Library of Congress

I/O Considerations in Big Data Analytics

26 September 2011

Marshall Presser
Federal Field CTO
EMC, Data Computing Division

Paradigms in Big Data

Structured (relational) data

- Very Large Databases (100’s TB +)
- SQL is the access method
- Can be monolithic or distributed/parallel
- Vendors distribute software only or appliance

Unstructured (non-relational data)

- Hadoop Clusters (100’s + nodes)
- Map/Reduce is the access method
- Vendors distribute software only (mostly)

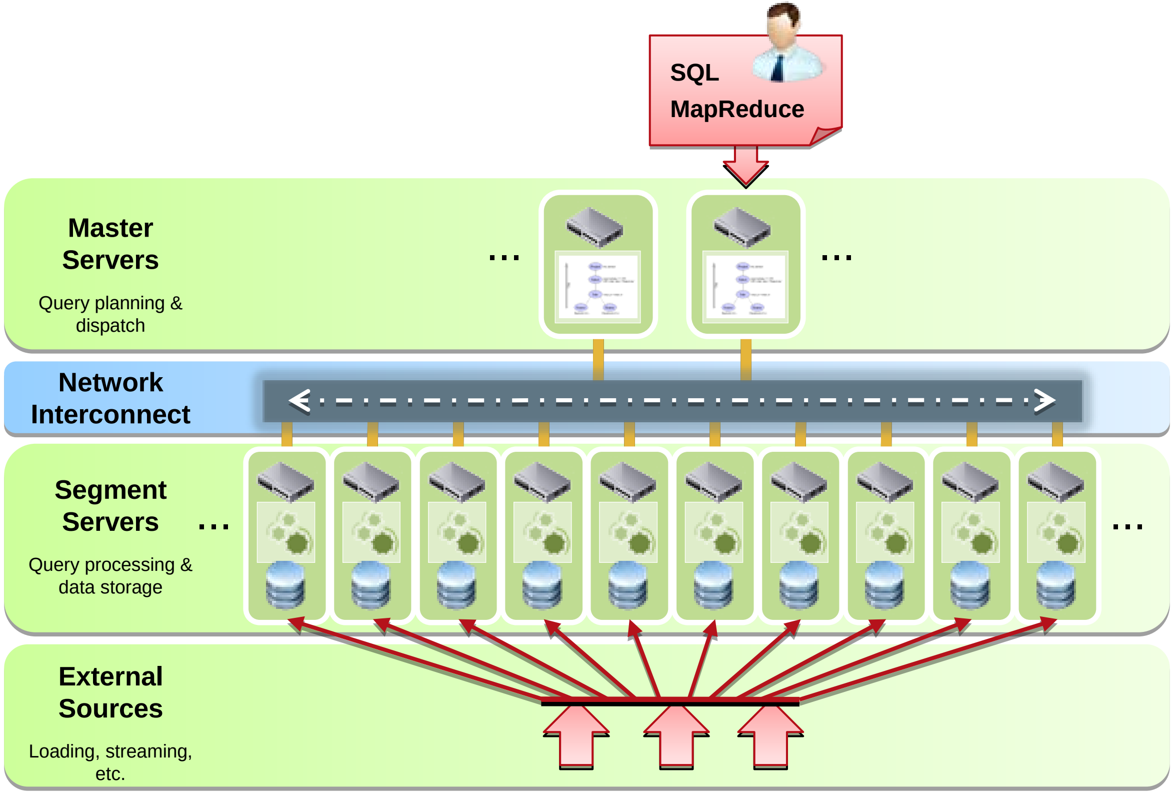
Obstacles in Big Data

Both Relational and Non Relational Approaches must deal with I/O issues:

- Latency
- Bandwidth
- Data movement in/out of cluster
- Backup/Recovery
- High Availability

MPP (Massively Parallel Processing) Shared-Nothing Architecture

- MPP has extreme scalability on general purpose systems
- Provides automatic parallelization
  - Just load and query like any database
  - Map/Reduce jobs run in parallel
- All nodes can scan and process in parallel
  - Extremely scalable and I/O optimized
- Linear scalability by adding nodes
  - Each adds storage, query, processing and loading performance



Software and Appliances in Relational Big Data

Greenplum DCA – EMC   (software and appliance)

