the Availability Digest

www.availabilitydigest.com

Poor Documentation Snags Google

April 2010

What should have been a ten-minute outage at a major Google data center hosting the Google App Engine turned into a two-and-a-half hour ordeal simply because of faulty failover documentation. Of course, the fact that the failover procedures were incorrectly documented also implies that they were never tested and that the staff was never trained.

Kudos to Google for its transparency during and after the outage. It published a detailed post-mortem study explaining minute-by-minute exactly what happened, what the underlying causes of the outage were, and what it plans to do to avoid this situation in the future.

The incident is an excellent example of a failure chain. Many failures occur because of a sequence of events. If any one event does not happen, the failure chain is broken; and the failure does not occur. In this case, the failure chain included a power failure, a backup power fault, recent failover enhancements, faulty failover documentation of the new procedures, and the unavailability of the knowledgeable technical people who could have untangled the documentation. If any one of these events had not happened, this major outage would have been a minor inconvenience.

The Google App Engine

The Google App Engine is a compute cloud service for users to develop and host their web applications in Google's data centers. The App Engine virtualizes applications across not only multiple servers but also across multiple data centers to ensure that no fault will take down the applications. Should a server fail due to equipment problems or even due to an entire data center outage, the applications on that server will be rapidly migrated to a surviving server, where they will continue to run. Or so Google thought.

The Data Center Power Failure

The Power Goes Out

On Wednesday, February 24, 2010, a Google data center hosting the Google App Engine suffered a major power outage.[1] The power was down for about thirty minutes, but Google's backup power kicked in and continued to power the data center.

The only problem was that for some reason (never disclosed), about 25% of the servers did not receive backup power and subsequently went down. This caused their applications to fail over to the surviving servers in the data center, which were not configured to handle that much additional load; and the server failures cascaded.

[1] When the Power Goes Out at Google, Data Center Knowledge; March 8, 2010.

Failover

This unusual failure pattern was never envisioned by the Google technical staff. If a server failed, its applications were migrated to surviving servers. It was never anticipated that a large group of servers would fail and would overload the remaining servers, thus in essence taking down the entire data center.

Should a server fail, failover to a surviving server in the same data center is automatic and transparent. However, if a data center fails, failover to a backup data center is a complex process requiring careful analysis and manual intervention. The decision-making process in this situation was made all the more complex by the failure mode that had never been thought through. The result was that there was no established procedure to determine that the data center had failed and when it might recover. The fact that the failure of a server group could take down an entire data center was a revelation to the technical staff.

This oversight was compounded by the fact that recent work had just been completed to enhance the data center failover procedures. Multihoming was introduced so that access to the Google App Engine could be provided by a virtual IP address over multiple communication links serving many servers. If a server or a network link failed, traffic could be easily rerouted by the network to a surviving server in the same or different data center. This change was intended to significantly simplify failover procedures.

Unfortunately, parts of the documentation of the new failover procedures incorrectly referred to the old data center configuration rather than the upgraded configuration. Clearly, the newly documented failover procedures had not been tested; nor had the staff been trained. Otherwise, these errors would have been found.

The Failover Fault

Thus, there was a great deal of confusion about how to handle the multiple server failures. The first decision was to fail over the data center. The documented procedures were followed but led to an unsuccessful failover.

It then looked like the primary data center had returned to operation because of its reduced load, so processing was returned to it. This turned out to be an erroneous observation since the primary data center could still not handle the full load, and it once again failed. Finally, knowledgeable technical personnel were reached; and the backup data center was brought successfully online. Two-and-a-half hours had passed since the initial failure. If things had gone properly, it would have taken only about ten minutes to fail over to the backup data center.

Post-Mortem

As it does with any outage, Google staff thoroughly analyzed the sequence of events that led to this extended outage. To Google's credit, it published the results of its post-mortem analysis for all to see. The post-mortem showed the following timeline of the outage and the corrective actions that Google plans to take to avoid this situation in the future.

