Today

Internet scale Computing (Lesson 9)

* giant scale services

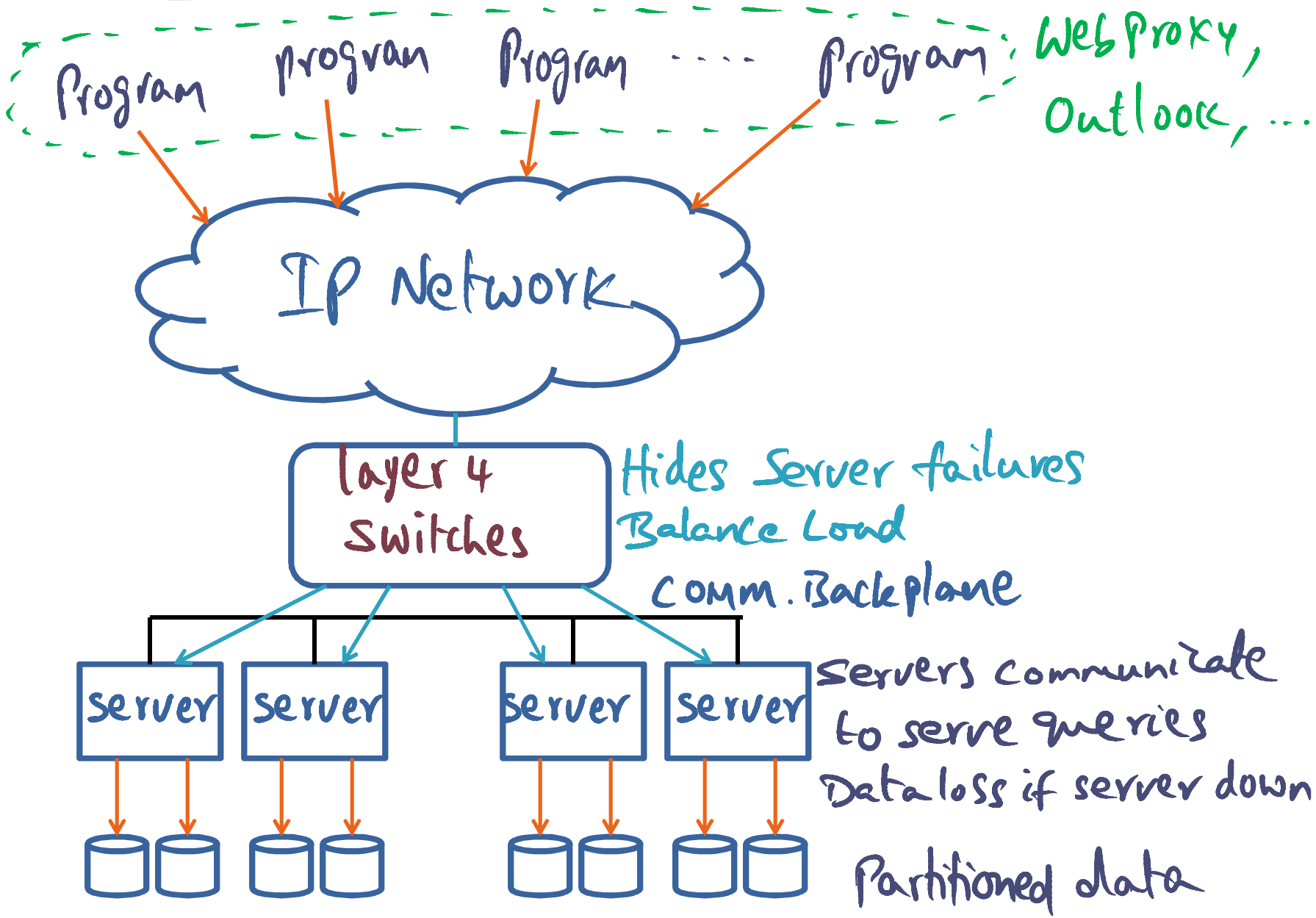
⇒ * Map-reduce programming model

Friday

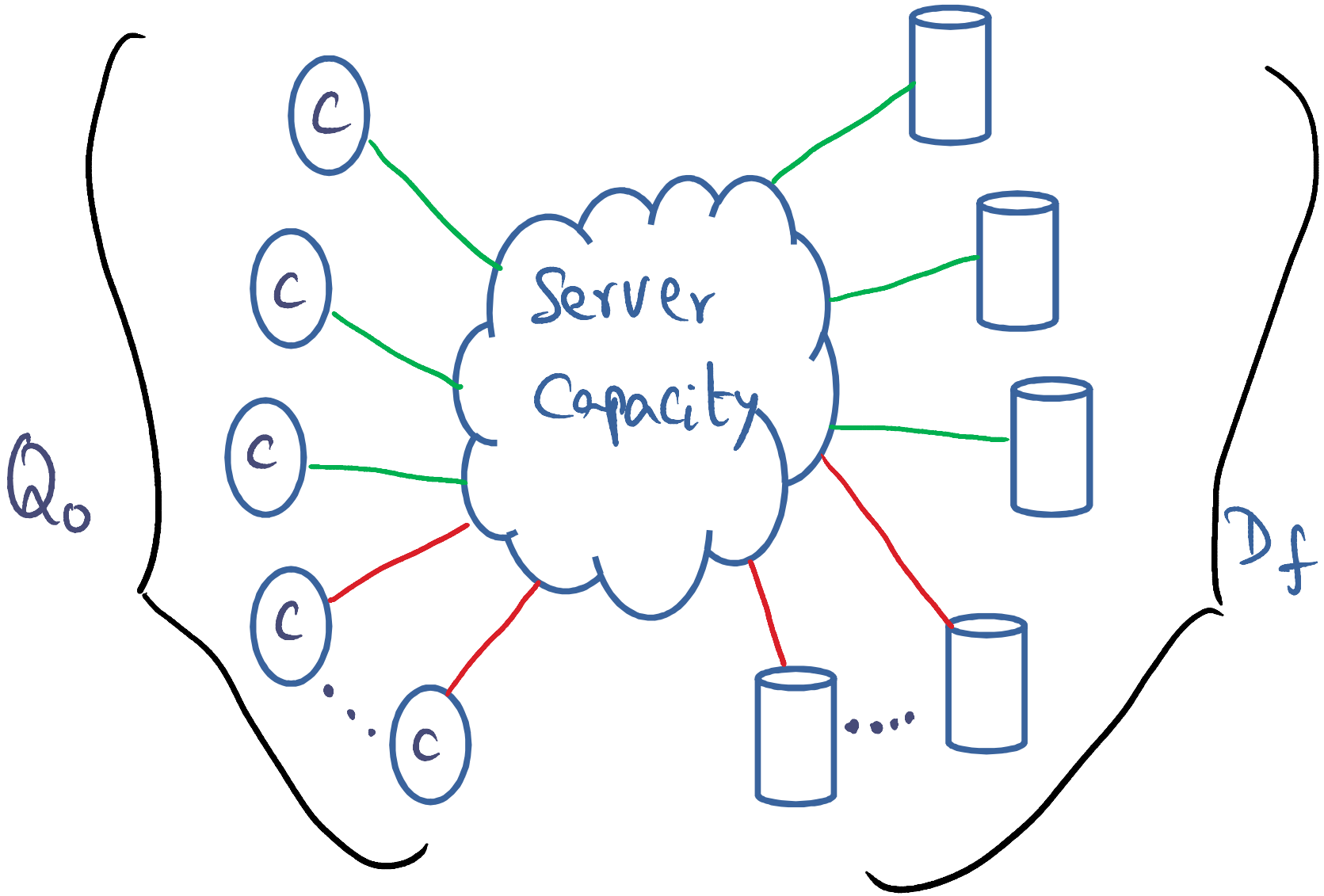
* CDN (Coral)

