

A decorative graphic consisting of multiple parallel, wavy lines in various colors (purple, blue, orange, grey, green) that flow from the left side of the slide towards the right, curving upwards and then downwards.

MASSIVELY SCALABLE FILE STORAGE

Philippe Nicolas, Scality

SNIA Legal Notice

- ◆ The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- ◆ Member companies and individual members may use this material in presentations and literature under the following conditions:
 - ◆ Any slide or slides used must be reproduced in their entirety without modification
 - ◆ The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- ◆ This presentation is a project of the SNIA Education Committee.
- ◆ Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- ◆ The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

➤ 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, Distributed and P2P approaches with Lustre, PVFS 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 for Massive Scalability
 - ◆ Fundamental Drivers
 - ◆ Needs and Limitations
 - ◆ Multiple Usages

- Implementations and Philosophies
 - ◆ Conventions, Approaches
 - ◆ Industry and Market Examples
 - ◆ Data Protection

- Conclusion

A decorative graphic consisting of multiple parallel, wavy lines in various colors (purple, blue, orange, grey, green) that flow from the left side of the slide towards the right, creating a sense of movement and depth.

Needs for Massive Scalability

Fundamental Drivers

Needs and Limitations

Multiple usages

Fundamental Drivers

- **Avalanche of data, increase demand for unstructured data storage**
 - ◆ x1000 in 15 years (800EB 2009, 44ZB+ 2020)
 - ◆ 80% of this volume is unstructured information
 - ◆ 80% of file data growth every year, 80% of inactive data after 30 days ! (file size grows as well)

- **Too many discrete data servers silos and point products**
 - ◆ File Server/NAS, Backup & Archiving, Application, VM, BI/Analytics...
 - ◆ Too low Storage utilization, cost of infrastructure management
 - ◆ File System limitations to support millions/100s of millions/billions of files, 10s-100s of PBs, thousands of clients and storage servers (IOPS & BW)
 - ◆ Poor automation

Fundamental Drivers

- Protection too complex, too long and too expensive
 - ◆ RTO and RPO not aligned to business
 - ◆ Too many tape restore errors (30%), no time to backup
 - ◆ Policies enforcement, Load balancing, Migration, Replication, Mirroring, [I|D|F]LM/Tiering...
- Business Continuity (HA/BC/DR)
 - ◆ 100% ubiquitous data access
 - ◆ SLA not always set, controlled and measured

File Storage Scalability Needs

Always-on, Everywhere for Everyone

◆ 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...

A horizontal line composed of several colored segments: purple, grey, yellow, blue, orange, grey, white, purple, grey, orange, grey, blue, grey, and yellow.

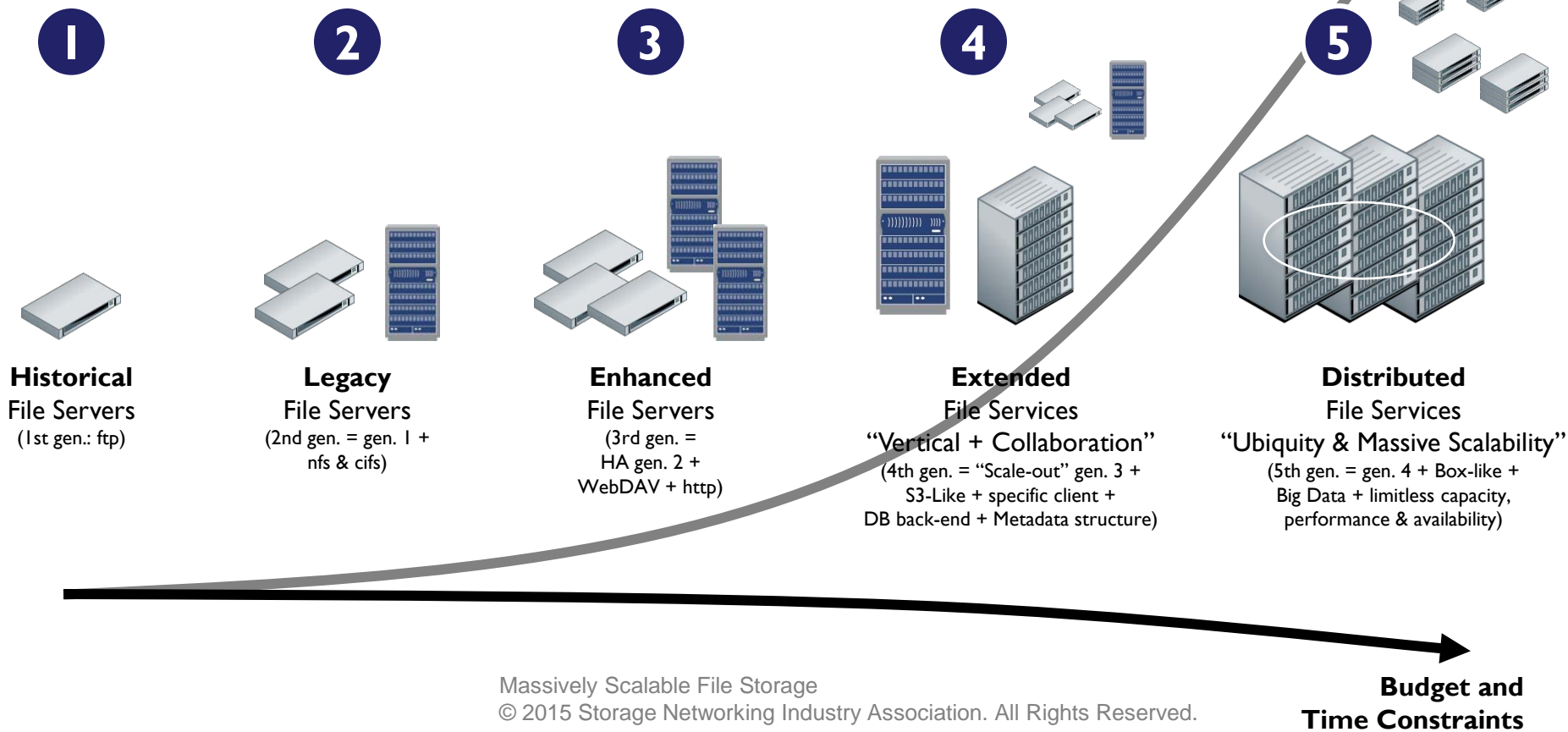
Where do we come from ?

