Massively Scalable File Storage

Philippe Nicolas - OpenIO

SNIA Legal Notice

SNIA Legal Notice

- Massively Scalable File Storage
  - Internet changed the world and continues to revolutionize how people are connected, exchange data and do business. This radical change is one of the cause of the rapid explosion of data volume that required a new data storage approach and design. One of the common element is that unstructured data rules the IT world. How famous Internet services we all use everyday can support and scale with thousands of new users added daily and continue to deliver an enterprise-class SLA ? What are various technologies behind a Cloud Storage service to support hundreds of millions users ? This tutorial covers technologies introduced by famous papers about Google File System and BigTable, Amazon Dynamo or Apache Hadoop. In addition, Parallel, Scale-out, Decentralized, Distributed, Object and P2P approaches such open source Lustre, PVFS, HDFS, Ceph, Swift, Minio, OpenIO and pNFS with several proprietary ones are presented as well. This tutorial adds also some key features essential at large scale to help understand and differentiate industry vendors offering.

Agenda

- Needs and Challenges
- Classic approaches and limitations
- Key industry shift
- Data Protection
- Some perspective

Needs and Challenges

Needs and Challenges

Volume of Data

Complexity, Silos

Infrastructure Cost

- x1000 in 15 years (800EB 2009, 44ZB+ 2020)
- Too many discrete data servers silos
- Too low Storage utilization, cost of infrastructure & management
- Protection too complex, too long and too expensive (fragile RTO and RPO)
- Business Continuity (HA/BC/DR)

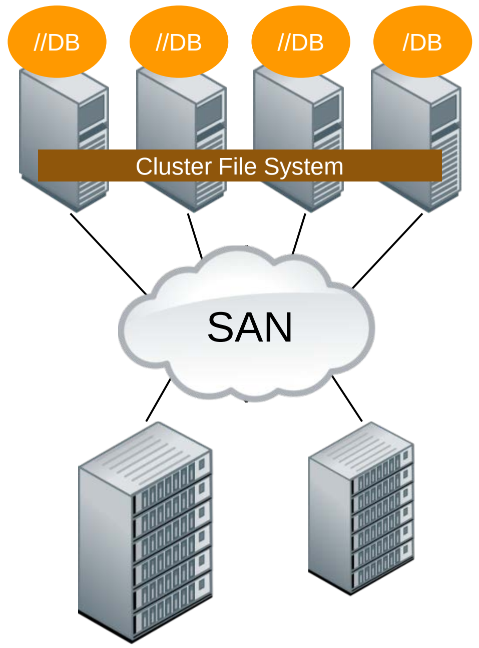
Usages & Consolidation

- Gigantic NAS/File Servers
- Ubiquitous (Geo) data access with Sync-n-Share tools
  - Multi-device, multi-user and geo-independent
- VM, Parked VM… and [Hyper]converged
- Backup and Archive (« Big Unstructured Data »)
  - « Passive, Dormant or Cold » or Active
- Reality of the merge of Backup, Sync and Share
- Video, Media, Broadcast, CDN…. especially with HD…
- Email services
- BI and Analytics (« Big Data »)
- …

Classic approaches and limitations

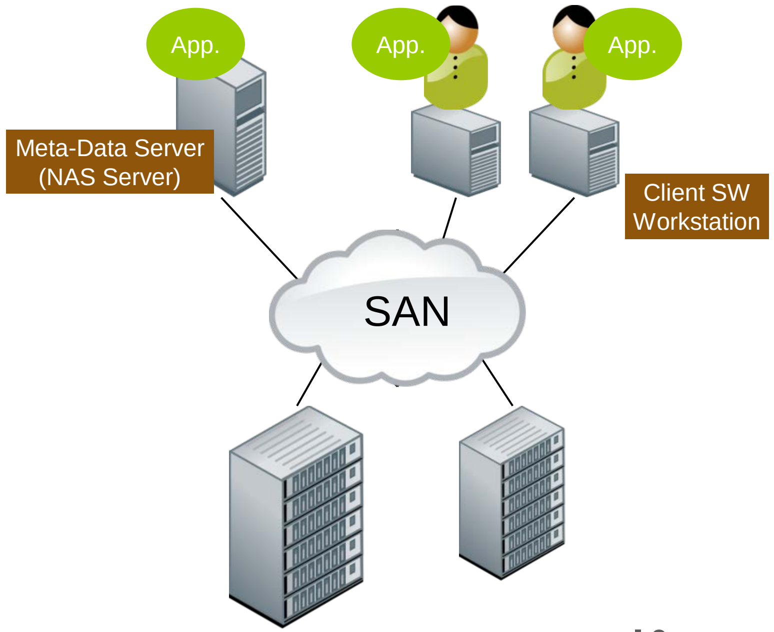
Cluster File System

- **Cluster File System** (CFS), also named Shared Data Cluster
- A Cluster FS allows a **FS and files** to be shared
- Centralized (asymmetric) & Distributed (symmetric) implementation
  - Centralized uses master node for meta-data updates, logging, locking…
- All nodes understand Physical (on-disk) FS structure
  - The FS is mounted by all the nodes concurrently
  - Single FS Image (Cache Coherence)
- Homogeneous Operating System
- Lock Mechanism
  - Distributed or Global Lock Management (DLM/GLM)
- Limitations
  - Distance (server to storage devices)
  - A few dozens of servers (limited scalability)