* Please watch videos before class *

Load management at Transport level or higher

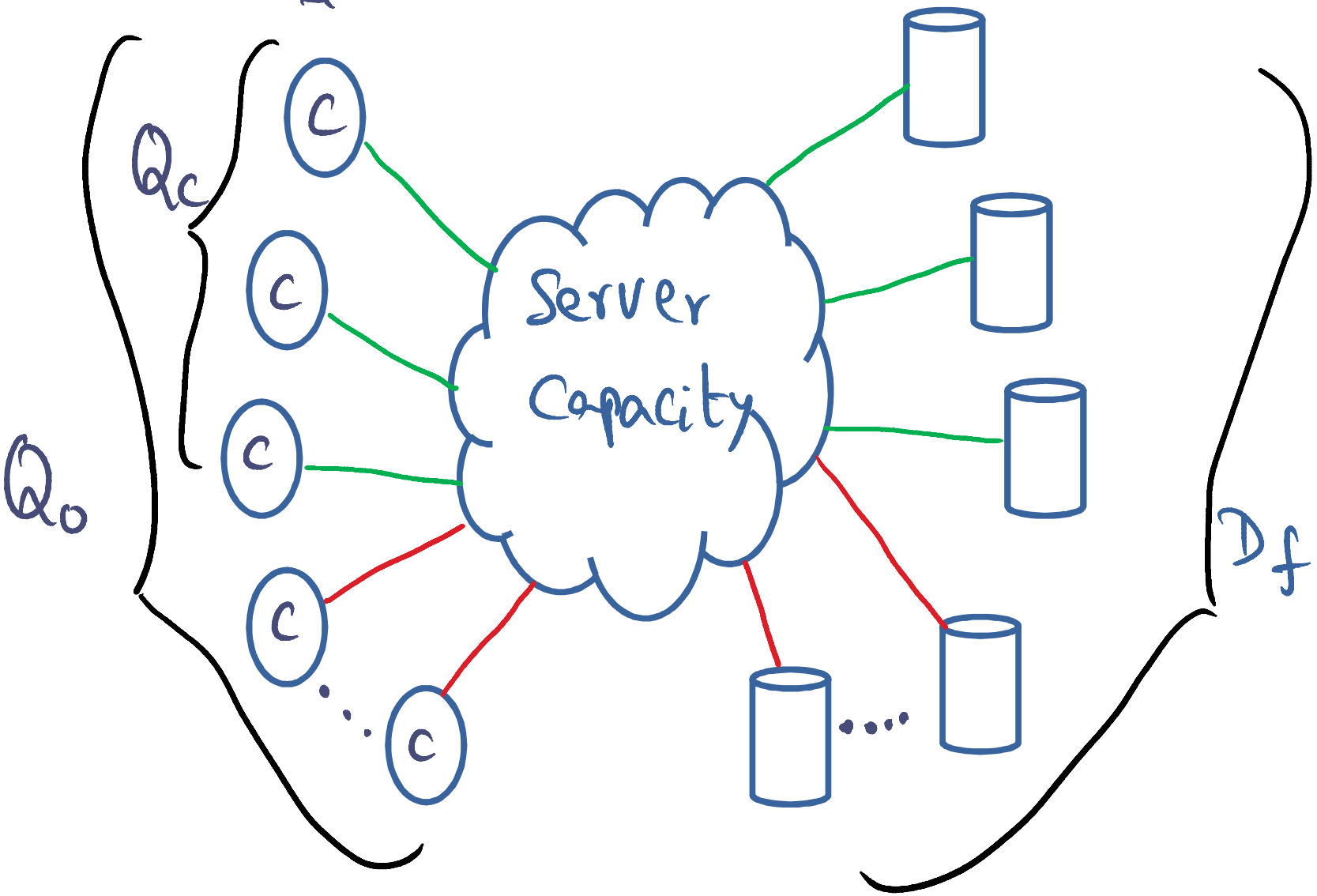


DQ Principle



DQ Principle

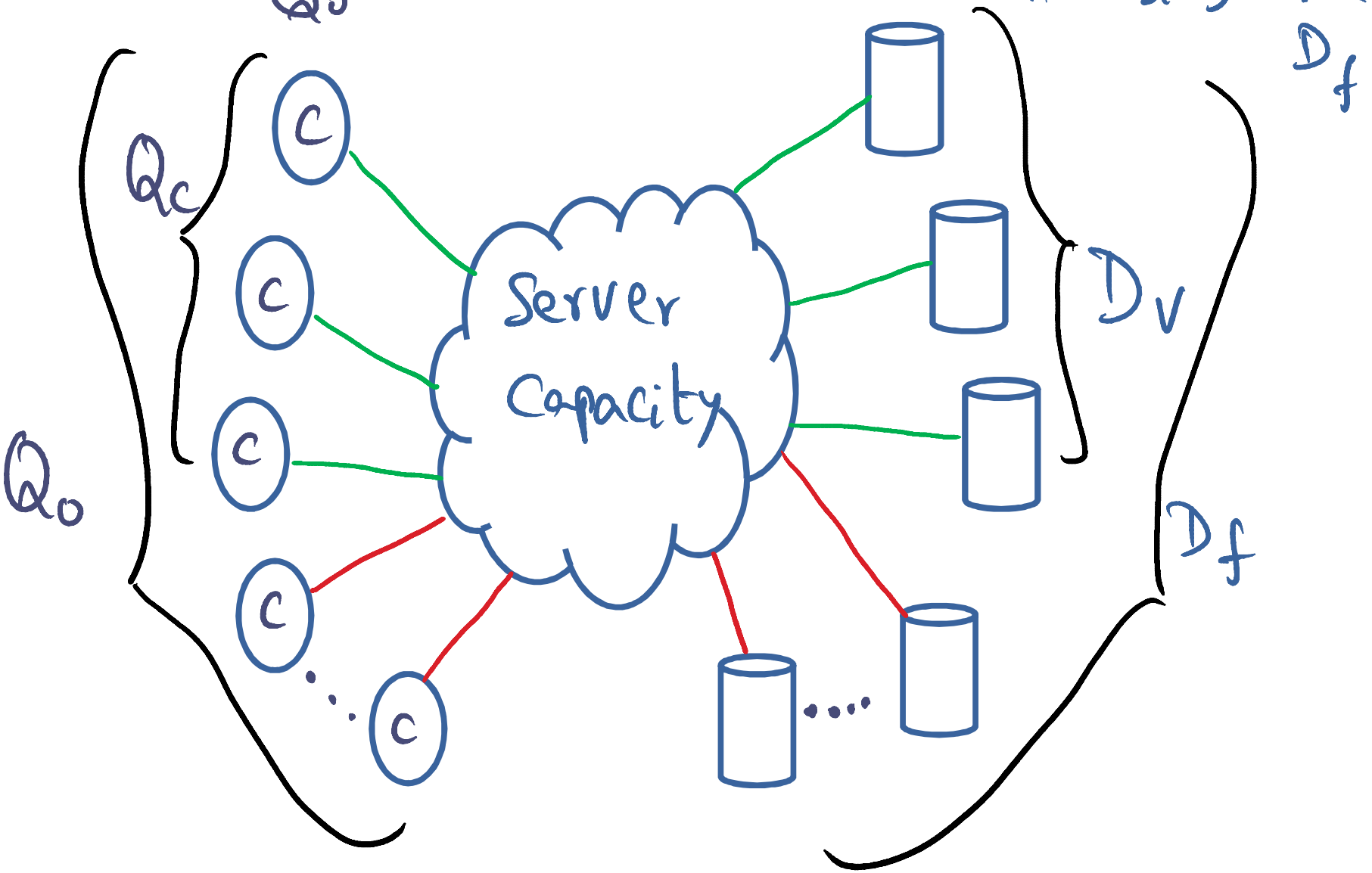
Yield: $Q = \frac{Q_c}{Q_o}$



DQ Principle

Yield: $Q = \frac{Q_c}{Q_o}$

