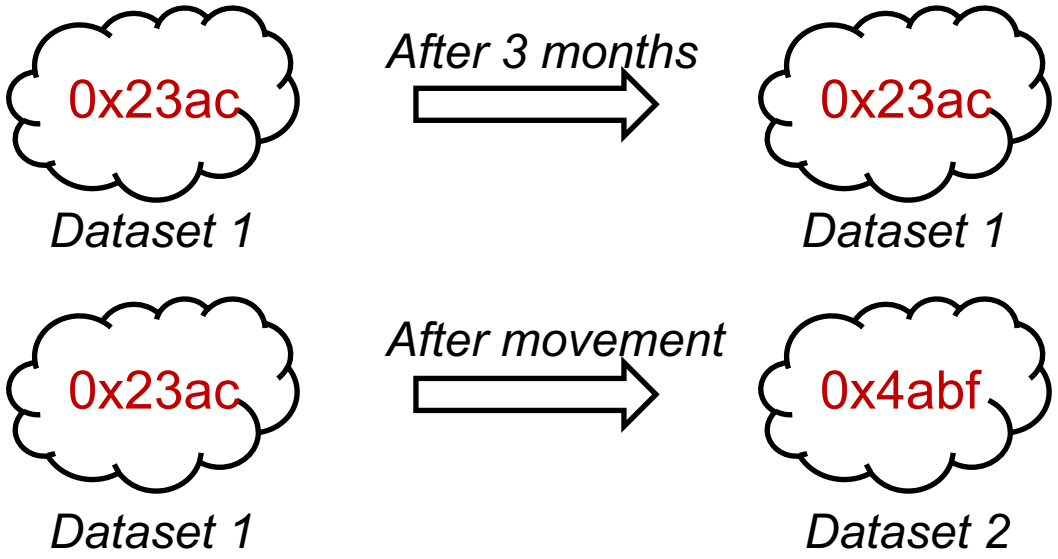
A Bloom Filter Based Scalable Data Integrity Check Tool for Large-scale Dataset

Sisi Xiong*, Feiyi Wang[+] and Qing Cao*
*University of Tennessee Knoxville, Knoxville, TN, USA
[+]Oak Ridge National Laboratory, Oak Ridge, TN, USA

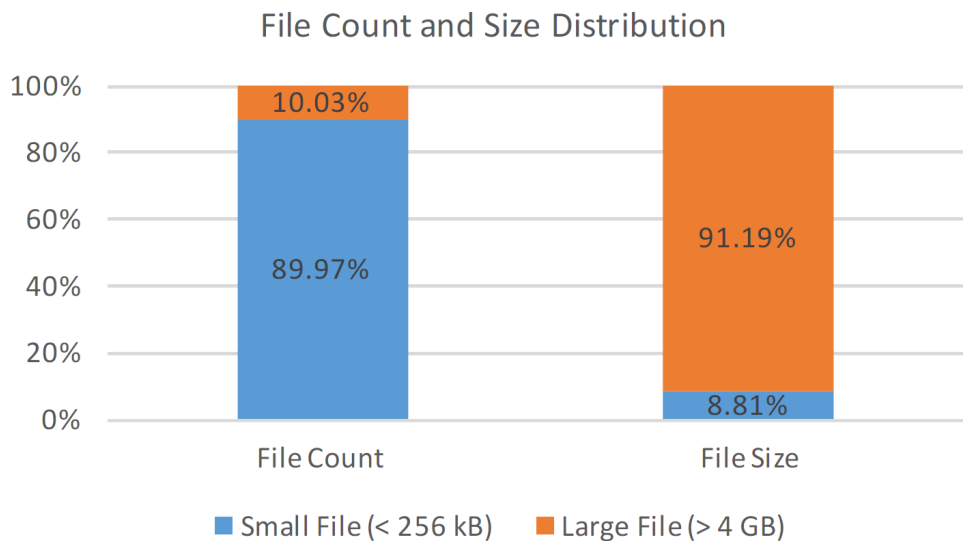
Data integrity check

- **Motivations**
  - Silent data corruption
  - Data movement
- **Checksumming**
  - Generate a signature for a dataset



