SCALITY

SDC

STORAGE DEVELOPER CONFERENCE

SNIA ■ SANTA CLARA, 2014

Scality RING: Scale Out File System & Hadoop over CDMI

Paul Speciale, Sr. Dir. Product Management

paul.speciale@scality.com

Agenda

- Scality Introduction
- Scality RING
- Core architecture
- Co-existence of Objects & Files
- Scale-out File System (SOFS) & CDMI
- Apache Hadoop
- Scality RING for Hadoop
- Conclusion

Scality – Quick Facts

SCALITY

- Founded in 2009
- Experienced management team
- Corporate HQ in San Francisco, Global reach
- ~85 employees, ~35 engineers in Paris
- 24 x 7 support team
- 3 US patents
- Privately held: Menlo Ventures, Galileo, IRIS Capital, Omnes Capital, Idinvest, FSN Prime
- Software-Defined Storage Solution: Scality RING

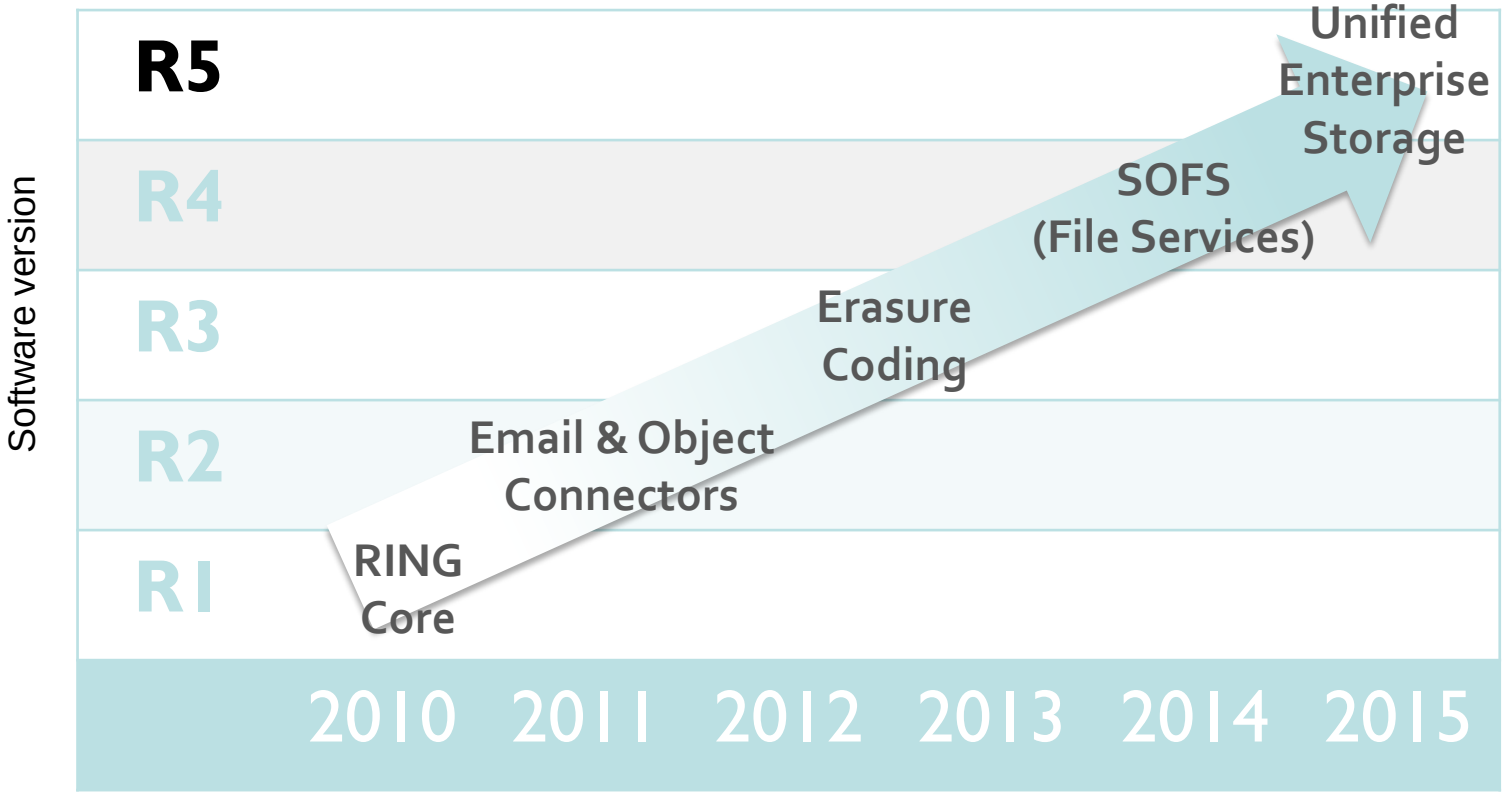
Our Use Cases & Customers

- Email
- Media Content Distribution
- Distributed Computing
- Active Archives
- Web & Cloud
- Enterprise Global Cloud