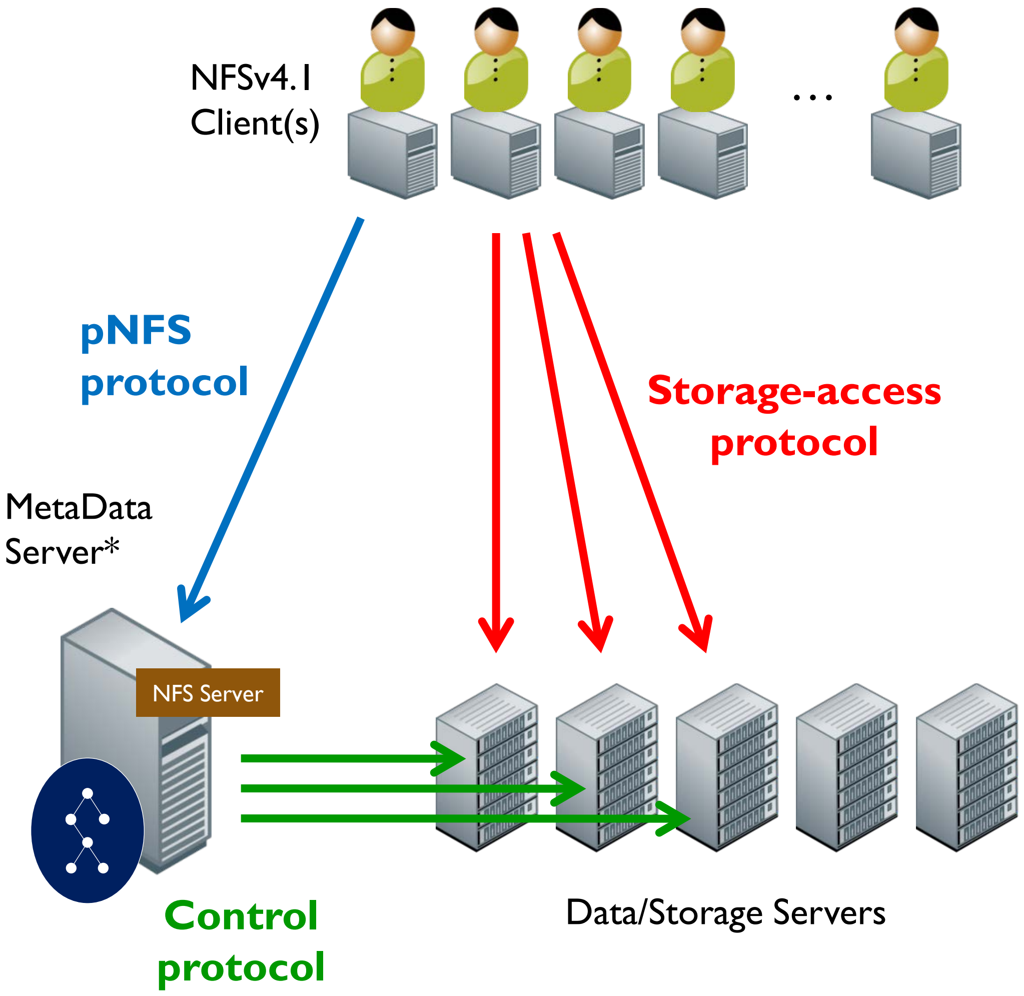
SAN File System

- **SAN File System** (SAN FS) aka **SAN File Sharing System**
- A SAN FS allows **files** to be shared across clients
  - The File System is not shared
- Client/Server or Master/Slave (aka asymmetric) model
  - Mixed role between direct data access with host based thin software and NAS access
- Flexibility of network FS at SAN speed
- Designed to support hundreds of nodes
- Lock Mechanism & Cache Coherency
- Limitations
  - A few hundreds of nodes
  - SAN connectivity & device contention
  - MDS bottleneck



Parallel NFS with NFSv4.1 (pNFS)

- pNFS is about scaling NFS and address file server bottleneck
  - Same philosophy as SAN FS (master/slave asymmetric philosophy) and data access in parallel
- Allow NFSv4.1 client to bypass NFS server for Data
  - No application changes, similar management model
- pNFS extensions to NFSv4 communicate data location to clients
  - Clients access data via FC, FCoE & iSCSI (block), OSD (object) or NFS (file)
- www.pnfs.com



* Could be one of the data servers
All DS could be MDS

Key industry shift

Key Industry Shift

- Historical Storage Industry Approach with DAS (internal disk)
- Then Dedicated External Storage with NAS and Disk Array thanks to storage networking technologies (SAN, iSCSI…) – Appliance Wave
- And key Storage Paradigm Shift: Storage with Servers thanks to Linux and Ethernet, HTTP as a “storage protocol”

