Teradata Database Architecture Overview

Todd Walter
Chief Technologist
Teradata

Session #637 – 11:30 – 12:15 Sunday, September 11, in C101
2500 BC
Building Pyramids

• Problem - Feed tens of thousands of workers building the pyramids
  – With little technology
  – Difficult communications

• Solution
  – Take the sole proprietor model and replicate it
  – Many individual bread makers
  – Many individual butchers
  – Each delivers their capacity
  – Collectively they build the pyramids

• How do you support an increase in the workforce?
Problem: Running a mail order retail business

Solution: Massively parallel processing individualized CRM
- Each customer recorded on index cards in giant card file
- Thousands of clerks
- Each order added to customer's card
- Suggestions handwritten on receipt
- Capturing and using detailed transaction history

How do you handle an increase in customers or orders?
1979
Birth of Teradata

- Microprocessor - 8086
- Personal computer - Commodore 64, Apple I, TRS80
- Disk drive - 200MB weighs 30lbs and requires a washing machine sized cabinet
- Enterprise computing - Blue or Bunch mainframe
- Client/Server computing - didn’t exist
- Internet and e-mail - Only if you are a university researcher or DoD

Teradata was founded to solve enterprise Terabyte problems utilizing microprocessors and commodity disks.
What Do These Have In Common?

“Shared Nothing”
- Take a big task
- Slice it vertically into a (large) number of smaller tasks
- Perform those tasks independently
- Balance the work so all the tasks complete simultaneously
- Assign the tasks evenly among the physical resources
- Communicate only at the beginning and end of a task

Benefits
- Large task completes in a short elapsed time
- Maximizes use of resources
- Minimizes communication bottlenecks
What Makes Teradata Unique

Key Components/Concepts
Teradata Data Warehouse Technology Is Unique

- Parallelism built-in from the ground up
- Dedicated to automatic management and operation
- Easy and fast data movement in and out
- Committed to the highest levels of reliability and availability
- High concurrency mixed workload management
- Parallel extensibility for high performance dynamic functionality
- Unequaled scalability in every dimension

Architected to deliver business value to users
What Makes Teradata Unique

Key Components/Concepts

- MPP Architecture
- Availability
- Data Management
- Workload Management
- Query Execution
- Extensibility
- Optimizer
As the system scales (e.g. add more nodes), it should be no additional administrative overhead.

As it scales, there should be linear improvement of performance.

The rest of the world is catching on to our MPP concept. We’ve done it from the day one.

Teradata no longer assumes that we always provide the hardware/platform that our database runs on.
Teradata MPP Server Architecture

• **Nodes**
  – Incrementally scalable to 2048 nodes
  – Current state of the art Intel chips

• **Operating System**
  – Linux (SUSE)

• **Storage**
  – Independent I/O
  – Scales per node

• **BYNET Interconnect**
  – Fully scalable bandwidth

• **Connectivity**
  – Fully scalable
  – Traffic spread across all nodes
  – Channel - ESCON/FICON
  – LAN, WAN

• **Server Management**
  – One console to view the entire system
Leveraging Extended Memory Space

- Sophisticated algorithms to track usage, measure temperature, and rank data
  - Temperature weightings for Tactical and Strategic I/O’s
  - Logical I/O’s affect data temperature
- Compliments FSG cache
- Dynamically adjusts to new query patterns

FSG Cache
Temporarily store data required for current queries, purges least recently used

Intelligent Memory
Hottest data placed and maintained in memory, aged out as it cools

most recently used data

most frequently used data

very hot in

cool out
BYNET™ Software

- **Optimized for Teradata Active Data Warehouse Performance**
  - Linearly scalable bandwidth
  - BYNET Low Latency Interface (BLJI) - Streamlined communication protocol
  - BYNET software provides unique Teradata features: broadcast message support, row merge support, multi-fabric message traffic shaping, software guaranteed message delivery (point to point and broadcast)
  - Broadcast reliability is implemented in software in the form of multiple virtual broadcast trees.
  - Supports several physical fabrics; InfiniBand, Ethernet (1Gb & 10Gb), Native BYNET.