Scalability

- **Traditional approaches**
  - Serial and file based
- **File distribution on Spider 2***
  - 50 million directories, half a billion files



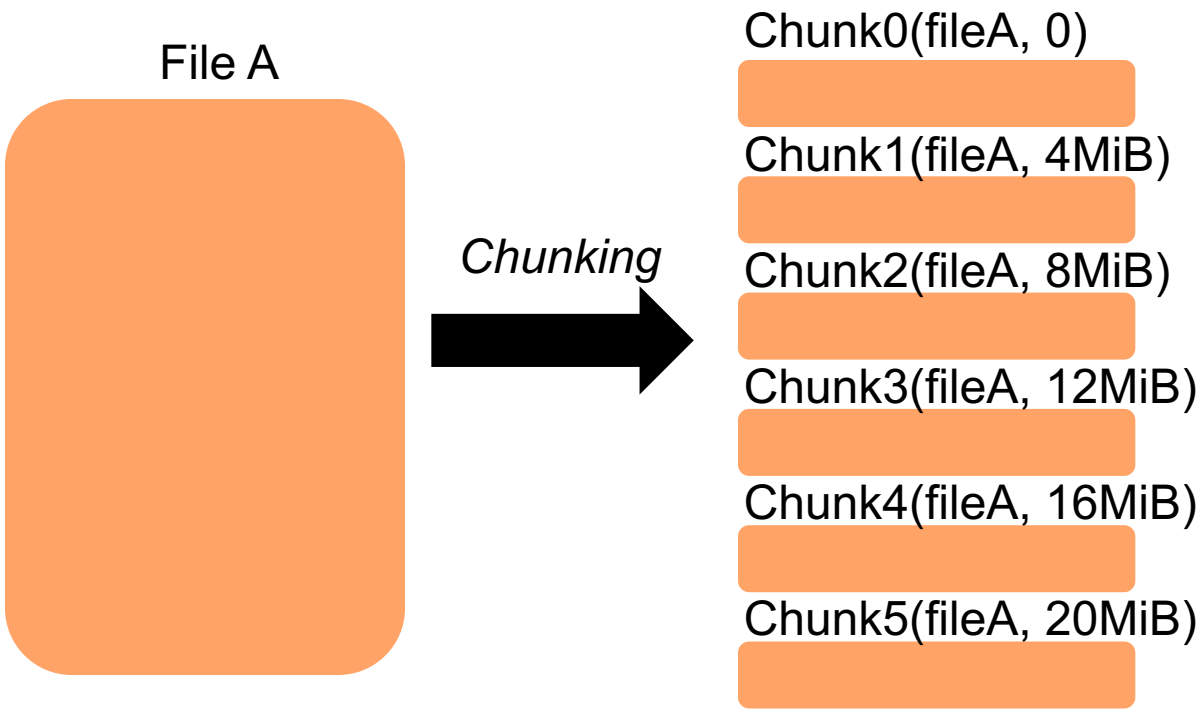
*Feiyi Wang, Veronica G. Vergara Larrea, Dustin Leverman, Sarp Oral. FCP: A Fast and Scalable Data Copy Tool for High Performance Parallel File Systems

Design goals

- **Develop a parallel checksumming tool**
  - Use multiple processes/hosts to achieve horizontal scaling
- **Generate signatures for large-scale datasets**
  - A two-step task:
    - Generate a signature for each file
    - Aggregate all the file-level signatures to a dataset-level signature
- **The resulting design: *fsum***