Neteeza Twin Fin – IBM (appliance only)

Teradata 2580 – Teradata (appliance only)

Vertica – HP (software and appliance)

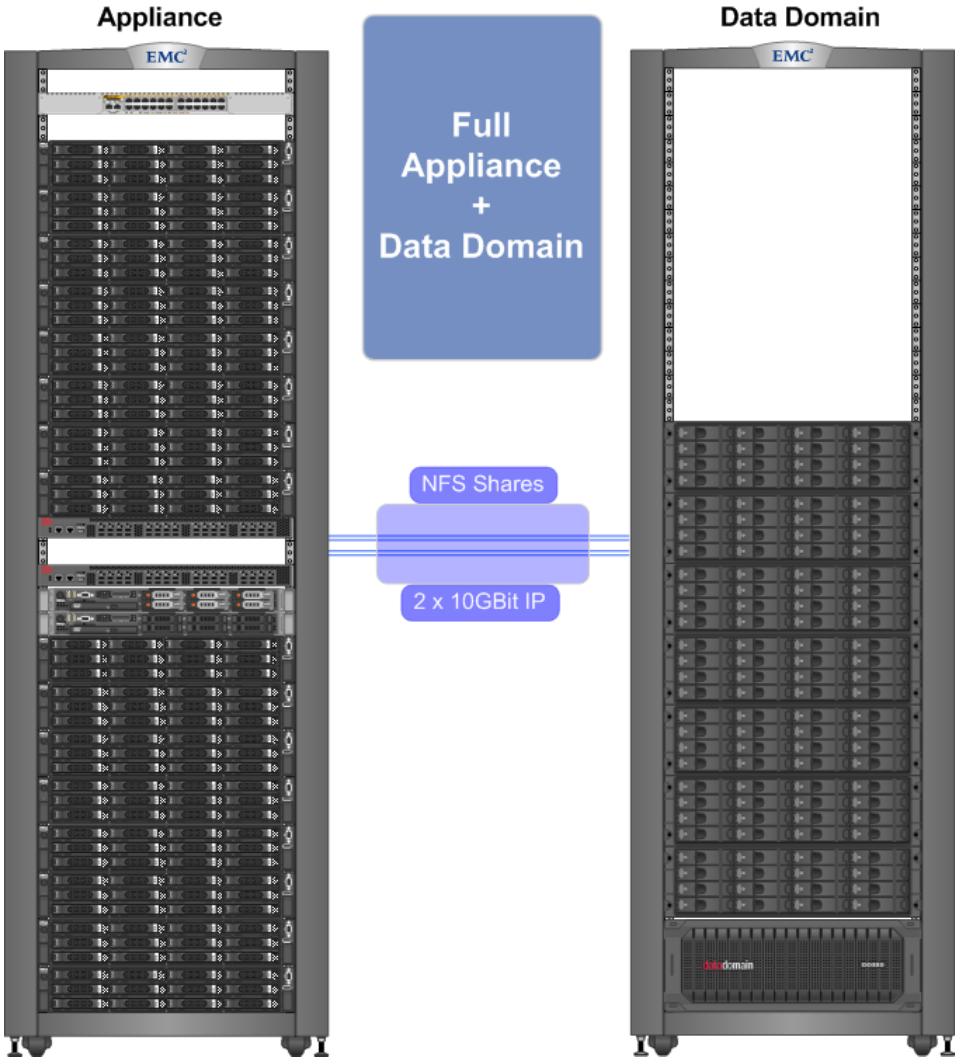
- All above use distributed data with conventional I/O
- Neteeza and Teradata virtual proprietary network s/w

Exadata – Oracle (appliance only)

- Oracle is the only vendor with a shared disk model
- Uses Infiniband to solve latency and bandwidth issues

