Holcombe Department of
ELECTRICAL AND COMPUTER ENGINEERING
Clemson University

# A Weak Scalability Study of File Aggregation in Asynchronous, Multi-Level Checkpointing

Mikaila Gossman, Bogdan Nicolae (advisor), Melissa C. Smith (advisor), Jon C. Calhoun (advisor)

Argonne NATIONAL LABORATORY

## Motivation: The Intersection of Checkpointing and I/O Aggregation

- Checkpointing captures a globally consistent state of an application during runtime for a variety of scenarios: fault tolerance, migration, on-demand preemption using suspend-resume, algorithms revisiting previous states (e.g., adjoint computations)
- Checkpoints need to be persisted to ***stable storage*** (e.g. PFS) with minimal overhead
- Asynchronous checkpointing techniques (e.g. VELOC [1]) reduce overhead by writing to local storage and then concurrently flush to stable storage in the background
- It is desirable to aggregate local checkpoints into a smaller number of files/objects on persistent storage (for easier management and/or better I/O performance)
- State-of-art I/O aggregation techniques are designed for synchronous I/O and is insufficiently explored in the context of checkpointing

## Contribution: Weak Scalability Study of I/O Aggregation

- Our custom benchmark spawns N (1 → 1024) total processes and performs a weak scalability study using 16 processes / compute node
- Initial studies presented at SC’21 (poster and paper at the SuperCHECK workshop) showed results obtained on small-scale run (N = 1 → 128) of weak and hard scalability studies
  - weak scalability study used 1 process / compute node
  - hard scalability iteratively divides 128 / $i$ , {$\forall\ i \in S \mid S = \{1,2,4,8,16,32\}$}
- This poster complements our previous studies with larger-scale weak scalability experiments

