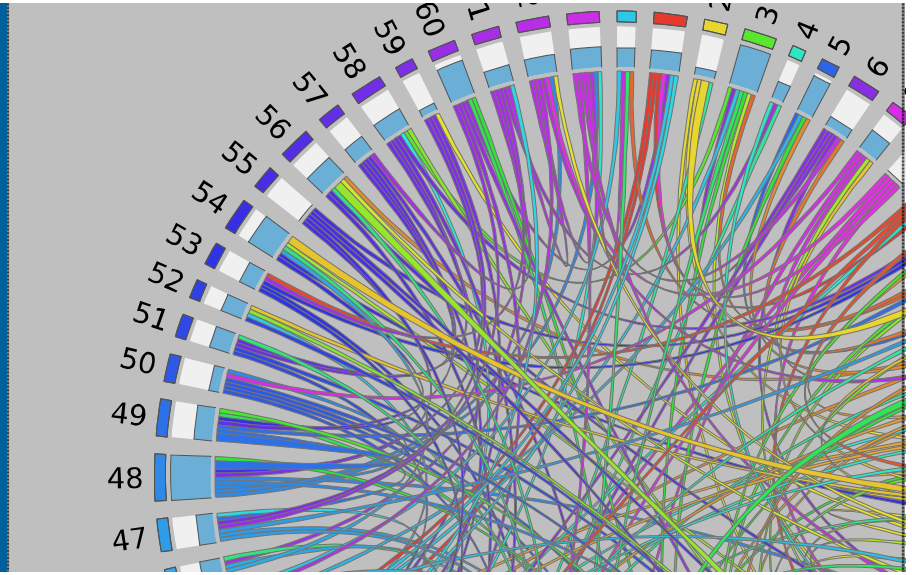
32ND INTERNATIONAL CONFERENCE ON MASSIVE STORAGE SYSTEMS AND TECHNOLOGY (MSST 2016)

Argonne NATIONAL LABORATORY

IMPACT OF DATA PLACEMENT ON RESILIENCE IN LARGE-SCALE OBJECT STORAGE SYSTEMS



PHILIP CARNS
Argonne National Laboratory
carns@mcs.anl.gov

KEVIN HARMS
JOHN JENKINS
MISBAH MUBARAK
ROBERT ROSS
Argonne National Laboratory

CHRISTOPHER CAROTHERS
Rensselaer Polytechnic Institute

May 6, 2016
Santa Clara, CA

MOTIVATION

Distributed object storage is an essential building block for large-scale data processing

- Replication is often employed to achieve resilience on commodity hardware
- Replicated systems must rebuild quickly after failures to limit MTTDL
- This leads to critical evaluation questions:
  - How long will it take to recover from a failure?
  - What are the weakest links in the architecture or algorithm?
  - Do data set characteristics affect performance?
- These questions are important but increasingly difficult to answer at scale:
  - Data paths and dependencies are more complex
  - Rigorous measurement of deployed systems requires considerable time and resources

APPROACH

Parallel Discrete Event Simulation with CODES and ROSS

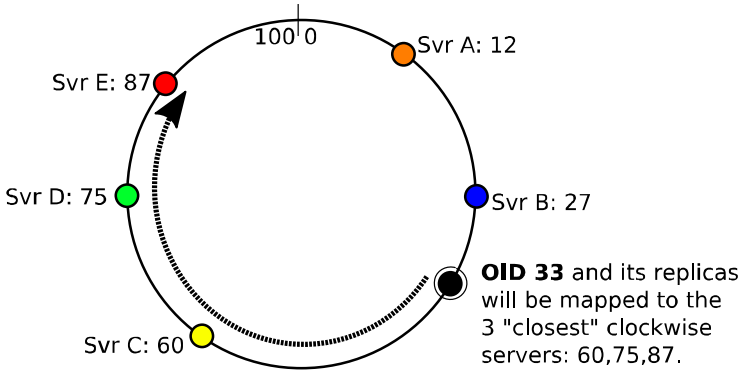
- CODES: Co-Design of Exascale Storage Architectures and Science Data Facilities
  - Toolkit for discrete event simulation of large storage and network systems
  - Modular configuration of algorithms, workloads, and hardware components
  - Includes several validated sub-models
- ROSS: Rensselaer's Optimistic Simulation System
  - Parallel discrete event simulator underlying CODES
  - Uses “Time Warp” synchronization to achieve scalable performance
- CODES and ROSS enable detailed design space exploration. In this case:
  - Real-world data population parameters
  - Simulate O(thousand) servers, O(billions) objects, O(petabytes) of data
  - Use device parameters (JBOD and IB) drawn from commodity data centers
  - Existing placement algorithms