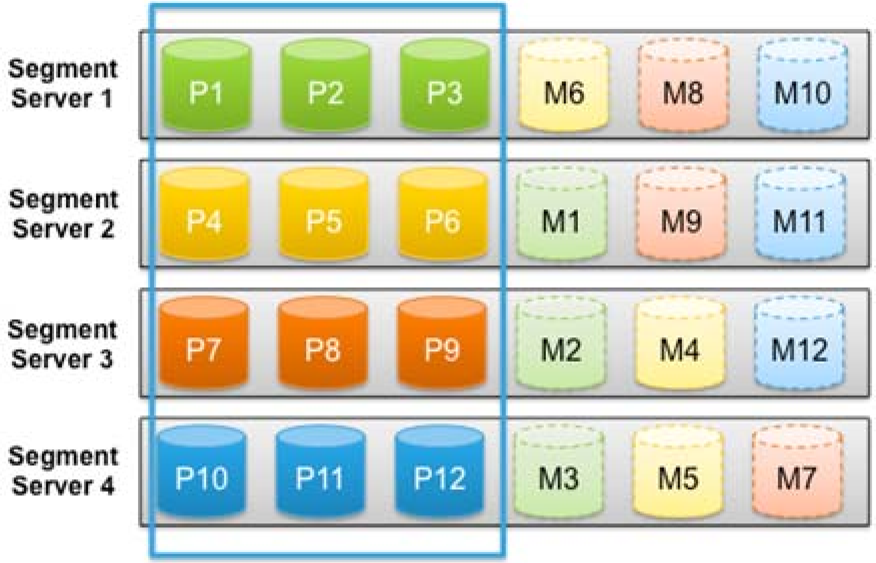
Backing up to from an Appliance

- Requirements:
- Parallel backup from all nodes, not just the master
- Incremental or dedup ability via NFS shares or similar
- Conneted to private network, not public



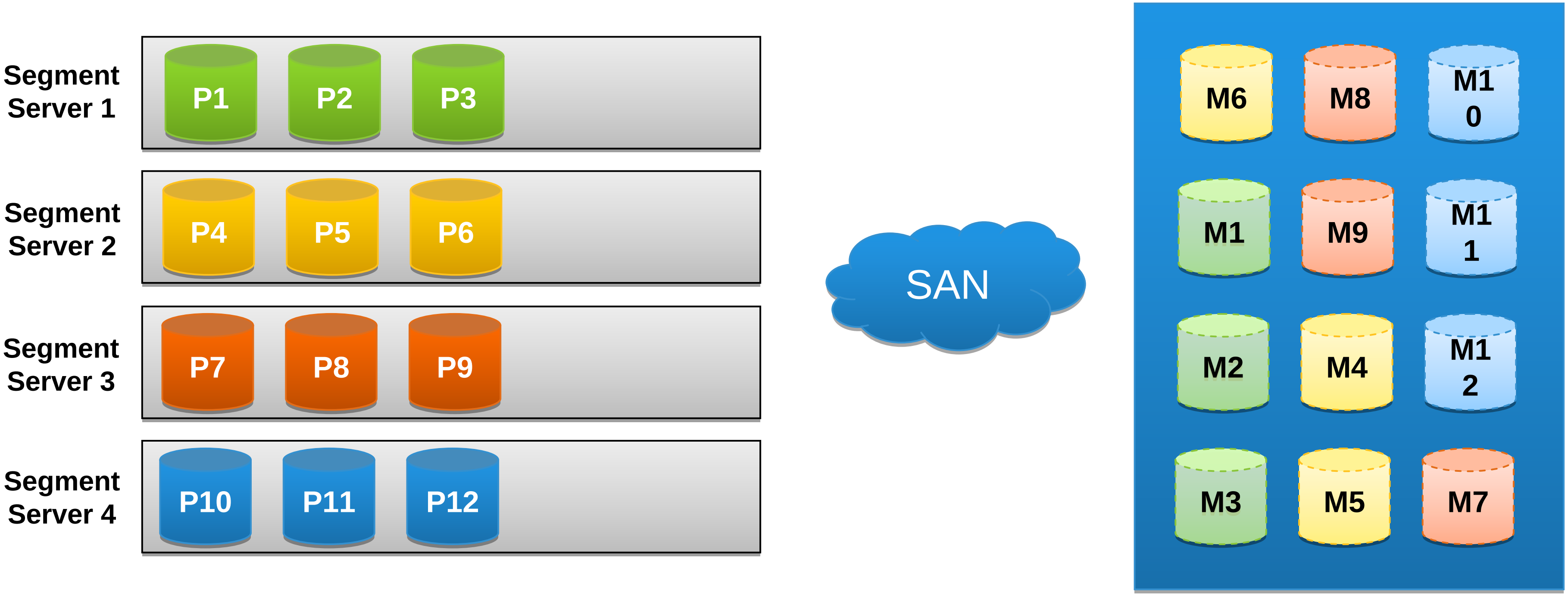
MPP Database Resilience Relies on In-Cluster Mirroring Logic

