Nonfunctional Requirements for System Design Interviews

educative

Nonfunctional Requirements (NFRs)

NFRs determine how well a system operates under specific conditions. For example, how fast can systems handle millions of requests per second? How secure is it designed?

Common NFRs

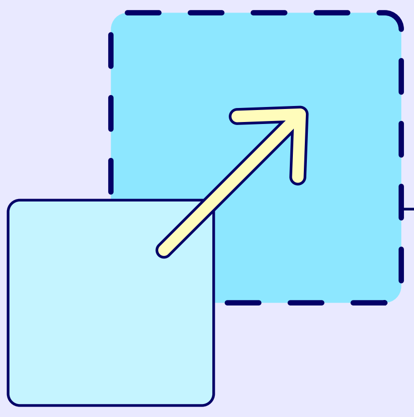
Availability:

The system's ability to ensure maximum uptime for users.

Techniques:
- Replication
- Fault tolerance
- Rate limiting
- CDNs
- Stress testing

Scalability:

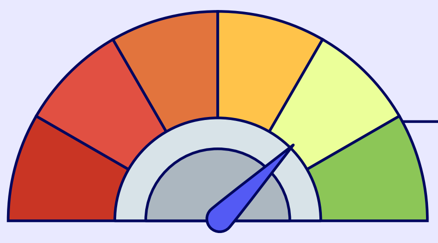
The system's ability to handle a growing number of users or requests.



Techniques:
- Vertical/Horizontal scaling
- Automated scaling
- Sharding
- Modular design
- Cache/CDNs

Performance:

The system's ability to respond to user requests and process data efficiently.



Techniques:
- Load balancing
- Efficient data structure/ Algorithm/Storage
- Caching
- CDNs
- Parallel processing

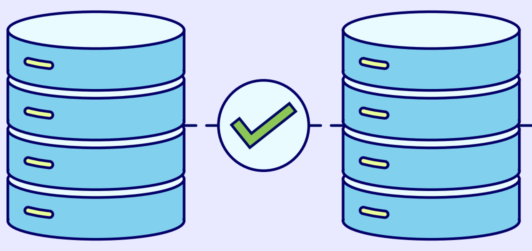
Security:

The system's ability to protect against unauthorized access and threats.

Techniques:
- Authorization using JWT
- Encryption
- Security protocols
- Login mechanism
- OAuth and OpenID Connect/PKCE

Consistency:

The system's ability to ensure that all users see the same data simultaneously.



Techniques:
- Distributed transaction
- Consistency models
- Conflict resolution techniques
- Versioning

Reliability:

The system's ability to recover quickly and smoothly from failure.