File-level signature vs chunk-level signature

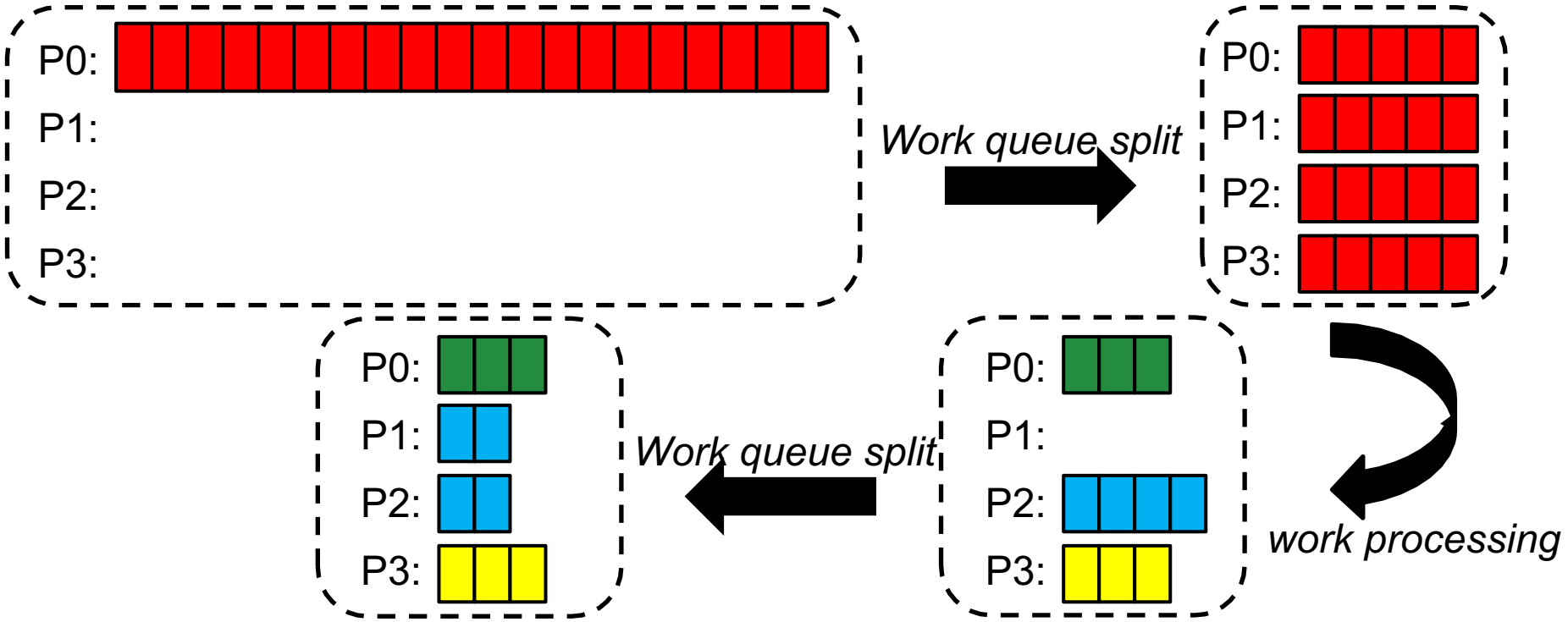
- **Increase parallelism: break files into chunks**



Chunk size: 4MiB

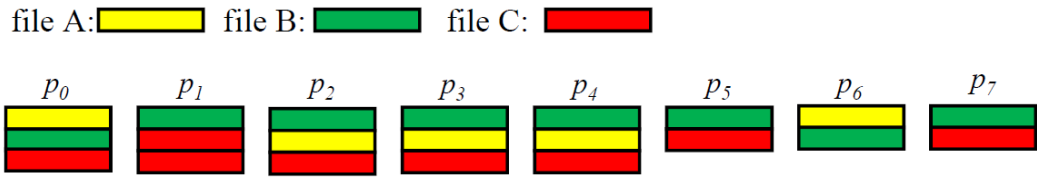
Workload distribution

- **Work stealing pattern**
  - An idle process sends out work request
  - A busy process distributes its work queue equally

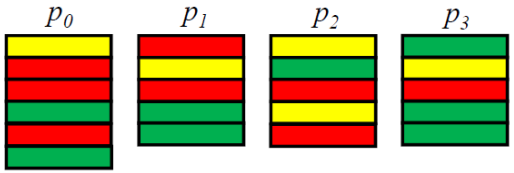


Work stealing pattern

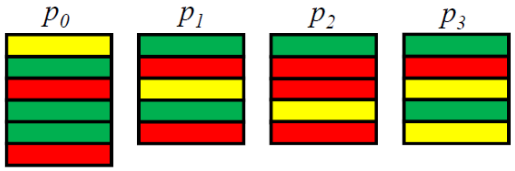
- **Random work distribution**
  - Same/different number of processes