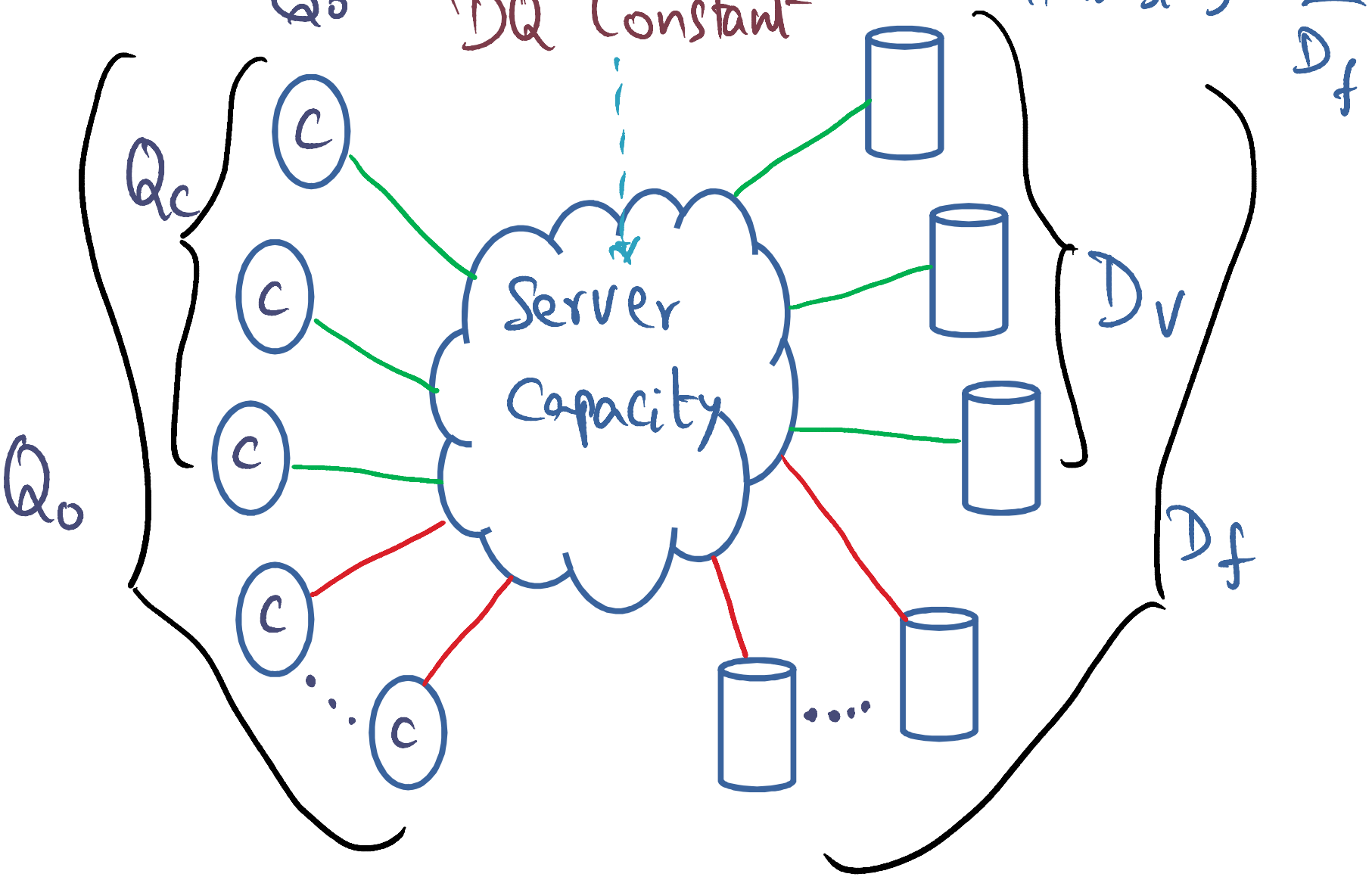
Harvest: $D = \frac{D_v}{D_f}$



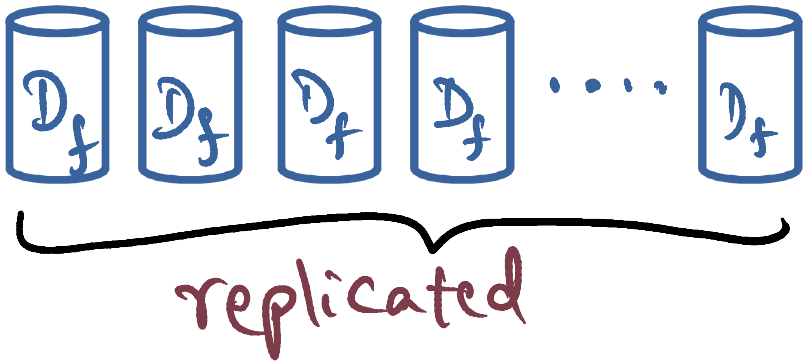
DQ Principle

Yield: $Q = \frac{Q_c}{Q_o}$

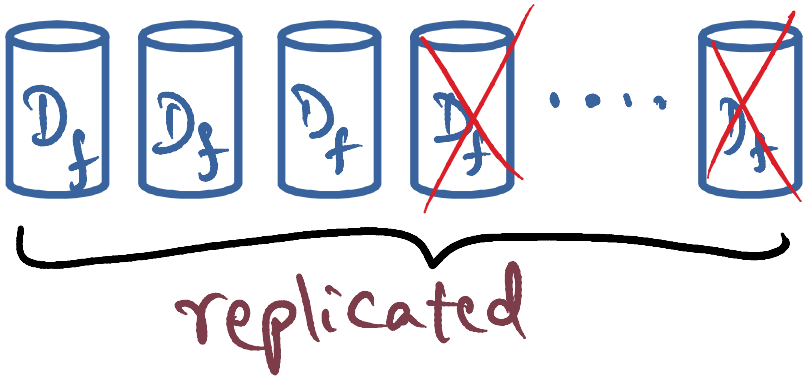
Harvest: $D = \frac{D_v}{D_f}$



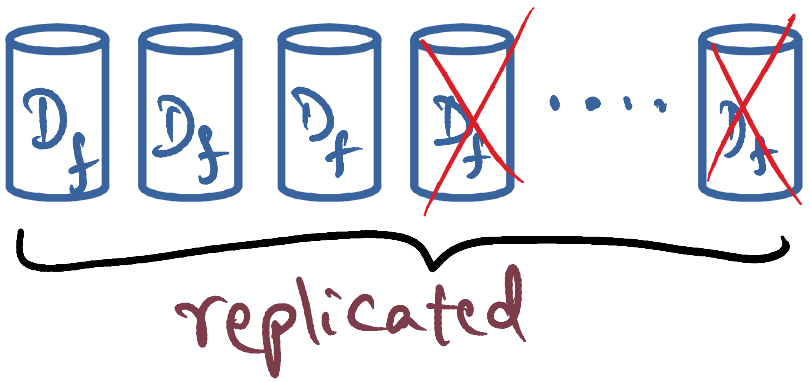
