Parallel Execution of SQL Statements

Introduction to Parallel Execution

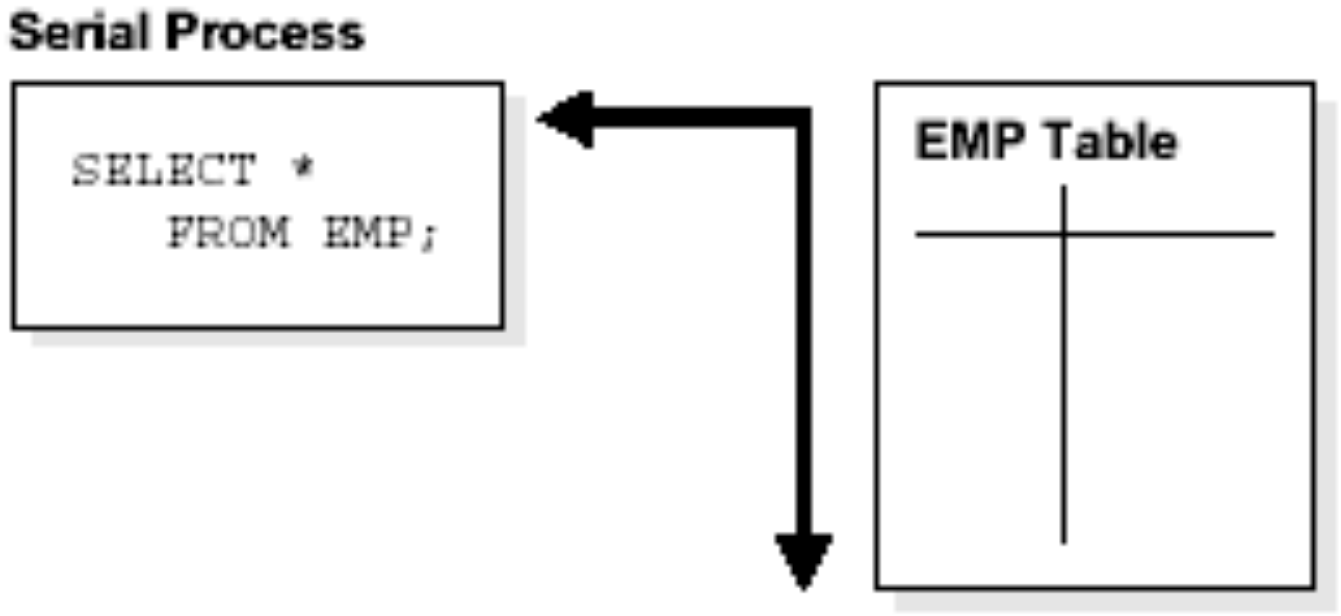
- When Oracle executes SQL statements in parallel, multiple processes work together simultaneously to execute a single SQL statement.
- By dividing the work necessary to execute a statement among multiple processes, Oracle can execute the statement more quickly than if only a single process executed it.
- This is called **parallel execution** or **parallel processing**.
- **Parallelism** is the idea of breaking down a task so that, instead of one process doing all of the work in a query, many processes do part of the work at the same time.
- An example of this is when 12 processes handle 12 different months in a year instead of one process handling all 12 months by itself.
- Parallel execution helps systems scale in performance by making optimal use of hardware resources. If your system's CPUs and disk controllers are already heavily loaded, you need to alleviate the system's load or increase these hardware resources before using parallel execution to improve performance.

When to Implement Parallel Execution

- During off-hours, however, parallel execution can effectively process high-volume batch operations.
- The most common example of using parallel execution is for DSS. Complex queries, such as those involving joins or searches of very large tables, are often best executed in parallel.
- Parallel execution is useful for many types of operations that access significant amounts of data. Parallel execution improves performance for:
  - Queries
  - Creation of large indexes
  - Bulk inserts, updates, and deletes
  - Aggregations and copying