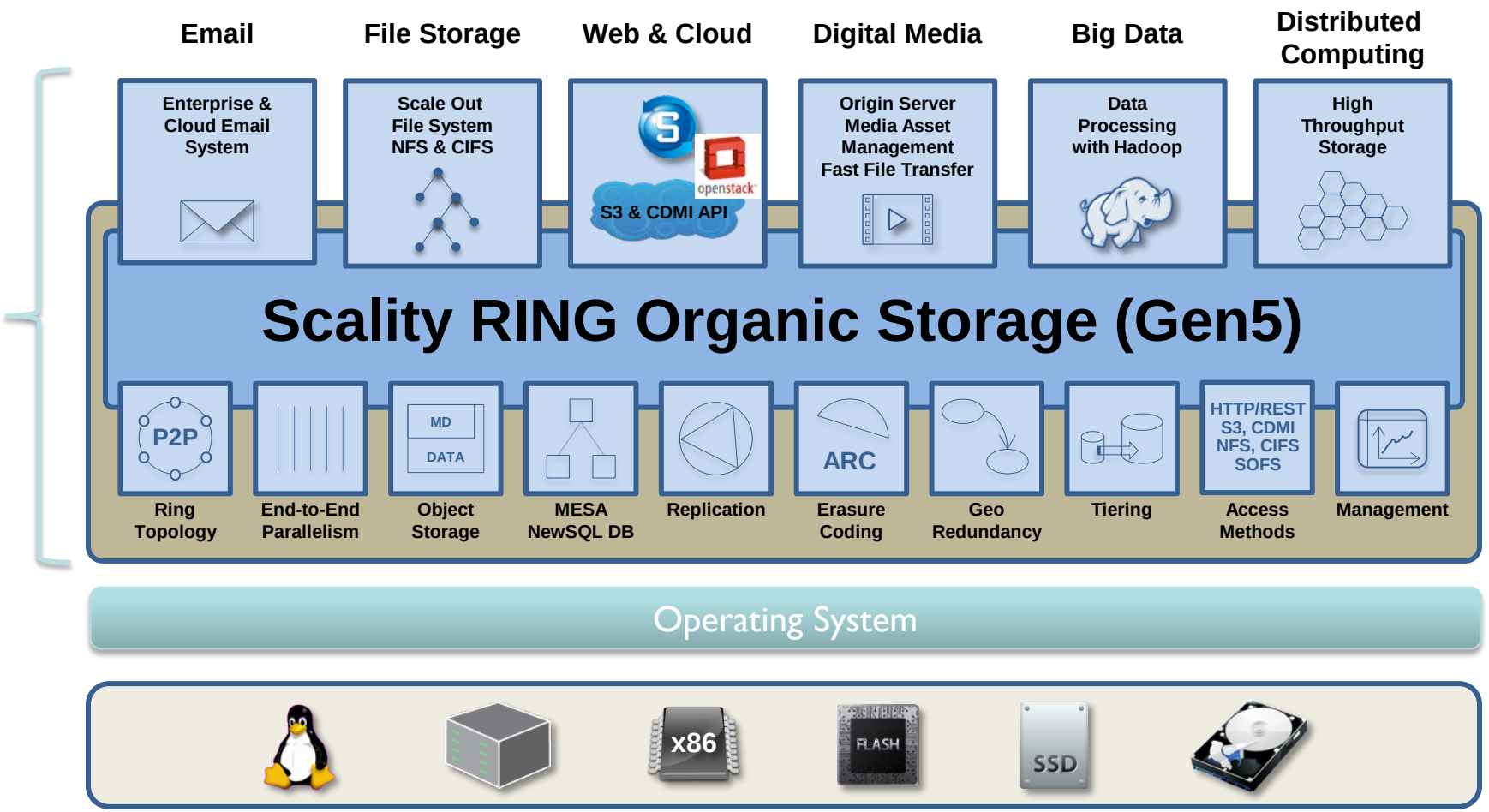
“Government Agencies”

Scality Release History

- Scality RING is currently 4th generation software with proven customer production deployments & refinement cycle (5th gen announced for late 2014)



RING Software Defined Storage



Scality RING Access Connectors

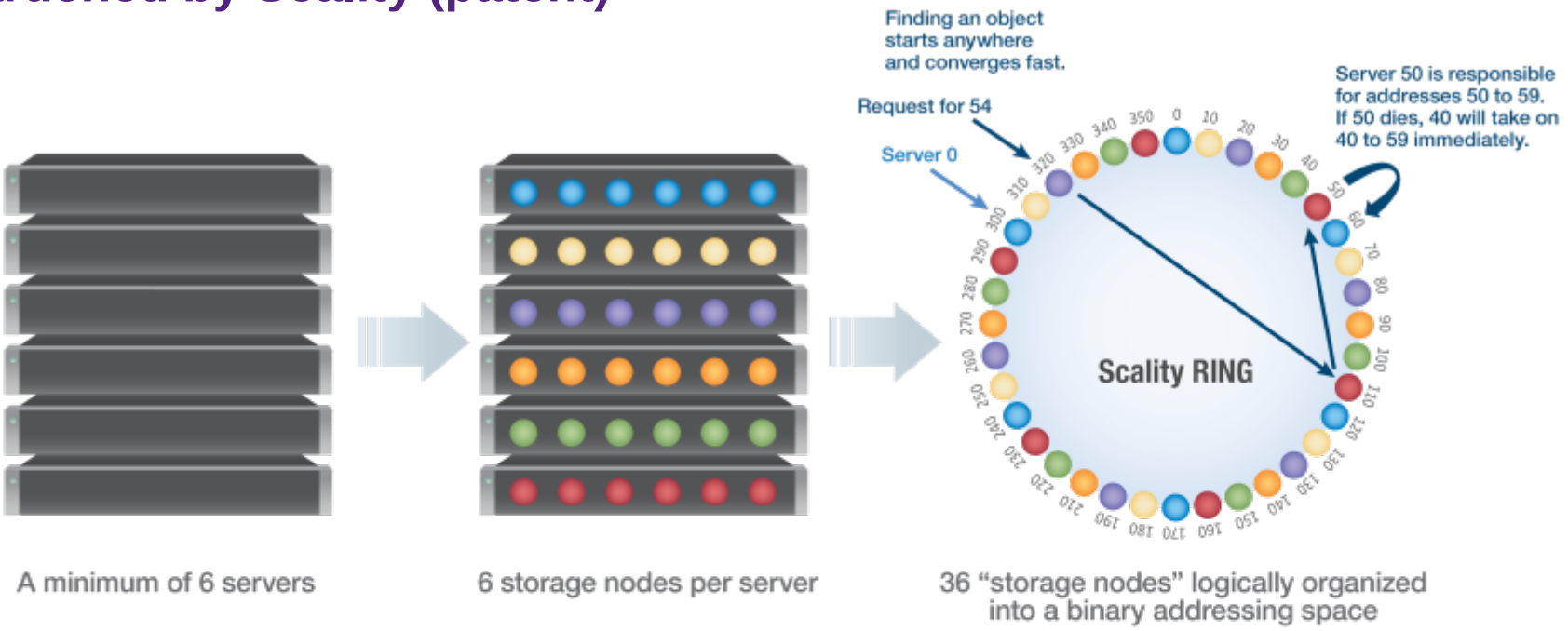
- Object Connectors (REST)
  - RS2 (S3 compatible)
  - SNIA CDMI
  - OpenStack Swift (Juno release fall 2014)
  - Native REST: sproxyd
- Native File Connectors (no external gateways required)
  - Native Scality Scale Out File System (SOFS)
  - NFS, SMB & FUSE (Namespace compatible with CDMI REST)
- Block Storage
  - OpenStack Cinder Driver: since Grizzly in 2013 (also Havana, Icehouse & Juno upcoming)
  - Open Source REST Block Driver (over CDMI)
- Hadoop HDFS
  - Standard HDFS API via Dewpoint Server (CDMI)