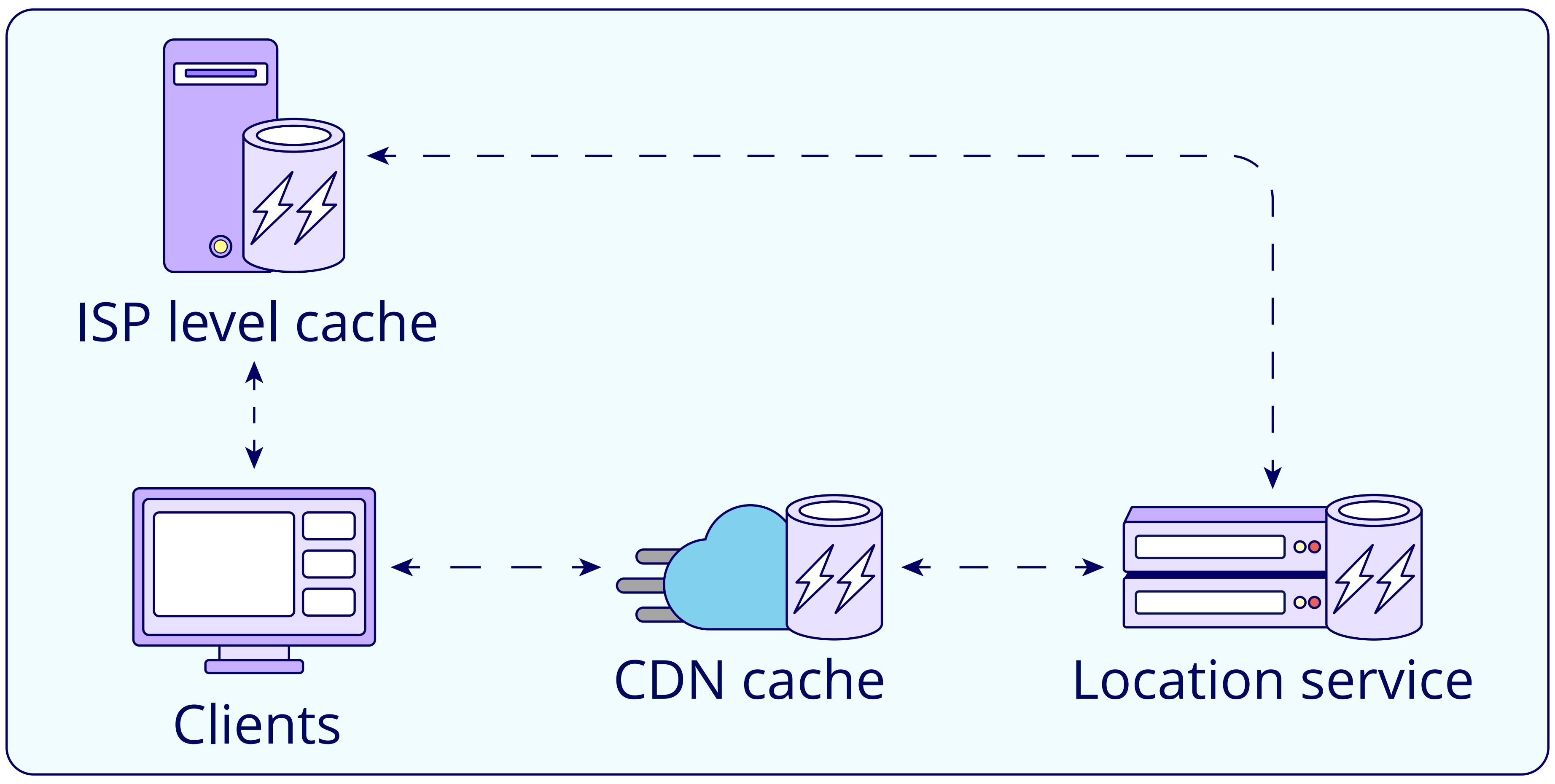
Techniques:
- Replication
- Data partitioning
- Circuit breakers
- Load balancing
- Automated monitoring