- **Proven High-Availability**
  - Each fabric is fault tolerant (multiple paths, redundant power & cooling)
  - 2 active and independent fabrics (no single point of failure)

The **Teradata Optimizer** chooses between Point-to-Point and Broadcast Messaging to select the most effective communication.
Infiniband
Integrated Communication for Teradata Unified Data Architecture

Corporate Network

Hadoop

Teradata

Aster

Infiniband

SAS

Node 1

Node 1

Node 1

Node 1

PE

Node A

Node B

Node C

Node D

Users, BI Tools, Visualization Tools

Teradata Parallel Transporter (TPT) / Teradata Data Mover

External Data

External Interfaces

Teradata Unity

Teradata

External Interfaces

13

© 2016 Teradata - Property of Teradata
Interact with Customer Support for proactive intervention and system metrics

System Administration

Escalate events to Enterprise Management Systems via SNMP or WMI

Filter and display relevant events locally

Distributed system management agents collect information
Collate and Recognize system level events

Secure Communication Gateway
Summary
Teradata is Teradata

Teradata Workload-Specific Platform Family
- On-Premises

Teradata Cloud
- Managed Cloud

Teradata Database on VMware®
- Private Cloud

Teradata Database on
- Microsoft Azure
- Amazon Web Services®
- Google Cloud Platform
- Public Cloud
What Makes Teradata Unique

Key Components/Concepts

- MPP Architecture
- Data Management
- Availability
- Workload Management
- Query Execution
- Extensibility
- Optimizer
Shared Nothing - Managing the Data

Key Concepts

• Basis of Teradata scalability
  – Each AMP owns an equal slice of the disk
  – Only that AMP reads that slice

• No single point of control for any operation
  – I/O, Buffers, Locking, Logging, Dictionary
  – Nothing centralized
  – Exponential communication costs avoided
Rows automatically distributed evenly by **hash partitioning**

- Even distribution results in **scalable performance**
- Done in real-time as data are loaded, appended, or changed.
- Hash map defined and maintained by the system
  - $2^{32}$ hash codes, 1,048,576 buckets distributed to AMPs

**Teradata Parallel Hash Function**

- Primary Index (PI) column(s) are hashed
- Hash is always the same - for the same values
- No reorgs, repartitioning, space management
The Magic of Row Hash

- Store rows in a table in row hash order (logically)
- Direct access via Primary Index
  - For continuous update or single lookup
  - Hash the value
  - Go direct to AMP (routed by BYNET)
  - MI (in memory) -> CI -> data block
  - Very sparse N* tree
  - MI and CI very small structures
  - Guarantees no more than 2 IOs to get to any record via PI
  - CI cached if frequent access
  - “First Index is free”

- Sort merge join with no sort
  - Joins which never require scan of the large table
  - E.g. Star Join doesn’t need to hash join or spool the fact table and sort it to join
File System

- Teradata wrote a new rule book
  - Old one written by IBM 40 years ago, used by most mainstream DBMSs today - except Teradata

- File system built of raw slices

- Rows stored in blocks
  - Variable length
  - Grow and shrink on demand
  - Rows located dynamically
    - May be moved to reclaim space, defrag
  - Maximum block size is configurable
    - System default or per table
    - 8K to 1MB
    - Change dynamically

- Blocks stored in 12MB allocation units
  - “cylinders”

- Indexes are just rows in tables
Space Allocation

• Space allocation is entirely dynamic
  - No tablespaces or journal spaces or any pre-allocation
  - Spool (temp) and tables share space pool, no fixed reserved allocations

• If no cylinder free, combine partial cylinders
  - Dynamic and automatic
  - Background compaction based on tunable threshold

• Quotas control disk space utilization
  - Increase quota (trivial online command) to allow user to use more space
Partitioned Primary Index (PPI) Comparison

No Partitions
Looking for a customer’s transactions across all time and states

Single Level Partition
Looking for two particular weeks

Multi-level Partitions
Looking for two states in two weeks
**Teradata Columnar**

**Key advantages:**
- Better query **performance** – read only the columns you need
- Smaller **data size** – better compression on homogeneous column values

**Query execution optimized for column processing**

**Full hybrid columnar**
- Row tables, column tables, mixed tables, and mixed index and table
- Physical design optimization option

---

**Row Store – Employee Table**

<table>
<thead>
<tr>
<th>Emp ID</th>
<th>Lastname</th>
<th>Firstname</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Smith</td>
<td>Joe</td>
<td>40000</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>Mary</td>
<td>50000</td>
</tr>
<tr>
<td>3</td>
<td>Johnson</td>
<td>Cathy</td>
<td>44000</td>
</tr>
</tbody>
</table>

