Several bottle necks exist for each of the services actually.

The most blatantly obvious one is ECS. Unless you run on ECS Fargate you will find out you can only compute a certain amount of data The solution is to define target tracking Auto Scaling which allows you to automatically adjust the number of running tasks or services in an ECS service based on defined criteria in CloudWatch alarms such as 50% CPU utilization. Or , With Step Scaling policies with which you can specify scaling adjustments based on the size of the alarm breach. This allows you to scale in or out in larger increments when demand changes suddenly.

The second is lambda concurrency. Say you handle an event ,this causes a lambda container to spin-up and execute the code, you can only run 5000 instances of lambda at any given moment. This is called a concurrency limit. The solution is to request AWS to increase it. The second solution is to decuple the lambda with SQS allowing asynchronous handling of the events. The second approach is also beneficial to ECS.

The third an final one is RDS. When you create an RDS it is recommended to use provisioned IO, also there is only one DB in the cluster that handles writes . The limitation exists when you go over the limit of IO or CPU util of the master in the cluster. The solution is to increase the provisioned io or increase instance size. alternatively utilize aurora serverless which is mysql compatible.also worth considering is Dynamo serverless which can hanle millions of requests per second and is indeed a popular choice in large scale data applications ,notable limitation of dynamo are:

1. Read Limitations:

a. Strongly Consistent Reads:

* + Limited to 4 KB per read operation.
  + Read Capacity Units (RCUs) determine the total read capacity, and each RCU provides up to 4 KB of strongly consistent read per second.

b. Eventual Consistent Reads:

* + Also limited to 4 KB per read operation.
  + Read Capacity Units (RCUs) determine the total read capacity, with each RCU providing up to 4 KB of eventually consistent read per second.

c. Burst Capacity:

* + DynamoDB provides burst capacity for reads, allowing you to exceed your provisioned RCUs for short durations.

1. Write Limitations:

a. Item Size:

* + Limited to 400 KB per write operation.

b. Write Capacity Units (WCUs):

* + Each WCU provides up to 1 KB of write capacity per second.
  + Write operations are billed in terms of WCUs.

c. Burst Capacity:

* + Similar to reads, DynamoDB provides burst capacity for writes, allowing you to exceed your provisioned WCUs for short durations.