Development of a Capacity ON Demand User Interaction Toolkit (CONDUIT) Job Launch Mechanism

Christa Collins (Liberty University)

Mentor: Kevin Pelzel

Abstract

Abstract: As High-Performance Computing (HPC) architectures progress towards of model of high throughput scratch space with the utilization of SSDs as opposed to traditional hard disks, multiple complexities occur. Such as the reduction of the overall size of scratch space available to store scientific data due to the cost difference between SSDs and hard disks. CONDUIT is a data transfer orchestration system, currently in development at Los Alamos National Laboratory. CONDUIT's primary purpose is to ease the burden on the user when it comes to the transferring of large datasets between various filesystems. Especially from scratch storage to campaign storage. This project's primary objective is to develop an internal CONDUIT job scheduler mechanism as an alternative to the current SLURM implementation. This was achieved through the development of a minimal client/server implementation that communicates over a gRPC API to coordinate secure job information exchange and launch. Through the development of a custom job scheduler, we were able to achieve the same end results faster than the previous implementation. Along with developing solutions for TLS support, as well as eliminating instability issues that the previous implementation experienced.

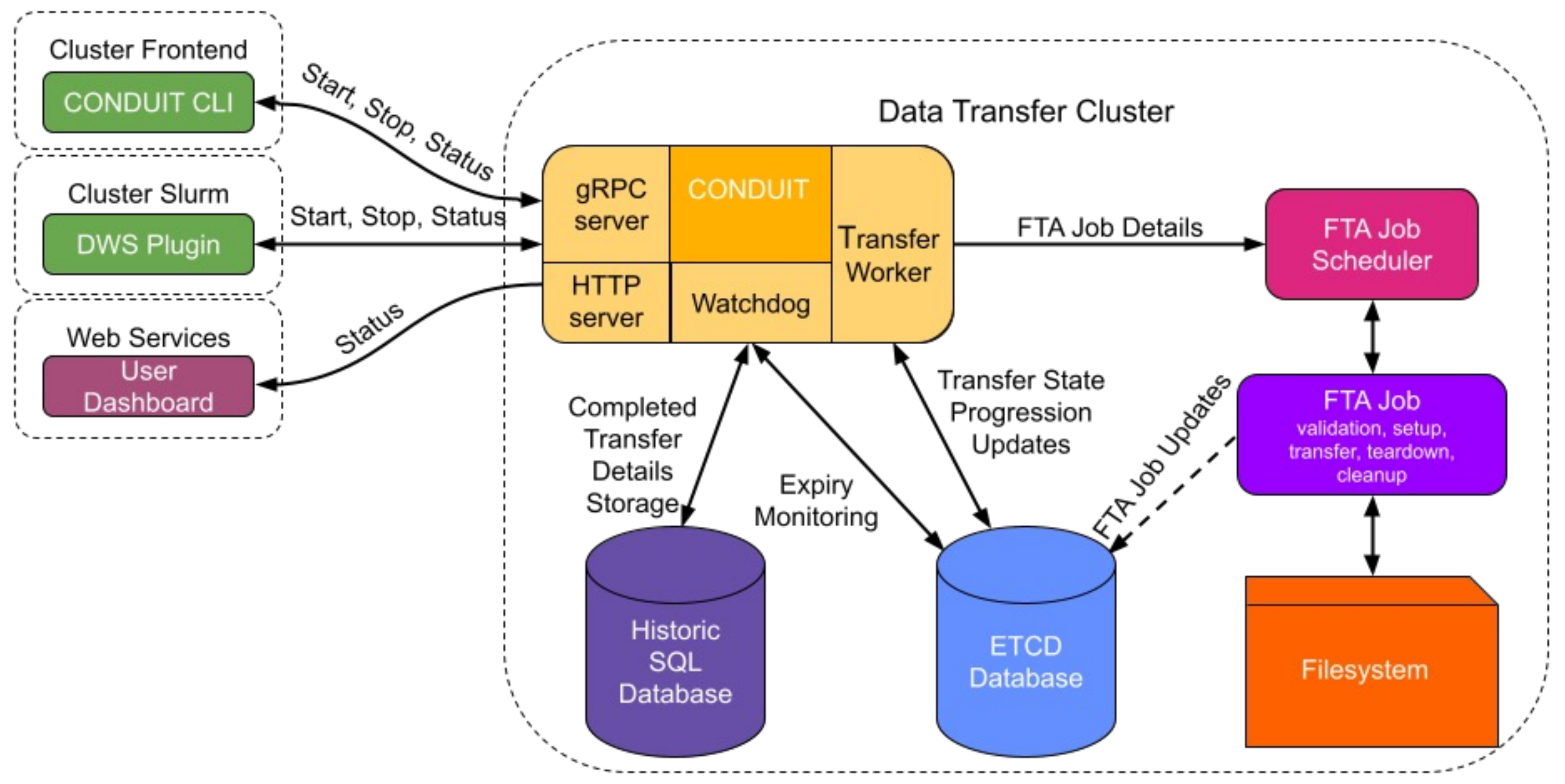

Figure 1. Detailed overview of CONDUIT configuration.