Mutation of File Services

Usage of File Servers in IT today

Data Volumes
& Storage Utilization

Towards a commodity File Service

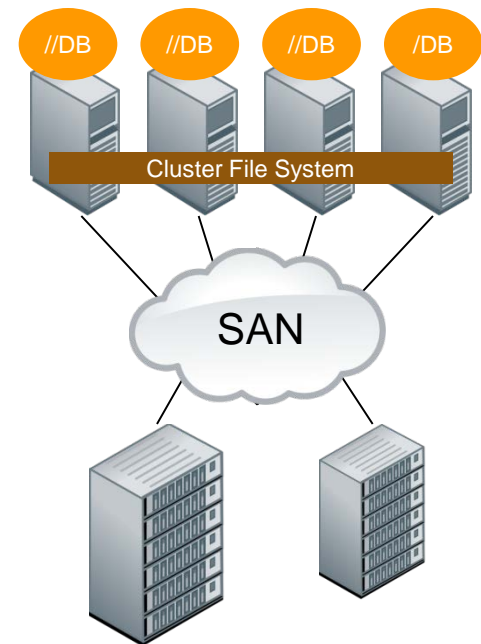


A horizontal line composed of several colored segments: purple, grey, yellow, blue, orange, grey, white, purple, grey, orange, grey, blue, grey, and yellow.

2 famous examples well deployed on the market

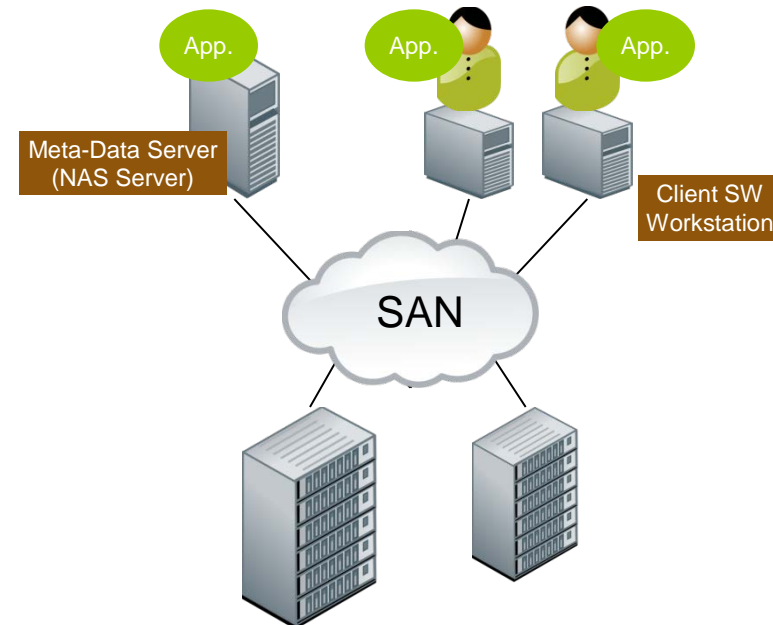
Cluster File System

- **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



Multiple usages

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

A decorative graphic consisting of multiple parallel, wavy lines in various colors (purple, blue, orange, grey, yellow) that flow from the left side of the slide towards the right, curving upwards as they go.

Implementations and Philosophies

**Conventions, Approaches
Industry & Market Examples
and Data Protection**

Implementations and Philosophies

- ◆ Industry standard components & servers (COTS*)
- ◆ Commercial or Open Source Software (Bare Metal or Hypervisor-based)
- ◆ Standards file sharing protocols (+ API, proprietary protocol)
- ◆ Online scalability (cluster can't be stopped, should be always online)
- ◆ Self-Managing, Self-Healing, Self-Aware, Self-Organizing...
- ◆ Data Management Services
- ◆ Scale-out approach with a distributed philosophy based on RAIN, Grid, new DB model...