Case Studies

1. Optimizing response time in the Google Maps navigation system

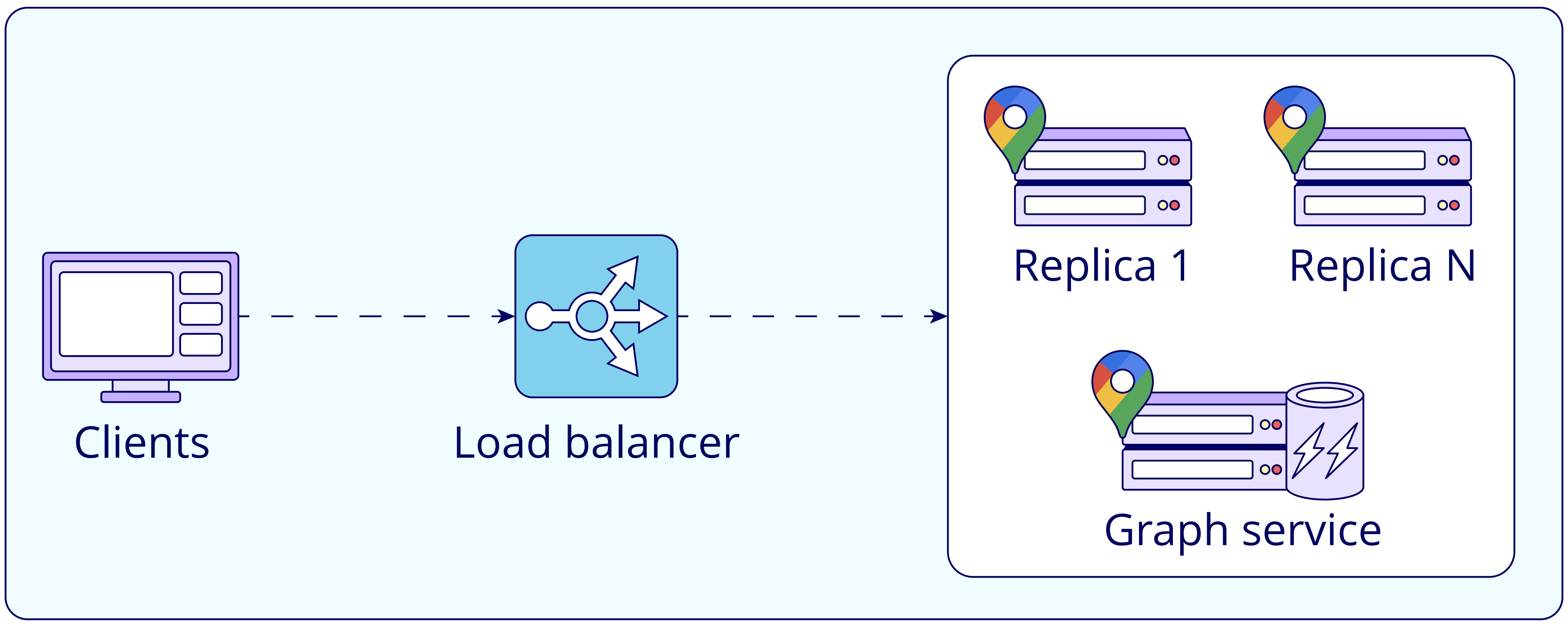
Techniques:
Caching, load balancing, and efficient data storage

Caching:
- Cache frequently searched locations at the ISP level
- CDNs cache to deliver data quickly
- Cache precomputed distances at the service level