RING Core CHORD protocol

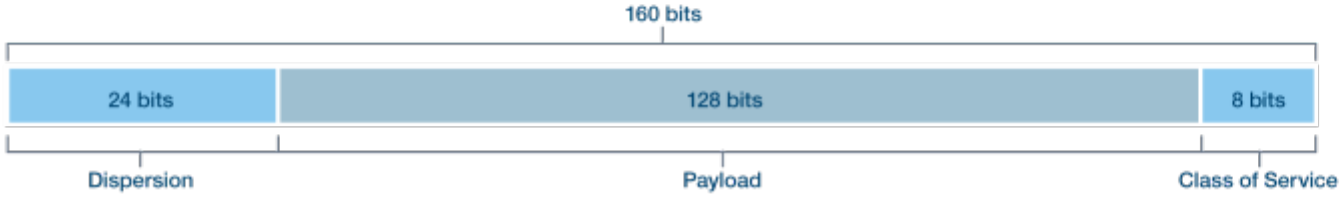
2nd generation peer-to-peer algorithm Originated at MIT

Designed for failure, fast lookup, and extremely resilient

Hardened by Scality (patent)



Data is stored and recalled based on a 128 bit address, which can be automatically mapped to URL.



Distributed Hash Table

- Scality's implementation of CHORD includes an efficient Distributed Hash Table (DHT) algorithm
  - Consistently and efficiently maps a particular key to a set of storage nodes
  - Mapping keys to values is distributed among the nodes
  - This is done in such a way that a change in the set of participants causes a minimal amount of disruption
- Provides a fast, decentralized distributed lookup
  - Key-value pairs are stored in the DHT and any participating node can efficiently retrieve the value associated with a given key
  - O log2(N) lookups (where N = number of nodes): e.g., 3 lookups for 1000 node RING, 4 lookups for 10,000 node RING
- The DHT is able to scale to extremely large numbers of nodes
  - Continuously handles sudden node arrivals, departures, and failures

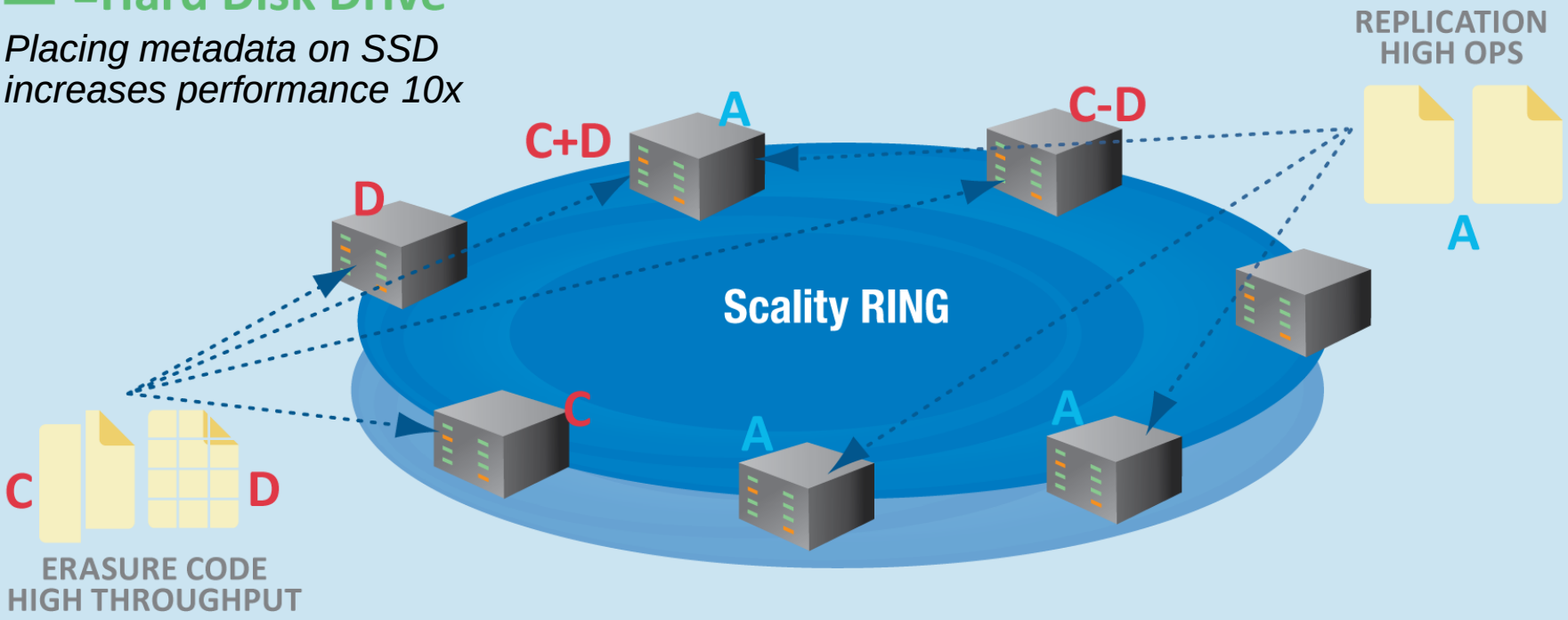
Two protection systems: replication and erasure coding