*COTS: Commodity Off-The-Self

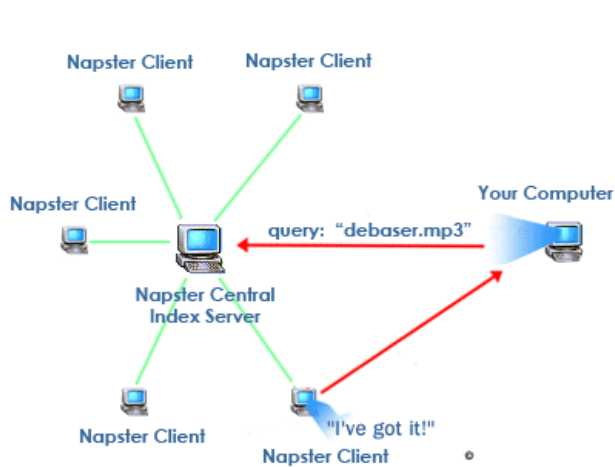
Distributed implementations

- “Aggregation | Union” of storage servers – Scale-out or Horizontal
- Symmetric, Asymmetric or P2P
 - ◆ Central Authority aka Master node or Masterless
- Parallel vs. non-Parallel
 - ◆ File striping and chunking, concurrent access, single node access
- Shared-nothing model
 - ◆ Aggregation of file storage/servers
- User-mode and Kernel/System-mode
- File-based and object-based
- Notion of Shared/Private Namespace (storage nodes wide)
- RAIN, Grid implementation with embedded data prot. and resiliency
 - ◆ No storage nodes are fully trusted and are expected to fail at any time
- New data model with NoSQL DB for Metadata and/or small objects
- Extended features
 - ◆ Policies enforcement for Automation, Snapshot, CDP, Replication, Mirroring, ILM/FLM/Tiering, Migration, Load balancing...

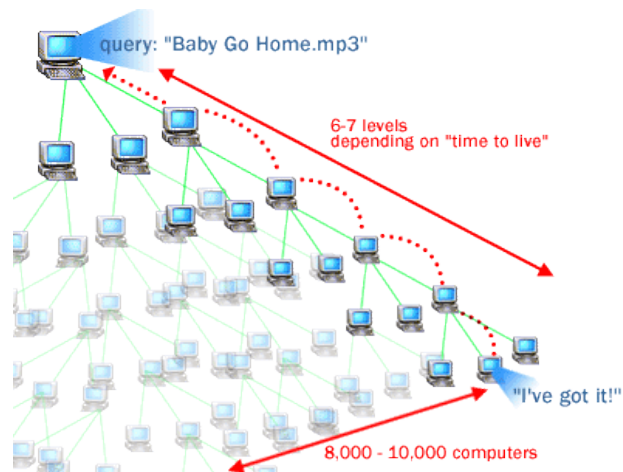
P2P Implementations

« Online/Internet/Cloud Service for dedicated usage »

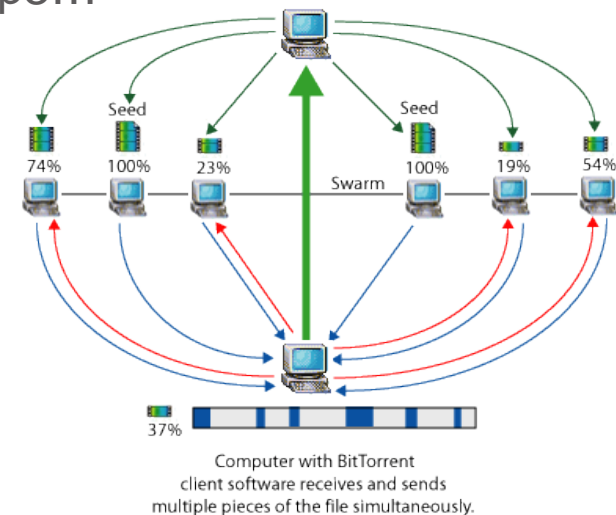
- Interesting implementation with aggregation of other machines' storage space (consumer/contributor) with or w/o a central server (asym. or sym. P2P)
 - ◆ Napster (central dir.), Gnutella (broadcast), BitTorrent (central list but parallel chunk download)
 - ◆ Ex: P2P File Sharing System such as music,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