Replication Vs. Partitioning



Replication Vs. Partitioning



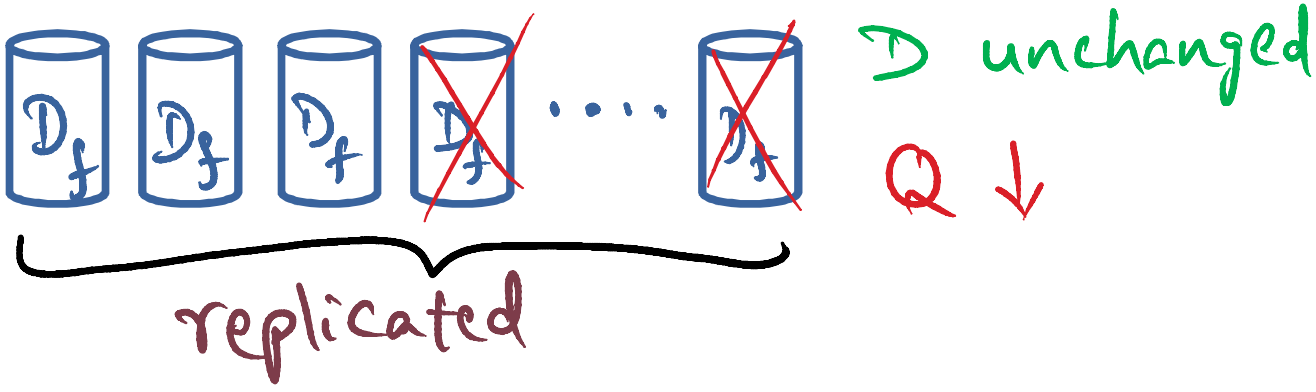
Replication Vs. Partitioning



D unchanged

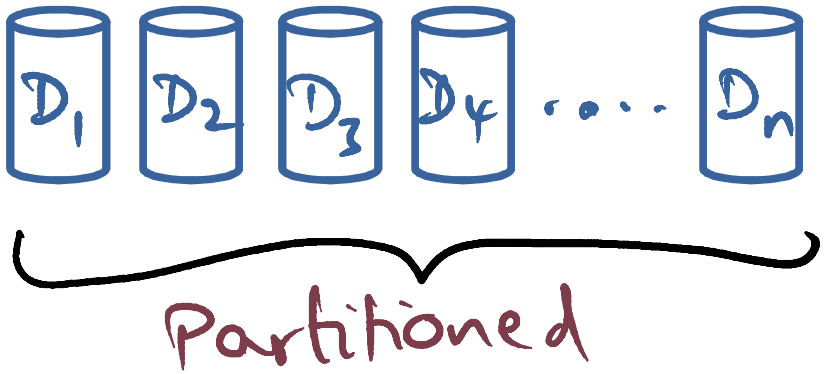
Q ↓

Replication Vs. Partitioning

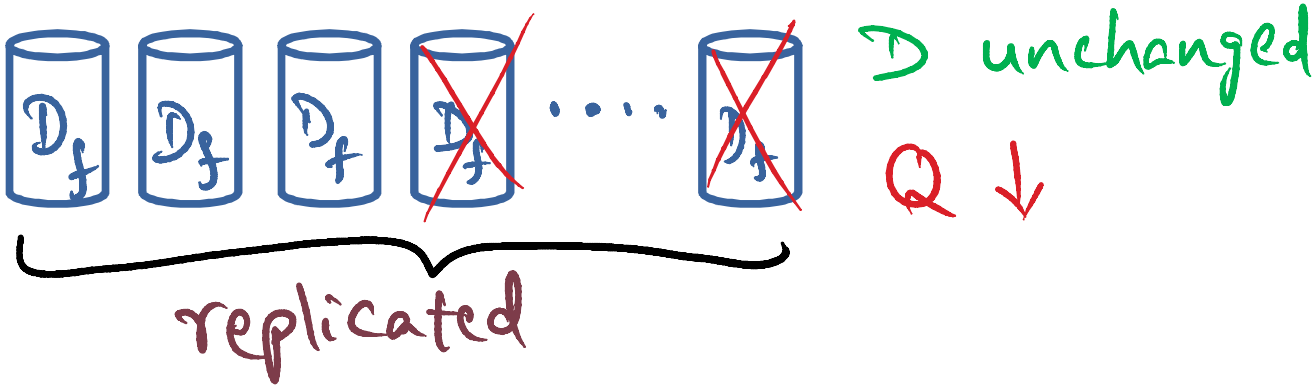