- Cluster comprises
  - Master servers
  - Multiple Segment servers
- Segment servers support multiple database instances
  - Active primary instances
  - Standby mirror instances
- 1:1 mapping between Primary and Mirror instances
- Synchronous mirroring



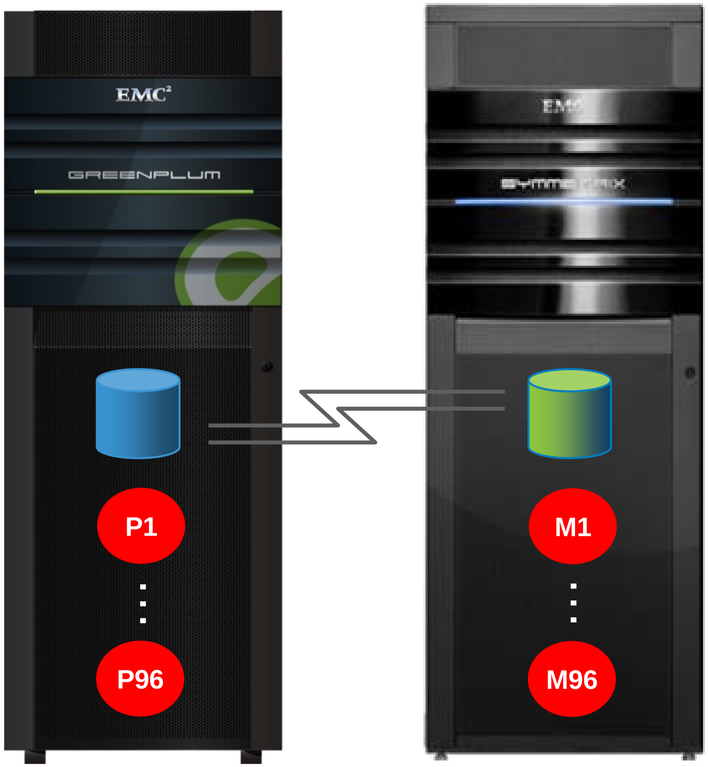
Set of Active Segment Instances

SAN Mirror Configuration: Mirrors Placed on SAN Storage



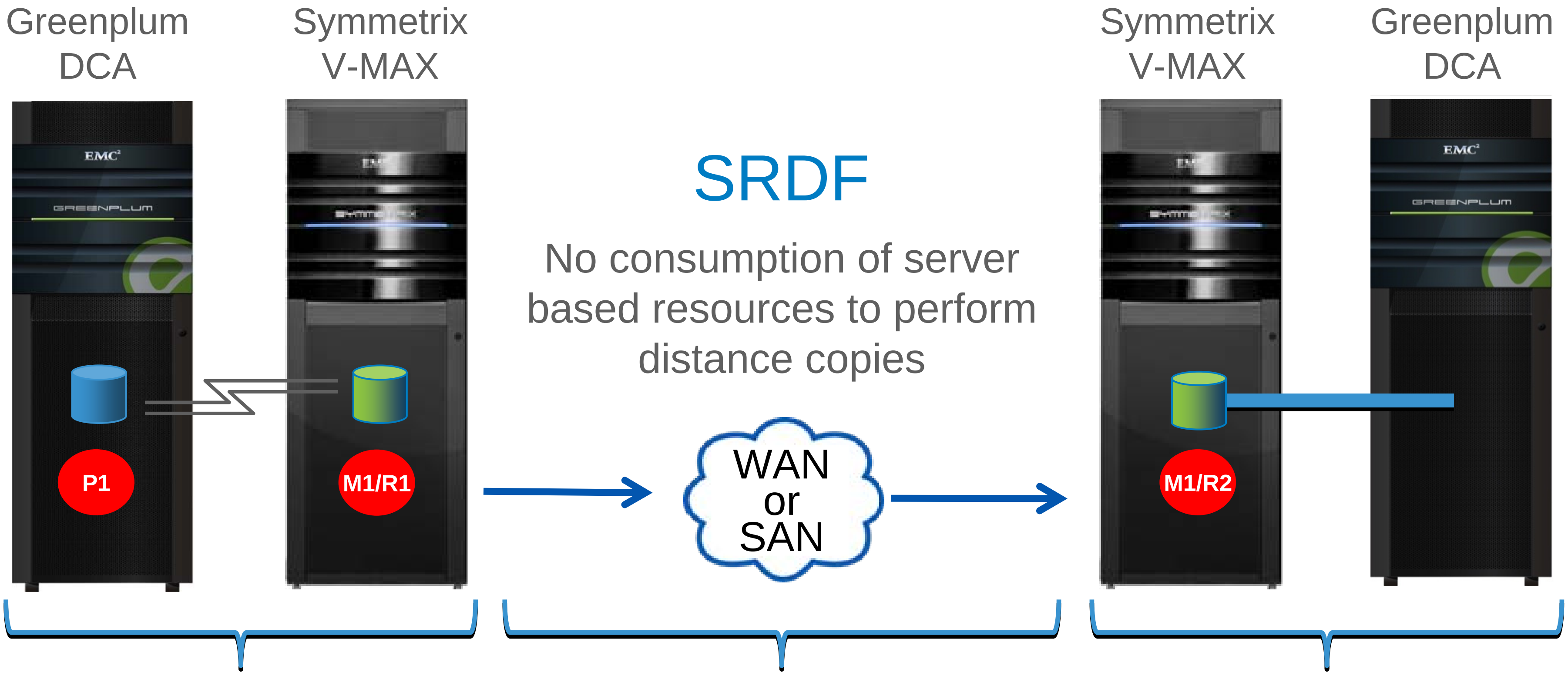
Doesn’t this violate principle of all local storage? Maybe, maybe not.

One Example: SAN Mirror to VMAX SAN



- Default DCA configuration has Segment Primaries and Segment Mirrors on internal storage
- SAN Mirror offloads Segment Mirrors to VMAX SAN storage
  - Doubles effective capacity of a DCA
  - Foundation of SAN leverage
  - Seamless off-host backups
  - Data replication
- No performance impact
  - Primaries on internal storage
  - SAN sized for load and failed segment server

One Example: SAN Mirror –With SAN based replication for DR



SRDF

No consumption of server based resources to perform distance copies

SAN Mirror

Synchronous or Asynchronous
Performed by SRDF / V-MAX

Database on SAN

What is Hadoop?

Three major components

- An infrastructure for running Map/Reduce jobs
  - Mappers produce name/value pairs
  - Reducers aggregate Mapper Output
- HDFS - A distributed file system for holding input data, output data, and intermediate result
- An ecosystem of higher level tools overlaid on MapReduce and HDFS
  - Hive
  - Pig
  - Hbase