Napster



Gnutella



BitTorrent

P2P Implementations

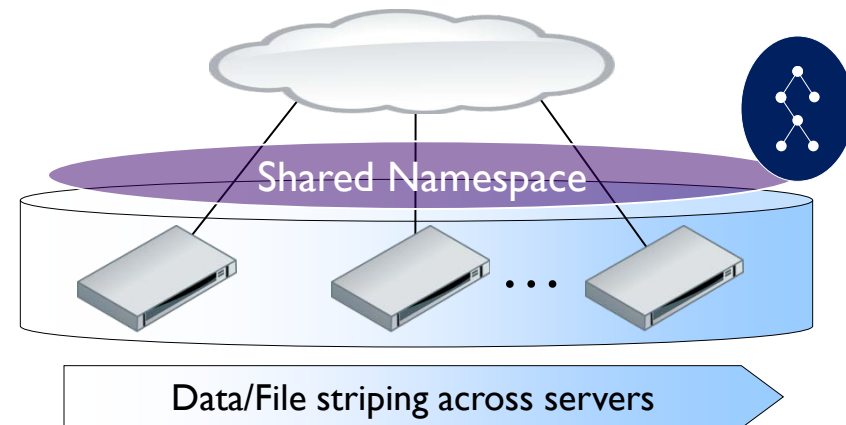
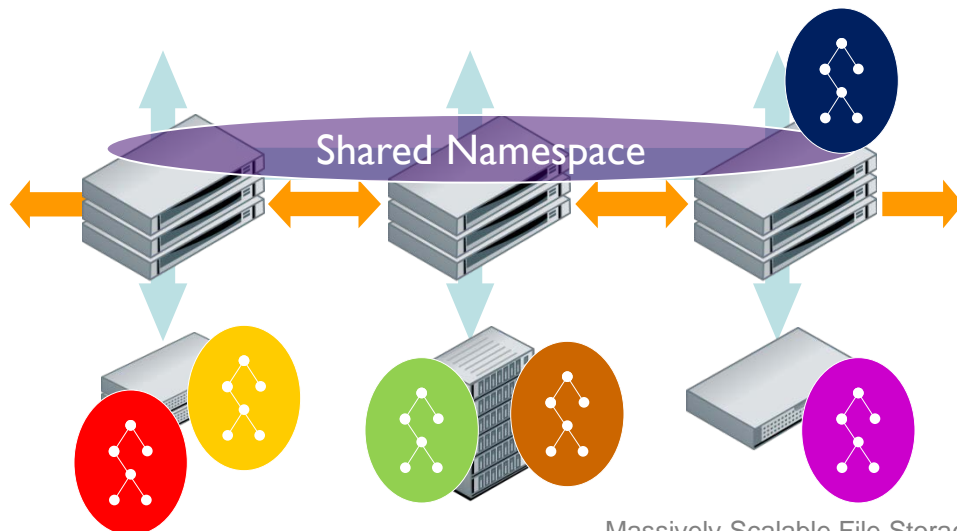
« Online/Internet/Cloud Service for generic usage »

- 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

Symmetric Philosophy

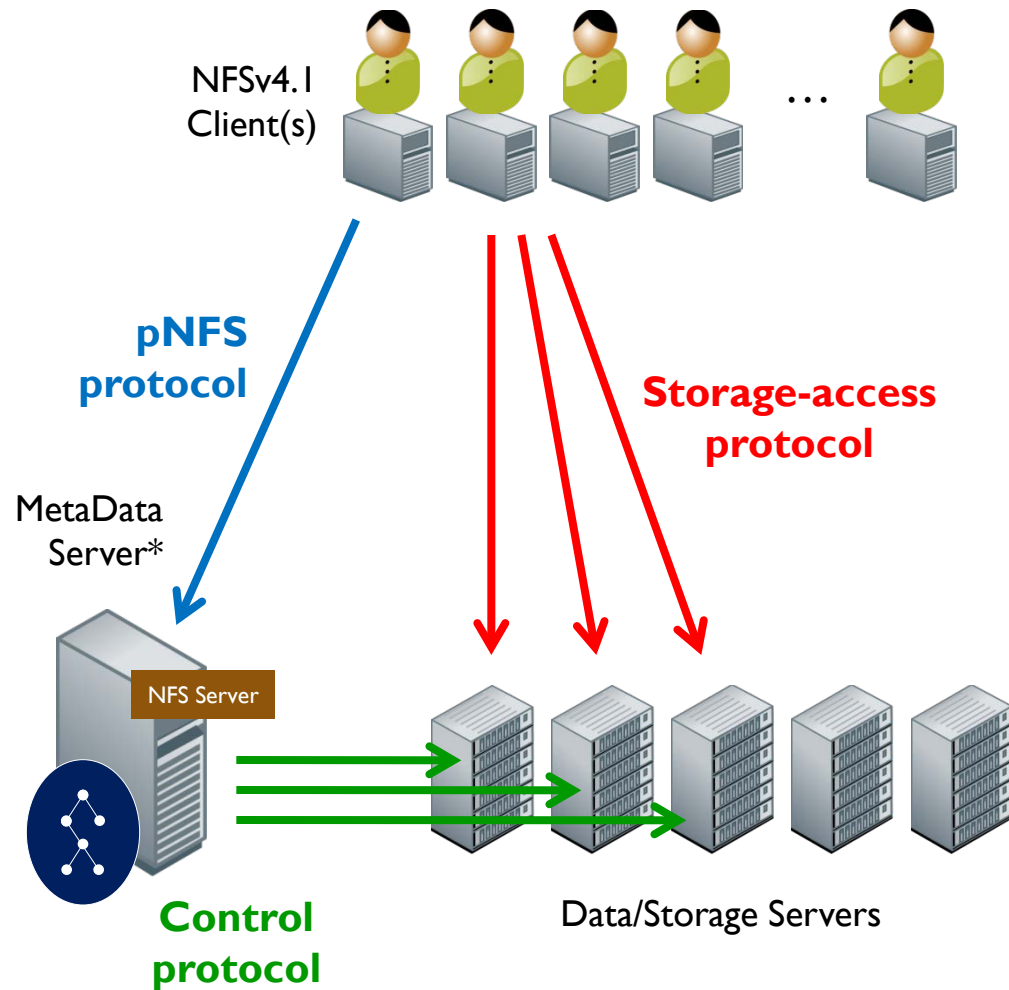
- No MDS or special Master server
- Aggregation of independent storage servers
- Shared Nothing
- 2 modes
 - ◆ File entirely stored by 1 storage server (no file striping or chunking)
 - ◆ File is striped across storage servers