**Column Store – Employee Table**

<table>
<thead>
<tr>
<th>Emp ID</th>
<th>Lastname</th>
<th>Firstname</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Smith</td>
<td>Joe</td>
<td>40000</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>Mary</td>
<td>50000</td>
</tr>
<tr>
<td>3</td>
<td>Johnson</td>
<td>Cathy</td>
<td>44000</td>
</tr>
</tbody>
</table>
## Compression

<table>
<thead>
<tr>
<th></th>
<th>Multi Value</th>
<th>Algorithmic</th>
<th>Data Block</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What it is</strong></td>
<td>• Dictionary-based compression that replaces values specified by the user</td>
<td>• Make your own - It applies a compression / decompression algorithm to a column of data</td>
<td>• It compresses all types of data in a data block before it’s stored on a disk</td>
</tr>
<tr>
<td><strong>Suited for</strong></td>
<td>• Data with repeated values</td>
<td>• Data with well-known attributes</td>
<td>• Data with a low frequency of access (cold data)</td>
</tr>
<tr>
<td><strong>Type of Data</strong></td>
<td>• NULLs + Common values</td>
<td>• UTF-16 to UTF-8, long strings</td>
<td>• Works on any/all data</td>
</tr>
<tr>
<td><strong>Characteristic</strong></td>
<td>• Zero CPU overhead</td>
<td>• Can write custom procedures</td>
<td>• Built-in library</td>
</tr>
<tr>
<td><strong>Compression Ratio</strong></td>
<td>• 30-40%</td>
<td>• 50% (2x)</td>
<td>• 3X or more</td>
</tr>
</tbody>
</table>
Integrated Management of Hybrid Storage

- Automatic
- Transparent
- No DBA effort
- No SQL changes
- Maintain user access to ALL data for analysis
- Avoid separate systems and copies of data for each use case

Teradata Intelligent Memory

- Internal memory
- SSD
- HDD

Teradata Virtual Storage

Teradata Compress On Cold

Cool

Hot

Warm

Cool
Summary

Automate and Eliminate

- Database should manage the data, not DBA or the user
- Data management should be as invisible as possible
  - Define and forget
- Enables user self-service

- No reorgs
  - Don’t even have a reorg utility
- No index rebuilds
- No re-partitioning
- No detailed space management
- Easy database and table definition
- Minimum ongoing maintenance
  - All performed automatically
What Makes Teradata Unique

Key Components/Concepts

- MPP Architecture
- Data Management
- Workload Management
- Extensibility
- Optimizer
- Query Execution
- Availability
Overview: Query Execution

• Teradata query execution is designed to make every analytic query scalable
• Enabling in-database execution is a primary goal
• Parallelization is built-in throughout all operations and is fully automatic
What's on a Node

• **Gateway**
  - Connect sessions to outside world
  - Balance external traffic workload across nodes

• **Parsing Engine (PE)**
  - Parse & Optimize
  - Dispatcher to AMPs

• **AMP (Access Module Processor)**
  - Execution engine
  - Logs & locks
  - Data dictionary
  - I/O management

• **"Vprocs"**
  - Virtual "processors" sharing one physical node
Query Parallelization

- Query parsing, management is fully distributed across the nodes
  - No head node/coordinator node
- All operations fully parallel
  - No single threaded operations
  - Scans, Joins, Index access, Aggregation, Sort, Insert, Update, Delete
  - Ordered Analytics
  - Extensibility functions
  - Result return

“Conditional Parallelism”

“Unconditional Parallelism”
**Data Access Methods**

- **Scan**
  - Read all rows of a table
  - Executed on all AMPs simultaneously, each AMP reads the rows hashed to that AMP
  - A table or spool will be read, rows will be qualified (Row Selection)
  - Unneeded columns will be removed (Projection)
  - Result will be written to a spool file
  - Only interaction between the AMPs is for each of them to report completion

- **Primary Index**
  - Unique (UPI) and Non-Unique (NUPI)
  - Allows efficient direct access by value or via join

- **Additional Indexes**
  - Allow optimization of access
  - Used to support workloads that are high frequency, high SLA
  - Selected for use automatically by the query optimizer
  - PI, PPI and high performance scan eliminate need for many indexes
Secondary Index: Non Unique Secondary Index (NUSI)