Transform a farm of servers (with internal disks)
into a large storage pool

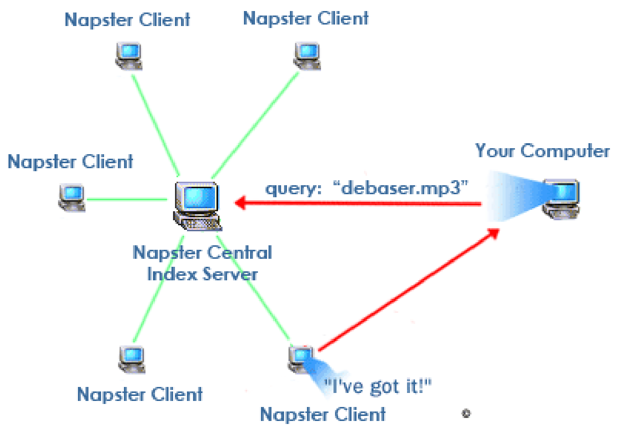
File Storage Scalability Needs

- By numbers and features
  - Clients connections: 10000s (and 10-100s millions in Cloud mode)
  - Storage Servers/Nodes: 1000s
  - Storage capacity: 10-100-1000 PBs of aggregated capacity
  - Number of Files: 100s of Billions
  - Throughput: 10s-100s-1000s GB/s of aggregated bandwidth & 10-100s millions of IOPS
  - Self-Everything (local and remote data protection, repair & healing, integrity checking) – Elastic Resiliency
  - Global, permanent and ubiquity access and presence
  - Notion of global/shared namespace
  - Standard (POSIX, NFS, CIFS, HTTP/REST...) and/or prop. file access
  - Security and privacy
  - Metering, Auditing, Monitoring, Billing/Chargeback…

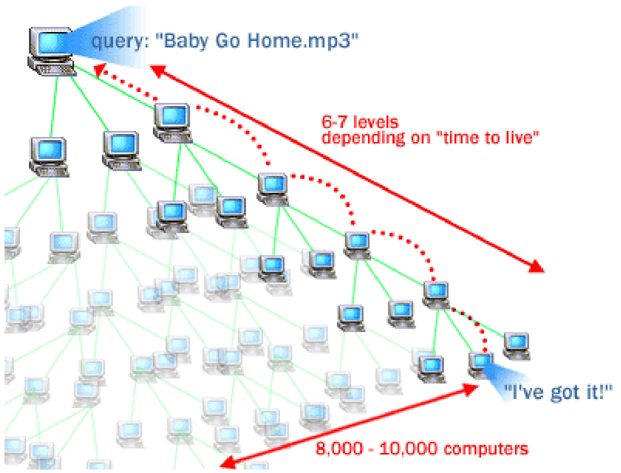
P2P Implementations

« Online/Internet/Cloud Service for dedicated usage »

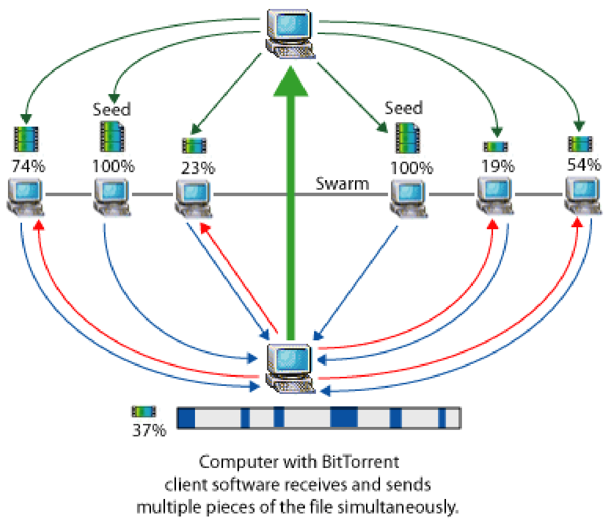
- Interesting implementation with agg. of other machines' storage space (consumer/contributor) with or w/o a central server (asym. or sym. P2P)
  - Napster (central dir.), Gnutella (broadcast), BitTorrent (central list but parallel chunk download)
  - Ex: P2P File Sharing System such as music, mp3…



Napster


query: "Baby Go Home.mp3"
6-7 levels depending on "time to live"
"I've got it!"
8,000 - 10,000 computers

Gnutella


Seed
Seed
74%
100%
23%
100%
19%
54%
Swarm
37%
Computer with BitTorrent client software receives and sends multiple pieces of the file simultaneously.