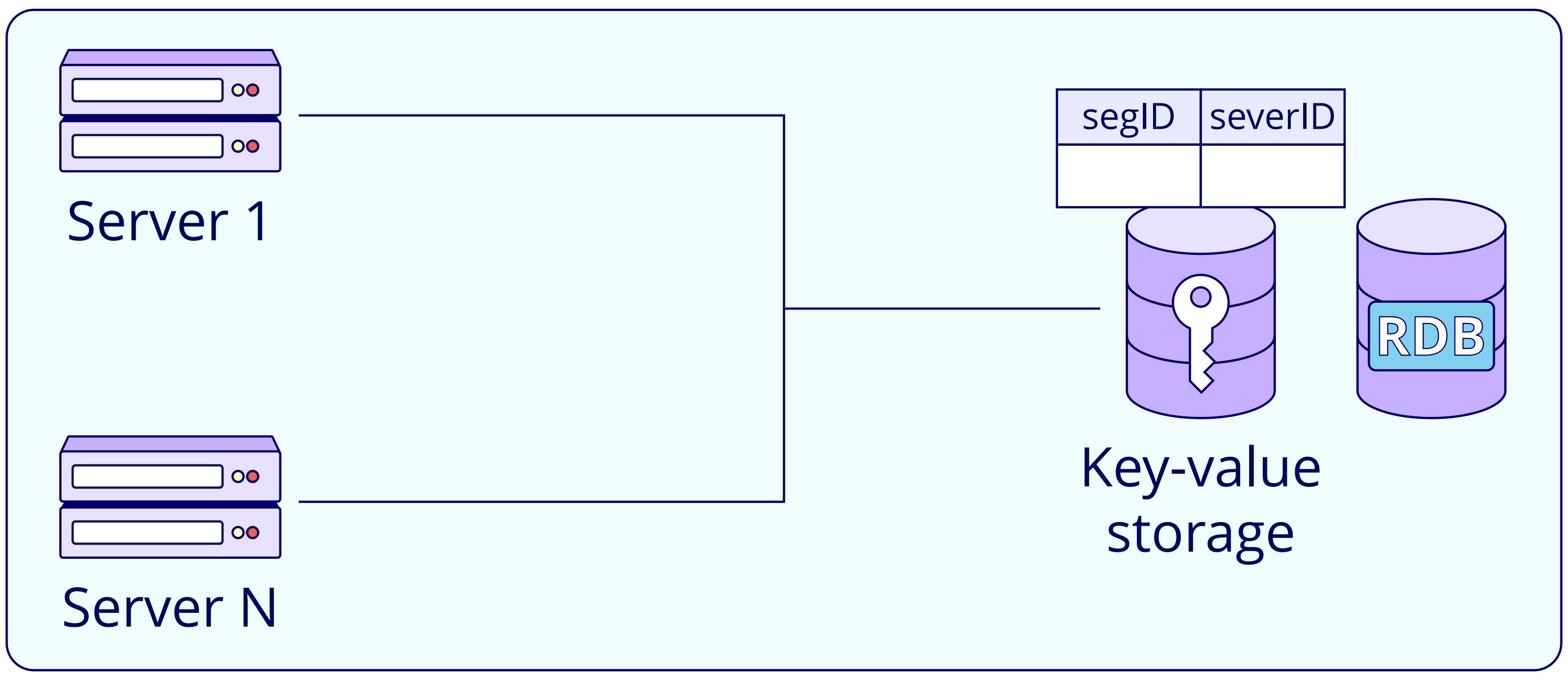
Load balancing:

- Divide the road network graph into smaller graphs or segments
- Host road network graph segments on different servers
- Route requests to the appropriate server rather than a single server



Efficient data storage:

- Use key-value storage for segment information



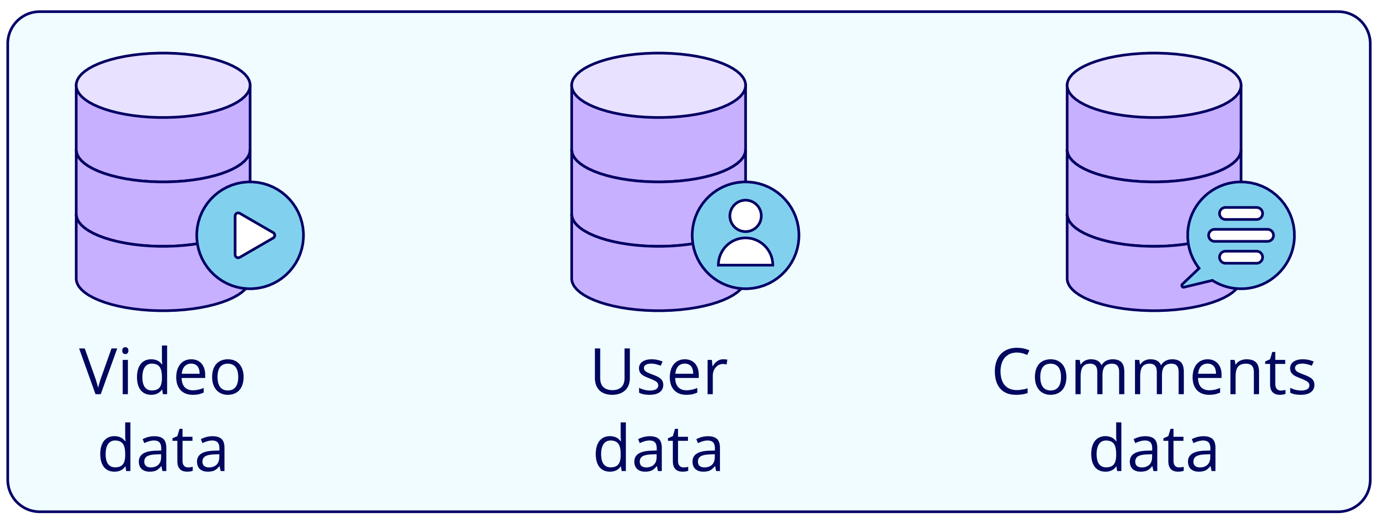
2. Ensuring reliability in YouTube video streaming platforms