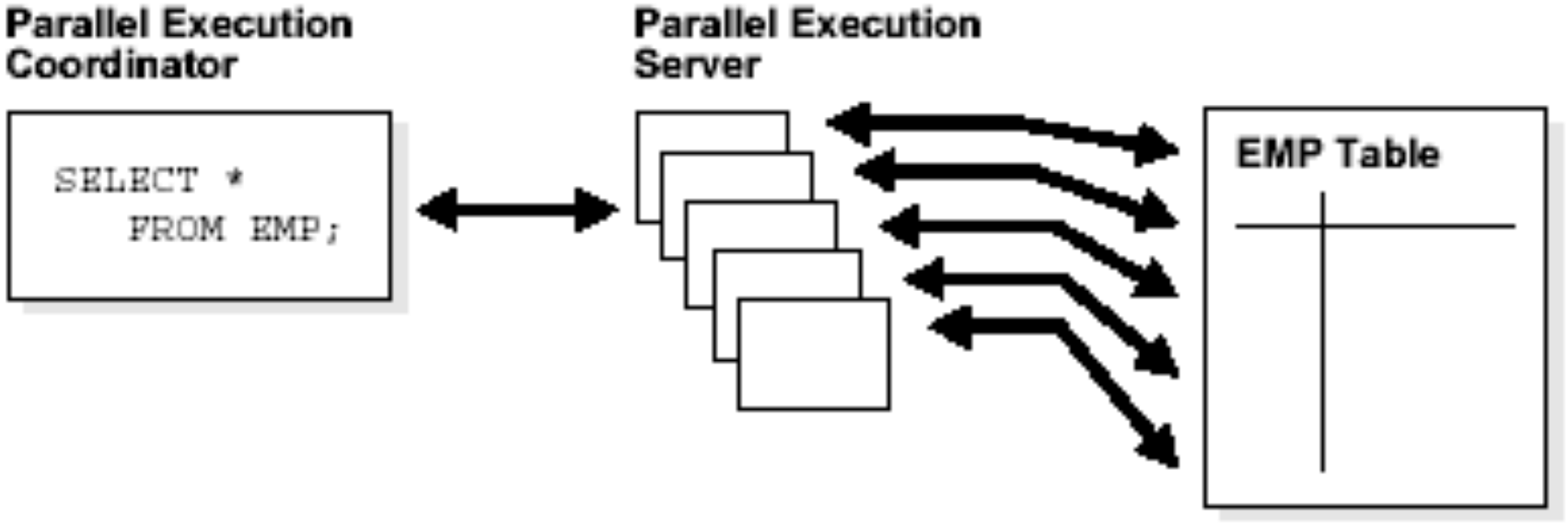
How Parallel Execution Works

- When *serial execution* is being used, a single server process performs all necessary processing for the sequential execution of a SQL statement.
- For example, to perform a full table scan (such as SELECT * FROM emp), one process performs the entire operation, as illustrated below:



How Parallel Execution Works

- Parallel execution performs these operations in parallel using multiple **parallel processes**.
- One process, known as the **parallel execution coordinator**, dispatches the execution of a statement to several **parallel execution servers** and coordinates the results from all of the server processes to send the results back to the user.



How Parallel Execution Works

- The figure illustrates several parallel execution servers performing a scan of the table emp.
- The table is divided dynamically (**dynamic partitioning**) into load units called granules and each granule is read by a single parallel execution server.
- The granules are generated by the coordinator. Each granule is a range of physical blocks of the table emp.
- The mapping of granules to execution servers is not static, but is determined at execution time.
- When an execution server finishes reading the rows of the table emp corresponding to a granule, it gets another granule from the coordinator if there are any granules remaining.
- This continues till all granules are exhausted, in other words. the entire table emp has been read.
- The parallel execution servers send results back to the parallel execution coordinator, which assembles the pieces into the desired full table scan.

How Parallel Execution Works

- Given a query plan for a SQL query, the parallel execution coordinator breaks down each operator in a SQL query into parallel pieces, executes them in the right order as specified in the query, and then integrates the partial results produced by the parallel execution servers executing the operators.
- The number of parallel execution servers assigned to a single operation is the degree of parallelism (DOP) for an operation.
- Multiple operations within the same SQL statement all have the same degree of parallelism.

Parallelized SQL Statements

- Each SQL statement undergoes an optimization and parallelization process when it is parsed.
- Therefore, when the data changes, if a more optimal execution plan or parallelization plan becomes available, Oracle can automatically adapt to the new situation.
- After the optimizer determines the execution plan of a statement, the parallel execution coordinator determines the parallelization method for each operation in the execution plan.
- The coordinator must decide whether an operation can be performed in parallel and, if so, how many parallel execution servers to enlist.
- The number of parallel execution servers used for an operation is the degree of parallelism.