BitTorrent

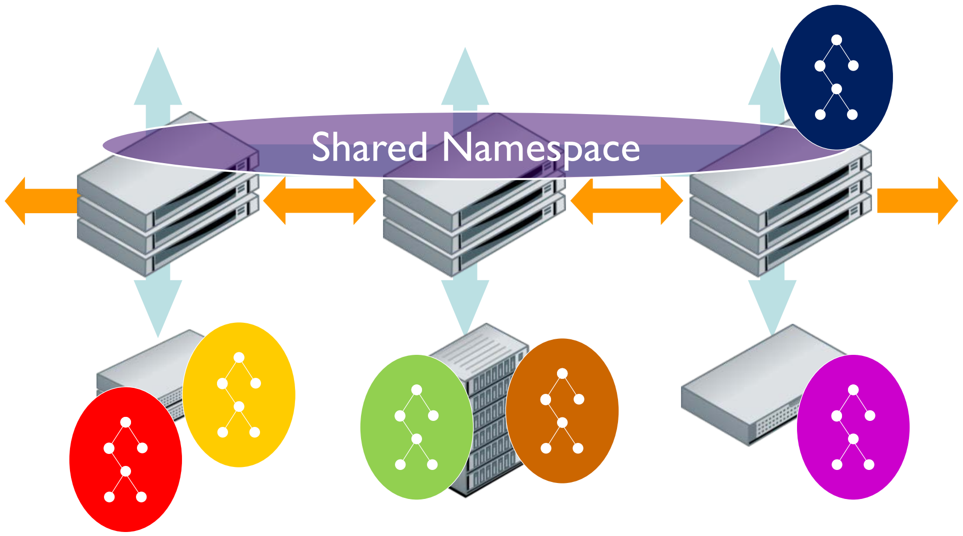
P2P Implementations

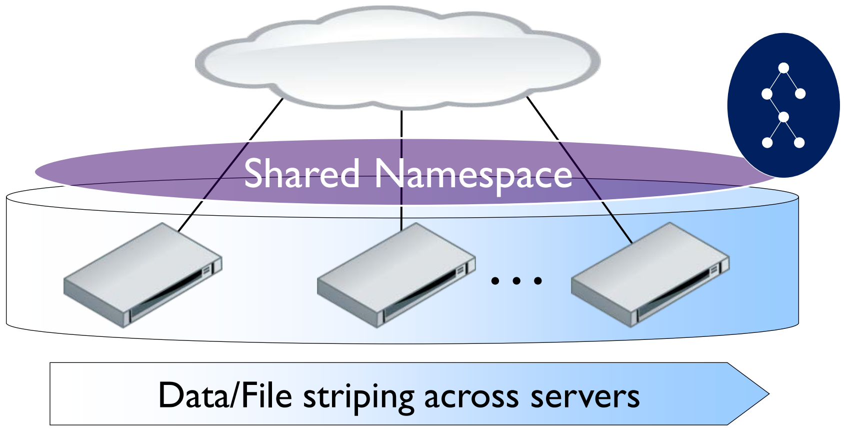
« Online/Internet/Cloud Service for generic usage »

- Mutual Storage as a New Flavor
  - Real innovation with only a few vendors
  - Generic and not “limited” to a specific application (video, mp3...)
- Idea: “Each data consumer can offer storage space”
  - No central data storage server
  - Different business model
- “A client is a server !!”
  - No external storage
  - Data is [geo]distributed/dispersed among consumers’ machine
  - Aggregation of internal storage (container...)
  - Total Capacity = Sum. individual storage - protection
- [Geo] Erasure Coding
- New market trend with a few vendors

Industry Server Implementations

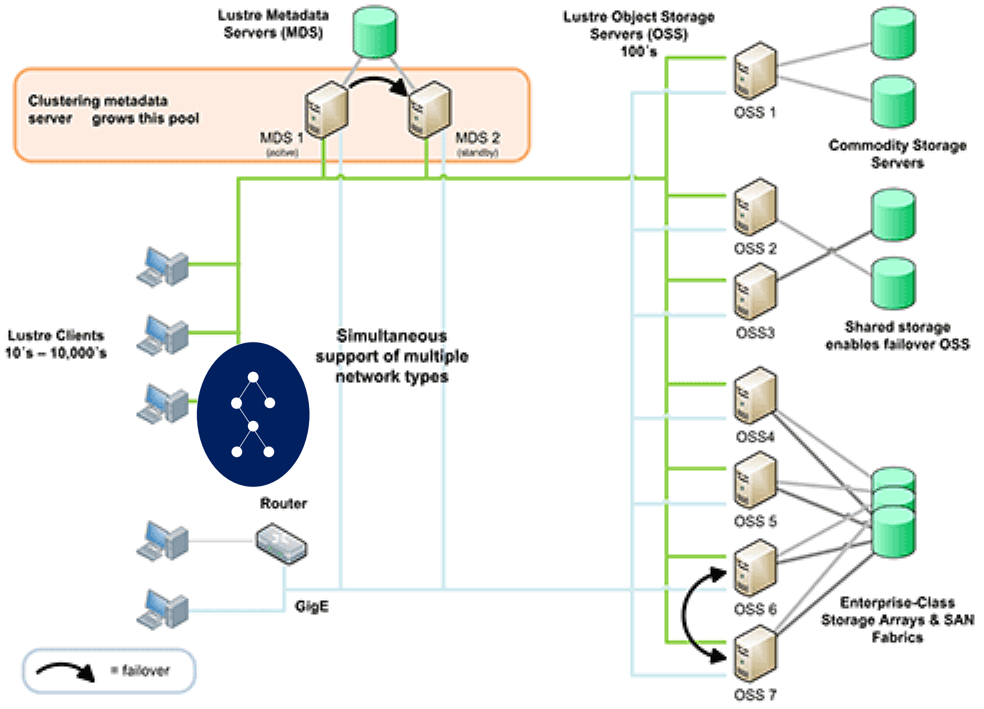
- No MDS or special Master server – Symmetric model
- Aggregation of independent storage servers
- Shared Nothing
- 2 modes
  - File entirely stored by 1 storage server (no file striping or chunking)
  - File is striped across storage servers