**More details available in our previous work**

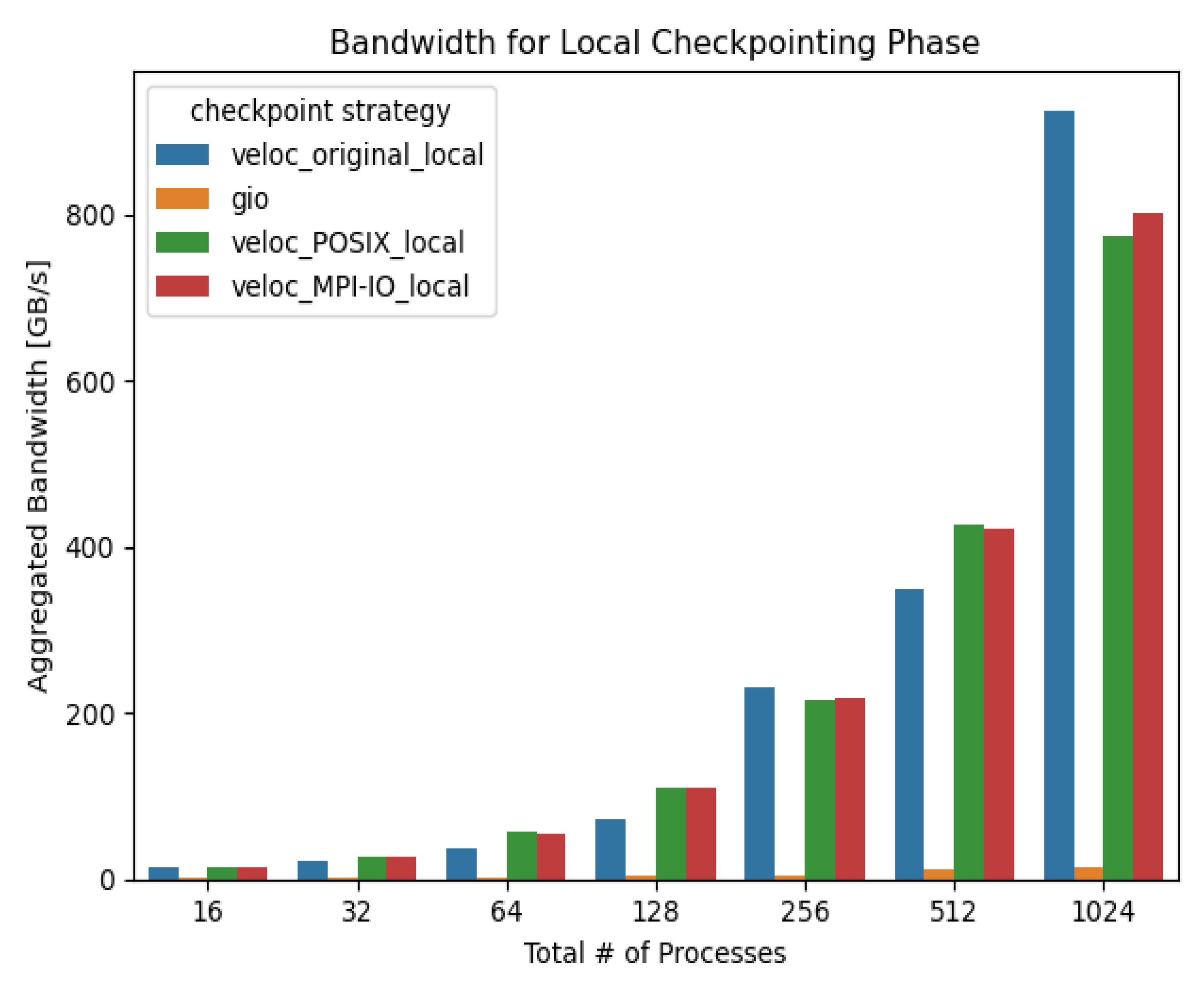
**SC’21 Poster**

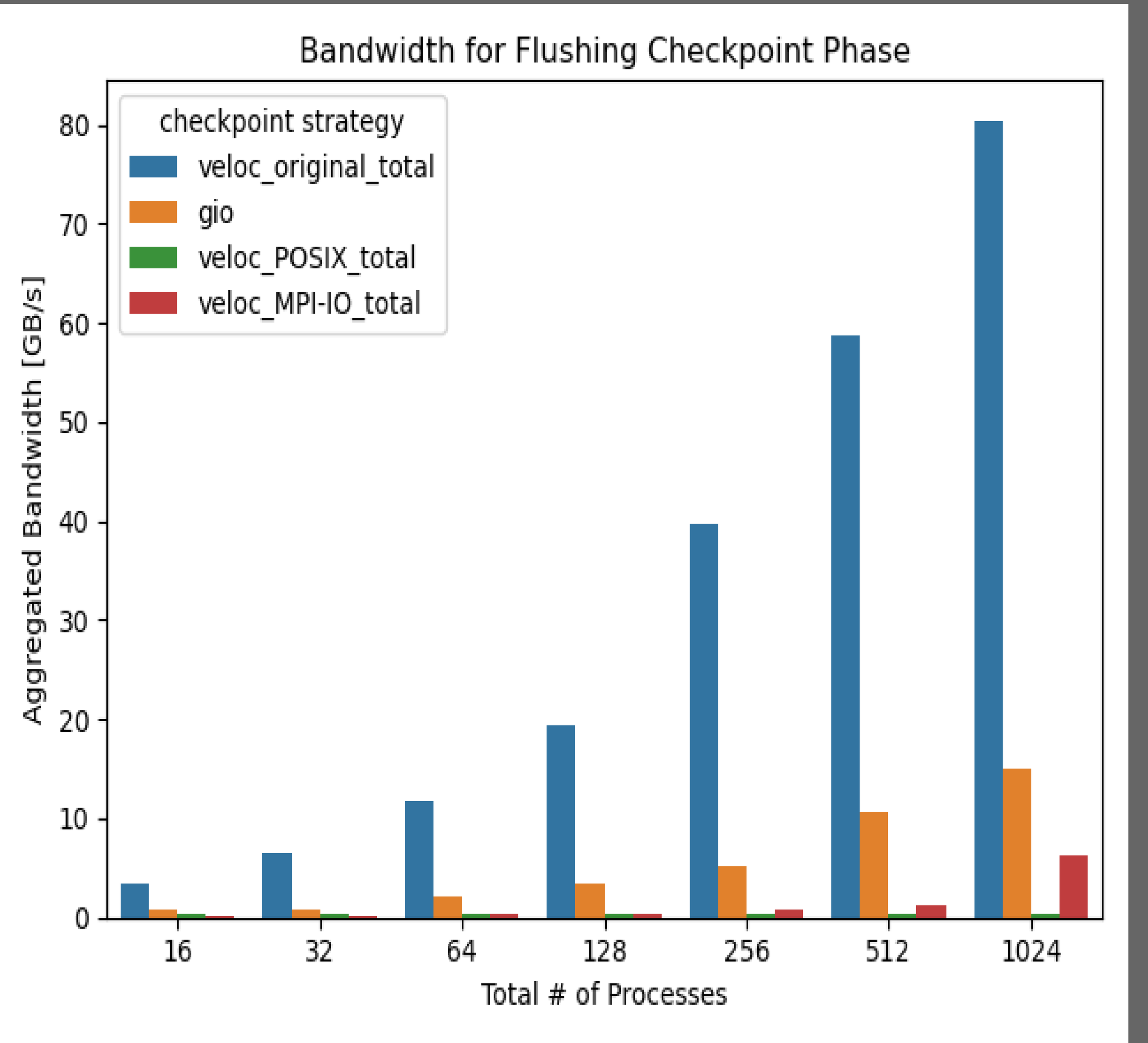
**SuperCHECK@SC'21**

## Experimental Setup

- Approaches: three aggregation strategies compared with no aggregation
  - **VELOC Default**: one file per process, (async)
  - **POSIX**: aggregates all checkpoints into 1 file; calculates each offset in global file via parallel-prefix-sum and issues a POSIX write for each checkpoint, (async)
  - **MPI-IO**: Same as POSIX but issues an MPI collective write for each checkpoint instead, (async)
  - **GenericIO**: aggregates all checkpoints into 1 file; single rank calculates total size and offset for each process and divides work among the processes, processes issue MPI collective writes to global file, (sync)
- Platform: Argonne’s Theta supercomputer
  - Persistent storage: Intel Enterprise Edition Lustre PFS (172 GB/s)
  - Local storage: in-memory temporary file system (/dev/shm)
- Weak scalability study: 16 processes / compute node, 1 GB checkpoint / process
  - Increasing number of processes (16 → 1024)

## Results




## Discussion

- The figure on the **left** shows the resulting aggregated bandwidth for the local checkpointing phase:
  - GenericIO (gio) in yellow does not use local storage resulting in significantly lower throughput
  - All VELOC methods, even with aggregation, mask the flushing speed to the application since it promptly resumes-thus we can further show that the prefix-sum operation (which requires synchronization) introduces negligible overhead
- The figure on the **right** shows the time it takes to write all checkpoints to the parallel file system (PFS):
