# Reliability

## Resilient to hardware or software faults

- Hardware faults
  - disk, memory, network, power grid
  - solutions: **redundancy** of hardware components, software fault-tolerance techniques (tolerate  the loss of entire machines, rolling-upgrade)
- Software errors
  - A systematic error: software bug, resource leak, problematic service, cascading failures
  - alleviations: right assumptions, testing, process isolation, allowing processes to crash and restart, monitoring and alerting

# Reliability

## Fault-tolerant for human errors

- Design systems in a way that minimizes opportunities for error
  - e.g., add constraints to interface to encourage the right things
- Decouple the places where people make the most mistakes from those where they can cause failures
  - e.g., try the change first in fully featured non-production sandbox environment
- Test thoroughly at all levels, from unit tests, integration tests, and manual tests
  - e.g., automation is quite important
- Allow quick and easy recovery from human errors
  - e.g., possible to roll back changes, roll out changes gradually, each to recompute data
- Set up detailed and clear monitoring (telemetry)
  - e.g., performance metrics, error rates can provide hints for troubleshooting and fixing
- Implement good management practices and training