Software Development

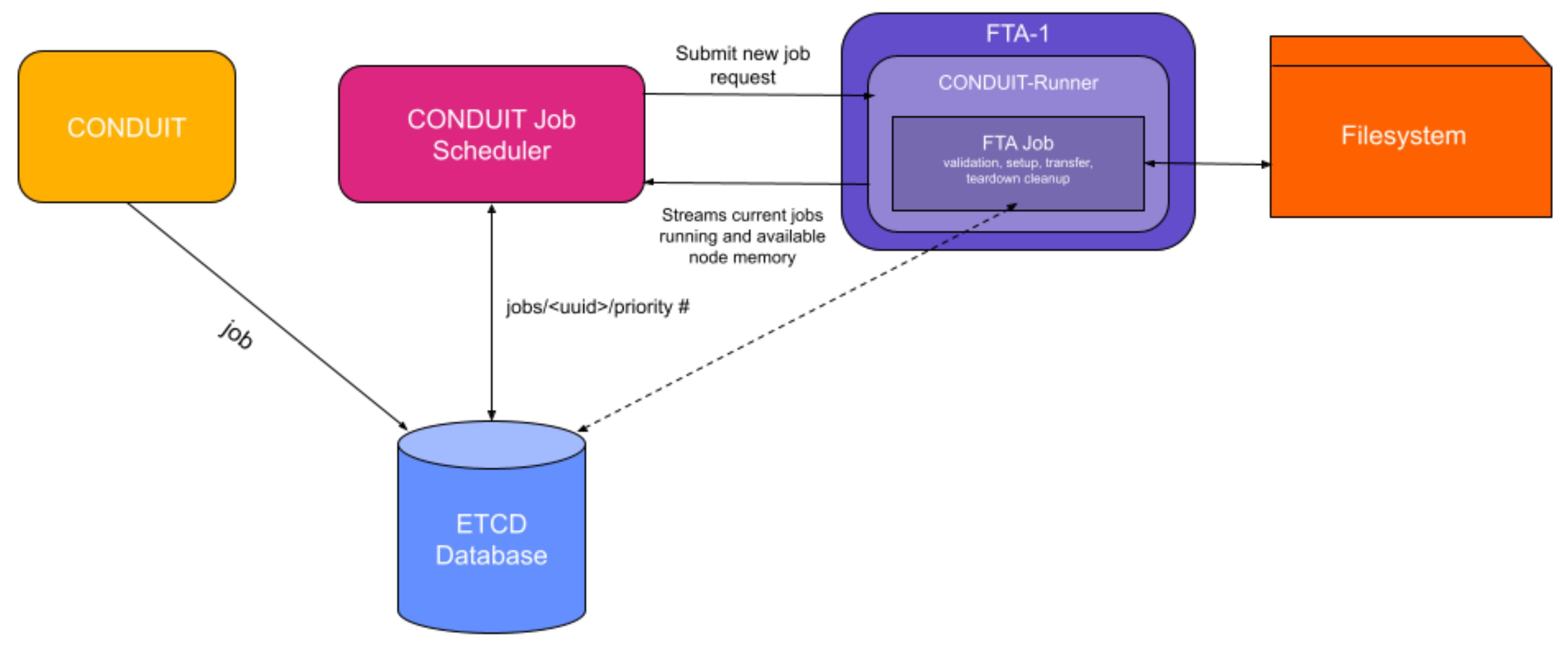

Figure 2. Detailed overview of new CONDUIT job scheduler configuration. CONDUIT submits a job to the scheduler, the scheduler then reads the job information from the ETCD and sends a job request to the available runners. The available