Work distribution among 8 processes



Work distribution among 4 processes

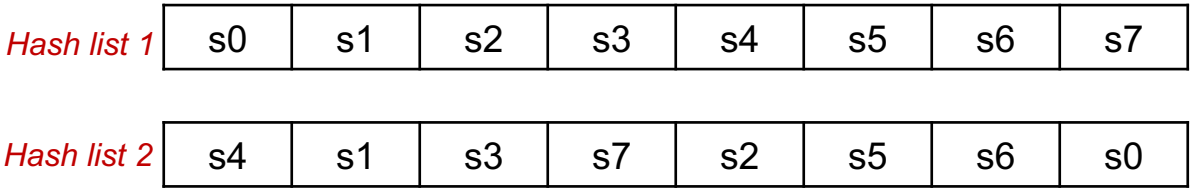


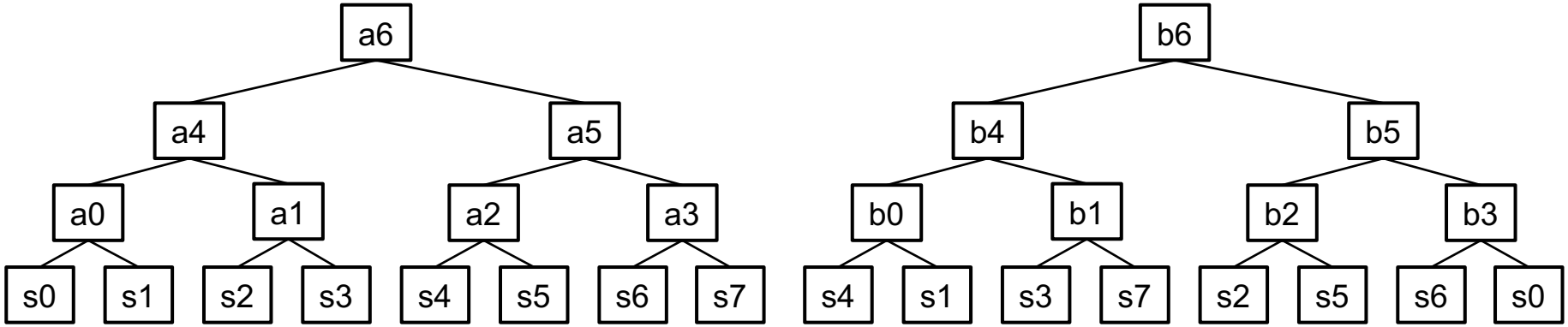
Work distribution among 4 processes, another execution

Sorted chunk-level checksums

Aggregation of chunk-level signatures

- **Two possible solutions:**
  - Hash list
  - Merkle tree: hash tree
- ***Sorting is necessary!***

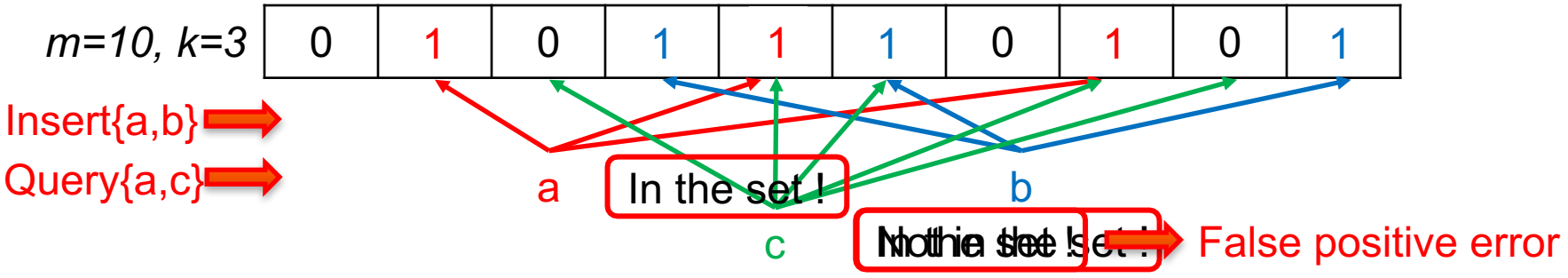




Hash tree1 *Hash tree2*

Bloom filter based signature aggregation approach

- **Bloom filter**
  - An array of bits, initialized to all 0s
  - Check membership
  - Two operations: insert and query
- **Configuration parameters**
  - Size: m
  - Number of hash functions: k



