To create a meaningful **Grafana single-view dashboard** for monitoring S3 buckets using these AWS SDK HTTP client concurrency metrics, we can focus on the following aspects:

1. **Concurrency Utilization**: Monitor available, leased, and pending concurrency to understand resource usage.
2. **Concurrency Acquisition Performance**: Measure acquisition times and detect bottlenecks.
3. **Error Trends**: Highlight unusual behavior through failed or excessive concurrency acquisition attempts.

Here's a breakdown of panels, the suitable metrics to use, and how to combine them using PromQL.

**1. Concurrency Utilization Panel**

**Goal**: Track how concurrency resources are being used.

* **Metrics**:
  + awsSdk\_HttpClient\_AvailableConcurrency\_counter\_total
  + awsSdk\_HttpClient\_LeasedConcurrency\_counter\_total
  + awsSdk\_HttpClient\_MaxConcurrency\_counter\_total
* **PromQL Queries**:
  + **Available Concurrency**:

promql

Copy code

rate(awsSdk\_HttpClient\_AvailableConcurrency\_counter\_total[5m])

* + **Leased Concurrency**:

promql

Copy code

rate(awsSdk\_HttpClient\_LeasedConcurrency\_counter\_total[5m])

* + **Max Concurrency**:

promql

Copy code

rate(awsSdk\_HttpClient\_MaxConcurrency\_counter\_total[5m])

* **Visualization**:
  + Use a **stacked bar chart** or a **time series graph**.
  + Show all three metrics together to compare the availability, utilization (leased), and maximum concurrency limits.

**2. Pending Concurrency Acquires**

**Goal**: Identify how often requests are blocked, waiting for concurrency slots.

* **Metric**:
  + awsSdk\_HttpClient\_PendingConcurrencyAcquires\_counter\_total
* **PromQL Query**:

promql

Copy code

rate(awsSdk\_HttpClient\_PendingConcurrencyAcquires\_counter\_total[5m])

* **Visualization**:
  + Use a **line graph** to display trends in pending requests over time.
  + Overlay alert thresholds (e.g., when pending requests exceed an acceptable limit for sustained periods).

**3. Concurrency Acquire Duration Analysis**

**Goal**: Monitor how long it takes to acquire concurrency slots and detect latency issues.

* **Metrics**:
  + awsSdk\_HttpClient\_ConcurrencyAcquireDuration\_seconds\_bucket
  + awsSdk\_HttpClient\_ConcurrencyAcquireDuration\_seconds\_sum
  + awsSdk\_HttpClient\_ConcurrencyAcquireDuration\_seconds\_count
* **PromQL Queries**:
  + **Average Acquisition Duration**:

promql

Copy code

rate(awsSdk\_HttpClient\_ConcurrencyAcquireDuration\_seconds\_sum[5m])

/

rate(awsSdk\_HttpClient\_ConcurrencyAcquireDuration\_seconds\_count[5m])

* + **Request Distribution (Histogram)**:

promql

Copy code

rate(awsSdk\_HttpClient\_ConcurrencyAcquireDuration\_seconds\_bucket[5m])

* **Visualization**:
  + Use a **time series graph** for average acquisition duration.
  + Use a **heatmap** or **histogram panel** to visualize the distribution of durations by bucket.

**4. Error/Anomaly Detection**

**Goal**: Highlight excessive durations or unusual behavior in concurrency acquisition.

* **Metric**:
  + awsSdk\_HttpClient\_ConcurrencyAcquireDuration\_seconds\_max
* **PromQL Query**:

promql

Copy code

awsSdk\_HttpClient\_ConcurrencyAcquireDuration\_seconds\_max

* **Visualization**:
  + Use a **single-stat panel** to display the maximum concurrency acquisition duration.
  + Set color-coded thresholds to indicate excessive durations.

**5. Overall Resource Efficiency**

**Goal**: Calculate and display the overall efficiency of concurrency usage.

* **Metrics**:
  + Combine awsSdk\_HttpClient\_LeasedConcurrency\_counter\_total and awsSdk\_HttpClient\_MaxConcurrency\_counter\_total.
* **PromQL Query**:

promql

Copy code

100 \* (rate(awsSdk\_HttpClient\_LeasedConcurrency\_counter\_total[5m])

/

rate(awsSdk\_HttpClient\_MaxConcurrency\_counter\_total[5m]))

* **Visualization**:
  + Use a **gauge panel** to show concurrency utilization percentage.
  + Set thresholds for high efficiency (>80%) or inefficiency (<50%).

**Dashboard Layout**

**Top Row (Overview)**

* **Stat Panels**:
  + Total available concurrency.
  + Total leased concurrency.
  + Maximum concurrency.

**Middle Row (Trends and Latency)**

* **Line Charts**:
  + Leased vs available concurrency trends.
  + Pending concurrency acquires rate.
* **Heatmap**:
  + Concurrency acquisition duration distribution.

**Bottom Row (Alerts and Anomalies)**

* **Single Stat Panels**:
  + Maximum concurrency acquisition duration.
* **Gauge**:
  + Concurrency utilization efficiency.

**Alerts to Configure**

1. **Pending Requests Alert**:
   * Trigger when the rate of awsSdk\_HttpClient\_PendingConcurrencyAcquires\_counter\_total exceeds a threshold (e.g., 100/s).
2. **High Acquisition Duration**:
   * Alert if awsSdk\_HttpClient\_ConcurrencyAcquireDuration\_seconds\_max exceeds a critical value.
3. **Resource Saturation**:
   * Alert if LeasedConcurrency approaches a high percentage of MaxConcurrency.

**Outcome**

This dashboard provides:

1. **Concurrency health monitoring**: Tracks availability, usage, and limits.
2. **Performance analysis**: Highlights latency in acquiring slots.
3. **Anomaly detection**: Identifies bottlenecks and saturation in S3 usage.

4o

Top of Form

Bottom of Form