Performance tuning AWS RDS for PostgreSQL involves several key considerations to ensure optimal performance, reliability, and cost-effectiveness. Here are the important points to consider:

1. Instance Type and Size
   - Choose the Right Instance Type: Select an instance type that matches your workload requirements. Consider factors such as CPU, memory, network performance, and I/O capacity.
   - Scale Appropriately: Monitor the performance and scale up or down based on your application's needs. Use Amazon CloudWatch metrics to identify when to resize your instance.

2. Storage Optimization
   - Use Provisioned IOPS (SSD): For high-performance applications, consider using Provisioned IOPS (SSD) storage to achieve consistent, low-latency performance.
   - Optimize Storage Size: Ensure that your storage size matches your needs. Over-provisioning can lead to unnecessary costs, while under-provisioning can lead to performance issues.

3. Parameter Group Tuning

Customize Parameter Groups: Adjust PostgreSQL configuration settings using parameter groups. Key parameters to tune include:
   - work_mem: Memory used for internal sort operations and hash tables.
   - maintenance_work_mem: Memory used for maintenance tasks like VACUUM.
   - shared_buffers: Amount of memory the database uses for shared memory buffers.
   - effective_cache_size: Estimate of how much memory is available for disk caching by the operating system and within the database itself.
   - max_connections: Maximum number of connections to the database.

4. Indexing
   - Proper Indexing: Ensure that your tables have appropriate indexes to speed up query performance. Regularly monitor and analyze slow queries to identify indexing opportunities.
   - Use the Right Index Types: Use B-tree indexes for equality and range queries, and consider other types such as GiST, GIN, and BRIN for specific use cases.

5. Query Optimization
   - Analyze and Optimize Queries: Use the EXPLAIN command to analyze query execution plans and optimize queries accordingly.
   - Avoid Long-Running Transactions: Long-running transactions can hold locks and lead to contention, impacting performance.

6. Monitoring and Alerts
   - Enable Monitoring: Utilize Amazon CloudWatch and RDS Enhanced Monitoring to track key performance metrics like CPU utilization, memory usage, I/O operations, and query performance.
   - Set Alerts: Configure CloudWatch alarms to notify you of performance issues or resource constraints.

7. Connection Management
   - Use Connection Pooling: Implement connection pooling to manage database connections efficiently and reduce overhead. Tools like PgBouncer or Amazon RDS Proxy can help.
   - Manage Connection Limits: Ensure that your application respects the maximum connection limits configured for your RDS instance.

8. Backup and Maintenance
   - Automate Backups: Ensure that automated backups are enabled and configured according to your RTO (Recovery Time Objective) and RPO (Recovery Point Objective).
   - Regular Maintenance: Schedule maintenance tasks like VACUUM, ANALYZE, and REINDEX to keep the database performance optimal.

9. Replication and High Availability
   - Read Replicas: Use read replicas to offload read traffic from the primary instance and improve read scalability.
   - Multi-AZ Deployments: Enable Multi-AZ deployments for high availability and automated failover support.

10. Network Optimization
   - Place RDS in the Right VPC: Ensure your RDS instance is in the same VPC and availability zone as your application to minimize network latency.
   - Use Enhanced Networking: Enable Enhanced Networking for instances that support it to achieve higher throughput and lower latency.

11. Application Tuning
   - Optimize Application Logic: Ensure that the application is efficiently interacting with the database, minimizing unnecessary queries and optimizing transaction handling.

12. Regular Performance Reviews
   - Conduct Performance Audits: Regularly review and audit the performance of your RDS instance to identify and address any potential bottlenecks.