Bloom filter based signature aggregation approach

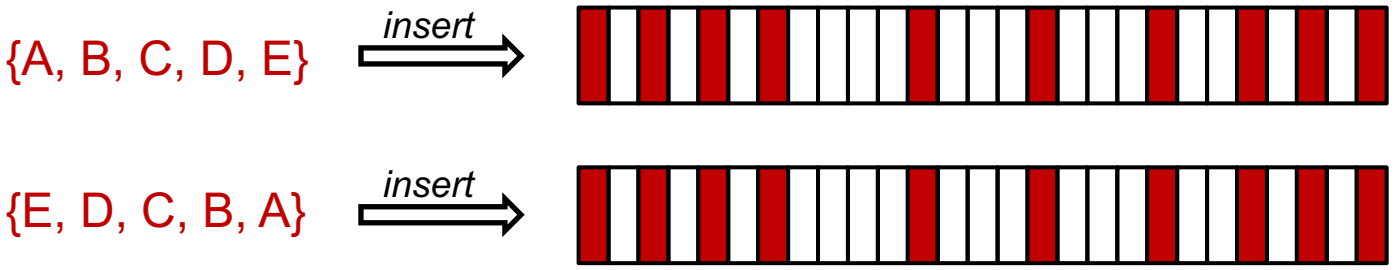
- **Probabilistic nature: errors**
  - False negative error never happens
  - False positive error - with a probability ***p***:
    - m: bloom filter size, n: number of elements
    - Given n, larger m results in a smaller p.
    - Give p, m increases linearly with respect to n

