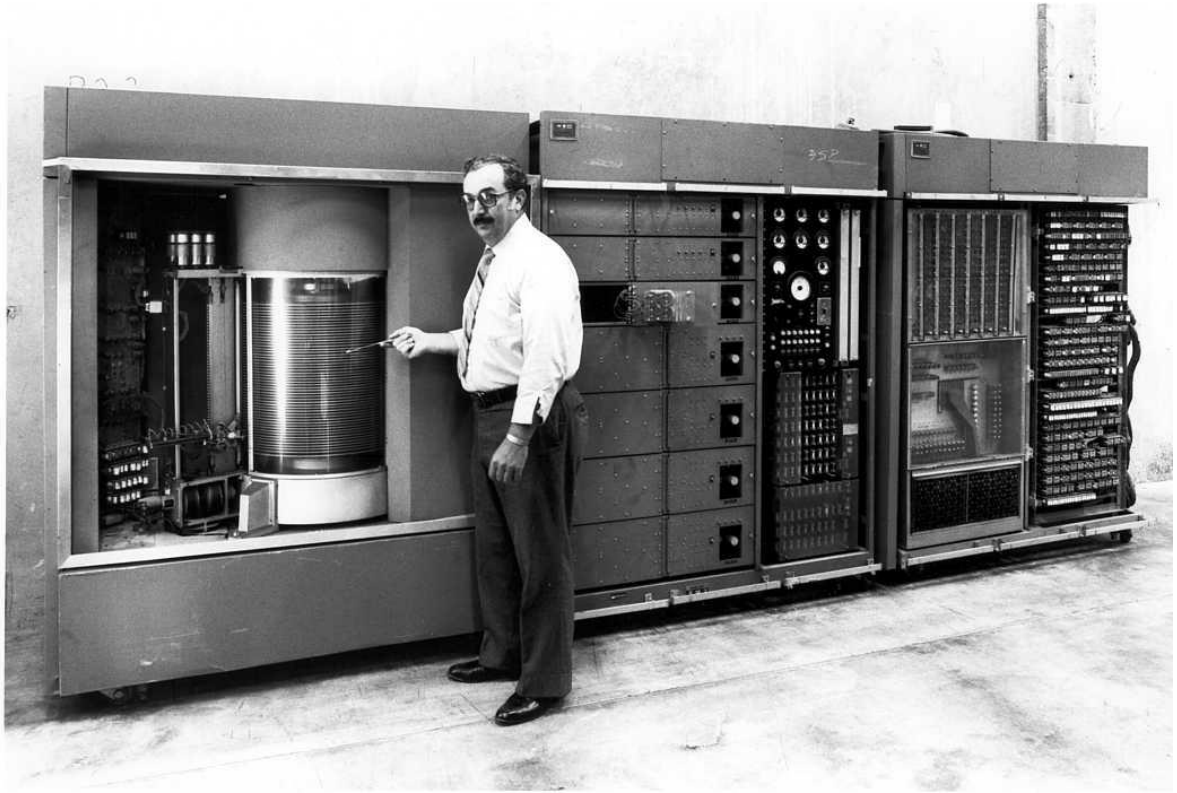
Data Evolution: 101



Parallel Filesystem vs Object Stores

Summary: Parallel Filesystem vs Object Stores

	Parallel Filesystem	Object Store
Client access method	Native RDMA Posix Client. Posix reads and writes, MPIIO	API (plus gateways): PUT, GET, DELETE
Object Types allowed	Files and Directories	Objects with defined structure (data, metadata, data protection policy, checksum)
IO types supported	Parallel IO from single clients, parallel IO from multiple clients	Single stream IO from very large numbers of clients
Geographical distribution	Usually local with some limited WAN capability	Global
Filesystem metadata managament	Usually use Inodes with pointers to data blocks and pointers to pointers	No inodes, object ID returned to client which encodes object location in a hash

WOS Testing Limits

System Attributes	WOS 2.5
Maximum # of Unique Objects	**256 Billion**
Maximum Total Cluster Capacity (using 4TB HDDs)	30.72 PB
Maximum Aggregate R/W Performance (SAS HDDs)	8M Object Reads; 2M Object Writes
Latency (4KB Read; 2-way Replication, SAS HDD)	9ms (Flash is even better)
Latency (4KB Write; 2-way Replication, SAS HDD)	25ms (Flash is even better)
Maximum Object Size; Minimum Object Size	5TB; 512B
Maximum Number of Nodes / Cluster	256 (2 nodes per WOS6000)
Maximum Metadata Space per ObjectID	64MB
Maximum # of Storage Zones	64
Maximum # of Replication/Protection Policies	64