Non Unique Secondary Index (NUSI)

- Designed for accessing one or a small number of values where many rows have the same value
- Index entries are stored on the same AMP with the data rows
- An index entry is a value followed by a list of rowids of rows containing that value
- Ordered by hash of value being indexed
- Execution via a NUSI reads the index first, then uses the rowids to retrieve the data rows
  - Executed on All-AMPs simultaneously
Secondary Index: Unique Secondary Index (USI)

**Unique Secondary Index (USI)**

- Designed for accessing one or a small number of rows by value
  - Enforces Uniqueness
- Index entry is stored on hash AMP of value
- Index entry contains the value and the single rowid
- Execution via a USI accesses the index entry on the hash AMP(s) of the index value(s), then sends a message to the AMP of the rowid to retrieve the row
  - Executed on the minimum number of AMPs possible
Join Preparation

- Join preparation works in parallel across all AMPs
  - Selection to eliminate rows
  - Projection to eliminate columns
  - Eliminate bytes as early in the plan as possible

- Leverages BYNET to get data co-located for join
  - Redistribute: Hash by some set of attributes, often the join columns
  - Duplicate: Make a full copy of the prepared set on every AMP

- Collect results in receivers and write to spool or feed directly to first stage of sort if needed

- Uses Spool for intermediate results
  - Intermediate spools can be released after use (marked “last use” in explain)
Joins

- Joins in Teradata all work in parallel and scale linearly
  - Must have data co-located in an AMP to join

- Join operates on two relations producing a join result which can be spooled locally or piped into redistribution and or sort as required
  - Queries with more than two tables are made into a series of two relation joins

- Many types of join operations available
  - Merge Join
  - Hash Join
  - Product Join
  - Outer Join versions of each
  - Exclusion and Inclusion Join for sub-query processing

- Each AMP performs its part of the join independently
  - Communicate only when completely done
Parallel Aggregation Assisted by the Optimizer

- Aggregation makes full parallel use of all AMPs
  - Final result and final aggregation distributed across the platform
  - Linearly scalable across any data size
- Perform local aggregation resulting in (at most) one record per group per AMP
  - Skip if we know that little or no reduction will occur
- Redistribute by hash on GROUP BY columns to get all rows for a group together and distribute groups evenly across all AMPs
  - Skip if we know already distributed by GROUP BY columns
- Receive redistributed rows and perform Global Aggregation
- Caches at both local and global level do aggregation in-memory unless number of groups is very large
  - Cache spill that adapts to data pattern (e.g. most frequent keys don’t spill)
Merge of an Answer Set

- Each AMP sorts and spools its part of answer set
- Query complete reported to application
- Application performs fetch, interface (JDBC, ODBC,...) requests buffer of rows
- Each node merges its contributing AMPs answer set into a buffer for the BYNET
- BYNET merges the sorted buffers and fills a buffer to return to the interface
- All levels merge in parallel, and data is transferred using point-to-point messages, maintaining linear scalability as the size of the answer set or number of nodes increases
- The "big sort" penalty has never existed (N(log(N) algorithm, N divided by # of AMPs)
- Final answer set never has to be brought together into a single node for a large final sort - only merge a buffer at a time on demand
Thank You

Questions/Comments
Email: Todd.Walter@Teradata.com

Follow Me
Twitter @toddawalter

Rate This Session # 637
with the PARTNERS Mobile App
Remember To Share Your Virtual Passes
Teradata Database Architecture Overview Part 2

Todd Walter
Chief Technologist
Teradata

#TDPARTNERS16

Session #643 – 14:00 – 14:45 Sunday, September 11, in C101
What Makes Teradata Unique

Key Components/Concepts

- MPP Architecture
- Data Management
- Workload Management
- Query Execution
- Extensibility
- Optimizer
- Availability
Primary Goals of the Optimizer:

• In-database
• Minimum tuning efforts
  – No hints
  – Only indexes and statistics
• Use the platform effectively and efficiently
• Turn a SQL statement into a series of steps for execution
Running Queries on Teradata Database

• ANSI compliant SQL
  – Small number of Teradata specific analytic extensions

• High performance algorithms
  – Efficient use of resources not just counting on parallelism
  – Join, Aggregation, Sort, ...
  – Compiled expressions

