**Confusion 1: Does Redshift Follow an MPP Architecture?**

* **User's Confusion**: The user was unsure whether Amazon Redshift follows a Massively Parallel Processing (MPP) architecture.
* **Solution**: Yes, Redshift follows an MPP architecture. It distributes data and query workloads across multiple compute nodes for parallel processing, enhancing scalability and performance. The leader node coordinates query execution, while compute nodes process data in parallel across their slices.
  + **Validation**: Redshift Developer Guide (p. 8): "Amazon Redshift is a fully managed, petabyte-scale data warehouse service... based on a massively parallel processing (MPP) architecture."

**Confusion 2: Where is Data Stored in Compute Nodes vs. Redshift Managed Storage (RMS)?**

* **User's Confusion**: The user initially thought compute nodes might store persistent data and was unclear whether the majority of data resides in RMS or compute node SSDs. They also questioned when AWS S3 is used for storage.
* **Solution**:
  + Persistent data is stored in **Redshift Managed Storage (RMS)**, not compute nodes. RMS uses SSDs for caching frequently accessed ("hot") data and offloads less-used ("cold") data to S3 automatically based on usage patterns.
  + Compute nodes have local SSDs, but these are used only for **temporary storage** during query execution (e.g., sorting, joins, intermediate results), not persistent data.
  + **S3 Usage Scenarios**:
    - Cold data offloaded by RMS for cost-effective storage.
    - Data backups and snapshots stored in S3.
    - External data queried via Redshift Spectrum without loading into Redshift.
    - Data loading (e.g., via COPY) or unloading from Redshift.
  + **Validation**: Redshift Developer Guide (p. 23): "Redshift Managed Storage (RMS) uses a combination of high-speed SSD storage for frequently accessed data and Amazon S3 for scalable storage of less frequently accessed data."

**Confusion 3: How Do Compute Nodes Access Data from S3?**

* **User's Confusion**: The user wondered if compute nodes fetch data directly from S3 when querying external data, or if the leader node intermediates this process.
* **Solution**:
  + For internal data in RMS, compute nodes fetch data directly from RMS (SSD or S3, managed transparently by RMS) based on the leader node’s query plan, not via the leader node as a bottleneck.
  + For external data in S3 (via Redshift Spectrum), compute nodes directly access S3 to scan and process the data, bypassing RMS. The leader node creates the execution plan, but compute nodes interact with S3 independently.
  + **Validation**: Redshift Developer Guide (p. 253): "Amazon Redshift Spectrum enables you to query data directly from files on Amazon S3... Compute nodes query the data in parallel."

**Confusion 4: Does Compute Node SSD Storage Persist Data During Loading?**

* **User's Confusion**: The user questioned whether compute node SSDs store data persistently during data loading or if it’s only temporary, and how data moves to RMS.
* **Solution**:
  + Compute node SSDs are used **temporarily** during data loading and query execution (e.g., for buffering, sorting, or joins). Persistent data is stored in RMS.
  + During loading (e.g., COPY), compute nodes process and sort data in-memory or on SSDs, then write it to RMS. RMS then decides whether to keep it in SSD cache or offload to S3 based on usage.
  + **Validation**: Redshift Developer Guide (p. 23): "Compute nodes use local SSDs for temporary storage during query execution, but persistent data is managed by RMS."

**Confusion 5: How Does RMS Decide Between Hot (SSD) and Cold (S3) Storage?**

* **User's Confusion**: The user was unsure how RMS determines which data stays in SSD (hot) vs. S3 (cold), and whether this involves block vs. object storage concepts.
* **Solution**:
  + RMS automatically manages data placement based on **access patterns**. Frequently accessed data ("hot") stays in SSD for fast retrieval, while infrequently accessed ("cold") data is offloaded to S3 for cost efficiency.
  + The leader node does not decide this; RMS handles it transparently. SSDs use block storage for low-latency access, while S3 uses object storage for scalability.
  + **Validation**: Redshift Developer Guide (p. 23): "RMS automatically moves data between SSDs and Amazon S3 based on usage patterns to optimize performance and cost."

**Confusion 6: What Are Node Slices and How Do They Work?**

* **User's Confusion**: The user was unclear about the concept of node slices and initially thought they might be virtual machines with persistent storage.
* **Solution**:
  + Slices are **virtualized processing units** within a compute node, each allocated a portion of CPU, RAM, and SSD. They enable parallel processing but do not store persistent data (RMS does that).
  + The number of slices per node depends on the instance type (e.g., dc2.large has 2 slices). Data is assigned to slices during loading based on distribution style, not at query time.
  + **Validation**: Redshift Developer Guide (p. 52): "Each compute node is divided into slices... Each slice is allocated a portion of the node’s memory and disk space."