D unchanged

Q ↓

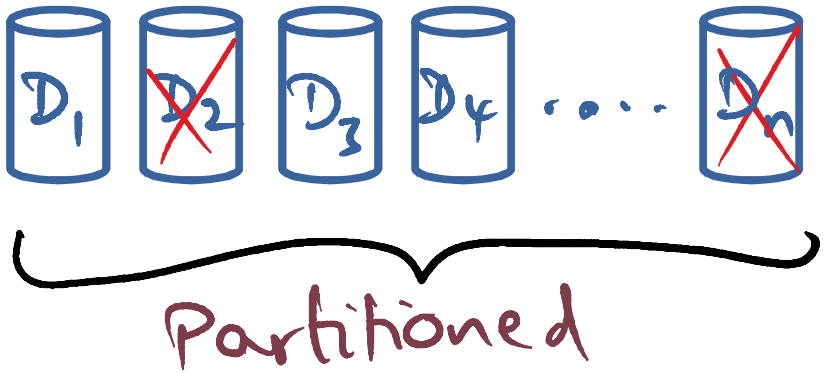


Replication Vs. Partitioning

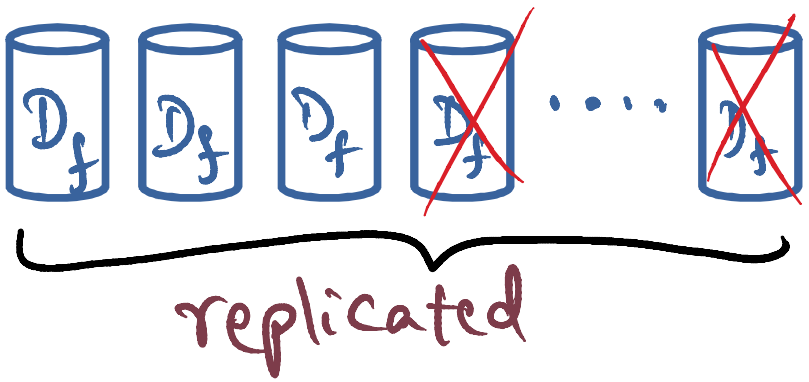


D unchanged

Q ↓

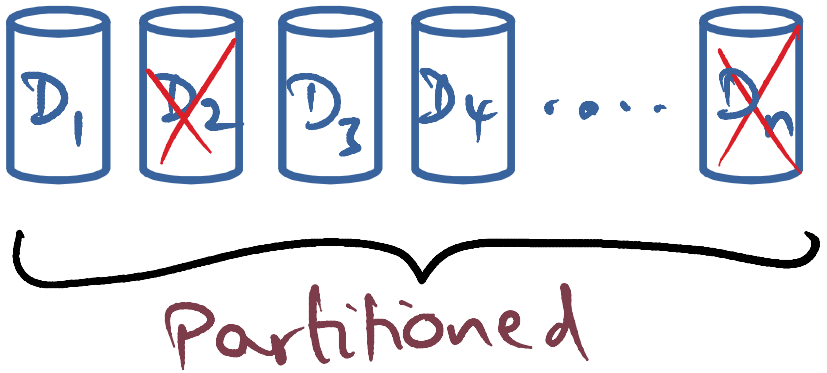


Replication Vs. Partitioning



D unchanged

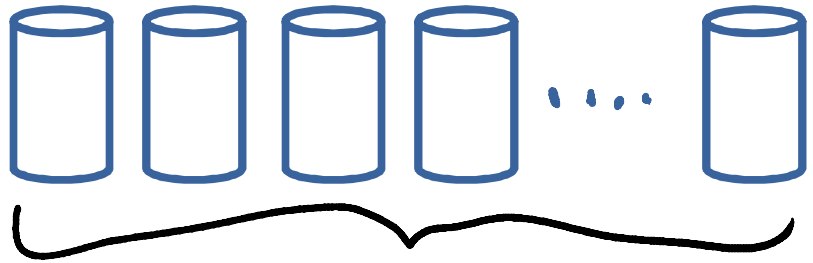