Techniques:

Data partitioning, automated monitoring, and consistent hashing models

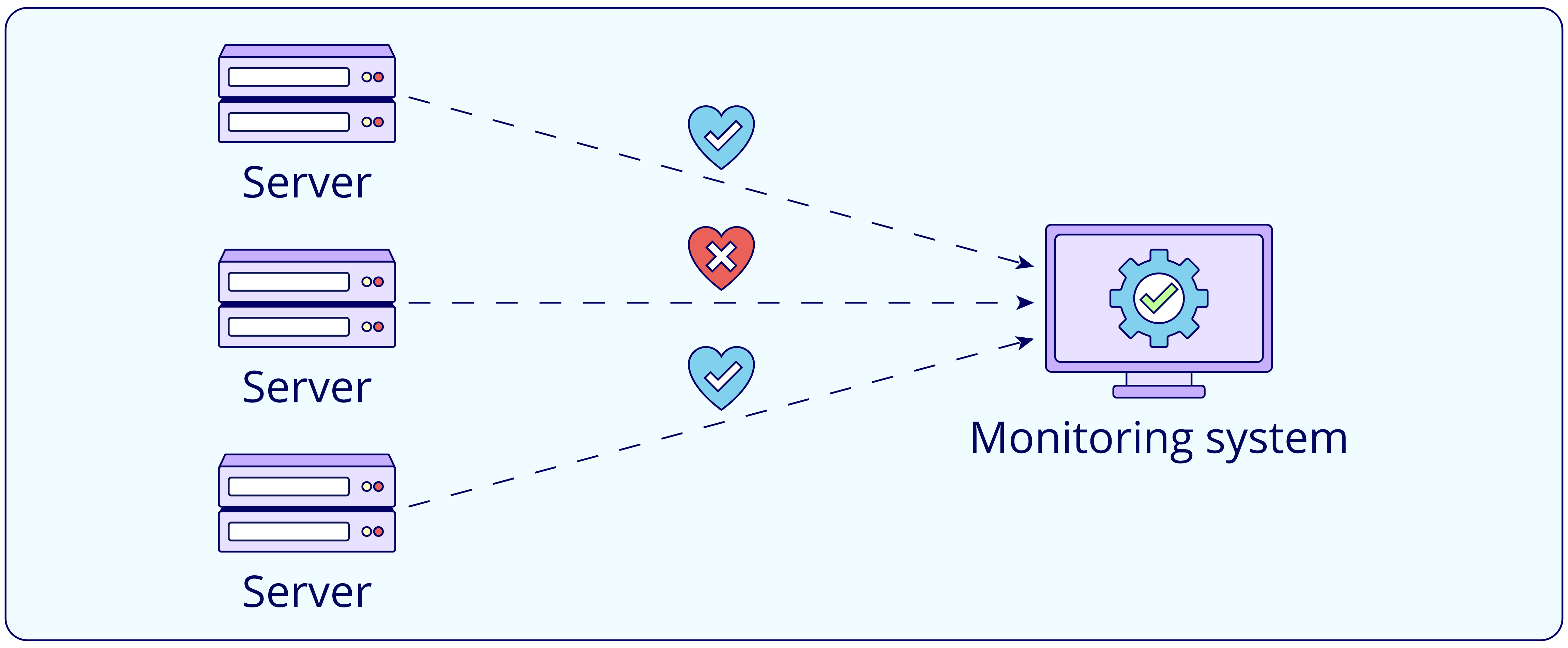
Data partitioning:

- Data partitioning isolates types, preventing cross-impact issues



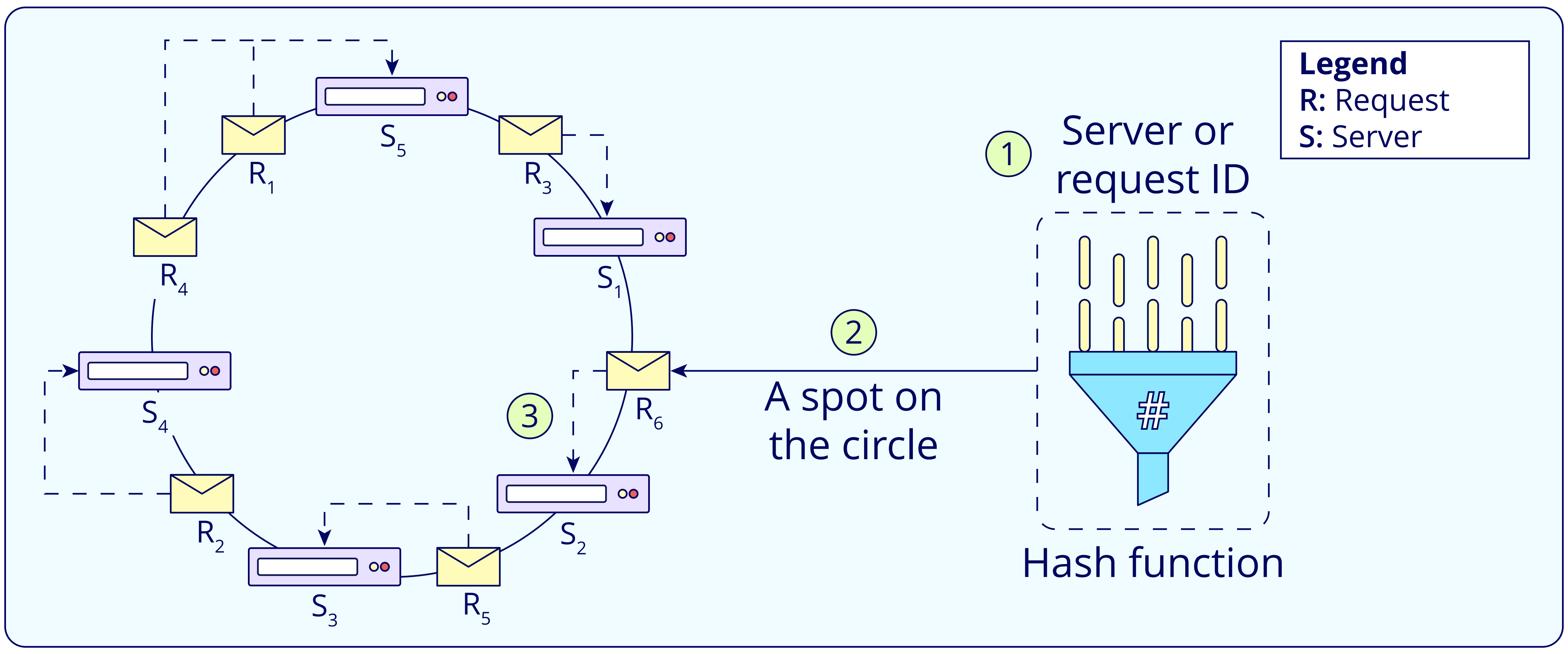
Automated monitoring:

- The heartbeat protocol monitors server health and triggers alerts



Consistent hashing models:

- Consistent hashing for seamless server addition/removal


Legend
R: Request
S: Server
1
Server or request ID
2
A spot on the circle
3
Hash function
#
S_5
S_1
S_2
S_3
S_4
R_1
R_2
R_3
R_4
R_5
R_6