**Confusion 7: When Does Distribution Style Apply?**

* **User's Confusion**: The user initially thought distribution style applies only during query execution, not data loading, and questioned its impact on RMS.
* **Solution**:
  + Distribution style (EVEN, KEY, ALL) is applied **during data loading**, determining how data is distributed across compute nodes and slices. The leader node uses this style to assign data, which is then stored in RMS.
  + During query execution, the pre-applied distribution style ensures compute nodes fetch only their assigned data from RMS, optimizing parallelism.
  + RMS stores data according to this distribution but doesn’t dynamically reassign it at query time.
  + **Validation**: Redshift Developer Guide (p. 63): "When you load data into a table, Amazon Redshift distributes the rows... according to the distribution style."

**Confusion 8: Does RMS Handle Data Slicing Dynamically?**

* **User's Confusion**: The user was unsure whether slicing happens dynamically at query time or during loading, and how RMS interacts with slices.
* **Solution**:
  + Slicing occurs **during data loading**, not dynamically at query time. The distribution style pre-assigns data to slices, and RMS stores it accordingly.
  + At query time, compute nodes fetch pre-sliced data from RMS based on the leader node’s plan, with no additional slicing needed.
  + **Validation**: Redshift Developer Guide (p. 52): "Data is distributed to slices when it is loaded into the table based on the distribution style."

**Confusion 9: Where Does Sorting Happen for Sort Keys?**

* **User's Confusion**: The user wondered whether sorting for sort keys occurs in RMS or compute nodes, and when it happens (loading vs. querying).
* **Solution**:
  + Sorting happens **during data loading** by compute nodes. They sort the data in-memory or on SSDs based on the sort key, then write it to RMS in sorted order.
  + RMS stores data as received (sorted) and doesn’t sort it. During queries, the pre-sorted order allows Redshift to skip irrelevant blocks, speeding up execution.
  + **Validation**: Redshift Developer Guide (p. 66): "When data is loaded into the table, Amazon Redshift sorts the data by the columns designated as sort keys."

**Confusion 10: How Does Data Movement Occur in Joins?**

* **User's Confusion**: The user wanted to understand data movement between compute nodes during joins and its impact on performance.
* **Solution**:
  + **Co-Located Joins**: No movement if tables share the same distribution key; data is already on the same nodes (fastest).
  + **Broadcast Joins**: Small table is replicated to all nodes; minimal movement, good for small tables.
  + **Redistribution Joins**: Data is shuffled between nodes if distribution keys differ, slowing performance due to high data movement.
  + Optimizing distribution keys minimizes movement.
  + **Validation**: Redshift Developer Guide (p. 70): "If tables aren’t co-located, Redshift might need to redistribute or broadcast data across nodes."

**Confusion 11: What is Redshift Spectrum and Why Use It Over Athena?**

* **User's Confusion**: The user was unclear whether Redshift Spectrum is a separate service, its integration with Redshift, and why it’s needed when Athena exists.
* **Solution**:
  + Redshift Spectrum is a **feature** of Redshift (not a separate service), available in both provisioned and serverless clusters. It allows querying external data in S3 without loading it into RMS.
  + **Why Use It**: Combines internal Redshift data with external S3 data, leverages Redshift’s performance optimizations, and avoids loading large datasets into RMS.
  + **Athena Comparison**:
    - Athena is serverless, charges per data scanned, and queries only S3 data.
    - Redshift Spectrum uses Redshift’s compute nodes, charges for data scanned plus compute, and integrates internal and external data.
  + **Validation**: Redshift Developer Guide (p. 253): "Amazon Redshift Spectrum enables you to query data directly in Amazon S3... leveraging your existing Redshift cluster."

**Confusion 12: Are External Tables and Redshift Spectrum the Same?**

* **User's Confusion**: The user questioned whether creating external tables in Redshift is the same as using Redshift Spectrum, and how metadata (DDL) is managed.
* **Solution**:
  + External tables are defined in Redshift to reference S3 data, and Redshift Spectrum is the engine that queries them. They’re related but distinct: external tables are the structure, Spectrum is the querying mechanism.
  + Metadata (DDL) can be:
    - Defined directly in Redshift via CREATE EXTERNAL TABLE.
    - Referenced from Glue Catalog via CREATE EXTERNAL SCHEMA, reusing schemas from Glue or Athena.
  + No separate DDL is needed for Spectrum; it uses the DDL from Redshift or Glue.
  + **Validation**: Redshift Developer Guide (p. 256): "You create external tables using the CREATE EXTERNAL TABLE command... Redshift Spectrum queries this data."

**Confusion 13: Can Athena or Redshift Spectrum Query Non-S3 Data?**

* **User's Confusion**: The user asked if Athena or Redshift Spectrum can query data from non-S3 sources (e.g., Oracle, on-prem) cataloged in Glue.
* **Solution**:
  + **Athena**: Limited to querying data in S3. It cannot query non-S3 sources directly, even if cataloged in Glue.
  + **Redshift Spectrum**: Also limited to S3 data. External tables in Redshift must point to S3; non-S3 data requires ETL (e.g., via Glue) to move it to S3 first.
  + Glue can catalog non-S3 data, but Athena and Spectrum need it in S3 to query.
  + **Validation**: Redshift Developer Guide (p. 253): "Redshift Spectrum queries data stored in Amazon S3."

