System Architecture

Memi Lavi
www.memilavi.com

System Architecture

- The Big Picture
- Answers the questions:
  - How will the system work under heavy load?
  - What will happen if the system will crash at this exact moment in the business flow?
  - How complicated can be the update process?
  - And more…

System Architecture

- Includes:
  - Defining the Software Components (Services)
  - Defining the way these components communicate
  - Designing the system's capabilities (scalability, redundancy, performance, etc.)

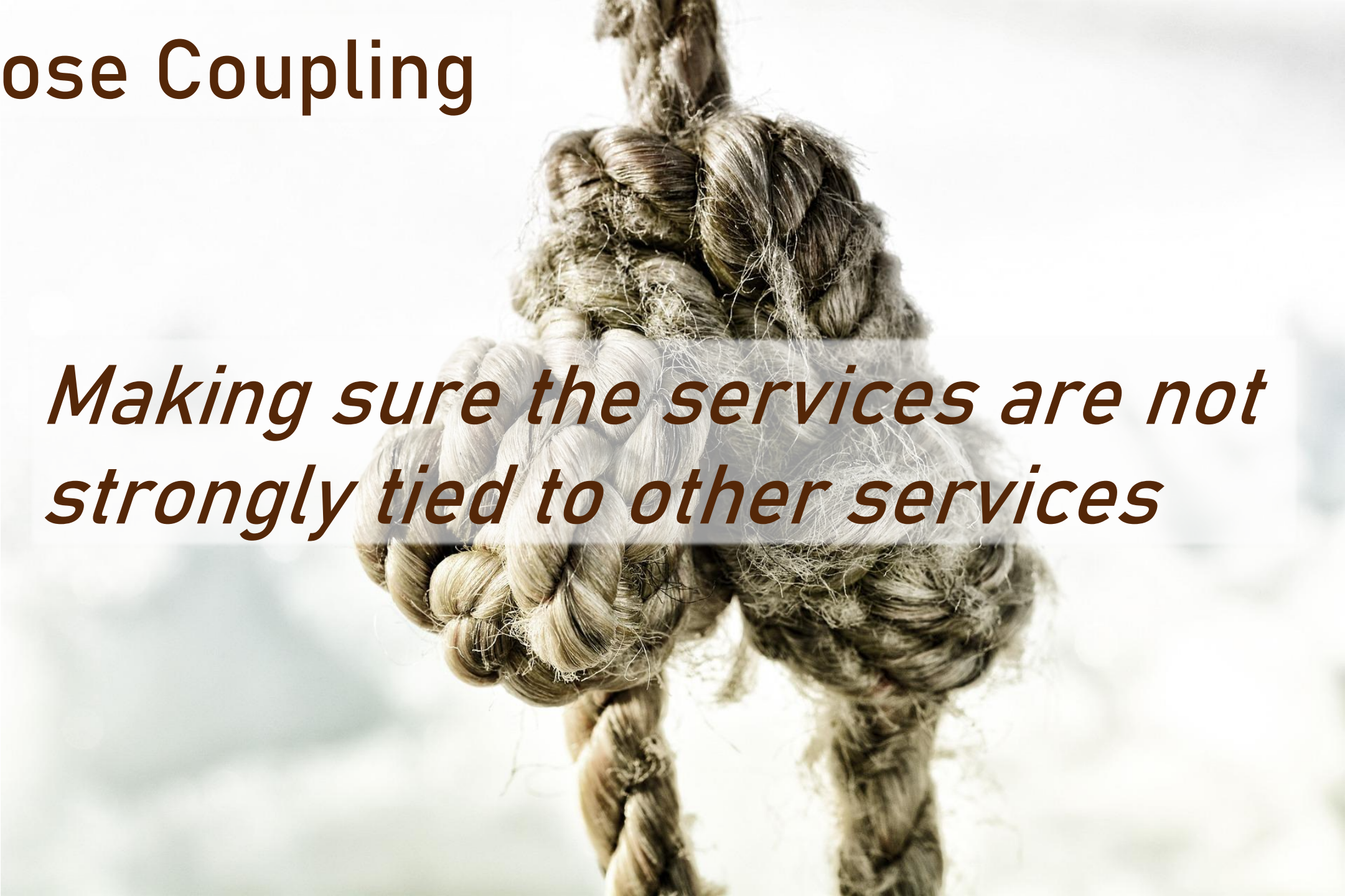
System Architecture

- Includes:
  - Defining the Software Components (Services)
  - Defining the way these components communicate
  - Designing the system's capabilities (scalability, redundancy, performance, etc.)