Lustre

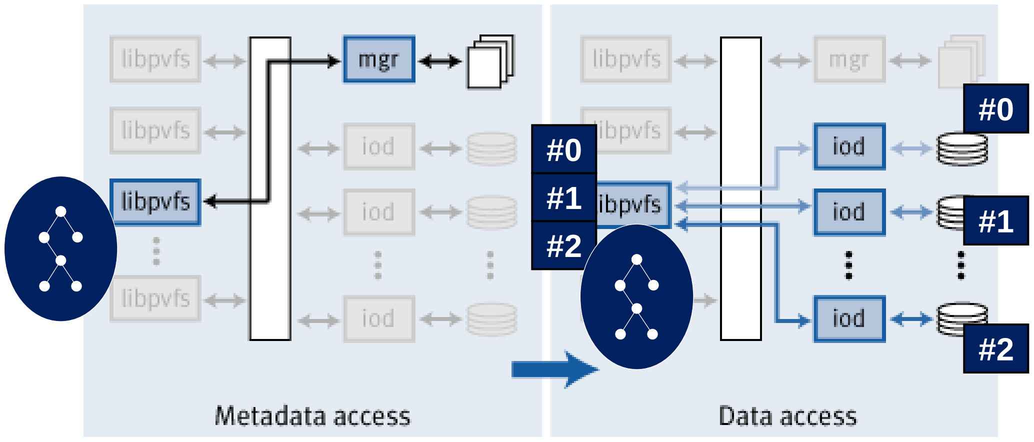
- Open source object-based storage system
  - Based on NASD study from Carnegie Mellon Univ.
- University project
- Asymmetric model
- Notion of Object (OST/OSS)
- Largely adopted in HPC
- www.lustre.org



Lustre Metadata Servers (MDS)
Lustre Object Storage Servers (OSS) 100's
Clustering metadata server grows this pool
MDS 1 (active)
MDS 2 (standby)
OSS 1
Commodity Storage Servers
OSS 2
OSS3
Shared storage enables failover OSS
Lustre Clients 10's – 10,000's
Simultaneous support of multiple network types
OSS4
OSS 5
OSS 6
OSS 7
Router
GigE
Enterprise-Class Storage Arrays & SAN Fabrics
= failover

PVFS (Parallel Virtual File System)

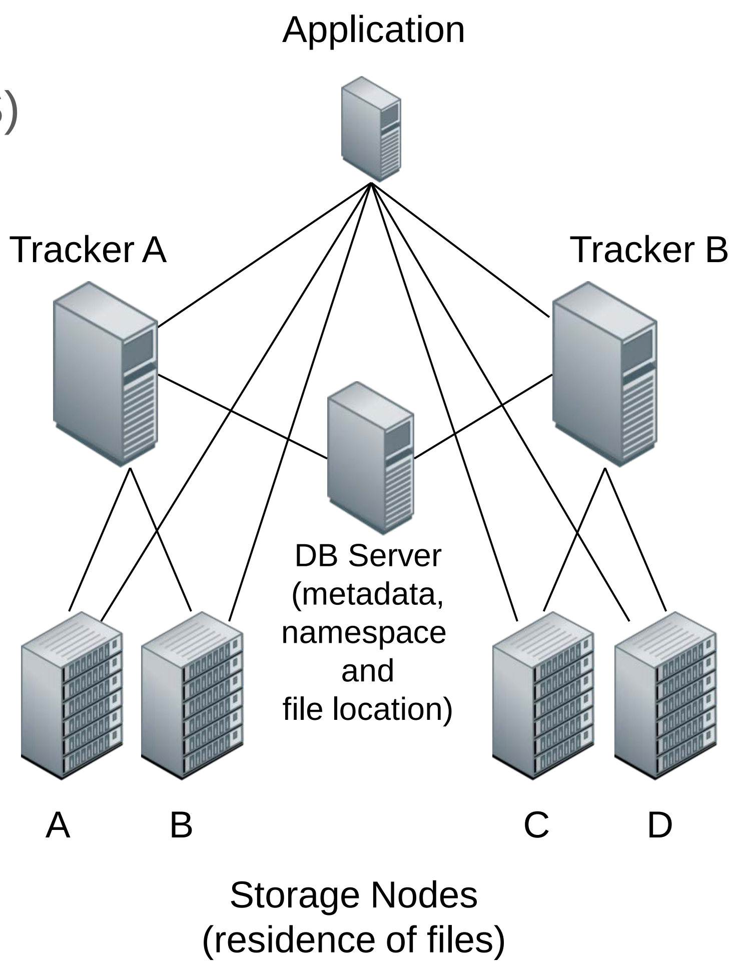
- Now PVFS2
- Project from Clemson University and Argonne National Lab
- Open source and based on Linux
- Asymmetric model
- File striping
- www.pvfs.org
- Special flavor (www.orangefs.org)



Other Distributed examples

- Asymmetric
  - BeeGFS (was FhGFS or FraunhoferFS)
  - XtreemFS
  - MogileFS
  - ...
  - Plenty in Open Source and Education/Research env.