Parallelism Between Operations

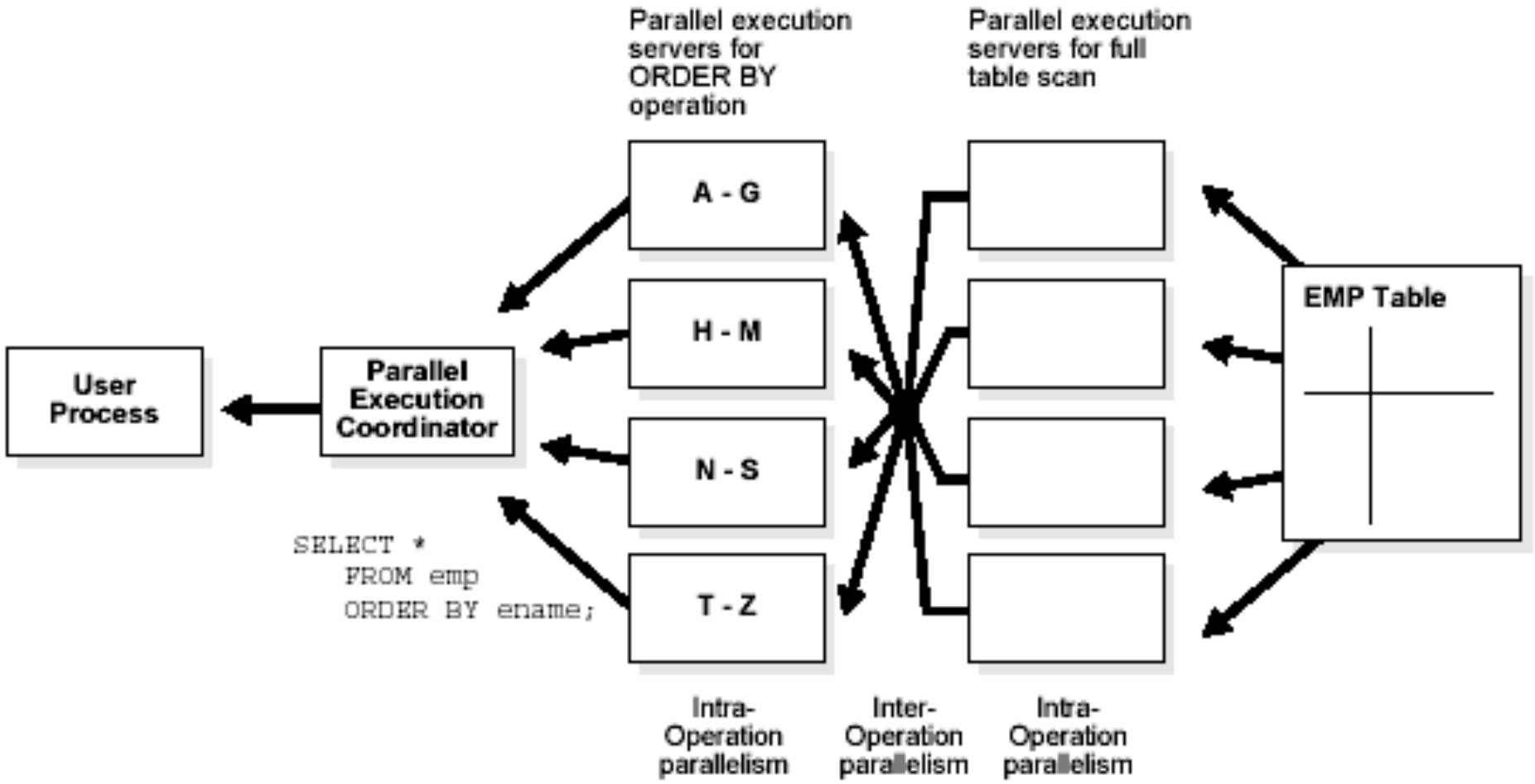
- To illustrate intra-operation parallelism and inter-operation parallelism, consider the following statement:

```
SELECT * FROM emp ORDER BY ename;
```

- The execution plan implements a full scan of the emp table followed by a sorting of the retrieved rows based on the value of the ename column.
- Assume the ename column is not indexed. Also assume that the degree of parallelism for the query is set to four, which means that four parallel execution servers can be active for any given operation.
- Each of the two operations (scan and sort) performed concurrently is given its own set of parallel execution servers.
- Parallelization of an individual operation where the same operation is performed on smaller sets of rows by parallel execution servers achieves what is termed **intra-operation parallelism**.
- When two operations run concurrently on different sets of parallel execution servers with data flowing from one operation into the other, we achieve what is termed **inter-operation parallelism**.

Parallelism Between Operations

Figure 20–3 Inter-Operation Parallelism and Dynamic Partitioning



Parallelism Between Operations

- As you can see from the previous figure, there are actually eight parallel execution servers involved in the query even though the degree of parallelism is four.
- This is because a parent and child operator can be performed at the same time (inter-operation parallelism).
- Also note that all of the parallel execution servers involved in the scan operation send rows to the appropriate parallel execution server performing the sort operation.
- If a row scanned by a parallel execution server contains a value for the ename column between A and G, that row gets sent to the first ORDER BY parallel execution server.
- When the scan operation is complete, the sorting processes can return the sorted results to the coordinator, which in turn returns the complete query results to the user.

SQL Operations That Can Be Parallelized

- The following are commonly parallelized to improve performance:
  - **Parallel Query** - You can parallelize queries and subqueries in SELECT statements, as well as the query portions of DDL statements and DML statements (INSERT, UPDATE, and DELETE). However, you cannot parallelize the query portion of a DDL or DML statement if it references a remote object.
  - **Parallel DDL**-You can parallelize the CREATE TABLE AS SELECT, CREATE INDEX, and ALTER INDEX REBUILD statements.
  - **Parallel DML**-Parallel DML (parallel insert, update, and delete) uses parallel execution mechanisms to speed up or scale up large DML operations against large database tables and indexes. You can also use INSERT ... SELECT statements to insert rows into multiple tables as part of a single DML statement. You can normally use parallel DML where you use regular DML.