See paper for model validation details

REBUILD MODEL

- We focus on the simulation of a critical scenario:
  - Initial state: a collection of servers storing a large replicated object population
  - **One random server fails**
  - Simulate the data transfers necessary to rebuild missing replicas
- Object placement is crucial to performance

Basic object placement example: consistent hashing



- Placement algorithms with good ***declustering*** properties enable the system to leverage more aggregate bandwidth during rebuild
- We used CRUSH [1] as our baseline :
  - Algorithmic and deterministic
  - Hierarchical organization of resources
  - Pluggable “bucket” algorithms
  - Flexible placement rules

[1] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, “CRUSH: Controlled, scalable, decentralized placement of replicated data,” in Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (SC06)

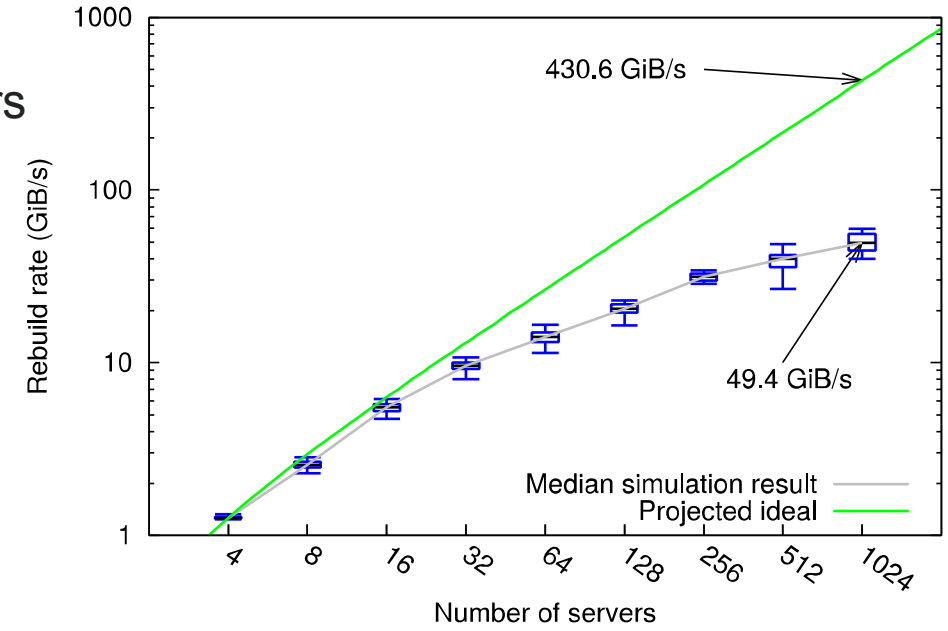
EXAMPLE CASE STUDY

Argonne NATIONAL LABORATORY

AGGREGATE REBUILD BANDWIDTH EXAMPLE

CRUSH straw bucket placement algorithm with placement groups

- System:
  - Generalized object storage model
  - Data can be streamed between pairs of servers at ~1.5 GiB/s
  - Vary the server count and data volume
- Data set:
  - Extrapolated from “1000 Genomes” [2] file size characteristics
  - 60 TiB (counting replication) of data per server
- Graph:
  - Shows aggregate rebuild rate on a log-log scale
  - Ideally, aggregate rebuild bandwidth would increase linearly as more servers are added



Simulated rate tracks ideal rate roughly at small scale, but not at large scale

[2] 1000 Genomes Project Consortium and others, “A map of human genome variation from population-scale sequencing,” Nature, vol. 467, no. 7319, pp. 1061–1073, 2010