Q ↓



D ↓

Q unchanged

Graceful Degradation



Defines capacity: DQ

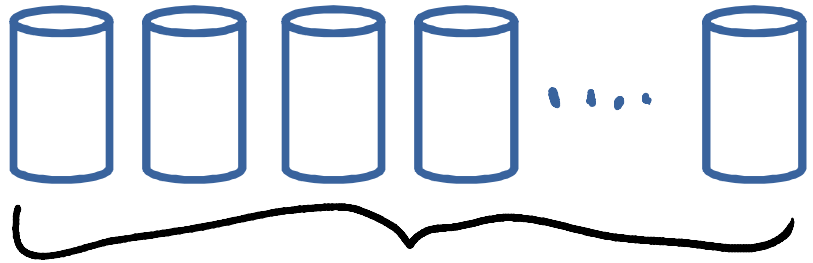
How do we deal with server saturation

DQ constant

⇒

- D fixed; Q ↓

Graceful Degradation



Defines capacity: DQ

How do we deal with server saturation

DQ constant

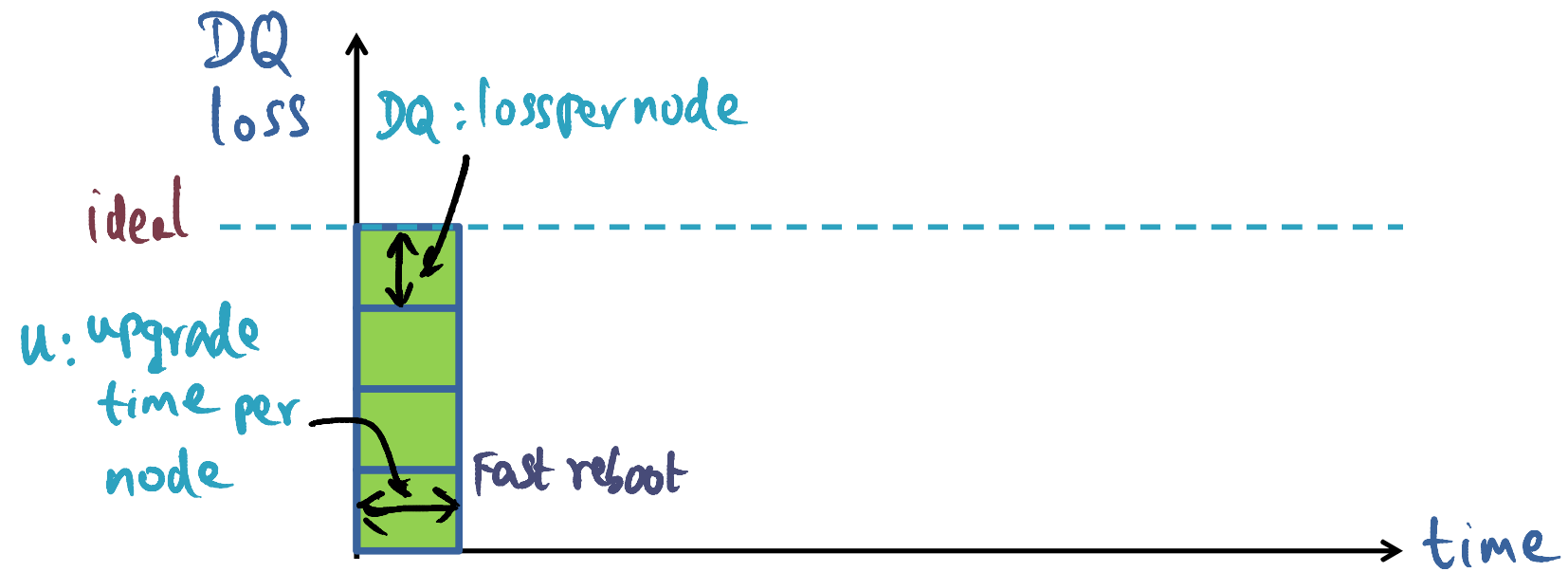
$\Rightarrow$

- D fixed; Q $\downarrow$

or

- D $\downarrow$; Q fixed

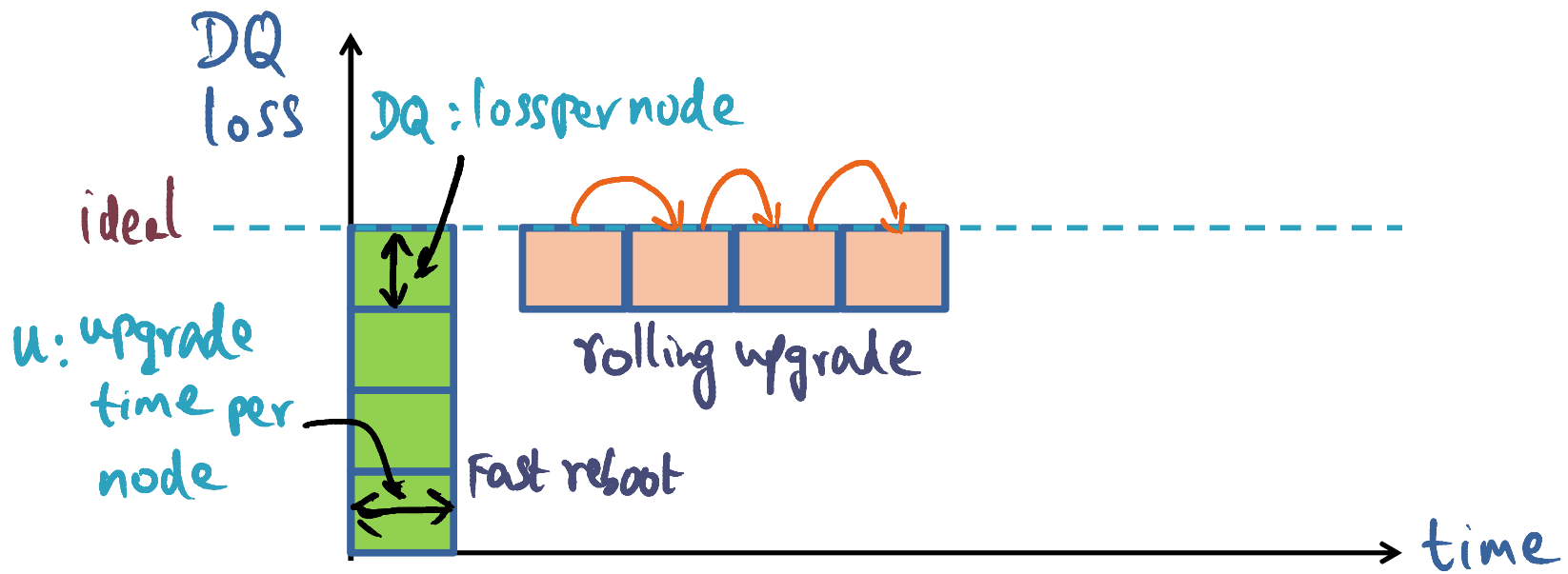
Online Evolution and growth



Fast

- off peak (e.g., use diurnal Server property)

