**1. Selection**

* Compute Resources: Select the optimal compute solution for your workload, considering instances, containers, and functions. Choose based on performance needs, cost efficiency, and usage patterns.
* Storage Options: Use the appropriate storage type (object, block, file) based on your workload requirements. Amazon S3 for object storage, EBS for block storage, and Amazon EFS for file storage are examples.
* Database Solutions: Select the right database technology (relational, NoSQL, in-memory, graph) depending on your application needs and access patterns. For example, Amazon RDS for relational databases or Amazon DynamoDB for NoSQL.
* Network Configuration: Optimize network settings and choose the right network architecture, such as using AWS Direct Connect for dedicated network connections or CloudFront for content delivery.
* Right-Sizing: Continuously evaluate and adjust resource sizes to meet demand without overprovisioning.
* Leverage Managed Services: Use AWS-managed services like RDS, DynamoDB, and Lambda to reduce the operational overhead and optimize performance.

**2. Review**

* Continuous Improvement: Regularly review and optimize your architecture by taking advantage of the latest AWS services and features.
* Benchmarking: Perform regular benchmarking of your workload to identify performance bottlenecks and opportunities for optimization.
* Adopt New Technologies: Stay updated with AWS announcements and consider integrating new services that enhance performance or reduce costs.
* Cost vs. Performance: Regularly review trade-offs between cost and performance to ensure your architecture remains efficient and cost-effective.
* Experimentation: Conduct experiments with different configurations and services to find the best fit for your workload.
* Auto Scaling Reviews: Regularly review and adjust auto-scaling configurations to ensure they meet changing demand patterns.

**3. Monitoring**

* Metrics Collection: Continuously collect and monitor performance metrics, such as CPU utilization, latency, and throughput, using services like Amazon CloudWatch.
* Alerting: Set up alerts for key performance indicators (KPIs) to detect performance issues early and take corrective action.
* Analyzing Logs: Use log analysis tools like AWS CloudTrail and CloudWatch Logs to identify and troubleshoot performance-related issues.
* Service Health Monitoring: Monitor the health of AWS services that your workload depends on, ensuring that external dependencies do not impact your performance.
* User Experience Monitoring: Monitor end-user experience to ensure that performance remains acceptable from the user's perspective, using services like AWS X-Ray for tracing.

**4. Tradeoffs**

* Caching: Implement caching mechanisms, such as Amazon CloudFront or ElastiCache, to reduce latency and improve data retrieval times.
* Compression: Use data compression to reduce data transfer times and improve performance.
* Asynchronous Processing: Where possible, offload tasks to background processing to improve the responsiveness of your primary workloads.
* Eventual Consistency: In some cases, accepting eventual consistency instead of strong consistency can improve performance and scalability.
* Concurrency Limits: Adjust concurrency limits based on your workload’s needs to balance between performance and resource utilization.
* Resource Allocation: Make trade-offs in resource allocation by prioritizing performance in critical areas and optimizing costs elsewhere.