=Solid State Drive

=Hard Disk Drive

Placing metadata on SSD increases performance 10x

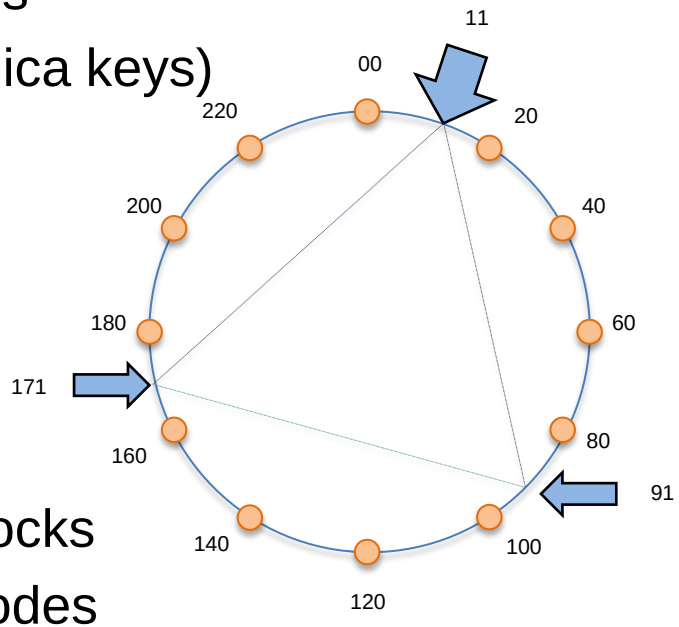
For small files, replication delivers high IOPS on HDD



For large files, erasure coding improves efficiency, saves money and increases throughput. Scality's erasure coding leaves data blocks in clear to increase read performance.

Data Protection: Replication

- No data transformation
  - Clear / native data format for fast access
  - Simple projection (no need to store replica keys)
- Class of Storage
  - From 0 to 5 replicas (1 to 6 copies)
- Multi-Rack & Site aware
  - Full independent object location
- Self-healing
  - Auto-heals missing replicas or parity blocks
  - Auto-rebalancing for new or dropped nodes
  - Transparent proxy for object being balanced
  - Permanent CRC of all contents (no silent data corruption)

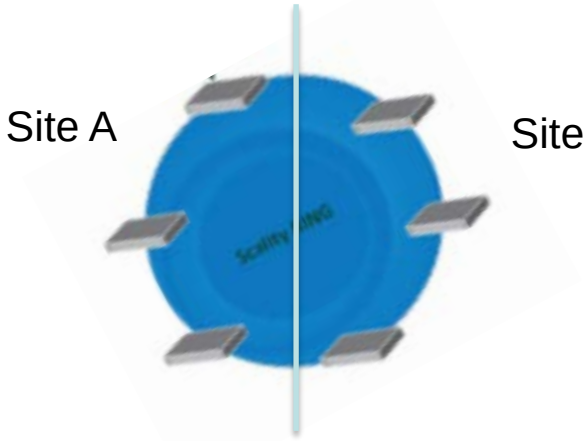


Data Protection: Erasure Coding

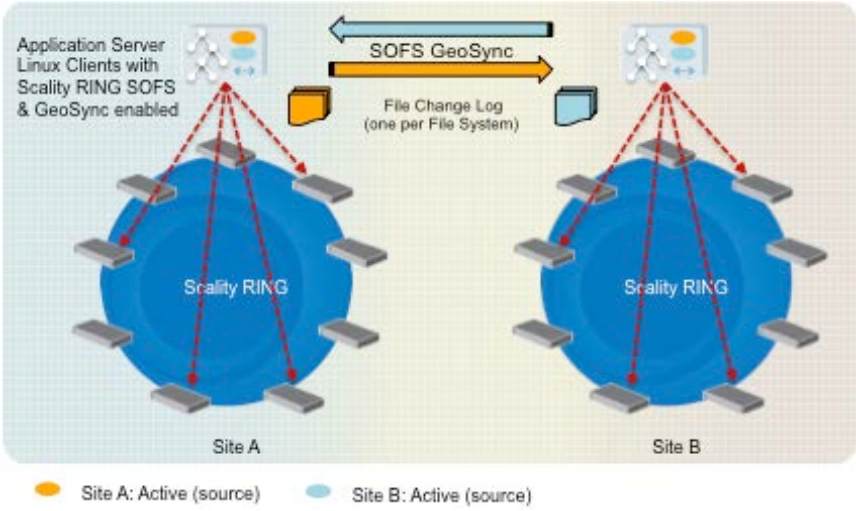
- Scality ARC (Advanced Resiliency Configuration)
  - Erasure coding for optimal data protection of large files and objects
  - Avoids the overhead of multiple replicas (data copies)
  - Variable data parity scheme to store data with improved storage efficiency
  - Protects against disk, server, rack or data center level failures
- ARC Schema
  - The combination of data chunks + parity chunks is the schema
  - Data chunks stored in clear to avoid read performance penalties
  - Can protect against more failures, e.g., 14+4 Schema provides four (4) disk failure protection with only 28% overhead
- Flexible & Dynamic Allocation
  - Choose Replication or ARC at the time an object is created
  - ARC schema is variable