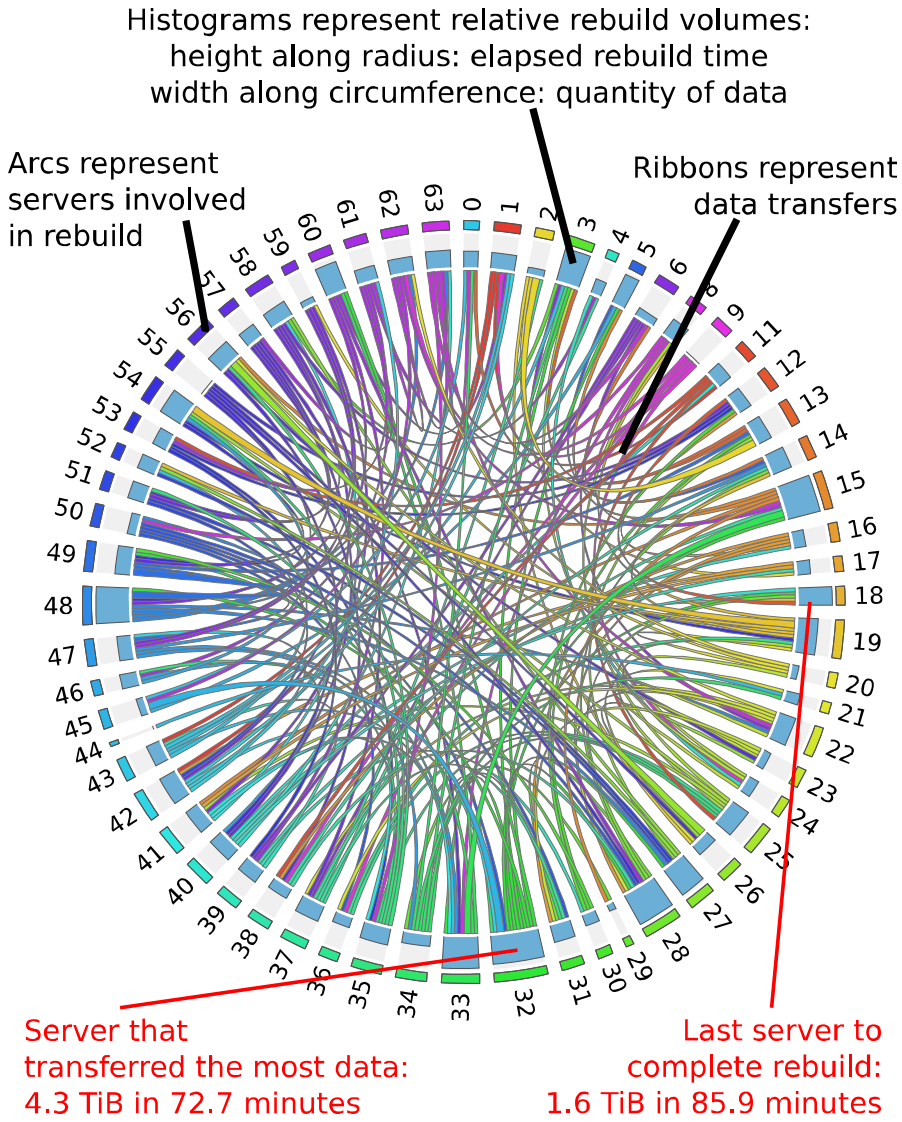
Argonne NATIONAL LABORATORY

AGGREGATE REBUILD

A closer look at inter-server traffic

- We examine the slowest 64-server sample in greater detail
- Plot the data transfers between pairs of servers using Circos [3]
- Server “10” is not shown: it is the failed server in this example

- The servers began the simulation with even utilization…
- … but traffic during rebuild is poorly balanced
- Servers with more active peers were generally able to sustain a higher rate