Extents-based Traditional File System Approach

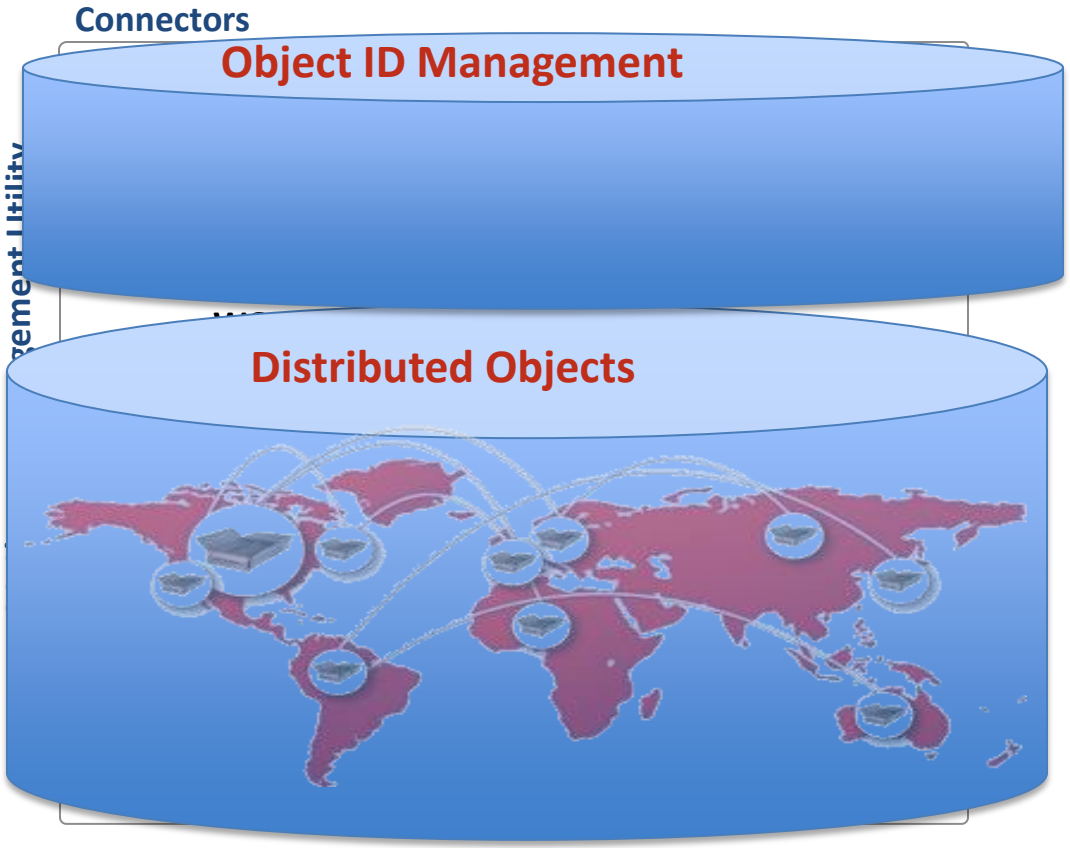
- Ease of use is limited at Scale
  - FSCK challenges
  - Inode data structures
  - Fragmentation issues
  - Filesystem expansion tricky
  - No native understanding of distribution of data
  - RAID management and hot-spare management

WOS (Web Object Scaler)

- Not POSIX-based
- Not RAID-based
- No Spare Drives
- No inode references, no FAT, no extent lists
- No more running fsck
- No more volume management
- Not based on single-site/box architecture

DDN | WoS

Software Stack



Data Protection

Replicate and/or Erasure Code

Global, Peer:Peer

Distribute data across 100s of sites in one namespace

Self-Healing

Intelligent Data Management system recovers from failures rapidly and autonomously

API-based

Integrate applications and devices more robustly

Policy driven

Manage truly via policy, rather than micromanaging mulitiple layers of traditional filesystems

Small files, large files, streaming files

Low seek times to get data

WOS cacheing servers for massive streaming data