runners then run CONDUIT-FTA and CONDUIT-FTA sends its status back to ETCD.

Testing

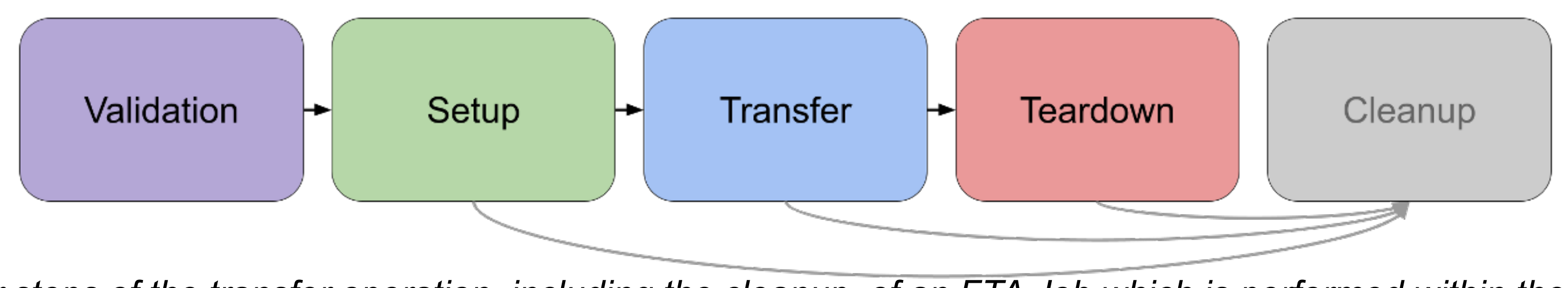

Figure 3. The 4 major steps of the transfer operation, including the cleanup, of an FTA Job which is performed within the CONDUIT-Runner portion of the scheduler configuration.