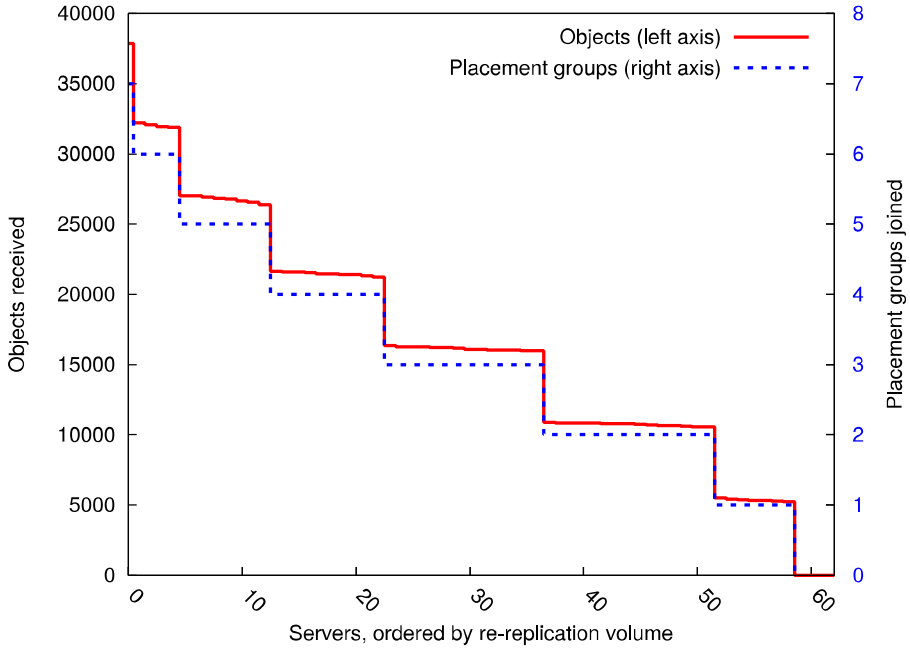


[3] Krzywinski et al., “Circos: An information aesthetic for comparative genomics,” Genome Research, vol. 19, no. 9, pp. 1639–1645, 2009.

AGGREGATE REBUILD

Where were objects reconstructed?

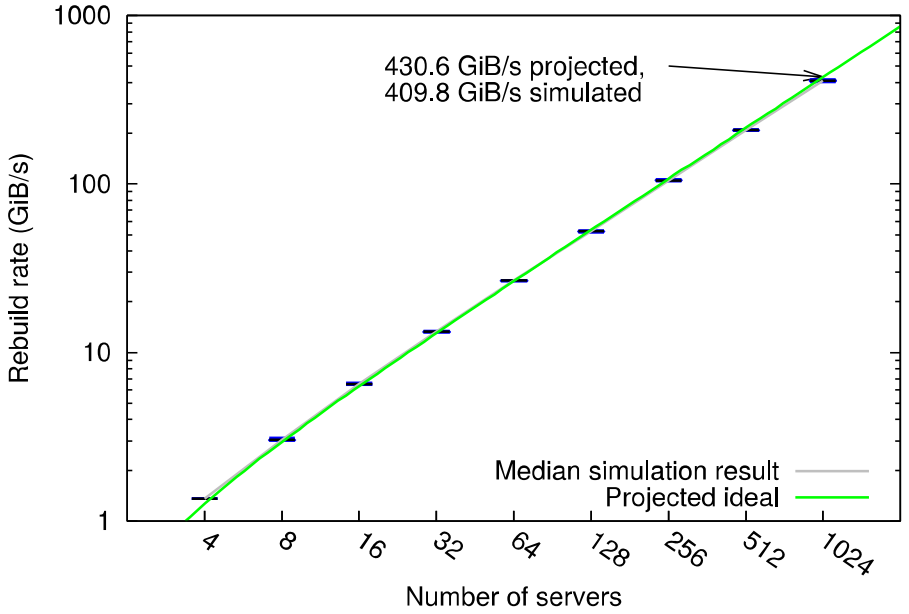
- Histogram (red) shows the number of *replicas* rebuilt per server
- Overlayed with the number of *placement groups* rebuilt per server
- The simulation followed the example of usage of CRUSH in Ceph:
  - Objects are mapped into a smaller number of placement groups
  - Placement group IDs are mapped to servers using CRUSH
  - Many objects share the same mapping to reduce placement cost