$$p = e^{-\frac{m}{n}(\ln 2)^2}$$

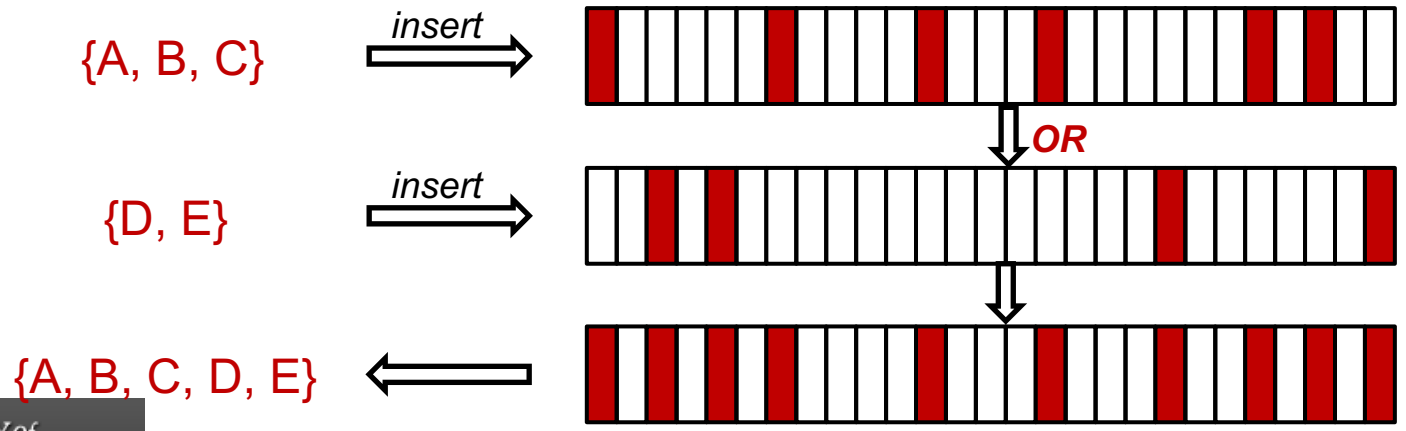
$$m = -n\frac{\ln p}{(\ln 2)^2}$$

Bloom filter based signature aggregation approach

- **Features**
  - Independent of insertion orders

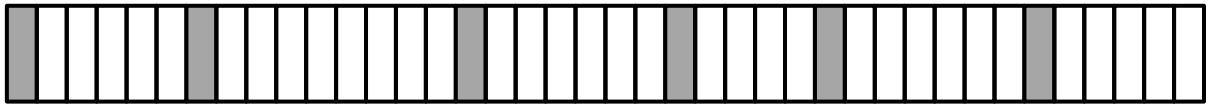


  - Use OR result to represent the union of multiple sets

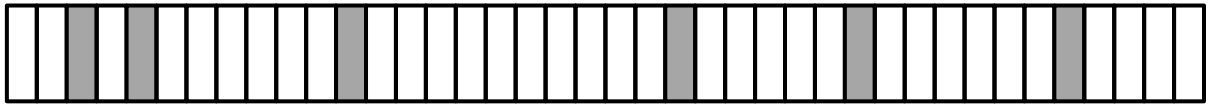


Bloom filter based signature aggregation approach

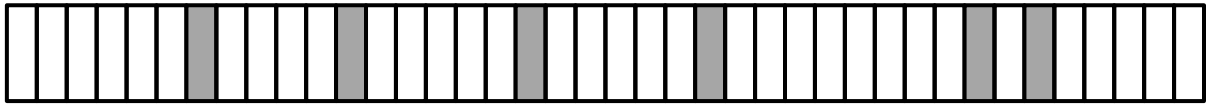
- P0: {(file1chunk1, s), (file2chunk2, s), (file2chunk3, s), (file3chunk2, s)}



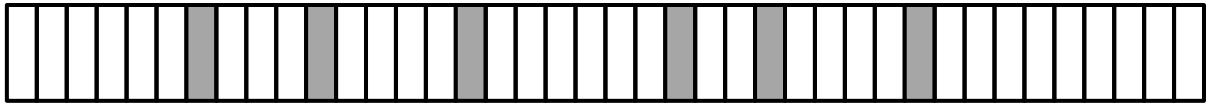
- P1: {(file1chunk2, s), (file1chunk3, s), (file2chunk1, s), (file3chunk4, s)}



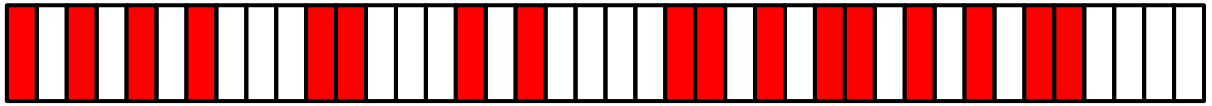
- P2: {(file1chunk4, s), (file1chunk6, s), (file2chunk6, s), (file3chunk3, s)}



- P3: {(file1chunk5, s), (file2chunk4, s), (file2chunk5, s), (file3chunk1, s)}

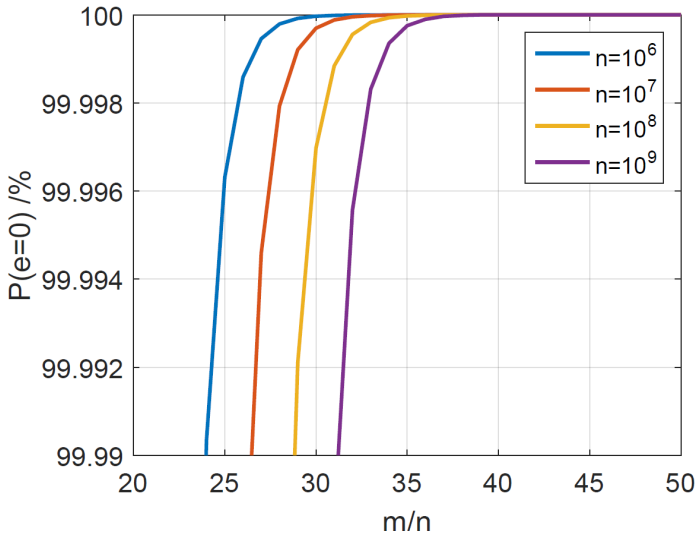


- OR result



Bloom filter based signature aggregation approach

- **Parameter setting and error probability**
  - Error case: two different datasets have the same signature
  - Suppose there are r different signatures
    - $C_1 = \{x_1, x_2, \dots, x_r, c_r, c_{r+1}, \dots, c_n\}$
    - $C_2 = \{y_1, y_2, \dots, y_r, c_r, c_{r+1}, \dots, c_n\}$
    - Given r, the error probability is p^{2r}, p is false positive probability of bloom filter
  - Worst case: r = 1
    - Number of errors follows a binomial distribution $e \sim B(n, p^2)$
    - $P(e = 0) = (1 - p^2)^n$
    - $p = e^{-\frac{m}{n}(\ln 2)^2}$
    - The relationship between P(e=0) and m/n