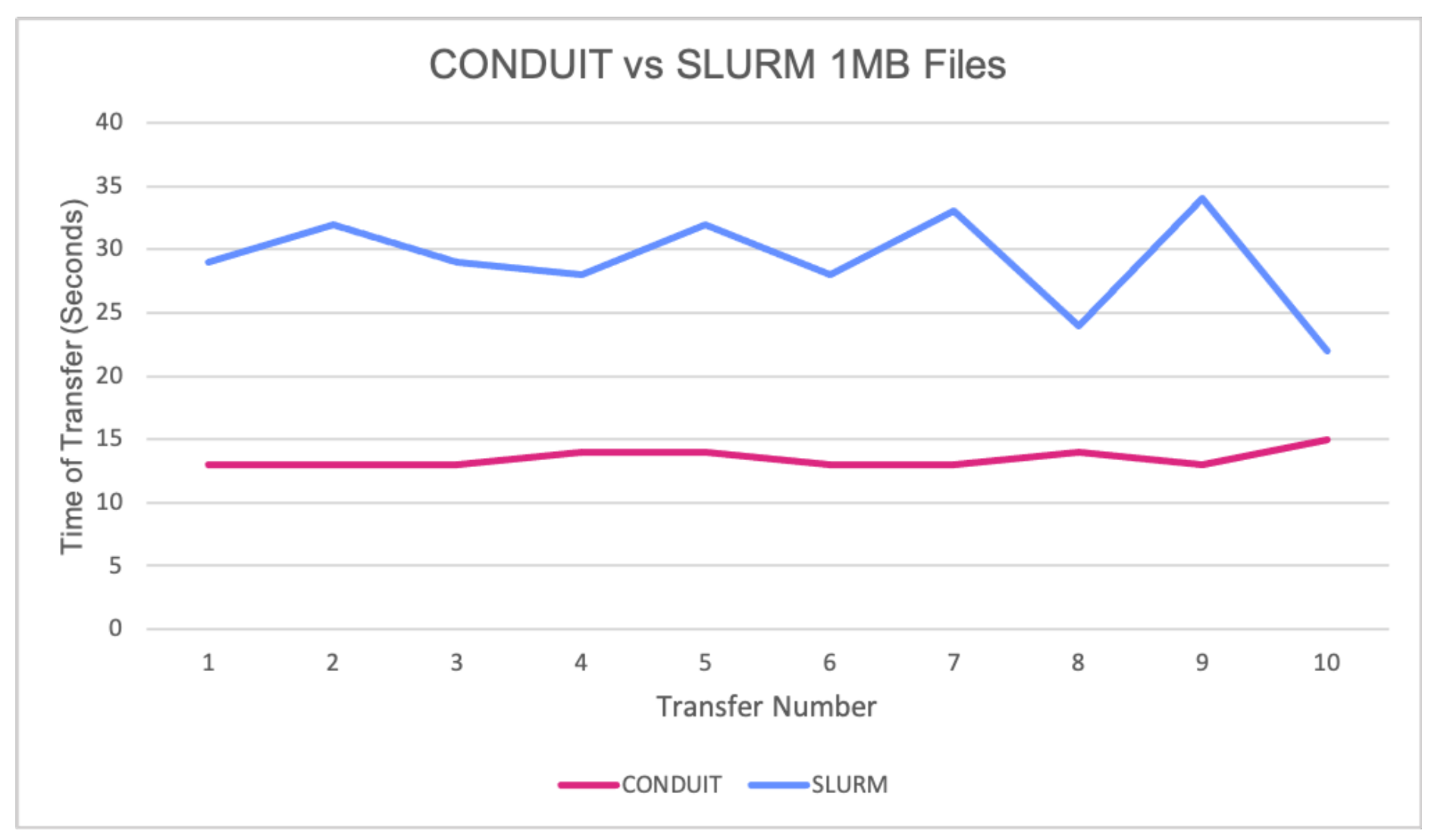

Figure 4. Comparison of CONDUIT scheduler and SLURM when a 1 MB transfer is being written to 10 files.