MogileFS example


Application
Tracker A
Tracker B
DB Server (metadata, namespace and file location)
A
B
C
D
Storage Nodes (residence of files)

But a Tsunami has occurred…

Internet changed everything !

...Internet introduced new challenges impossible to solve with traditional approaches:

100s Millions of users, 10s-100s PB of Data and Billions of files to store and serve

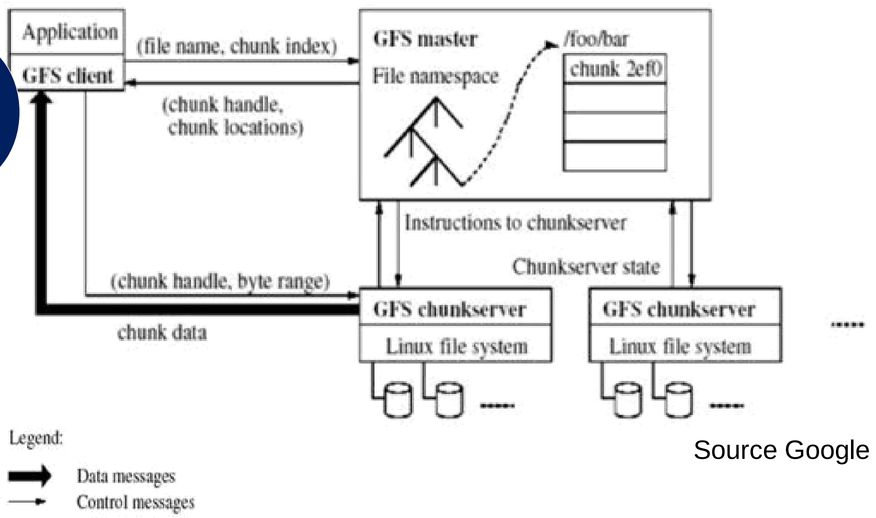
What all these companies have in common ?

Pure Software approaches on COTS

- Traditional File Storage not ready and designed to support Internet age !! (fast and massive growth, always on)
  - Limitations in Size and Numbers (total capacity, file size, file number - global and per directory)
    - Facebook runs a 100PB HDFS config. !!
    - Facebook develops its own storage backend for pictures/images (Haystack)
    - Yahoo! operates 170PB HDFS on 40K+ servers (multiple clusters)
  - Metadata wide-consistency is problematic
    - “Horizontal” consistency (volume, FS) - Recovery model
    - Asym design starts to address this
  - File System structure and disk layout, File Sharing Protocols “too old”
- New Ideas coming from Academic, Research and “Recent” IT companies
  - Google research papers about BigTable, GFS or Amazon about Dynamo
- New category for solutions: Webscale

Google File System

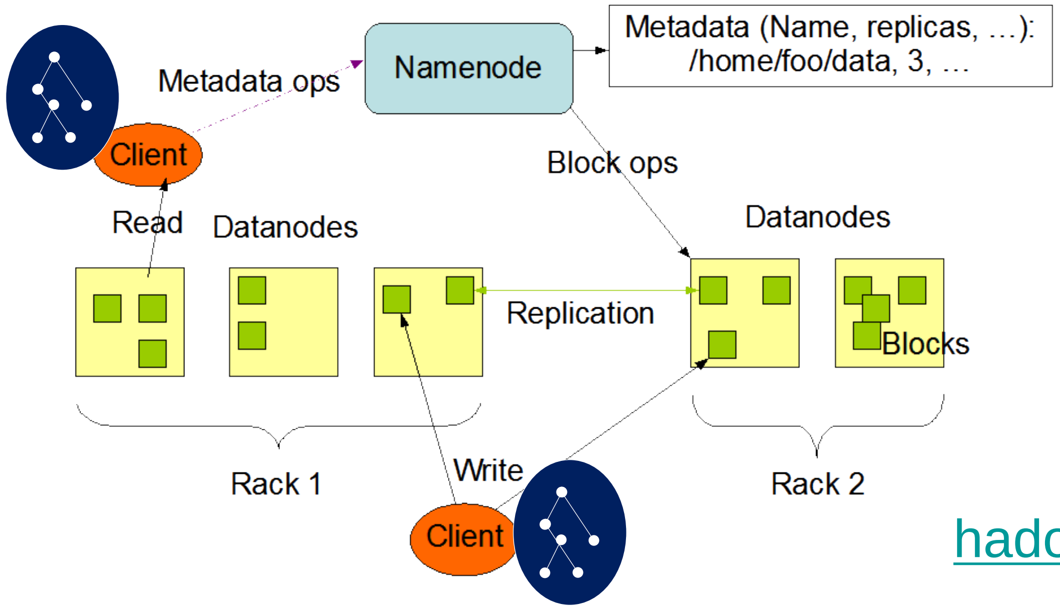
- Internal deployment used by WW end-users (Google Search, Gmail….)
  - Not available for external usage
- Proprietary approach
- Asymmetric model
- Thousands of chunk servers with 64MB chunk size (stripe unit)
- research.google.com/pubs/papers.html
- GFS v2
  - Chunk size = 1MB !!
  - Hundreds of Distributed Masters (100 millions files managed by 1 master)
- Other example based on same approach: MooseFS, CloudStore (Quantcast FS based on KosmosFS)