- The failed server in this example participated in 190 out of 4096 PGs
  - Pseudo-random distribution: one server took responsibility reconstructing 7 PGs, while four servers took responsibility for no PGs
- Imbalance of replica targets led to imbalance in data transfers

TUNING PLACEMENT TO IMPROVE AGGREGATE REBUILD RATE

- Can this be improved?
- We repeated the experiments with the same data set, same number of servers, and same hardware parameters, but with the following changes:
  - Eliminated placement groups (each object is placed independently)
  - Added a new bucket algorithm based on Chord-style consistent hashing algorithm with virtual nodes
- System achieves much higher and more consistent aggregate rebuild rate with object-granular placement
- New bucket algorithm is more computationally efficient while retaining key properties of CRUSH straw bucket



CASE STUDY DISCUSSION

Findings

- Sensible object placement policies at small scale can have unexpected consequences at large scale
- Object-granular replication enables near-ideal scalability in distributed rebuild
- Existing consistent hashing algorithms can be adapted for use in CRUSH to reduce CPU costs
- Simulation methodology was effective for design space exploration

Impact

- How would a file system be implemented by changing the placement granularity?
  - Ceph notably uses placement groups for a variety of purposes beyond placement calculation: also impacts peering, write-ahead logging, and fault detection, for example
  - Our simulation does not encompass the entire file system design
- Are there other benefits to object-granular placement?
  - Potential for fine-grained prioritization or scheduling of object reconstruction

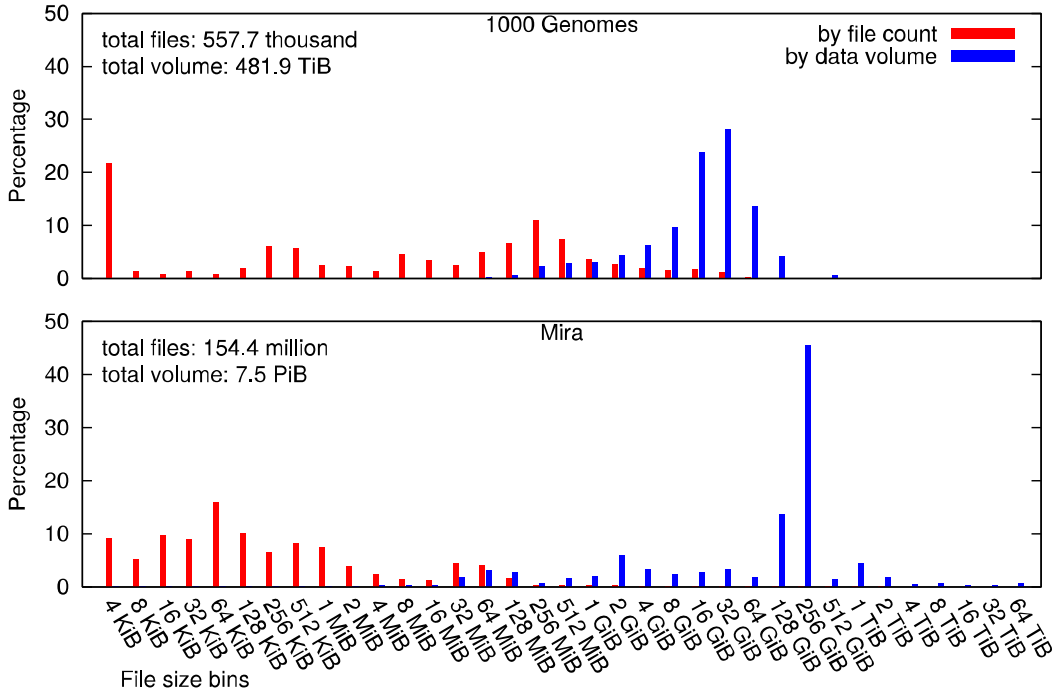
THE IMPACT OF DATA POPULATION CHARACTERISTICS

Argonne NATIONAL LABORATORY

CONTRASTING REAL-WORLD DATA POPULATIONS

The file-level perspective

- File size histogram comparing relative file counts and data volume
- Top: 1000 Genomes dataset (used in previous case study)
- Bottom: Mira file system (GPFS storage for IBM Blue Gene / Q system)