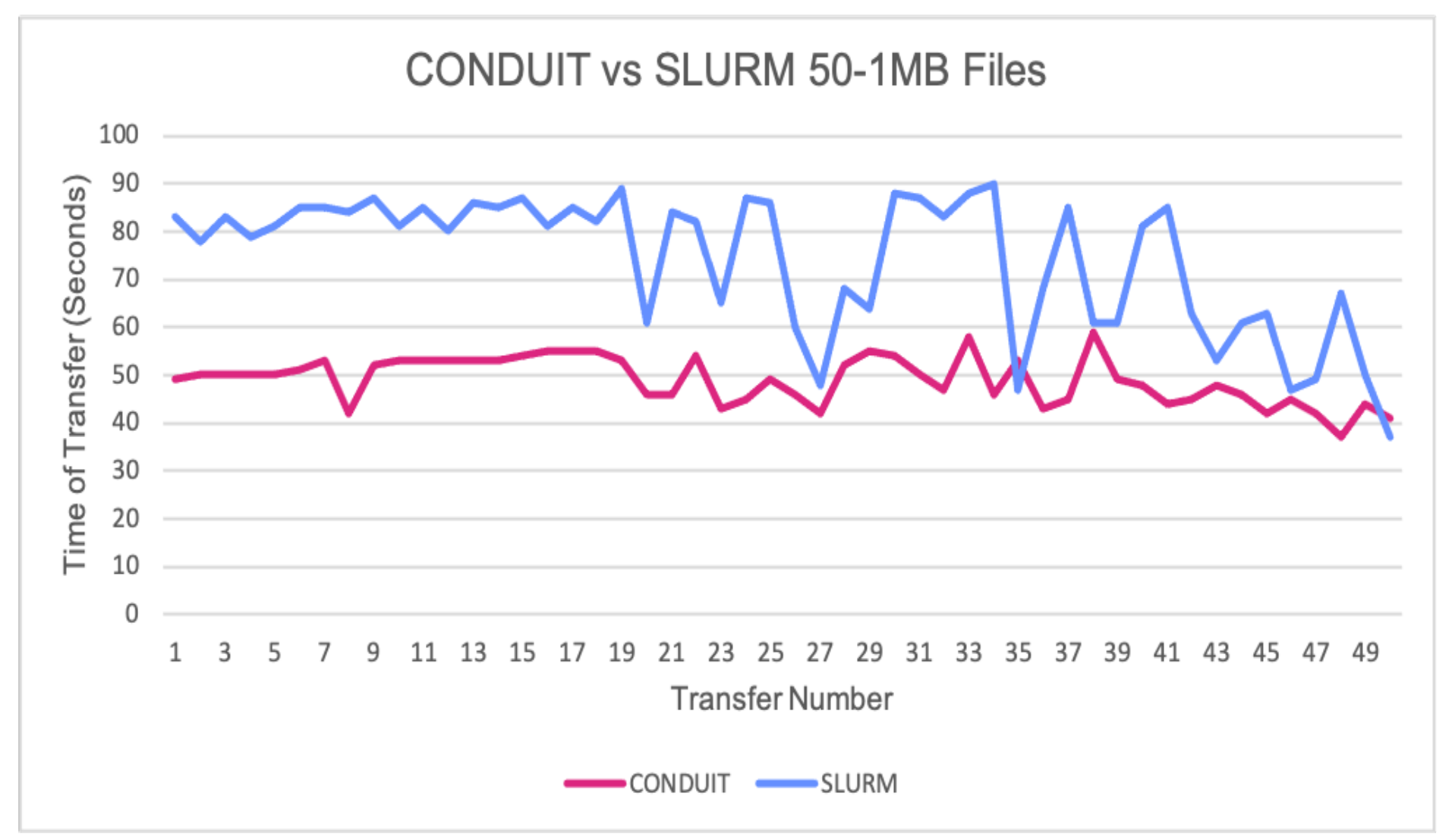

Figure 5. Comparison of CONDUIT scheduler and SLURM when a 1 MB transfer is being written to 50 files.

