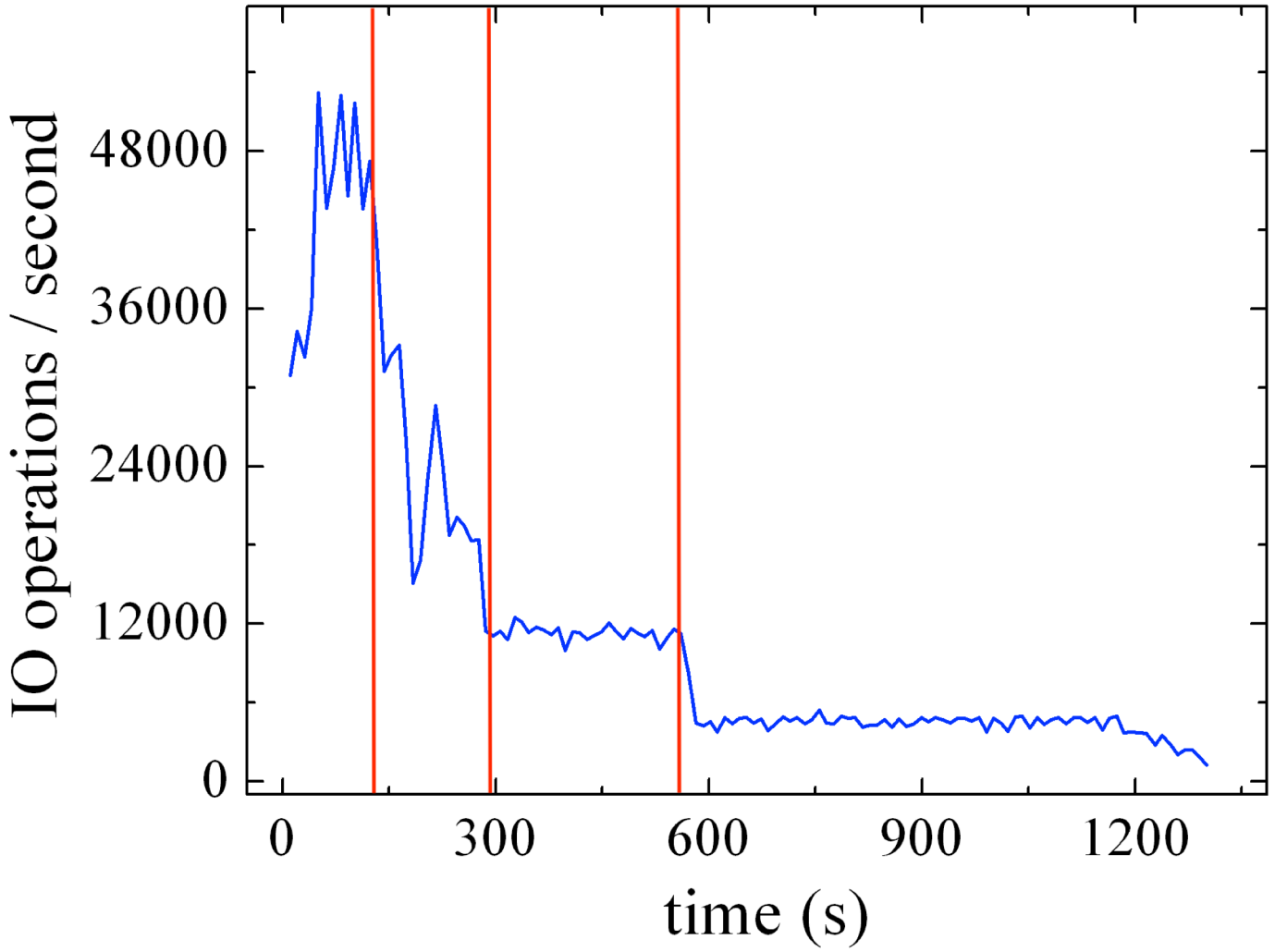
Example of Running OpenFOAM



Example of Running TensorFlow



Example of Dynamically Throttling IO Requests



OOOPS Highlights

- Convenient to HPC users
  - No source code modification at all on uses' side
  - Little/no workflow update on users' side
  - Self-driven slowdown IO work when necessary

- Valuable on supercomputers
  - Protect filesystem from overloaded IO requests
  - Little overhead: minimal/slight influence on performance except some jobs performing excessive IO work
  - Easy to deploy on an arbitrary cluster as long as file system is POSIX compliant
  - Scale up to any size of supercomputers
  - Little work for system administrators
  - Dynamically control running jobs' IO requests without interruption

Limitations

- The IO resource provisioning policy is too simple.
- OOOPS will lead to noticeable performance degradation for the jobs with very intensive IO for significant time.

Conclusion

- We developed a new tool (OOOPS) to help
  - ✓ users carry out heavy IO work that is originally not allowed
  - ✓ administrators protect the cluster from overload

- We enforce a fair-sharing IO resource provisioning policy on client side practically (instead of server side)
  - ✓ Treat IOPS/Metadata server throughput as a resource
  - ✓ Increase system capacity (applications with heavy IO load)

Acknowledgement

Colleagues at TACC

- Zhao Zhang
- Tommy Minyard
- Bill Barth
- Junseong Heo
- Robert McLay
- John Cazes

Stampede2 early users of OOOPS

Other HPC centers

- Davide Del Vento (NCAR)
- Kevin Manalo (JHU)