Accept Partial Failures, Minimize Service Loss

Daxin Wang
Diversified products
Promote
Experienced team
Too complicated to recover rapidly
Basic model of reduce service lost in incident

\[ service\_lost = \int_{0}^{t} Lost(t)dt \]
Root cause recovery VS partial recovery

\[ service\_lost = \int_{0}^{t} Lost(t)dt \]
Basic principles

• Deployment isolation
  – Limit failures in one cell, shift user queries rapidly

• Module isolation
  – Make the non-essential modules detachable

• User traffic isolation
  – Drop some of the queries to save the important ones
Deployment isolation
Deployment isolation
Deploy isolation – Global Single Point

Global single point service

single point module

ZooKeeper

Third Party Service

MySQL
Deploy isolation – Service Across Cells

Deployed across multi-cell

multi-cell service

---
Deploy isolation – Capacity Redundancy

Not enough capacity left in other cells

Realtime capacity measure
Periodic stress test
Deploy isolation – Reduce Change Risks

• Not only deploy, but also operation
• Do not change all cells at the same time, especially in automation!!
• Check system status after every stage of change, manually if necessary
• Pay attention to different operation entry, set global “locks”
Module isolation

- No service will never crash
- Detail loss is much better than totally outage
- Make every non-essential module detachable, even automatically
Module isolation -- External Dependencies

- CDN
- DNS
- HttpDNS
User traffic isolation

- When no sufficient capacity, sacrifice part of requests to save the more important part

- Which part?
  - Real user > Crawler
  - Paid user > Free user
  - Popular request > Long tail request
User traffic isolation – Distinguish Real-time

- Prepare for dropping at any time
- Crawlers may disguise the requests as human
- Machine learning attempt
Conclusion
THANKS

Daxin Wang
王达心

Enjoy the fighting against incidents