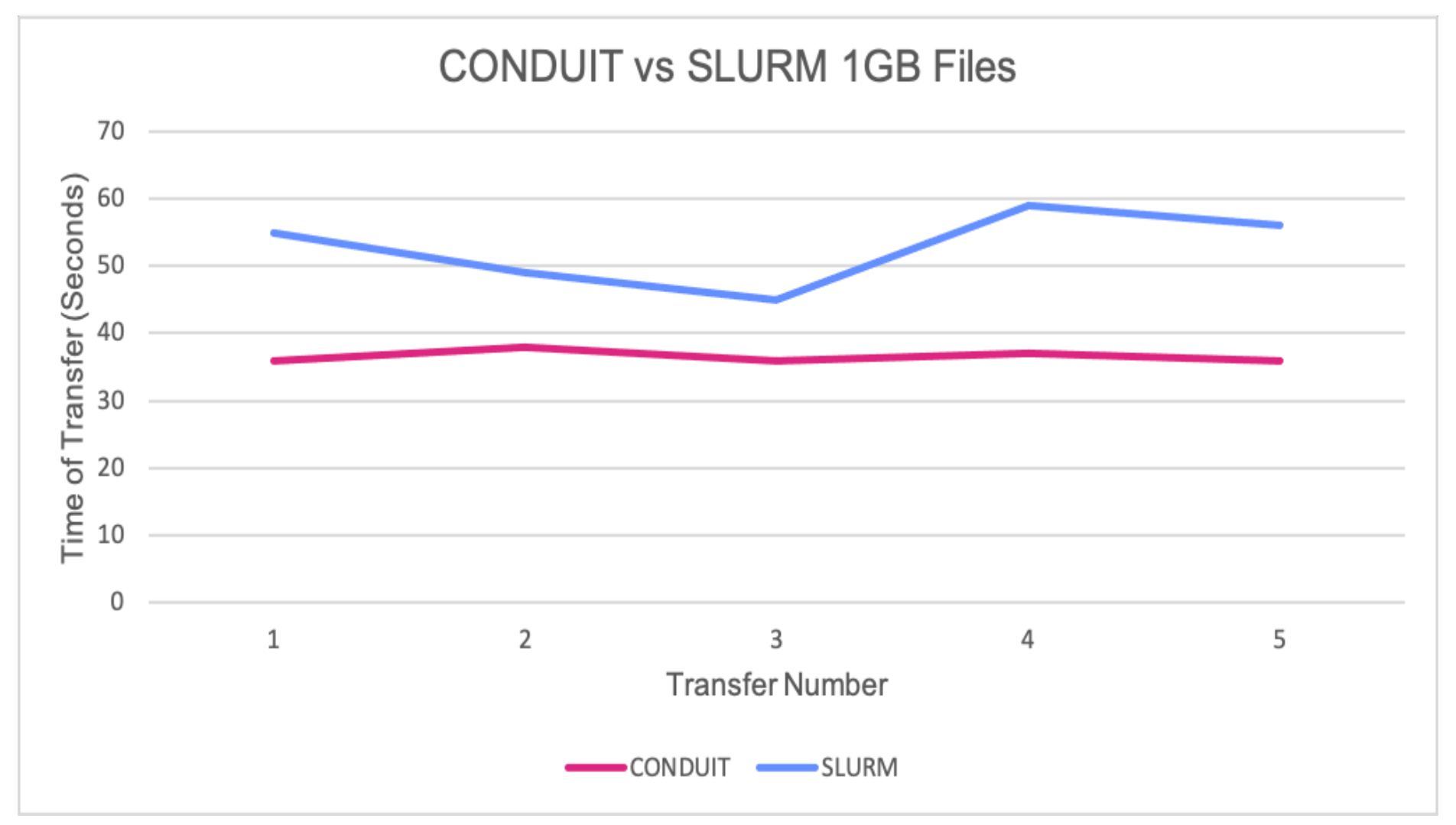

Figure 6. Comparison of CONDUIT scheduler and SLURM when a 1 GB transfer is being written to 5 files.