The Timeline

Google's published timeline[2] for the events of the morning of February 24th showed the following:

[2] Post-mortem for February 24th, 2010 outage, *Google Post-Mortem*, March 4, 2010.

7:48 AM – Internal monitoring graphs first began to show elevated errors in the primary data center.

7:53 AM – The on-call staff was notified that the primary data center had suffered a power outage and that about 25% of the servers had not received backup power.

8:01 AM – The primary on-call engineer determined that the Google App Engine was down. He paged product managers and engineering leads, requesting them to handle communication with the users about the outage.

8:22 AM – Though power had been restored to the data center, it was determined that many servers were down and that the surviving servers were not able to handle the traffic. Major clusters had lost enough machines that they were not able to carry the load. The on-call team agreed to invoke the unexpected failover procedure for an unplanned data center outage.

8:40 AM – Two conflicting sets of procedures were discovered. The team attempted to contact the specific engineers responsible for procedure changes so that the situation could be resolved.

8:44 AM – The primary on-call engineer attempted to move all traffic in a read-only state to the backup data center. Unexpected configuration problems from this procedure prevented the read-only backup from working properly.

9:08 AM – New data seemed to indicate that the primary data center had recovered. With no clear policy, the team was not aware that based on historical data, the primary data center was unlikely to have recovered to a usable state. An attempt was made to move traffic back to the primary data center while the read-only problems in the backup data center were debugged.

9:18 AM – The primary on-call engineer determined that the primary data center had not recovered. Traffic was failed back to the backup data center, and the unplanned failover procedure was reinitiated.

9:35 AM – An engineer familiar with the unplanned failover procedure was finally reached and began providing guidance about the procedure. Traffic was moved in read-only mode to the backup data center.

9:48 AM – Read-only mode became operational. Applications that handled read-only mode worked properly but in a reduced operational mode.

9:53 AM – Relevant engineers were now online. The correct procedure document was confirmed. The actual failover procedure for reads and writes began.

10:08 AM – The unplanned failover procedure completed with no problems. The App Engine was restored to service.

10:19 AM – a post to the App Engine downtime-notify group let users know that the App Engine was operating properly.

Modified Procedures

The post-mortem analysis concluded with the following plan to avoid such a situation in the future:

1. Additional drills will be scheduled for all on-call staff to review production procedures, including “rare and complicated procedures.”

2. A monthly audit of operation documents will be implemented. All documents out-of-date will be marked “deprecated.”

3. A clear policy about taking intrusive actions during a failure will be established to allow the operations staff to act confidently and without delay in emergency situations.

4. Two different App Engine configurations will be offered:
    a. The current low-latency option that has lower availability during unexpected failures
    b. A new, higher-availability option using synchronous replication that will have higher latency during normal operation.

Kudos for Good Communication

The rapid and transparent communication from Google about its problems did not go unnoticed among its users. There was a raft of favorable responses on many blogs:

> “The amount of time and thought put into this post-mortem is staggering, and the takeaways are useful to every organization.”
>
> “… a template for what the entire industry should do so that collectively we can learn from each other’s mistakes instead of having to learn the painful lessons individually.”[3]
>
> “Knowing what happened gives a large number of customers peace of mind, even if it is sometimes painful for internal customers/employees to admit to fault.”

Lessons Learned

The primary lesson to be learned from this experience is to try to identify all failure modes. Then for each such mode, plan, test, document, and train.

Another lesson is to ensure that all procedures are properly documented and the appropriate personnel trained in the new procedures. Documentation and training should both be an important part of change management.

A second lesson is one on which we have frequently commented. Be transparent, and communicate effectively with your users during the resolution of a problem. There is a great deal of forgiveness out there if you simply keep users informed as to what is happening.

[3] More to the point, if you have a story to share that will help others, even if it must be published anonymously, let us know so that we can publish it as a *Never Again* article for the benefit of all.