**Confusion 14: How Are Costs Applied for Redshift Spectrum?**

* **User's Confusion**: The user sought clarity on how costs are applied when using Redshift Spectrum with external tables.
* **Solution**:
  + **Redshift Spectrum Costs**: Charged per terabyte of data scanned from S3 (e.g., $5/TB in most regions), plus the cost of the Redshift cluster (provisioned) or compute usage (serverless).
  + **S3 Storage Costs**: Separate, based on data stored in S3.
  + **Glue Costs**: Additional if using Glue for cataloging or ETL.
  + In serverless, Spectrum costs are bundled into compute usage, not separately charged.
  + **Validation**: Redshift Developer Guide (p. 258): "You are charged for the amount of data scanned by Redshift Spectrum queries."

**Confusion 15: Does Redshift Spectrum Create the Query Plan?**

* **User's Confusion**: The user wondered if Redshift Spectrum creates the query plan or if it’s just for scanning, and how distribution styles/sort keys apply to external data.
* **Solution**:
  + The **leader node** creates the query plan for all queries, including those involving Spectrum. Spectrum executes the scanning of S3 data as directed by the plan.
  + Compute nodes use Spectrum to fetch external data into slices for processing.
  + **Distribution Styles/Sort Keys**: Apply only to internal RMS data, not external S3 data queried via Spectrum.
  + **Validation**: Redshift Developer Guide (p. 253): "The leader node creates the query plan... Redshift Spectrum processes external data in parallel."
* **Confusion 15:. Data Storage & Access Patterns**
* **How do compute nodes access data from S3?**
* Confusion: Unsure whether compute nodes access S3 directly or through the leader node.
* Solution: Compute nodes directly access S3 for external queries via Redshift Spectrum. For internal queries, they fetch data from RMS, which may use SSDs or offload cold data to S3.
* **Does compute node SSD storage persist data during loading?**
* Confusion: Unsure if SSDs store data persistently during data loading.
* Solution: SSDs are only used temporarily for buffering/sorting during loading. Persistent data is written to RMS.
* **How does RMS decide between hot (SSD) and cold (S3) storage?**
* Confusion: Unclear how Redshift determines whether data stays in SSD or gets offloaded to S3.
* Solution: RMS automatically manages this based on access frequency, storing frequently accessed data in SSDs and offloading infrequently used data to S3.
* **2. Data Distribution & Query Execution**
* **What are node slices and how do they work?**
* Confusion: Thought slices might be VMs with persistent storage.
* Solution: Slices are virtualized processing units within a node, assigned a portion of CPU, RAM, and SSD. Persistent data is managed by RMS, not slices.
* **When does distribution style apply?**
* Confusion: Thought distribution style is applied at query execution rather than during loading.
* Solution: Distribution style is set at data load time and determines how data is stored across slices/nodes. Query execution simply follows this distribution.
* **Does RMS handle data slicing dynamically?**
* Confusion: Unsure if slicing occurs dynamically at query time.
* Solution: Slicing is done during data loading. At query time, compute nodes fetch pre-assigned data slices from RMS without additional slicing.
* **3. Sorting & Joins**
* **Where does sorting happen for sort keys?**
* Confusion: Unsure if sorting happens in RMS or compute nodes and whether it occurs at load time or query time.
* Solution: Sorting happens during data loading within compute nodes. RMS stores data in sorted order for efficient query retrieval.
* **How does data movement occur in joins?**
* Confusion: Unclear on when and how data moves between nodes during joins.
* Solution: Data movement depends on join type:
* Co-located joins (same distribution key) → No movement.
* Broadcast joins (small table copied to all nodes) → Minimal movement.
* Redistribution joins (different distribution keys) → High movement, performance hit.
* **4. Redshift Spectrum & External Data**
* **What is Redshift Spectrum and why use it over Athena?**
* Confusion: Thought Spectrum might be a separate service and questioned its need when Athena exists.
* Solution: Spectrum is a feature of Redshift, allowing external S3 queries while integrating with internal Redshift data. Athena is a separate service for querying S3 data without Redshift.
* **Are external tables and Redshift Spectrum the same?**
* Confusion: Unsure if creating external tables in Redshift is equivalent to using Redshift Spectrum.
* Solution: External tables define S3 data structure, while Spectrum is the engine that queries them. Metadata can be managed in Glue or defined directly in Redshift.
* **Can Athena or Redshift Spectrum query non-S3 data?**
* Confusion: Questioned whether they can query sources like Oracle or on-prem data.
* Solution: By default, both query S3. Athena can connect to other sources via federated queries; Spectrum is limited to S3.