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
 - ◆ 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)

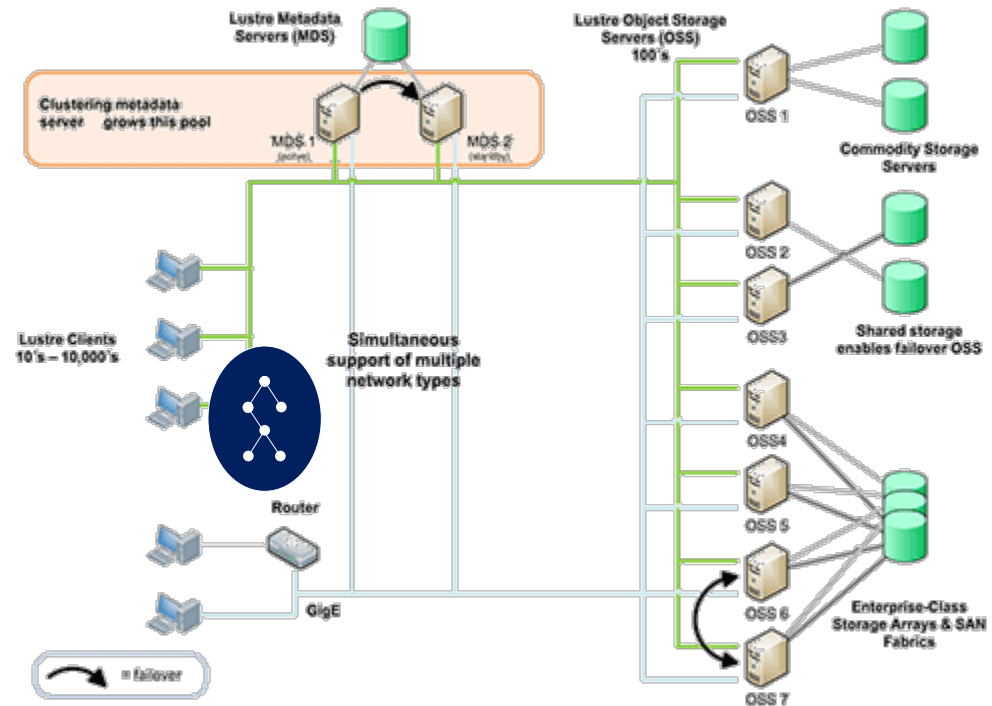
➤ www.pnfs.com



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

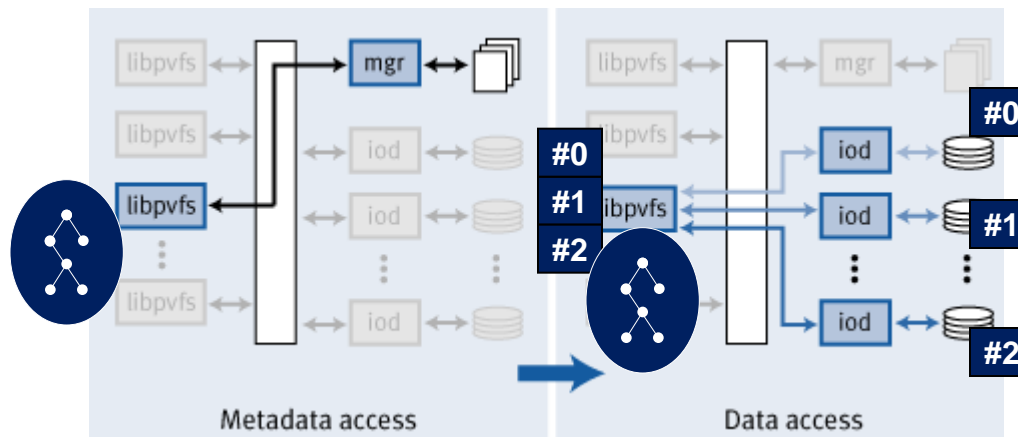
Lustre

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



PVFS (Parallel Virtual File System)

