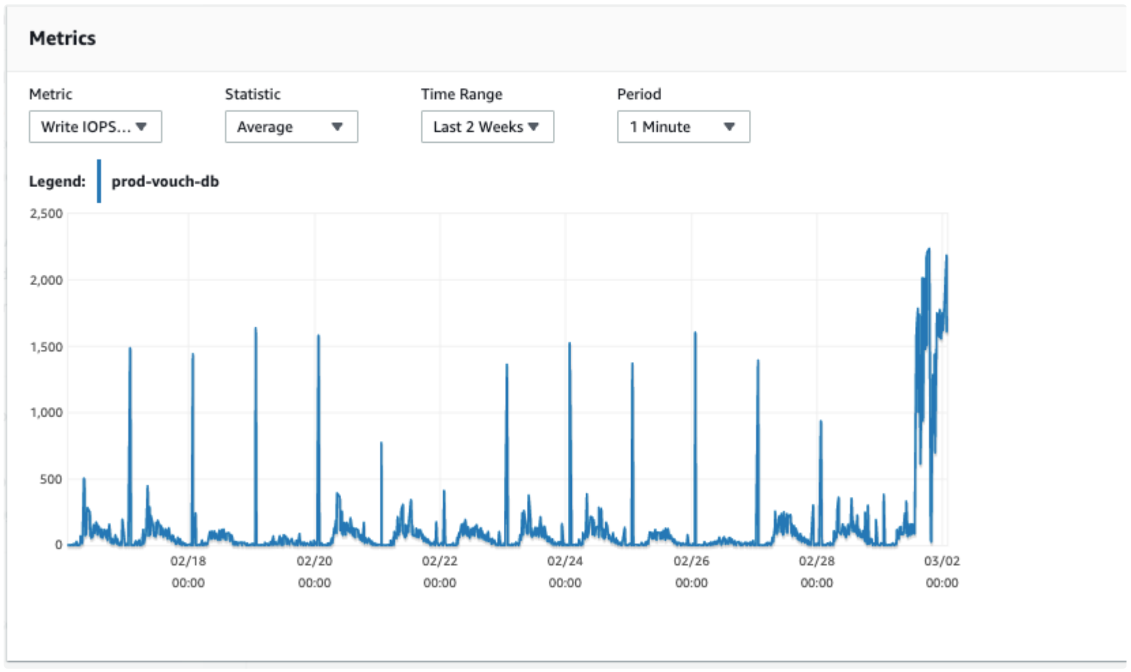
RCA: Slowness March 1

Overview

On 1st March 2023, our database experienced a significant increase in latency (majorly write IOPS), which caused delays in our application's response times. This impacted our users' experience and led to frustration and complaints.



Time to first response

The concern was raised by on-ground team at 18:30 and it was first acknowledged by @Yogesh Bhat at 18:40. First response time was 10 mins.

Number of team participants

5

Incident Description

How was the incident detected?

The concern was raised by the on-ground team on the slack channel. Actions were taken to identify:

- We checked our application logs to confirm that there was indeed an increase in Write IOPS during the time period in question.
- We reviewed the database logs to identify the queries running at the time of the Write IOPS spike.
- We used database monitoring tools to analyze the query performance and identify any bottlenecks.
- We examined the code and deployments to identify any potential issues.

What were the symptoms / impact?

The symptoms of the incident included increased write IOPS, degraded database performance, and intermittent unresponsive customer-facing services. The impact was a partial outage affecting the API calls.

What discovery or investigation was done?

After conducting a thorough investigation, we determined that the root cause of the issue was due to an infinite loop while disbursing management fees, which led to infinite write queries in the database. The code in question had a bug that caused it to enter into an infinite loop, resulting in an endless number of write queries being executed against the database. This led to the increased Write IOPS and impacted the overall performance of the database.

Timeline (IST)

6:30 pm - The incident was reported by @Int.

6:40 pm - Acknowledged the incident by @Yogesh Bhat .

7:23 pm - Restarted the services by @Yogesh Bhat .

1:30 am - Observed the DB spike.

8:39 am - Slowness again was reported by @aom.

8:43 am - Acknowledged by @Rishabh Gupta

9:00 am - Restarted the services by @Rishabh Gupta

11:43 am - Slowness again got reported by @rumbeer.

11:51 am - Restarted the services by @Rishabh Gupta.

12:37 pm - The PR causing the infinite loop was identified and rolled back.

5-whys

1. Why was there a spike in write IOPS?
2. Why were there so many write queries?
3. Why did the code enter into an infinite loop?
4. Why was the bug not caught during testing?
5. Why database connection timeout or API request timeout was not getting hit?

Root Cause

The root cause of the issue was an infinite loop in the code for disbursing management fees, resulting in an endless number of write queries being executed against the database.

How was the issue resolved and the impact mitigated?

We resolved the issue by identifying the code bug and temporarily reverting the code change.

Were there existing backlog items for this issue? Was this a known failure mode?

No

Did we already know about this potential failure mode? Did we have scheduled work? Were there backlog items that would have prevented this issue?

No

Retrospective

What went well?

- Identified a few configurations at db and API level which might cause issues later. API keeps performing the request until it is finished (no timeout), the database doesn’t seem to have a connection pool time limit, to kill connections after a threshold. Also, ELB timeout of 30 mins will be re thought.

What went wrong?

- The small code change was taken for granted.
- It took around 19-20 hours to identify and revert the PR.
- The review and testing process were not done judiciously.

Where did we get lucky?

- The write queries being sent were update queries, it didn’t corrupt the database.

Recommendations

What are the actionable tasks and follow-ups?

1. **Database Monitoring**: It's essential to monitor database performance, especially during peak usage times, to identify bottlenecks and proactively address them. This can include monitoring Write IOPS, read and write latencies, and other database metrics.
2. **Code Review**: A code review process has been introduced we have also mandated `CODEOWNERS` - Leads approval for merging PR.
3. **Code Testing**: A code testing process needs to be in place to prevent buggy code from being pushed into production. Automated testing of code can also help identify issues before they cause problems for users.
4. **Alerting and Response:** It's critical to have alerting in place that can notify the team when performance issues are detected. This can help ensure a quick response time to resolve any issues before they escalate and impact users.
5. **Review ELB Timeout:** In the process of debugging, it was realized that the ELB timeout is 30 minutes. It is recommended to re-evaluate this timeout and potentially reduce it to improve response times and prevent similar incidents from occurring.
6. **Implement DB and API timeout:** In the process of debugging, we realized there is no connection pool timeout or request timeout set at DB or API level.