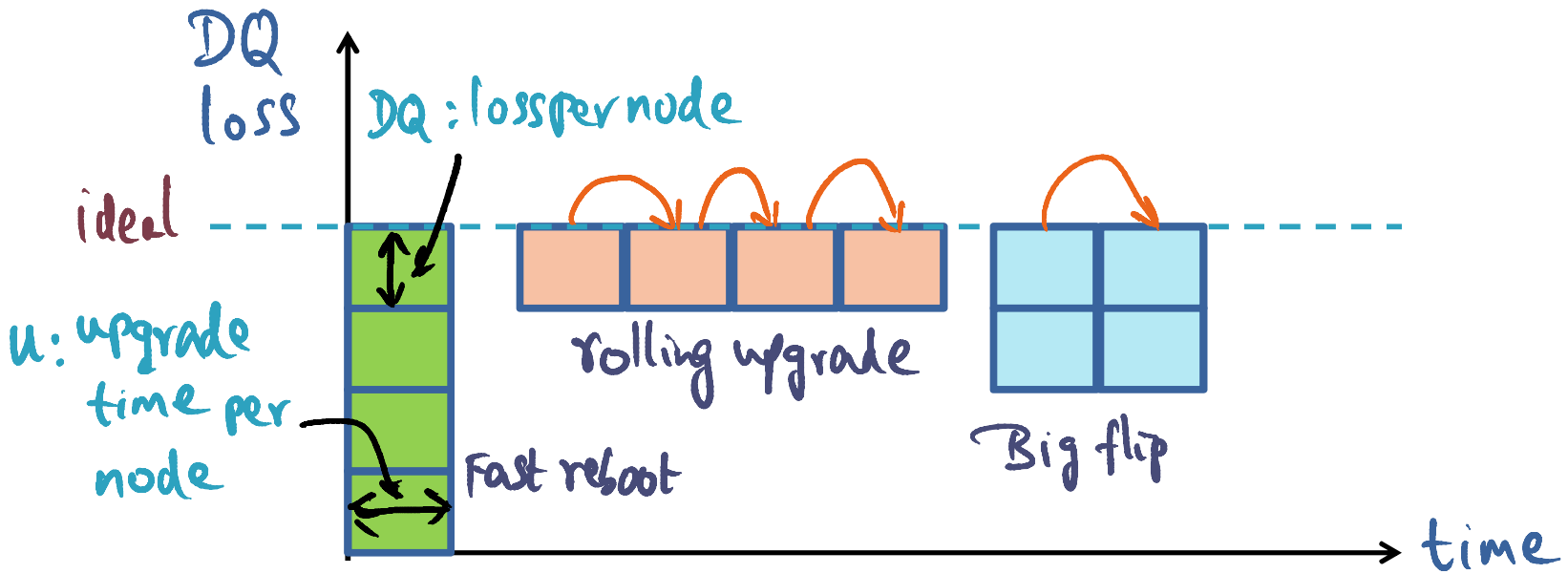
Online Evolution and growth



Fast
- off peak (e.g., use diurnal Server property)

Rolling
- wave upgrade

Online Evolution and growth

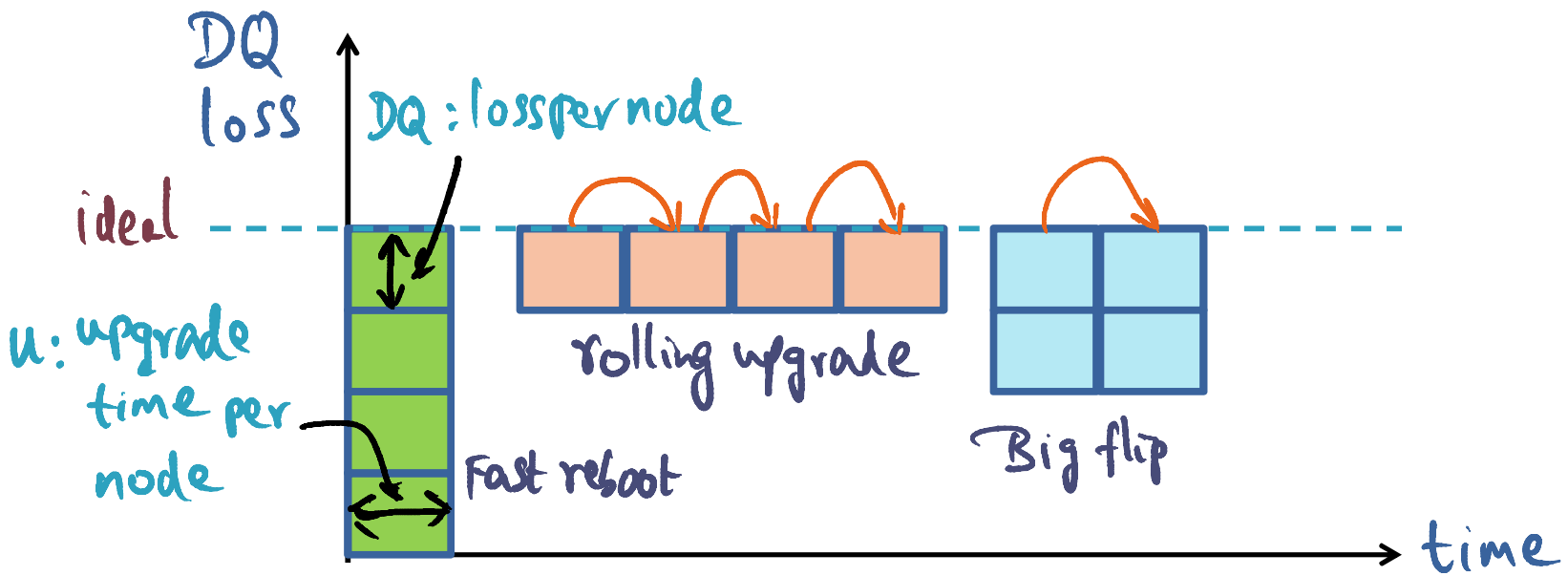


Fast
- off peak (e.g., use diurnal Server property)

Rolling
- wave upgrade

Big Flip
- Half the nodes at once

Online Evolution and growth



Fast
- off peak (e.g., use diurnal Server property)

Rolling
- wave upgrade

Big Flip
- Half the nodes at once

DQ loss same for all three strategies

$= DQ * U * n$

Key takeaways

- Giant-scale services are network bound and not disk I/O bound => DQ principle
- DQ principle:
  - optimizing for yield or harvest for a given system capacity
  - helps with explicit policies for graceful degradation when either failure of servers happen, or load saturation happens, or upgrades are planned