Geo-distributed RING Deployments

- Stretched RING
  - Single logical RING across multiple sites
  - For lower-latency WAN environment for synchronous operations
  - Location aware: Replication or ARC
  - tolerates 1 of 2 data center outages

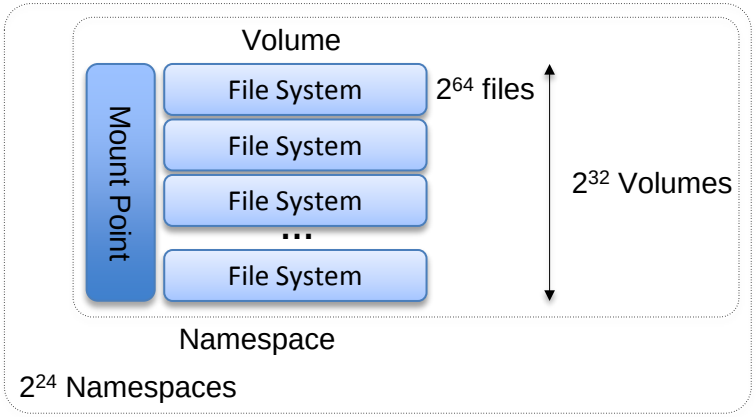


- Multi-Ring
  - Asynchronous replication across multiple sites
  - ARC or replication modes (can be mixed)
  - Best for higher-latency WANs
  - Can also be deployed as cascading replication



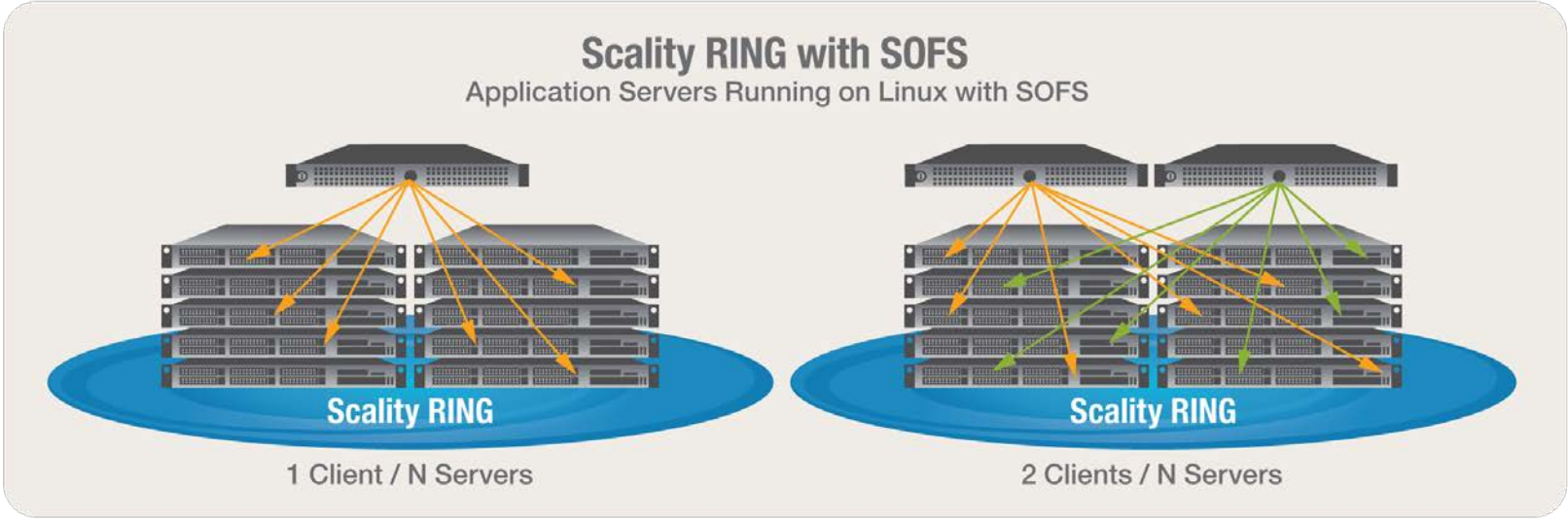
Scality Scale Out File System

- Parallel Network File System
- FUSE-based (POSIX compliant)
- Direct data access, no gateway
- Highly Scalable
  - 2^32 volumes (=FS), 2^24 namespaces
- Very High Capacity and Billions of Files
  - 10^20 addressable files (2^64 files)
- Sparse files to efficiently support large files
- Leverages Scality MESA Internal Distributed DB to enable a virtual file system
  - Provides the inode & directory structures
  - Fully transactional semantics across the RING



