Non-functional requirements - Durability

durability

once data is successfully submitted to the system, it is not lost

How do systems achieve durability?

by creating and maintaining **multiple copies** of data

backup RAID replication

Non-functional requirements - Durability

copy data periodically and store it elsewhere

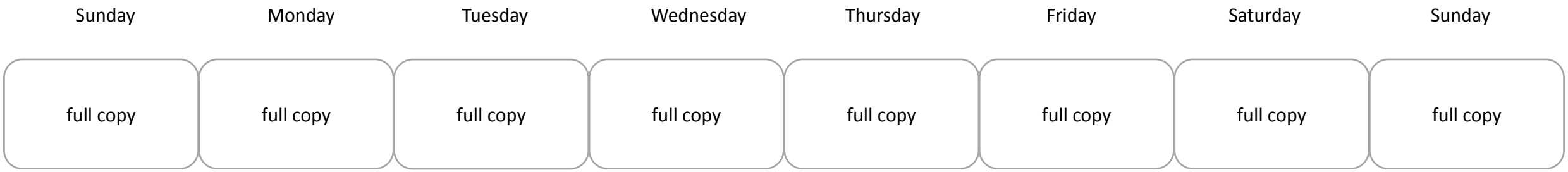
backup

three popular strategies to create backups

- **full** backup
- **differential** backup
- **incremental** backup

Non-functional requirements - Durability

full backup

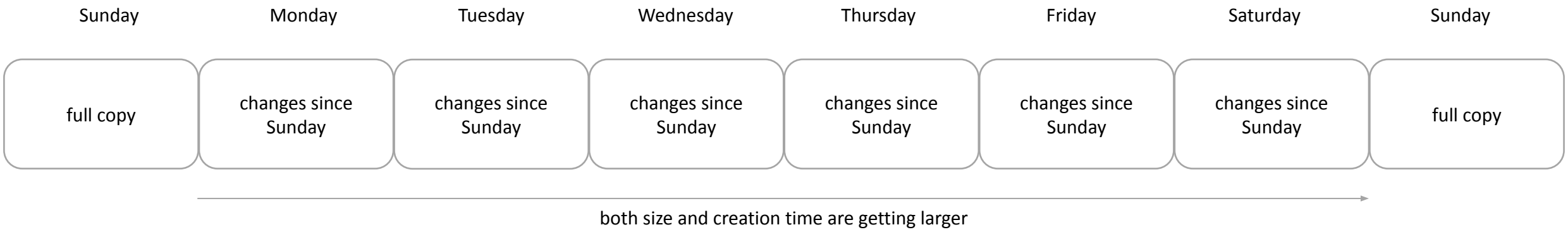


pros
- short restoration time

cons
- long creation time

Non-functional requirements - Durability

differential backup



pros

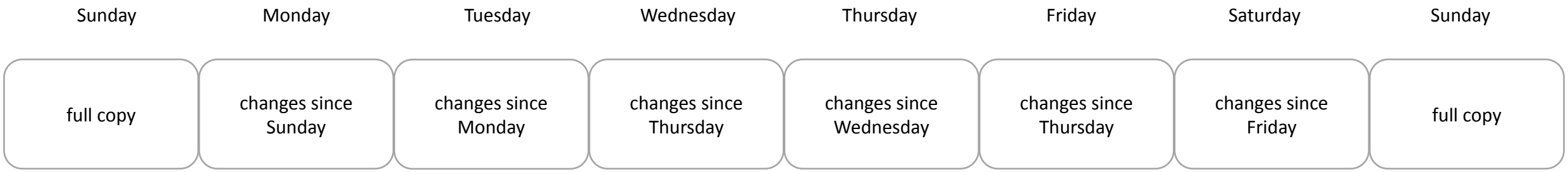
- smaller than a full backup
- shorter creation time

cons

- longer restoration time (than full backup)