• Complex query features
  – Derived tables, Case expressions, all forms of Sub-queries, Sample, ...
  – Big limits: 128 table joins, 128 nesting levels, ...
  – 1MB SQL/Views/Macros
Teradata Optimizer

- Cost based query planning
  - Balance costs of
    - Cardinalities
    - Sorts and joins
    - Redistribution
    - Disk I/Os
    - Networks and nodes
    - Compression
  - Lowest resource costs
  - Fastest elapsed time
- Automatic rewrites
- Automatic optimization
  - No hints
  - No degrees of parallelism
Planning

• Process of turning a query into a plan
• Optimizer deconstructs SQL into individual components
  – Relations, Join Conditions, Access Conditions
• Access Method: how to access each relation
  – Table Scan, Index Use, Bitmap Use
• Join Method: how pairs of table are joined
  – Merge Join, Product Join, Hash Join
  – Outer Joins, Inclusion and Exclusion Joins
• Join Order: sequence of table joins
• Join Geography: how rows are relocated prior to the join
  – Redistribute rows, duplicate rows, local spool
• Sort Order: order for merge join, ordered result, distinct
Join Order

- The order of joins is determined by identifying the minimum cost ordering
- Look at size (with conditions applied), distribution, available access methods, available join methods, ...
- Create each order and determine cost, pick best
- Thousands or millions of combinations
  - Game theory used to eliminate unlikely combinations
  - Look-ahead limits to control reviewed combinations
  - Heuristics for special cases, starting points
- Dependent on good Statistics!
- And the Optimizer makes the decision – does the hard work of figuring it out
Views and Physical Database Design

High performance views:
• No penalty for using views
• Enables wide use for ease of use, security, privacy
• Recommend using views rather than users accessing the base tables
• Compiled early so optimizer only sees base tables and complete condition sets

Physical database design affects planning:
• Indexes, Join Indexes, partitions – provide access methods
• PI, Referential Integrity, data types – guide join planning
• Columnar, Compression, row definition – guide size and I/O estimates
Examples of Optimizer’s Advanced Capabilities

- Aggregate Join Index (AJI)
- SATisfiability-Transitive Closure (SAT/TC)
- Query Re-Writes
  - Reorganize a query’s structure for better optimization/performance
- Partial Redistribution Partial Duplication (PRPD)
  - Skew optimization
- Incremental Planning and Execution (IPE)
  - Plan in stages based on partial results
- In-Memory and Chipset Optimizations
- Temporal sequencing/normalization
- Geospatial indexing/bounding
Three types of histograms are generated:

- **Equal-height interval histograms:**
  - Each interval has the same number of values

- **High-biased interval histograms:**
  - Each interval has no more than two values

- **Compression histograms:**
  - Up to 200 equal-height intervals, plus one or more high-biased intervals

**Statistics Collection**

- High bias values separated to recognize skew
  - Optimizer will generate different plans for different values or ranges

- Statistics calculated across all rows on all AMPs with a fully scalable algorithm

- Statistics sampled when not available, dynamic choice of single AMP or all AMPs

- Statistics extrapolated when stale relative to data change and table size relative to when statistics were collected
Auto-Stats Collection

Automates and provides intelligence to DBA tasks related to Optimizer Statistics Collections where such tasks include:

- Identifying and collecting missing statistics that are needed for query optimization
- Detecting stale statistics and promptly refreshing them
- Identifying and removing unused or unimportant statistics from routine maintenance
- Prioritizing the list of pending collections such that important and stale statistics are given precedence
- Executing needed collections in the background during scheduled time periods

- Dynamically issuing collections in response to key events, most notably the completion of bulk load operations
- Configuring the system to ensure that resource usage incurred by statistics collections is properly regulated and throttled through TASM or other mechanisms
Explain

- English description of plan
- Common code path through parser
- Time estimates relative not absolute
- Assumes run standalone
- Explain should be used by all SQL coders
  - Understand how the optimizer and query execution work
  - Avoid/diagnose performance issues
  - Keep for future comparison

explain select c_name, sum(o_totalprice) from customer, orders where customer.c_custkey = orders.o_custkey and o_clerk = 'Clerk#000008402' group by 1 order by 1;

4) We do an all-AMPs RETRIEVE step from TPCH10G.orders by way of index # 4 "TPCH10G.orders.O_CLERK = 'Clerk#000008402'" with a residual condition of ("TPCH10G.orders.O_CLERK = 'Clerk#000008402'") into Spool 4 (all_amps), which is redistributed by the hash code of (TPCH10G.orders.O_CUSTKEY) to all AMPs. Then we do a SORT to order Spool 4 by row hash. The size of Spool 4 is estimated with high confidence to be 1,503 rows (37,575 bytes). The estimated time for this step is 0.14 seconds.