  - Zookeeper
  - Mahout
  - Others

Why Hadoop?

- With massive growth of unstructured data Hadoop has quickly become an important new data platform and technology
  - We’ve seen this first-hand with customers deploying Hadoop alongside relational databases
- A large number of major business/government agency are evaluating Hadoop or have Hadoop in production
- Over 22 organizations running 1+ PB Hadoop Clusters
- Average Hadoop cluster is 30 nodes and growing.

Why Not Hadoop?

Hadoop still a “roll your own” technology

Appliances just appearing Sep/Oct 2011

- Wide scale acceptance requires
  - Better HA features
  - More performant I/O
  - Ease of use and management
- Access to HDFS through a single Name Node
  - Single point of failure
  - Possible contention in large clusters
  - All Name Node data held in memory, limiting number of files in cluster
- Unlike SQL, programming model via Hadoop API a rare skill
- Apache distribution written in Java, good for portability, less good for speed of execution

Storage Layer Improvements to Apache Hadoop Distribution

- HDFS optimizations
  - Recoded in C, not Java, different I/O philosophy
  - Completely API compatible
- NFS interface for data movement in/out of HDFS
- Distributed Name Node eliminates SPOF for Name-Node
- Remote Mirroring and Snapshots for HA
- Multiple readers/writers – lockless storage
- Built-in transparent compression/encryption

Thank you

Marshall Presser| 240.401.1750 – cell | marshall.presser@emc.com