Non-functional requirements - Durability

incremental backup



pros

- small size
- short creation time

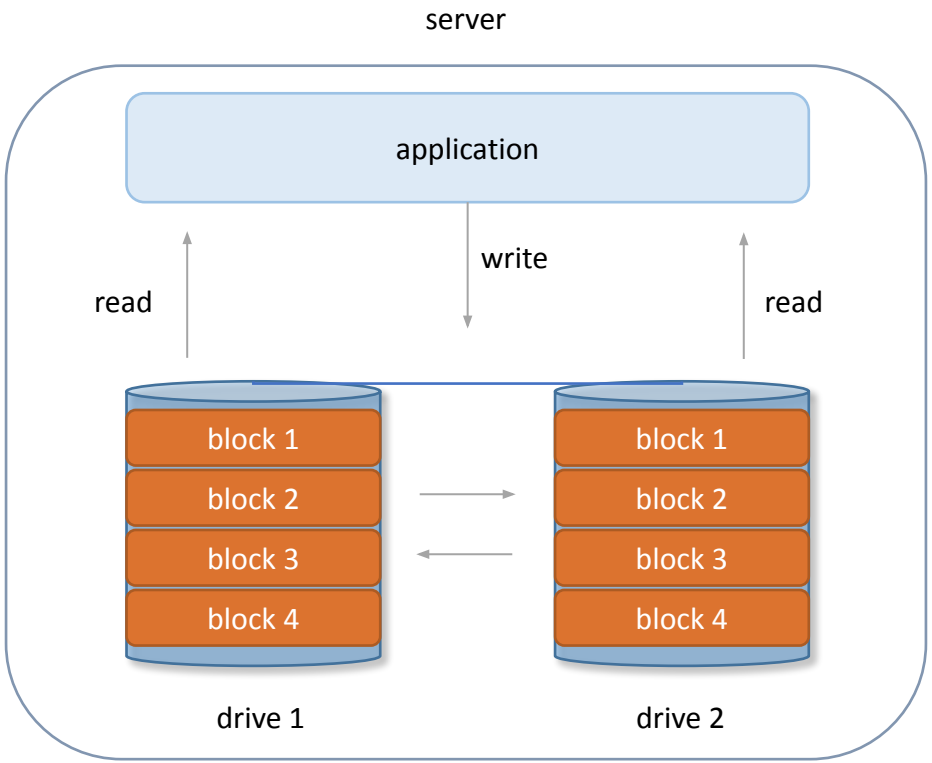
cons

- long restoration time
- more complex restoration process

Non-functional requirements - Durability

RAID

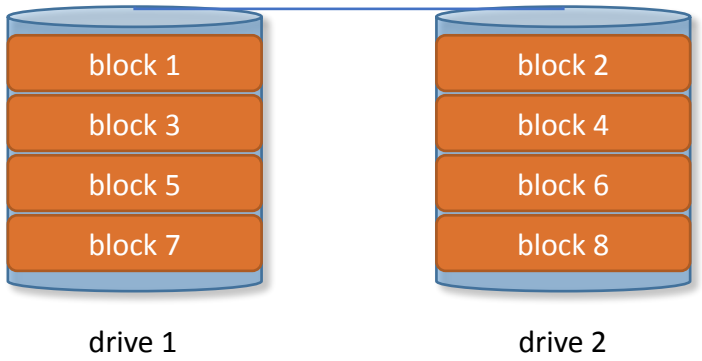
(redundant array of independent disks)



RAID 1
(aka mirrored volume)

Non-functional requirements - Durability

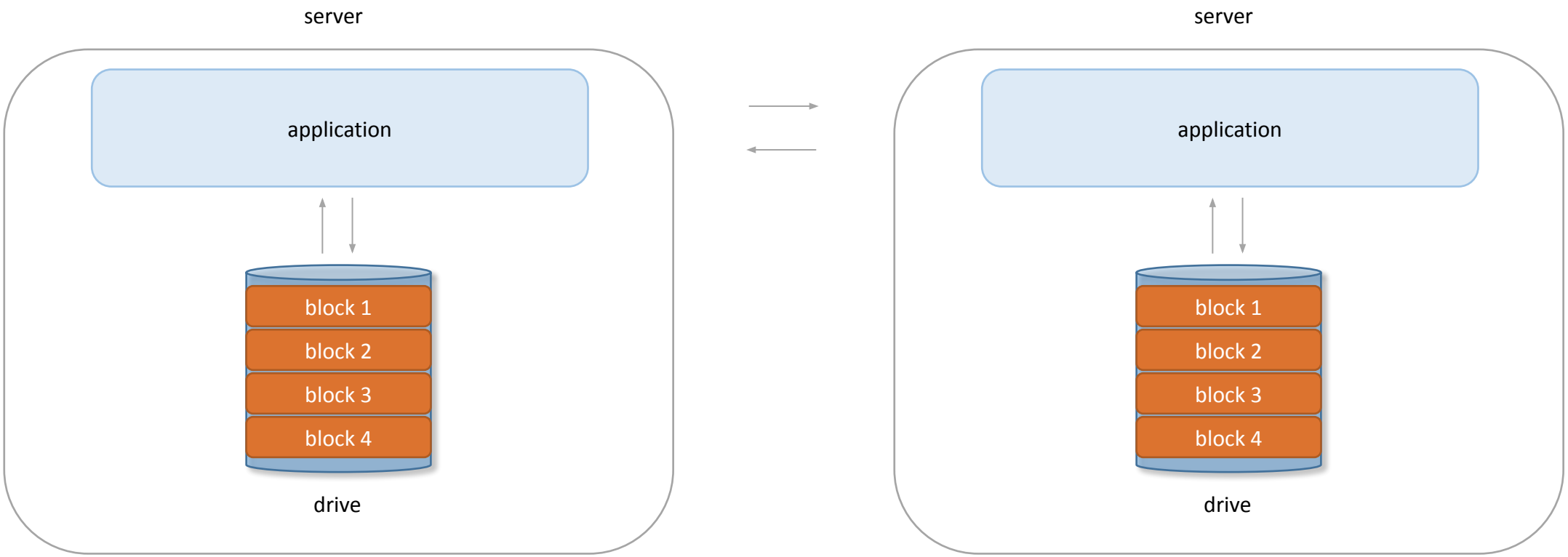
RAID 0



- employs the techniques of data striping
- no data redundancy
- increases throughput

Non-functional requirements - Durability

replication



Non-functional requirements - Durability

backup and/or replication and/or RAID

- versioning
- safeguards against accidental deletion
- and
  more…

Non-functional requirements - Durability

What if we have multiple copies of data, but they all got corrupted somehow?

software engineer

checksum

on write

data

checksum function

3782225421

on read

compare checksums (retrieved data checksum vs original checksum)

If corrupted

create one more copy from other healthy copies

Non-functional requirements - Durability

availability

about system uptime

“Can I access my data right now?”

≠

durability

about storing data without losing it

“Will my data still be there in the future?”