5) We do an all-AMPs JOIN step from TPCH10G.customer by way of a RowHash match scan with no residual conditions, which is joined to Spool 4 (Last Use) by way of a RowHash match scan. TPCH10G.customer and Spool 4 are joined using a merge join, with a join condition of ("TPCH10G.customer.C_CUSTKEY = O_CUSTKEY"). The result goes into Spool 3 (all_amps), which is built locally on the AMPs. The size of Spool 3 is estimated with low confidence to be 1,503 rows (49,599 bytes). The estimated time for this step is 0.07 seconds.

6) We do an all-AMPs SUM step to aggregate from Spool 3 (Last Use) by way of an all-rows scan, grouping by field1 (TPCH10G.customer.C_NAME). Aggregate Intermediate Results are computed globally, then placed in Spool 5. The size of Spool 5 is estimated with low confidence to be 1,503 rows (61,623 bytes). The estimated time for this step is 0.04 seconds.

7) We do an all-AMPs RETRIEVE step from Spool 5 (Last Use) by way of an all-rows scan into Spool 1 (group_amps), which is built locally on the AMPs. Then we do a SORT to order Spool 1 by the sort key in spool field1 (TPCH10G.customer.C_NAME). The size of Spool 1 is estimated with low confidence to be 1,503 rows (121,743 bytes). The estimated time for this step is 0.02 seconds.
What Makes Teradata Unique

Key Components/Concepts

- MPP Architecture
- Data Management
- Availability
- Query Execution
- Workload Management
- Extensibility
- Optimizer
Extensibility

Allowing new in-database Analytics capability:

- **User Defined Extensions**
  - User defined functions
  - User defined data types
  - User defined SQL Operators

- **Processing Extensions**
  - Stored procedures
  - Table functions
  - Queue tables

- **Programming Language**
  - SQL, C/C++, Java

- **JSON, XML, etc.**
- **In-DB Analytics**
  - Geospatial
  - Temporal
  - Data Lab

- **Teradata QueryGrid™**

- **Scripting Languages**
  - Python, Ruby, R
What Makes Teradata Unique

Key Components/Concepts
Teradata Integrated Workload Management (TIWM)

- Define a Workload for each type of work
- Map the workload to a priority category
- Add filters and throttles, as needed
- Classification criteria maps queries to workload management rules

Session and query characteristics link to workload management rules:

- System Filter
  - Classification Criteria
    - System Throttle
      - Classification Criteria
        - Workload Throttle
          - Workloads Control Priority
            - Workload Single-AMP
            - Workload Loading
            - Workload Adhoc-Short
            - Workload Data Mining
            - Workload Adhoc-Long

A session logs on, a query is submitted:

- Session and query characteristics link to workload management rules
- System Filter
  - Classification Criteria
  - System Throttle
    - Classification Criteria
    - Workload Throttle
      - Delay Queue
  - Reject
  - Delay Queue
Viewpoint Monitoring System

DVR for Your DBA
Centralized Management
Exception Based Portal Alerts
What Makes Teradata Unique

Key Components/Concepts

- MPP Architecture
- Data Management
- Query Execution
- Extensibility
- Optimizer
- Workload Management
- Availability
Teradata Availability

- **Teradata Dual Systems Unity**
  - Fallback
  - Proactive system monitoring
  - Business Critical Support Services
  - On-Site spares, Test System
  - Backup-Archive-Restore, Hot Standby Nodes
  - Teradata Recovery Centers
  - Disaster Recovery, Business Impact Analysis

- RAIDs, Clustering, Cliques, OS disk mirroring,
  - Redundant Hardware Components
  - On-line Upgrades, Documented Operational Procedures
  - Change Control Procedures, Proactively apply fixes
  - Restartability - Application and ETL

- **Best in Class**
- **Optional**
- **Built-in, Automatic**
Thank You

Questions/Comments
Email: Todd.Walter@Teradata.com

Follow Me
Twitter @toddawalter

Rate This Session # 643
with the PARTNERS Mobile App
Remember To Share Your Virtual Passes