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

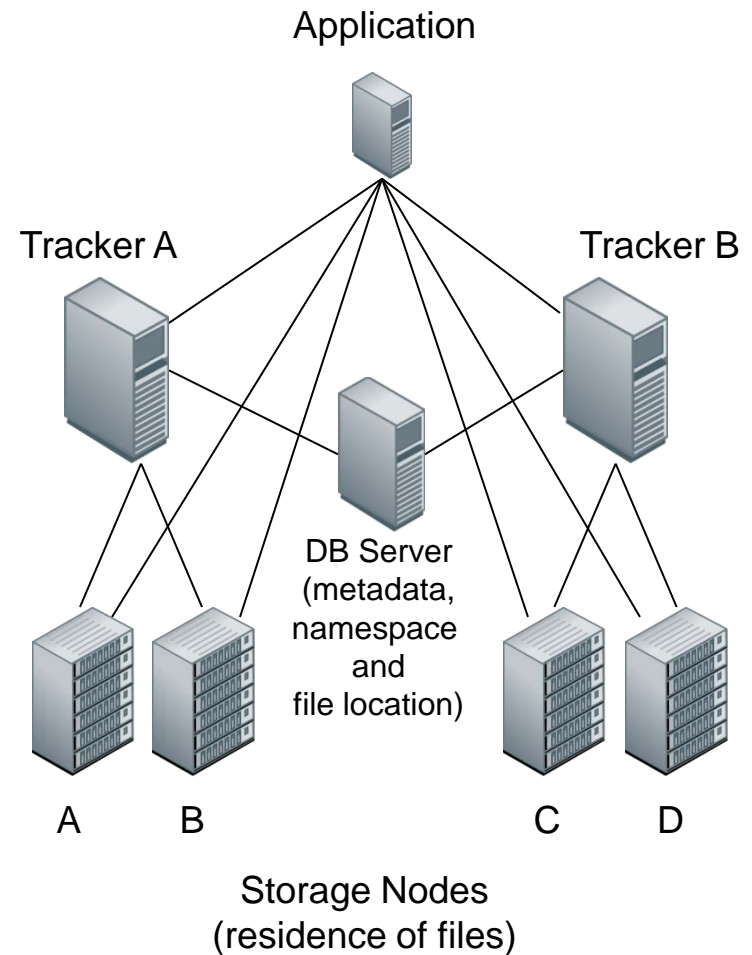


Other Distributed examples

➤ Asymmetric

- ◆ FhGFS (FraunhoferFS)
- ◆ XtremFS
- ◆ MogileFS
- ◆ ...

MogileFS
example



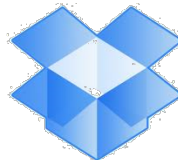
A horizontal line composed of several colored segments: purple, grey, yellow, blue, orange, grey, white, purple, grey, orange, grey, blue, white, and yellow.

But these approaches still have
some limitations...

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 ?

Radical new Software Approaches

◆ Why ?

- ◆ 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

Google File System

- Internal deployment but used by WW end-users (Google Search, Gmail....)

- ◆ Not available for external usage

- Proprietary approach

- Asymmetric philosophy

- Thousands of chunk servers with 64MB chunk size (stripe unit)

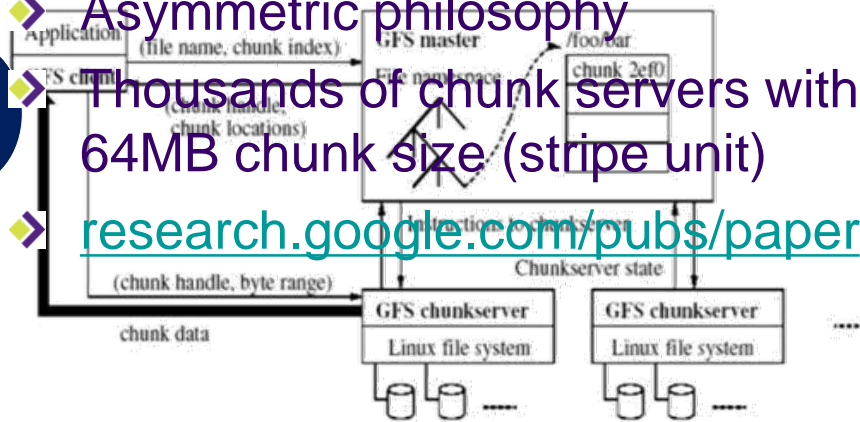
- research.google.com/pubs/paper

[s.html](#)

- GFS v2

- ◆ Chunk size = 1MB !!
- ◆ Hundreds of Distributed Masters (100 millions files managed by 1 master)