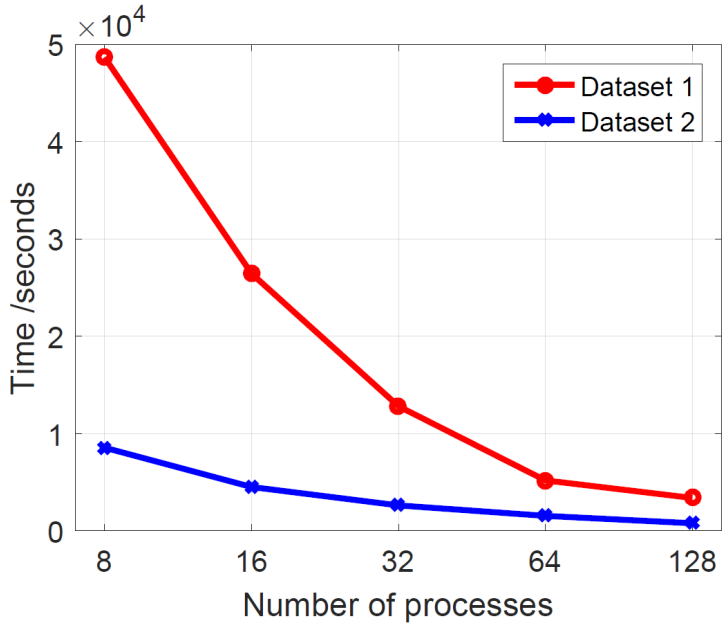
Evaluation

- **Two datasets**
  - Scientific datasets generated on Spider 2

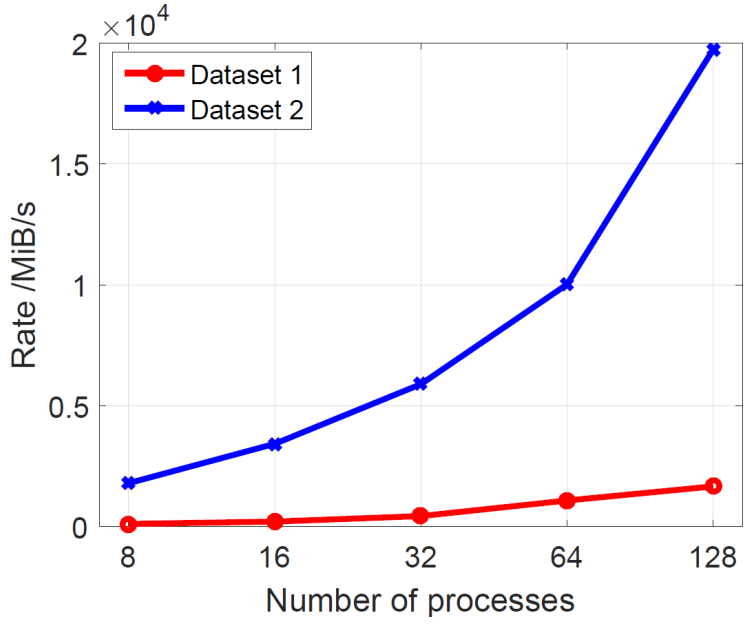
	Dataset 1	Dataset 2
Total size	5.39TiB	14.74TiB
Number of files	28,114,281	15,590
Average file size	205.83KiB	1.19GiB
Chunk size	16MiB	64MiB
Number of chunks	28,343,725	251,629

Evaluation

- **Scalability test**
  - Higher process rate when using more processes, until bounded by I/O bandwidth
  - Handling large files is more efficient than handling small files, potentially bounded by metadata retrieval in Lustre file system