Source Google

Hadoop Distributed File System (HDFS)

- Apache project
- Highly fault-tolerant built in Java
- Programmable (not CLI)
- Large data sets (6k nodes, 120PB)
- Asymmetric model
- HA NameNode (A/P Hadoop 2.0)
- Files striped across DataNodes
- Federation of Namenodes
- WebHDFS REST API, httpFS gw
- NFS layer, Data Services



hadoop.apache.org

New design and architecture

- Industry standard components & servers (COTS*)
- "Aggregation | Union" of independent storage servers – Scale-out or Horizontal
- Shared-nothing model, RAIN, Grid…
- Symmetric, Asymmetric or P2P (Central Authority, Master node or Masterless)
- New Distributed Topologies with Consistent-based Hashing, P2P, Hierarchical, Flat or Asym model and fast inter-node routing techniques (Chord, Pastry, Tapestry or CAN methods) and Data Placement methods
- Object Storage and Key/Value Store – simple consistency model
- Parallel vs. non-Parallel (striping or chunking…)
- User-mode
- File-based and object-based
- Notion of Shared/Private Namespace (storage nodes wide)
- Embedded DB to store and distribute Meta-Data and/or small objects (NoSQL)
- Extended features (Automation, Snapshot, CDP, Replication, Mirroring, ILM/FLM/Tiering, Migration, LB…)
- Commercial and/or Open Source Software

*COTS: Commodity Off-The-Self

Key Value Store and Object Storage

- Key Value Store (KVS)
  - No File System complexity
  - No inode dependency
  - No nested directories (hierarchy)
  - Flat namespace
  - Simple relation between Key and Value
  - 64-128 bit Univ. Uniq. ID
  - Very Scalable & Flexible
  - “Vertical” consistency
- Object Storage as extension and union of KVS
  - Some historical approach and tentatives with CAS, OSD…
  - Scale-out by nature
  - Object = Metadata + Data
  - Flat single global namespace
  - Super high capacity, Billions of objects, no real limit

- Data Protection controlled by the core engine (JBOD approach, No RAID or sophisticated Disk Array needed)
- Multiple topologies (Ring, P2P, Hierarchical, Flat or Asym...)
- Notion of Container with 3 basic operations: GET, PUT and DELETE
- Very simple interface with HTTP/REST (C++, Python, Java & PHP API)
  - De facto standard such as Amazon S3 API, community standard like Swift
- Many times associated with NoSQL DB for metadata or small objects
- File interface with In-Band Gateway or native File System layer
  - CIFS and NFS Gateway on top of Object Store
- CDMI: glue between client jungle and dozens of Obj. Storage (industry standard)

Open Source Object Storage

	ceph	openstack Swift Project	MINIO	openio
Open Source	Yes			
HW Agnostic	Yes			
Scalability	Webscale			
Data Access Methods	RADOS Objects APIs, RADOS Gateway for S3 and Swift, RBD and CephFS, FUSE, NFS, SMB	Swift API, S3, File Access (NFS/SMB) via SwiftStack File Gateway (Maldivica†)	S3	Objects APIs (HTTP/REST, S3, Swift), File sharing protocols (FUSE, NFS, SMB, AFP, FTP) and Editions (Mail, Video…)
Data Protection	Replication and Erasure Coding			
Compute + Storage	No, just Storage			Yes (Grid for Apps)
TCO	$			

SNIA CDMI (www.snia.org/CDMI)

- Today v1.1.1 - An ISO/IEC standard (17286)
- An Universal Data Access Method - “The NFS of The Internet”
- Local and Remote access
- File access (by Path) and object (by ID)
- Extensions with LTFS
- “Global” Namespace
- Independence of Clients and Servers
  - One client to connect to multiple storage entities
  - Evolution of server without changing client layer
- Need industry push and promotion (~10 CDMI servers available)
- **Join SNIA Cloud Storage Initiative to support CDMI development and worldwide adoption**