Scality Scale Out File System

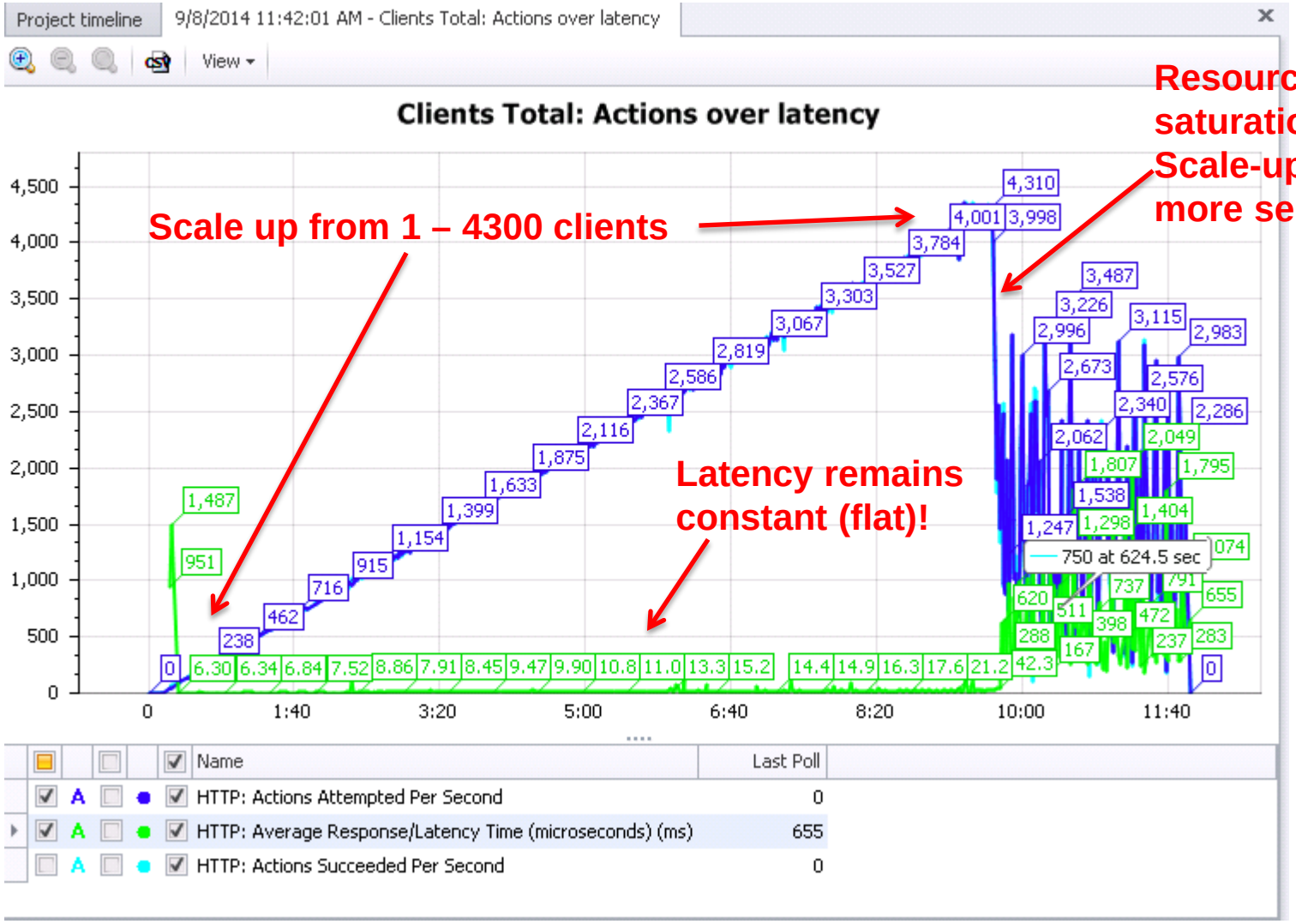
- Linux FUSE, NFS or SMB access
- High Aggregated Throughput (clients to RING nodes)
- Auto-Scaling with Storage or Connector Nodes addition
- GUI & CLI for Management