  - Flushing speeds improved dramatically for VELOC methods compared to works presented at SC'21 by deactivating optional modules that operate during the flush phase (e.g. error correction)
  - MPI-IO performance continues to dominate POSIX performance throughout this study but still suffers serious performance degradation, as neither implemented aggregation strategies reach 1/10th of the peak write BW on Theta and even GenericIO barely reaches the 1/10th mark @ 1024 processes
- This study shows that VELOC's file-per-processes flushing style is the most efficient way to flush checkpoints
- A large gap is observable between the one-file-per-process async I/O strategy (as implemented by VELOC) and the other compared async I/O aggregation strategies, which presents a research opportunity

## Conclusions and Future Work

- MPI-IO is optimized for synchronous parallel I/O to aggregate data into single files
- POSIX (and object-store based APIs) offers the most performance potential and flexibility for implementing optimized asynchronous aggregation strategies
- We are currently working to develop a novel, resource-aware aggregation strategy designed to meet the specific challenges of asynchronous checkpointing (i.e. resource contention)

ACKNOWLEDGEMENTS:
Clemson University and Argonne National Laboratory facilitated the experimental study thanks to the Palmetto cluster and, respectively, the Theta supercomputer. This material is based in part upon work supported by the National Science Foundation under Grant No. SHF-1910197 and the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11.357

FIND ME

CHECKOUT OUR CODE