SNIA Global Education

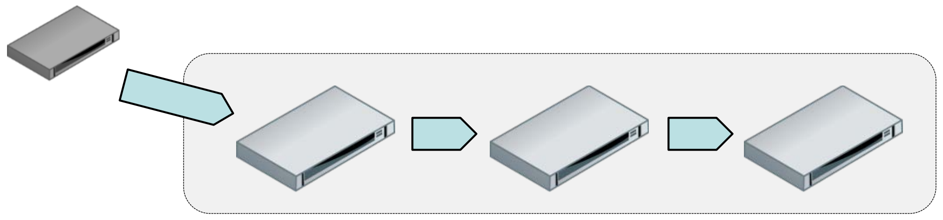
Data Protection

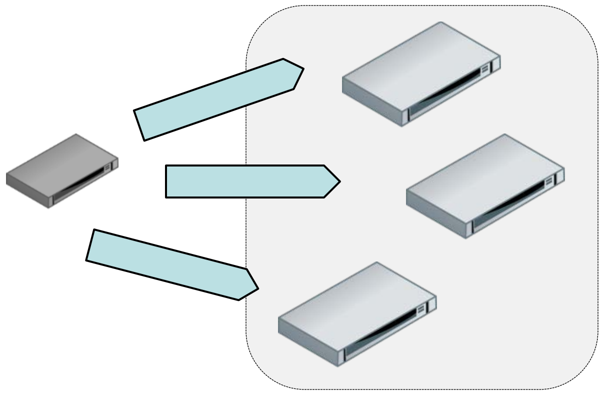
Data Protection

- Key metric: Durability think about Data Persistence
  - according to Amazon website, Durability is defined as follows: “durability (with respect to an object stored in S3) is defined as the probability that the object will remain intact and accessible after a period of one year. 100% durability would mean that there’s no possible way for the object to be lost, 90% durability would mean there’s a 1-in-10 chance, and so forth.”
- Nearly impossible to run backup
  - Too big data sets with many constant updates (backup job started but never finished)
- RAID Can’t scale, Risk Exposure
  - RAID6 not enough, Large disks today (10TB), very long to rebuild
  - Storage protection overhead & rebuild process (resource and time)
  - Potential impact of other users’ requests

1st solution: Data Replication

- Data Replication with Local and/or Geo mode
  - Multi-copies with special consistency approaches and considerations (CAP, Merkle tree, Paxos...)
  - No more consistency in the Write() operation with now control in the Read()
    - CAP Theorem: Strong Consistency when R + W > N (R: Number of Replicas in a Read, W: Number of Replicas that must send an ACK for Writes and N: Number of nodes for the operation) or Eventually Consistent when R + W <= N
      - Particular cases: W>= 1: Writes always possible, R>=2: Minimal redundancy
  - No data modification so data is easy to access
  - Ideal model but very costly at large scale (HW, large config, large files)





2nd solution: Erasure Coding

- Erasure Coding techniques
  - Cauchy/Reed-Solomon, Forward Error Correction, Fountain Coding, IDA, Dispersed Storage, XOR-based codes, X-Code, EVENODD, RDP, Mojette Transform…
  - Goal: tolerate high number of concurrent failures and continue to deliver data on requests (Examples: 4 failures among 10 data elements, 6 on 16...)
  - Various notation: Data (k) and Parity/Coding (m) chunks often noted (k,m) or (n,k) with n = k+m
  - Ratio (k+m/k) is key: common from 1.2 to 1.6 (capability to tolerate loss) and storage efficiency (k/k+m)
  - Data format: Clear (native, no transformation aka Systematic approach) or Encoded (scrambled, sort of encryption aka Non-Systematic approach)
  - Local within a Cluster or Geo (Dispersed), Local coupled with Replication for Geo
  - Cost effective solution to control the cost associated with the data volume growth

SNIA™
Global Education

Next industry shift

Next Industry Shift

Disaggregate Storage from Servers

- 2nd Storage Paradigm Shift
- Industry Standardization Effort

SEAGATE WD TOSHIBA

- Replace Servers by Ethernet Drives (Node = Drive)
- SNIA Object Drive TWG (www.snia.org/object-drives)

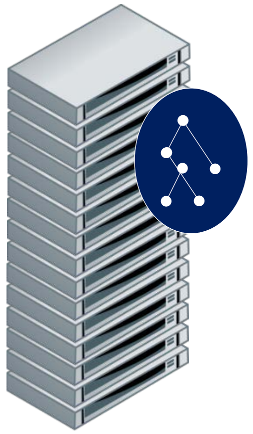
SNIA KINETIC Open Storage Project

- Phase 1: Ethernet Drives aligned with Kinetic Open Storage Project
- Phase 2: Embed and run intelligence directly inside Drives

Conclusion

Conclusion

- Massively Scalable File Storage is a reality
  - Cost effective approach with COTS
  - Acceptance of 10 of thousands of machines (clients, servers)
  - At Scale Asymmetric (Master/Slave) seems to be the more scalable philosophy
  - Jump into the Ethernet Drives industry initiative
  - Superiority of Object over File and Block approach at scale
  - Think de facto (S3 and Swift) or industry Standard (CDMI for instance)
  - Unified, Global Namespace is fundamental
  - Advanced Data Protection – local and geo w/ Replication and EC
  - Reliability & Security are key (authorization, authentication, privacy…)
  - Self-Everything (Autonomous system)
  - Other services: Monitoring, Billing, Multi-Tenancy...



Attribution & Feedback

The SNIA Education Committee thanks the following Individuals for their contributions to this Tutorial.

Authorship History

Philippe Nicolas – 2009

Recent Updates:
Ph. Nicolas April 2016, March 2015, 2014, August, Feb. 2013 & July 2012

Additional Contributors

Please send any questions or comments regarding this SNIA Tutorial to ***tracktutorials@snia.org***

Massively Scalable File Storage

Philippe Nicolas - OpenIO