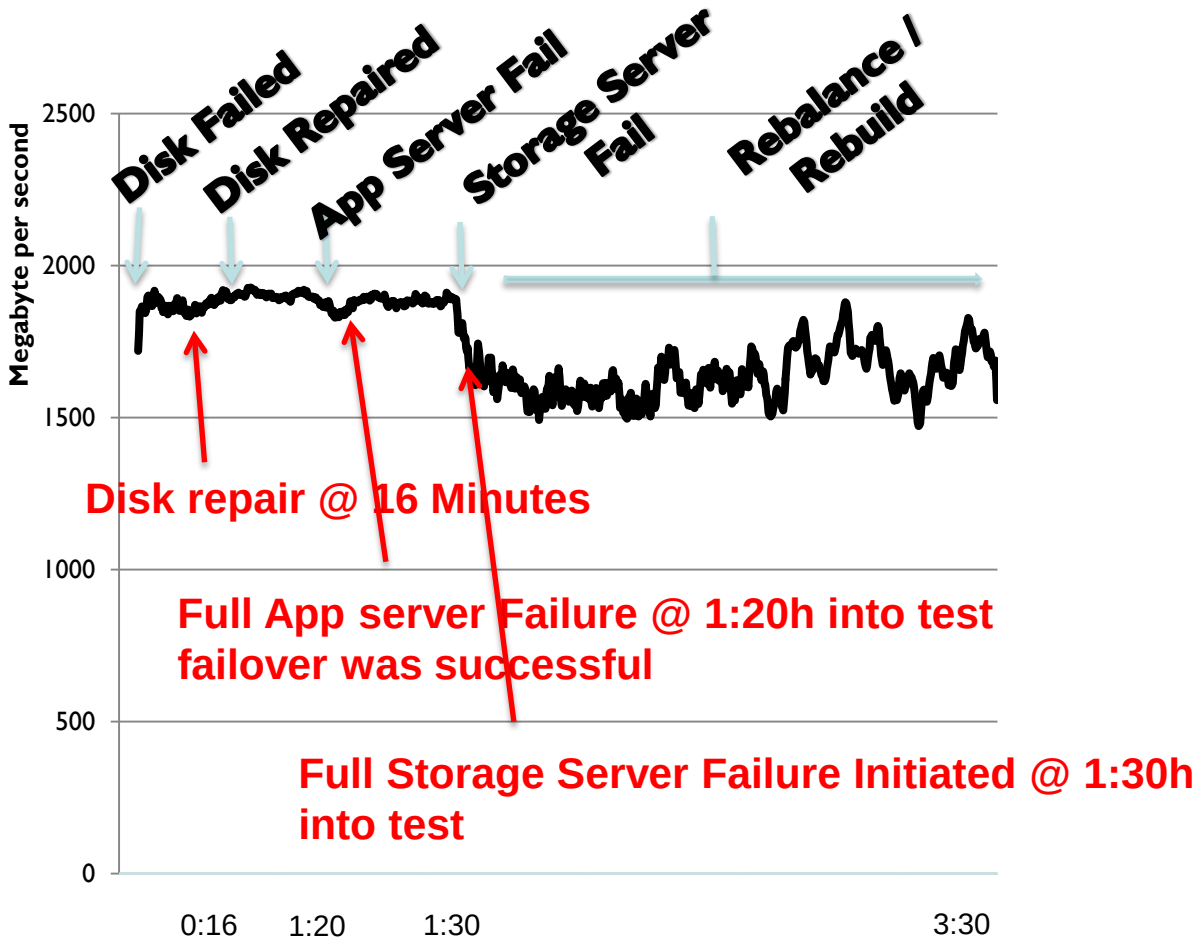
Performance & Scalability

- Throughput & IOPS
  - Line rate 10 GbE over object (sproxyd) & file (NFS) per client
  - ~100,000 (4K) read IOPS on 18 node cluster
  - Scales linearly as connectors are scaled out
- RING scales performance linearly at scale
  - Fully utilizes the pool of resources: CPU, memory, disk spindles
  - Maintains latency as #clients are increased
- Negligible degradation as system fills up
  - CHORD algorithm equally balances across nodes/disks
- Deterministic performance during component failure & rebuild
  - Proportional to affected resources

Why Parallelism Matters



Server rebuild performance impact



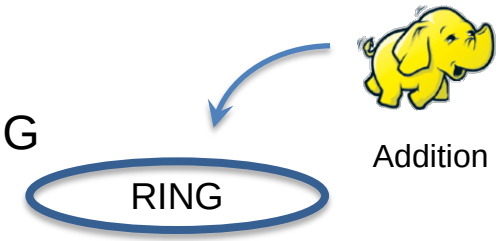
- Basic 6 Server RING config
- Load generator
  - HTTP PUT/GET 1GB Chunk Size
  - Continually running
  - 1800 MB/sec sustained before failure injections
- Sequence of failure injections:
  - Disk
  - App Server
  - Full storage server
- Self healing ran unrestricted for remainder of test
- 60TB total rebuilt/rebalanced during 2 hour period

Scality RING for Hadoop

- Data Compute and Storage Platform in ONE Cluster
  - RING is primarily a petabyte scale data platform – but can be leveraged for Hadoop
  - Leverage Scality RING to process data *in-place* (data already stored)
- Scality Scale Out File System instead of HDFS
  - No single point of failure, no performance bottleneck, no size limitation
  - Full symmetric model – no need for NameNode
  - First industry CDMI integration with Hadoop
- Advanced Data Protection
  - Data Replication (up to 6 copies)
  - Can also use with Scality ARC erasure coding (~30% overhead)
- Validated with standard Hadoop distributions (not another distro!)
  - Hortonworks HDP 1.x (2.x planned)
  - Cloudera CDH4 (5.x planned)

Use Cases for Convergence

- Existing RING extended to run Hadoop
  - Capability to run Analytics on data stored in the RING without any ETL process
  - Addition of computing on storage nodes
  - Good fit for “80% workloads” but not for real-time processing
- New RING dedicated to Hadoop
  - New specific Hadoop project but capability to access and store data outside of Hadoop
- RING for Hadoop and “Generic” data
  - Coexistence of analytics data plus regular object and file data