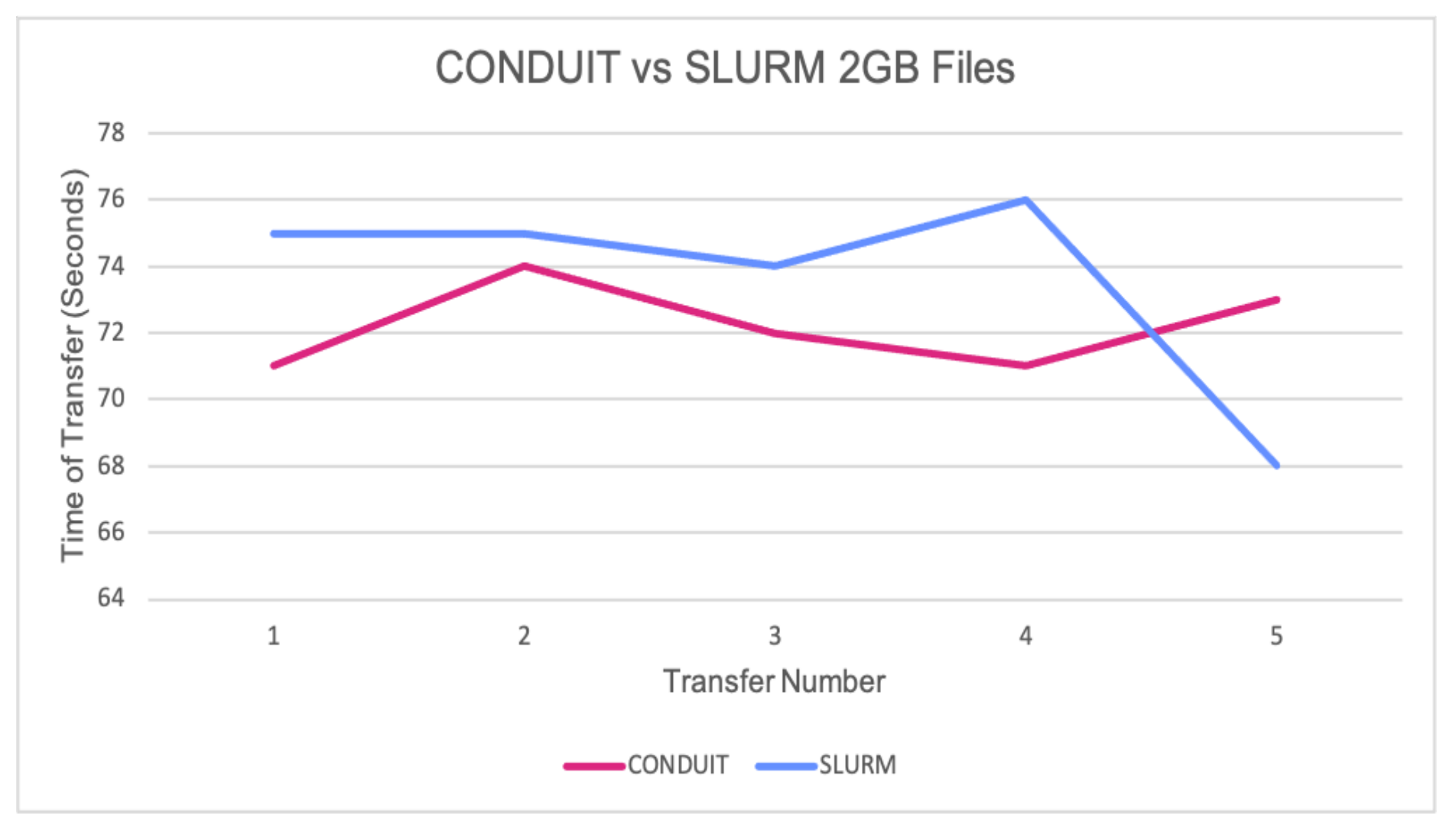

Figure 7. Comparison of CONDUIT scheduler and SLURM when a 2 GB transfer is being written to 5 files.

Motivation

Our primary purposes for developing a custom job scheduler mechanism for CONDUIT are as followed:

1. While SLURM provides a plethora of built in features, many of these features are not utilized in CONDUIT
2. The SLURM restAPI has shown signs of unreliability with periodic crashing and does not support TLS
3. By simplifying the CONDUIT scheduler, we increased performance while maintain the same end results

Conclusion

- CONDUIT job scheduler performed transfers faster than SLURM's job scheduler on an average of 28%
- The performance of pftool was similar between our scheduler and the SLURM scheduler
- Increase in responsiveness
- Increased speed in the validation, setup, teardown, and clean up stages
- Decreased complexity
- Improvements on stability and error reporting

Future Work

- More bug-fixing and testing
- Fully integrate the job scheduler into CONDUIT
- High availability with regard to the scheduler being able to run multiple instances of itself

LA-UR-23-28959