Processing time



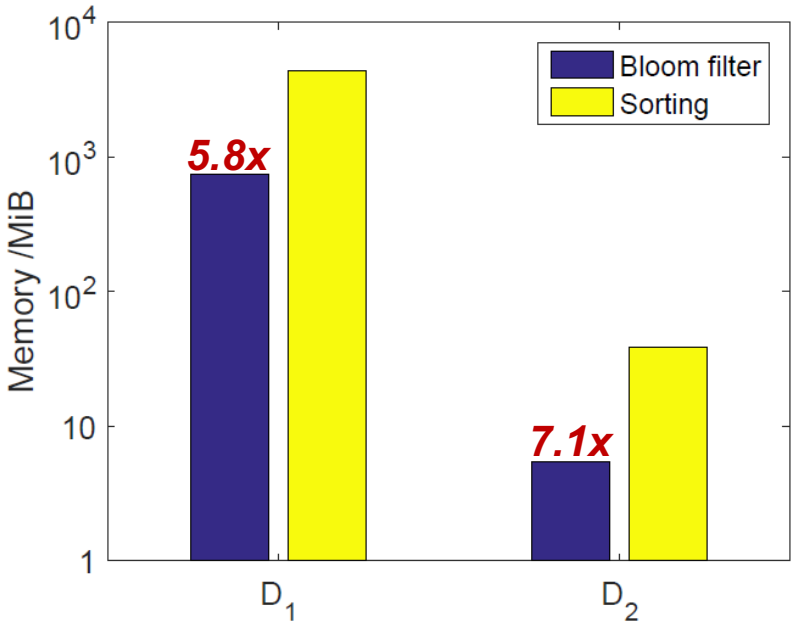
Processing rate

Evaluation

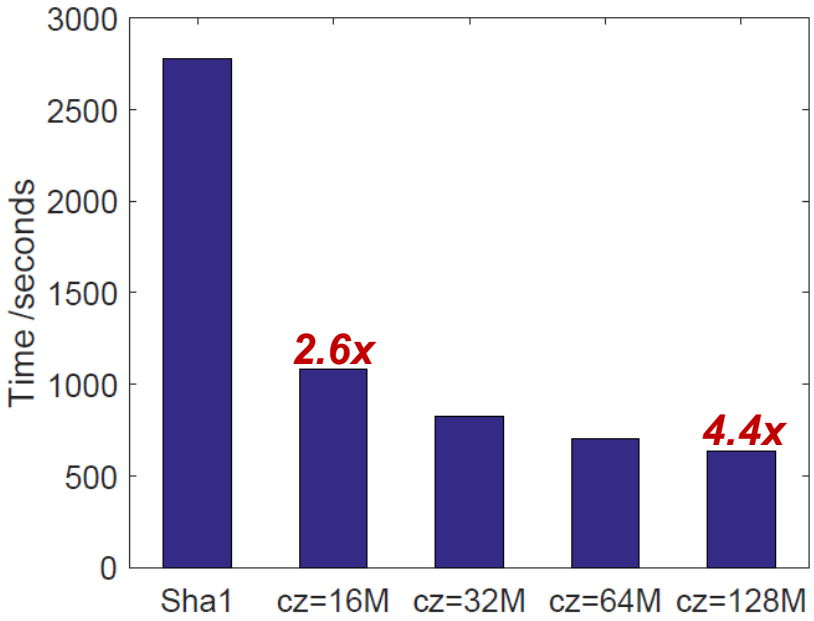
- **Function verification**
  - Corrupt the dataset
    - 100 times, corrupt a single byte in a single file each time
  - Compare two signatures of the original and the corrupted dataset
  - Results: all the signatures are different

Evaluation

- **Compare with related approaches**



Memory usage compared with sorting (P(e=0) = 99.99%)



Runtime compared with sha1 (a 500GiB file, 4 processes)

Conclusions

- Present the design and implementation of scalable parallel checksumming tool, ***fsum***, for large scale datasets
- Design a bloom filter based signature aggregation approach and analyze the relationship between error probability and parameter selection
- Test on representative and real production datasets and demonstrate that ***fsum*** exhibits near-linear scalability and is able to detect data corruption
- ***fsum*** is both memory and time efficient than other approaches

Code available at github.com/olcf/pcircle

Thanks!