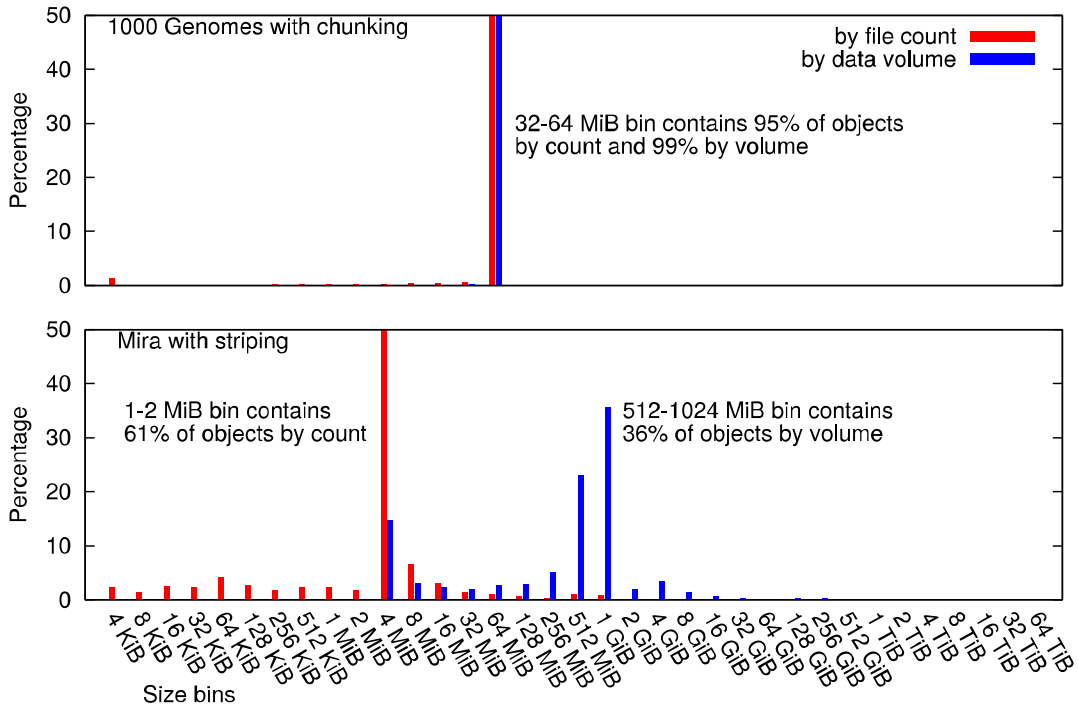


- Both exhibit a large *count* of small files, but most of the actual data *volume* is stored in large files
- On Mira, files between 256 and 512 GiB hold more data than any other file size bin

CONTRASTING REAL-WORLD DATA POPULATIONS

The object-level perspective

- This histogram shows the same data set as the previous slide…
- …but the histograms are in terms of underlying *object* sizes rather than file sizes
- 1000 Genomes: files are split into 64 MiB objects according to typical MapReduce strategy
- Mira: files are widely striped in round-robin fashion