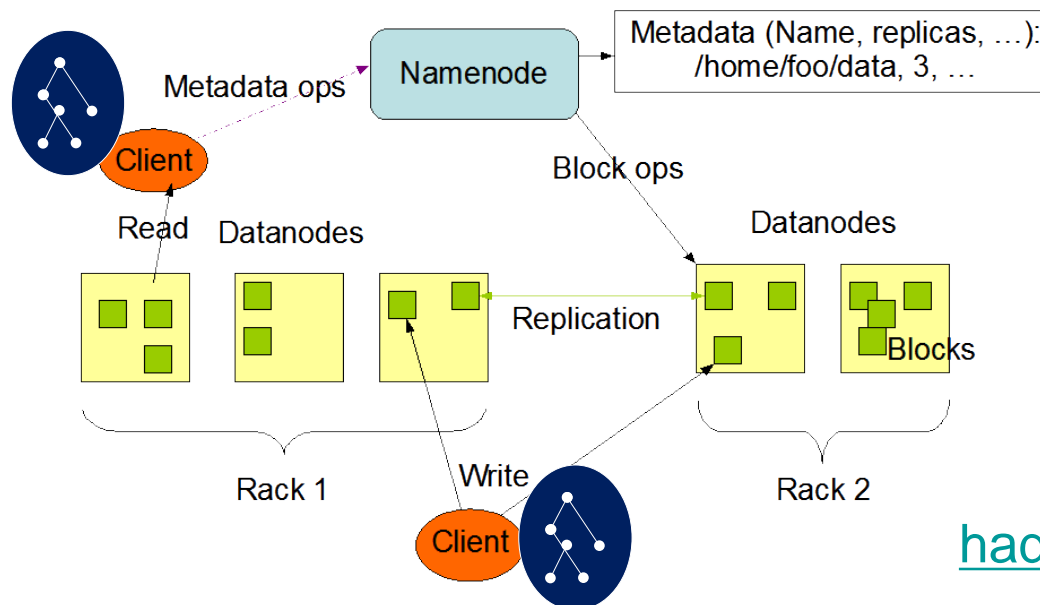
- Other example based on same approach: MooseFS, CloudStore (Quantcast FS based on KosmosFS)



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 philosophy
- HA NameNode (A/P Hadoop 2.0)
- Files striped across DataNodes
- Federation of Namenodes
- WebHDFS REST API, httpFS gw
- NFS layer, Data Services



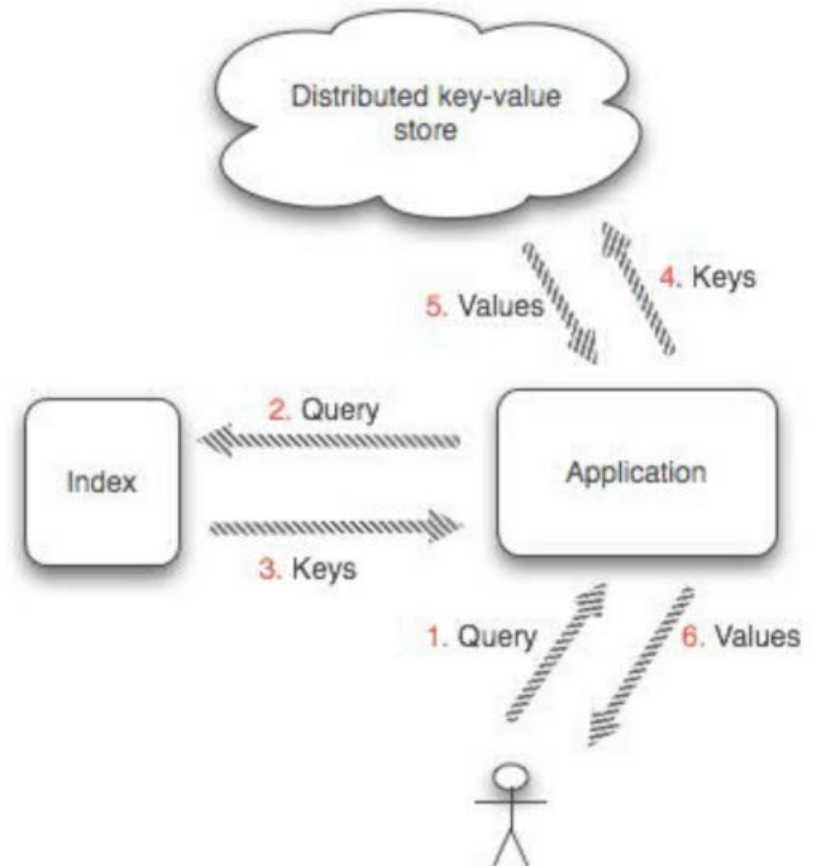
hadoop.apache.org

The new mandatory elements

- ◆ New (Updated) Distributed Topologies: Ring, P2P, Hierarchical, Flat or Asym
 - ◆ 1000s-10000s nodes with new fast inter-node routing techniques (Chord, Pastry, Tapestry or CAN methods)
 - ◆ New data placement model (some need to integrate some kind of MAID)
 - › Full Parallel Philosophy (“Divide and Conquer”)
- ◆ NoSQL as RDBMS can’t scale at that level but some maintain SQL (NewSQL)
- ◆ No more Relational data model
- ◆ Brewer CAP Theorem, Paxos algo...
- ◆ Object Storage and Key/Value Store
 - ◆ Consistency is very simple
 - ◆ True Scalability, no degradation
- ◆ Shared Nothing is the Rule
- ◆ User Mode
- ◆ No dedicated HW, everything is Commodities (at all level)
- ◆ Commercial and/or Open Source Software