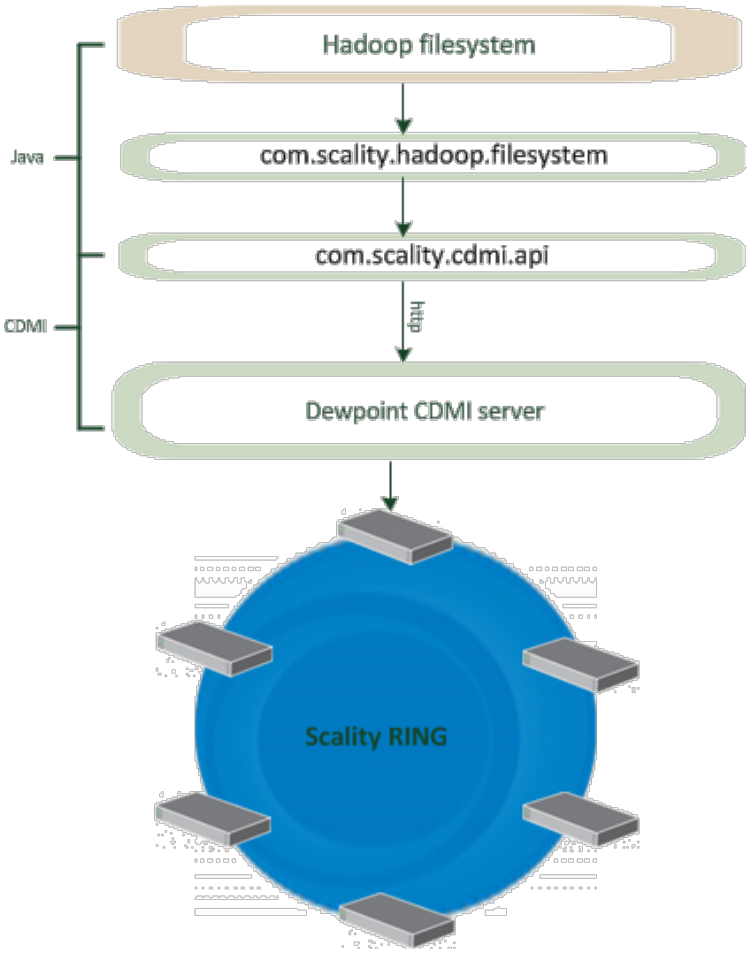


New combined project

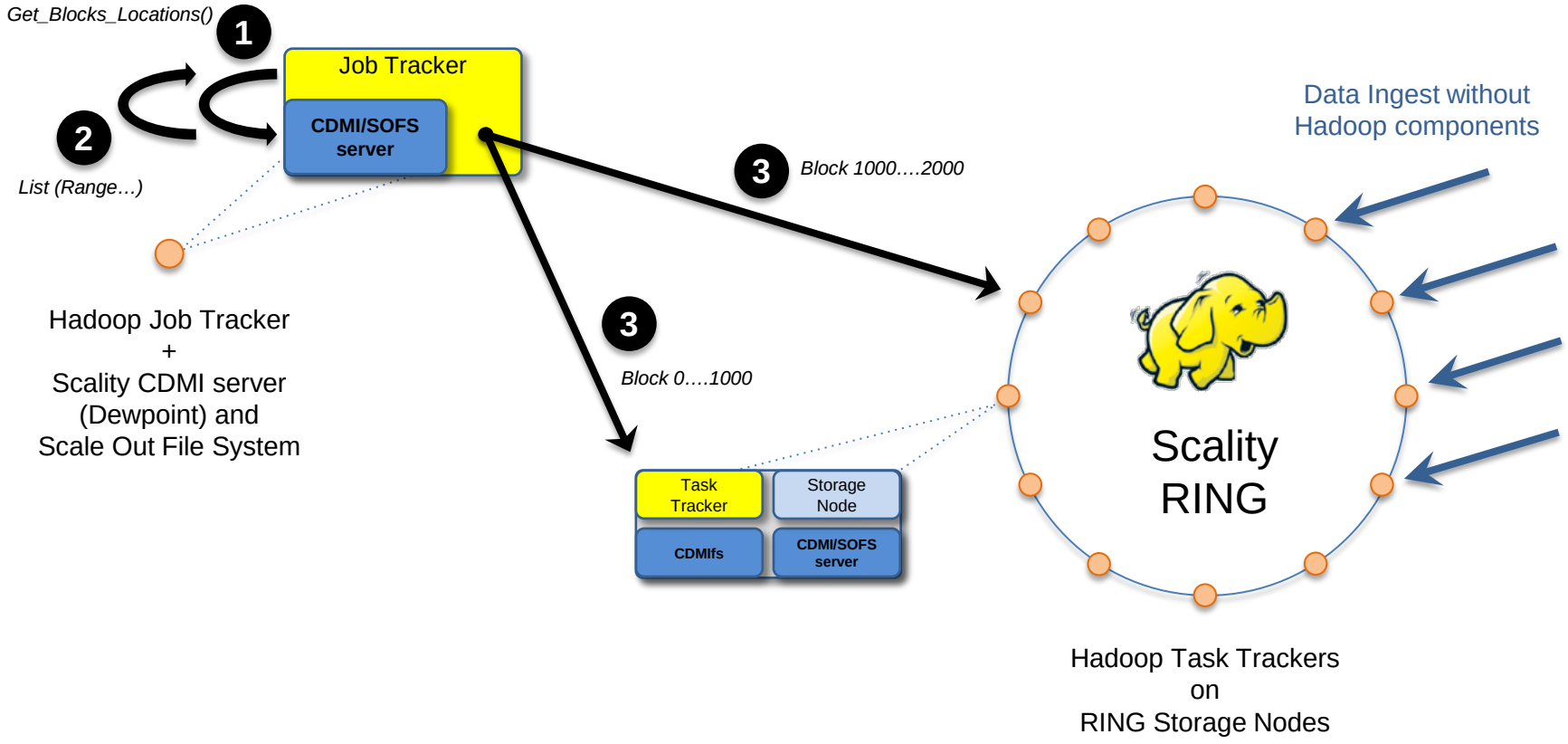
Coexistence: same storage infra.
for potentially different data usage

RING Components for Hadoop

- Key Scality components
  - com.scality.hadoop.filesystem
    - Behaves like Hadoop file system API (implements the org.apache.hadoop.fs FileSystem interface)
    - No changes required to application
    - Based on the com.scality.cdmi.api
    - Core value-added IP is GetBlockLocations()
  - com.scality.cdmi.api
    - Java library used to implement routine file system operations
    - Similar to a standard file system API
    - Glue between File System and CDMI server
    - Use of Apache HttpComponents library
  - Dewpoint Server (CDMI)
    - Maps REST calls into Scality SOFS
    - Also installed in the Hadoop JobTracker server



Scality RING for Hadoop



Hadoop

Scality RING

Scality extension for Hadoop

Hadoop Cluster on Scality RING

(example: 12 Hadoop nodes for 12 storage nodes)

Scality CDMI Server & Clients

- CDMI
  - Scality Dewpoint Server (CDMI Server)
    - Based on CDMI 1.0.1, 1.0.2
  - Scality Droplet Library (CDMI Client)
    - Open Source Cloud client lib (since 2010)
    - Portable C API with bindings for Perl, Python, Ruby…
- CaDMIum is a CDMI client java library
  - Open source file system tools such as distributed notifier
  - Examples: Content Indexing, Data Mover, Geo Replication
- Available for download on GitHub
  - https://github.com/scality

Summary

- Scality RING Software Defined Storage
  - Hardware-agnostic for industry standard servers
  - Flexible data protection schemes
  - Scalable to petabytes and supports mixed workloads
  - Geo-distributed deployment options
- Wide Application Support
  - Object applications
  - File applications
  - VM & Block Storage
  - Hadoop storage

Questions?

paul.speciale@scality.com