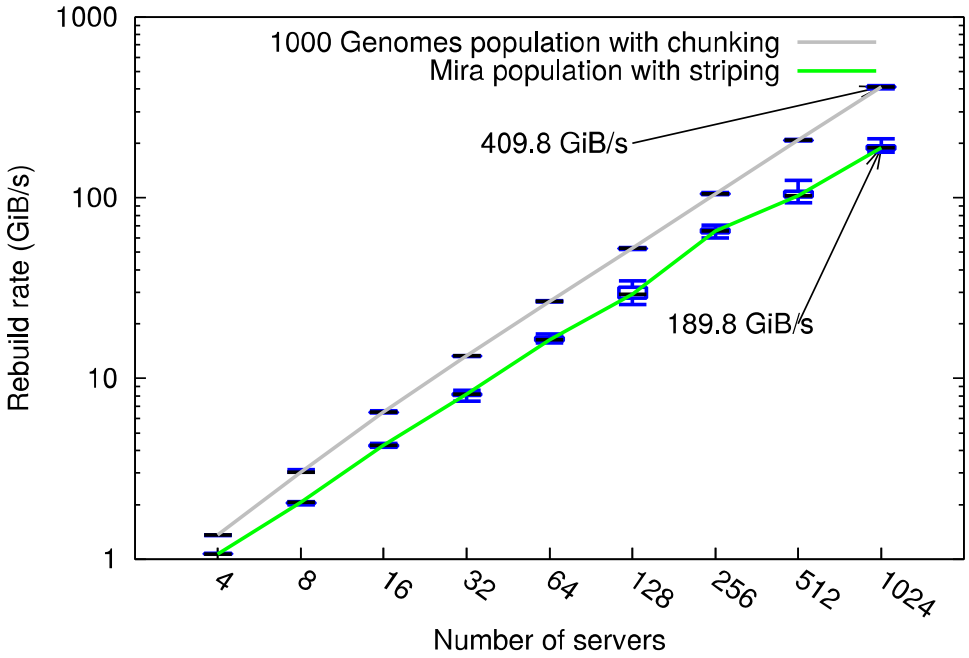


- This distinction in file decomposition leads to a pronounced difference in object size distribution
  - Top example dominated by a single bin: 64 MiB objects
  - Bottom example dominated by much larger objects

CONTRASTING REAL-WORLD DATA POPULATIONS

The rebuild algorithm perspective

- This plot shows the aggregate rebuild rate for both dataset examples
- Similar trends in performance as system scale increases, but 1000 Genomes examples is 2x faster



- Two notable reasons:
  - Mira data set has a higher proportion of small objects that cause lower messaging efficiency (ratio of control msg to data msg traffic; seek costs)
  - Extraordinarily large Mira objects (up to 100s of GiB) dominate transfers between pairs of servers and cause bottlenecks
- Data population characteristics can have a surprising impact on performance

ASSESSING THE METHODOLOGY

Argonne NATIONAL LABORATORY

THE USE OF PDES FOR ANALYSIS OF DISTRIBUTED STORAGE ALGORITHMS

- Simulation approach offered a number of advantages:
  - Ability to evaluate scenarios that would be difficult to recreate in a real-world test environment
  - Fast turn-around time enabled ensemble studies (see box-and-whisker plots) to discriminate typical behavior from outliers
  - Object and message level granularity allowed us to evaluate realistic, non-idealized data sets and account for transport efficiency and seek time
- Did we really need to run it in parallel?
  - Largest simulations tracked 3.9 billion replicas and issued over 200 million discrete events to rebuild a subset of them
  - We executed this scenario in roughly ~30 seconds with 256 MPI processes
  - The same model would not execute in serial at all due to memory limitations
  - We put more effort into model validation than performance tuning; more speed is likely possible

SUMMARY AND FUTURE WORK

We used parallel discrete event simulation to study the performance of replicated object storage reconstruction at scale. Key factors in performance included:

- Object placement algorithm (and it's declustering characteristics)
- Granularity of placement
- Nature of the data stored on the system

Possible future directions:

- Evaluate more complex failure modes
- Study additional hardware parameters and architectures
- What about erasure coding?
- The "big picture" of storage system design beyond rebuild behavior

THE CODE

All tools used in this study are available with permissive open source licenses:

- Libch-placement (placement algorithm library and CRUSH patch): https://xgitlab.cels.anl.gov/codes/ch-placement
- Codes-rebuild model (model of distributed object rebuild): https://xgitlab.cels.anl.gov/codes/codes-rebuild
- CODES project: http://www.mcs.anl.gov/projects/codes/
- ROSS project: http://carothersc.github.io/ROSS/

THANK YOU!

This material was based upon work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computer Research, under contract DE-AC02-06CH11357. The research used resources from the Argonne Leadership Computing Facility (ALCF).

Speaker: Phil Carns
carns@mcs.anl.gov