Key Value Stores

- 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



Implementation example

- ◆ Object Storage as extension and union of KVS
 - ◆ CAS, OSD also belong to this category
 - ◆ Scale-out by nature
 - ◆ Multiple topologies (Ring, P2P, Hierarchical, Flat or Asym...)
 - ◆ Object = Metadata + Data
 - ◆ Billions of objects, no real limit
 - ◆ Flat single global namespace (“vertical” consistency)
 - ◆ Notion of Container with 3 basic operations: GET, PUT and DELETE
 - › Minimal layers and simplified data access pattern
 - ◆ Very simple interface with HTTP/REST (C++, Python, Java & PHP API)
 - › De facto standard such as Amazon S3 API
 - ◆ 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

SNIA CDMI (www.snia.org/CDMI)

- Today v1.0.2 - 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**

➤ Key metric: Durability


- ◆ 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

➤ RAID Can’t scale, Risk Exposure

- ◆ RAID6 not enough, Large disks today (4TB), very long to rebuild
- ◆ Storage protection overhead & rebuild process (resource and time)
- ◆ Potential impact of other users’ requests

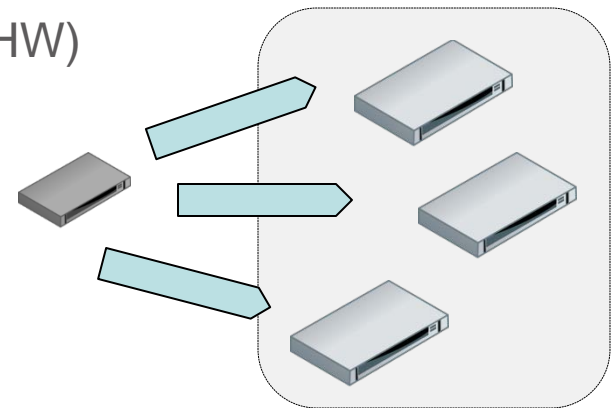
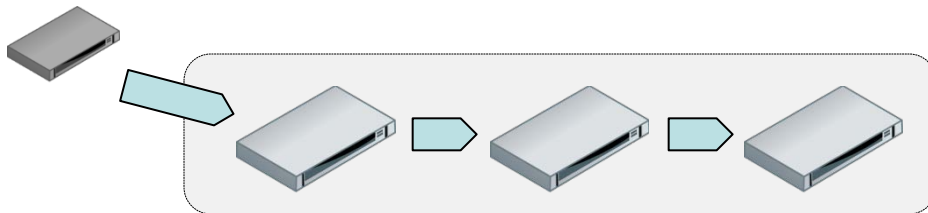
A horizontal line composed of several colored segments: purple, grey, yellow, blue, orange, grey, white, purple, grey, orange, grey, blue, grey, and yellow.

Large scale environments need other Data Protection mechanisms

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 $R + W \leq N$
 - Particular cases: $W \geq 1$: Writes always possible, $R \geq 2$: Minimal redundancy
- ◆ No data modification so data is easy to access
- ◆ Ideal model but very costly at large scale (HW)



2nd solution: Erasure Coding

➤ Erasure Coding techniques

- ◆ Reed Solomon, Forward Error Correction, Fountain Coding, IDA, Dispersed Storage...
- ◆ 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 (n) and Parity/Checksum (k) chunks noted (n,k)
- ◆ Ratio $(n+k/k)$ is key: common from 1.2 to 1.6 (capability to tolerate loss) and storage efficiency $(k/n+k)$
- ◆ Data format: Clear (native, no transformation) or scrambled (sort of encryption)
- ◆ Local within a Cluster/Ring or Geo (Dispersed), Local coupled with Replication for Geo
- ◆ Cost effective solution to control the cost associated with the data volume growth

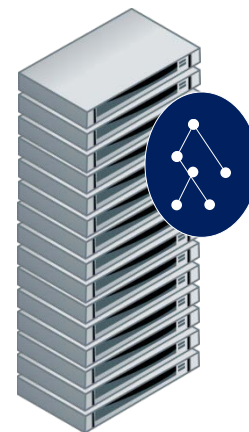
A decorative graphic consisting of multiple parallel, wavy lines in various colors including purple, blue, orange, grey, and yellow-green. The lines flow from the left side of the page, curving downwards and then upwards towards the right side, creating a sense of movement and flow.

Conclusion

Conclusion

➤ Massively Scalable File Storage is a reality

- ◆ Cost effective approach with COTS
- ◆ Acceptance of 10 of thousands of machines (clients, servers)
- ◆ Asymmetric (Master/Slave) and P2P seem to be the more scalable philosophies
- ◆ Unified, Global Namespace is fundamental
- ◆ Superiority of Object over File and Block approach at scale
- ◆ Reliability & Security are key (authorization, authentication, privacy...)
- ◆ Self-Everything (Autonomous system)
- ◆ Other services: Monitoring, Billing, Multi-Tenancy...
- ◆ Think de facto or industry Standard (CDMI for instance)



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 March 2015, 2014,
August, Feb. 2013
& July 2012

Additional Contributors

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

A decorative graphic consisting of multiple parallel, wavy lines in various colors including purple, blue, orange, and green, flowing from the left side of the slide towards the right.

MASSIVELY SCALABLE FILE STORAGE

Philippe Nicolas, Scality