Our Topics

- Loose Coupling
- Stateless
- Caching
- Messaging
- Logging & Monitoring

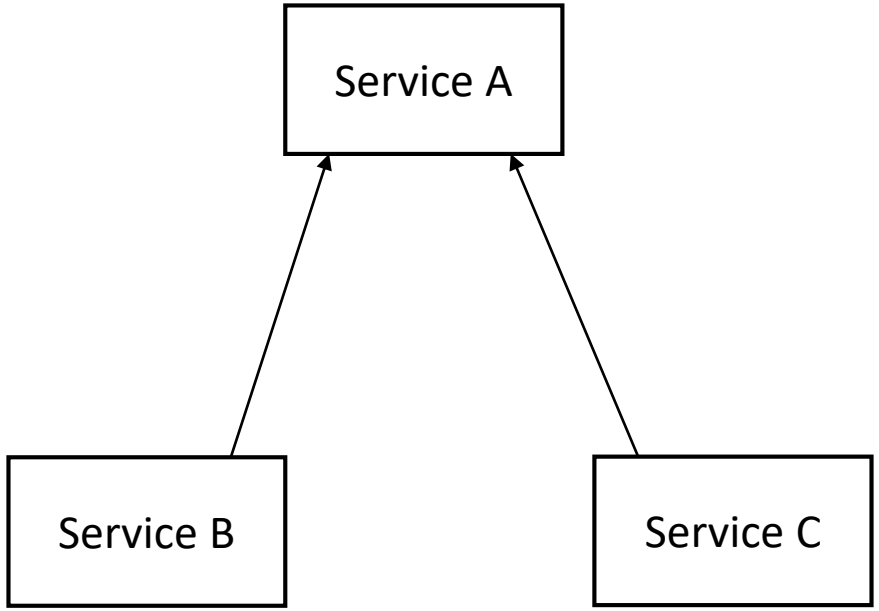
Loose Coupling



Making sure the services are not strongly tied to other services

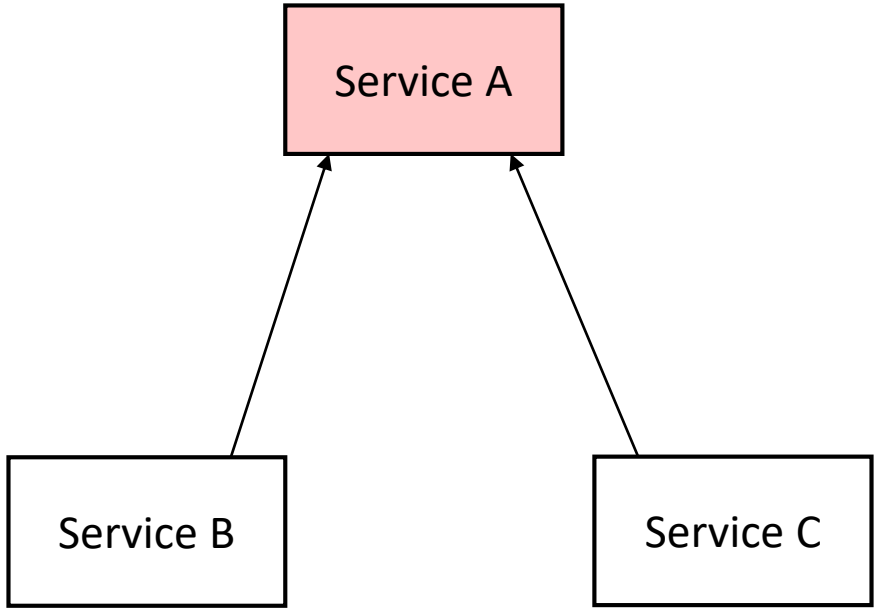
Loose Coupling

- Without:



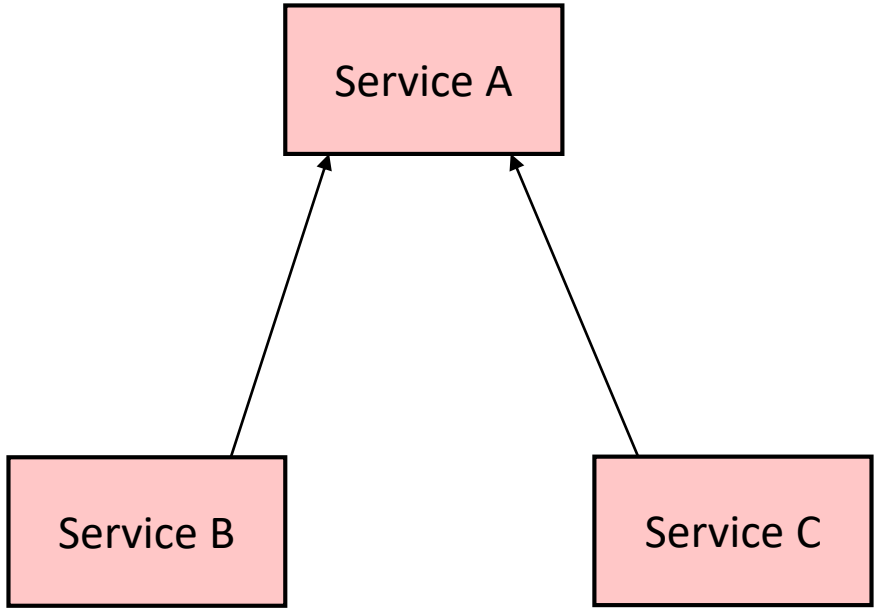
Loose Coupling

- Without:



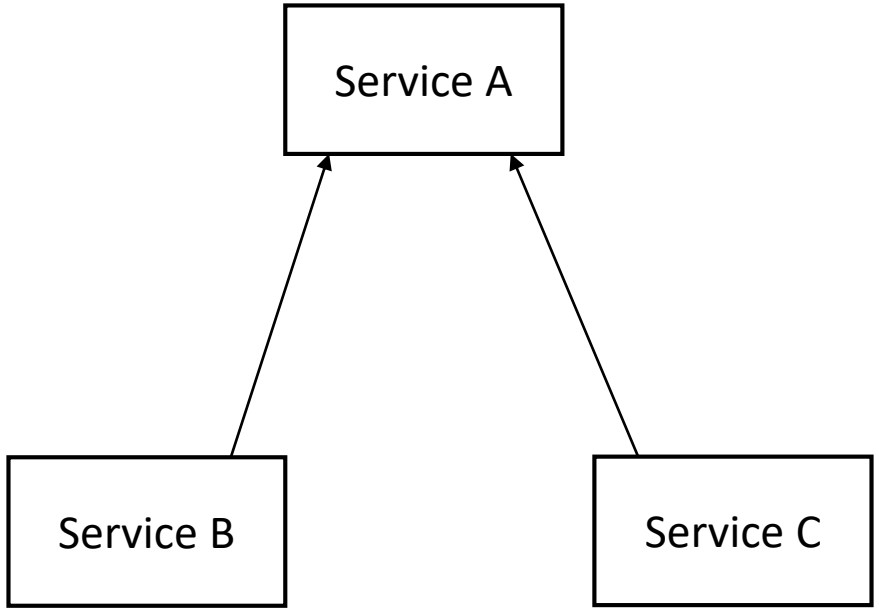
Loose Coupling

- Without:



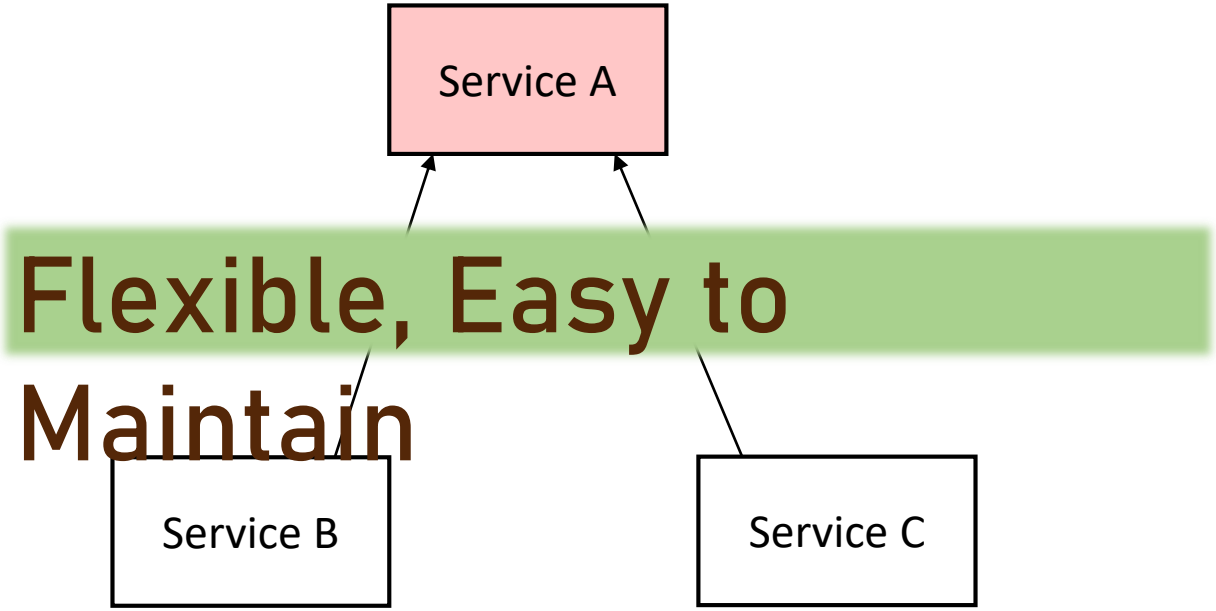
Loose Coupling

- With:



Loose Coupling

- With:

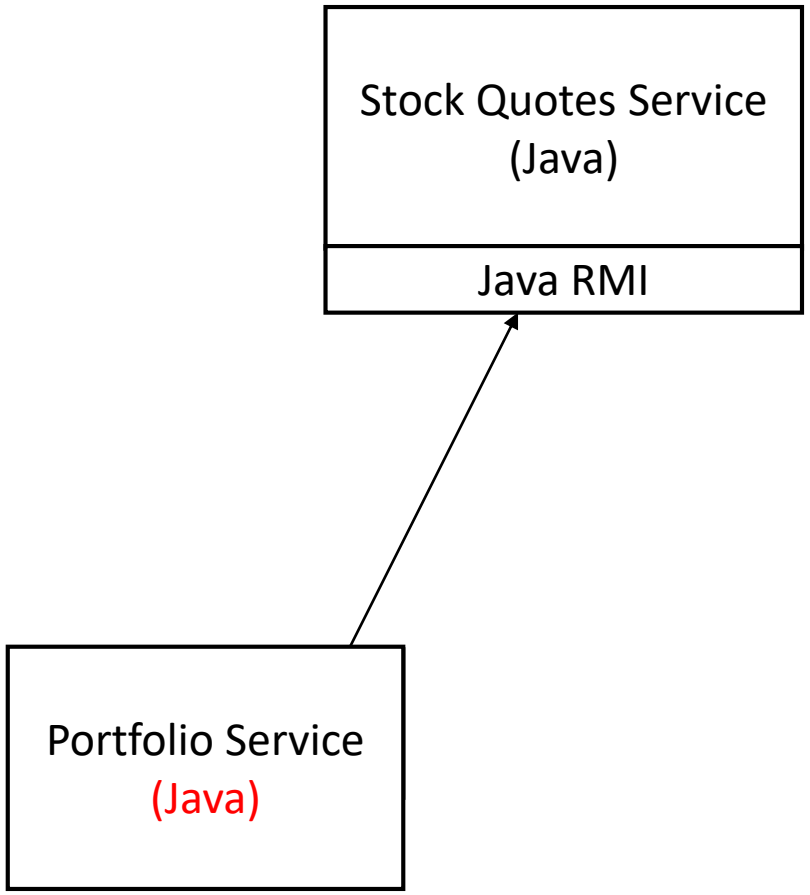


Flexible, Easy to Maintain

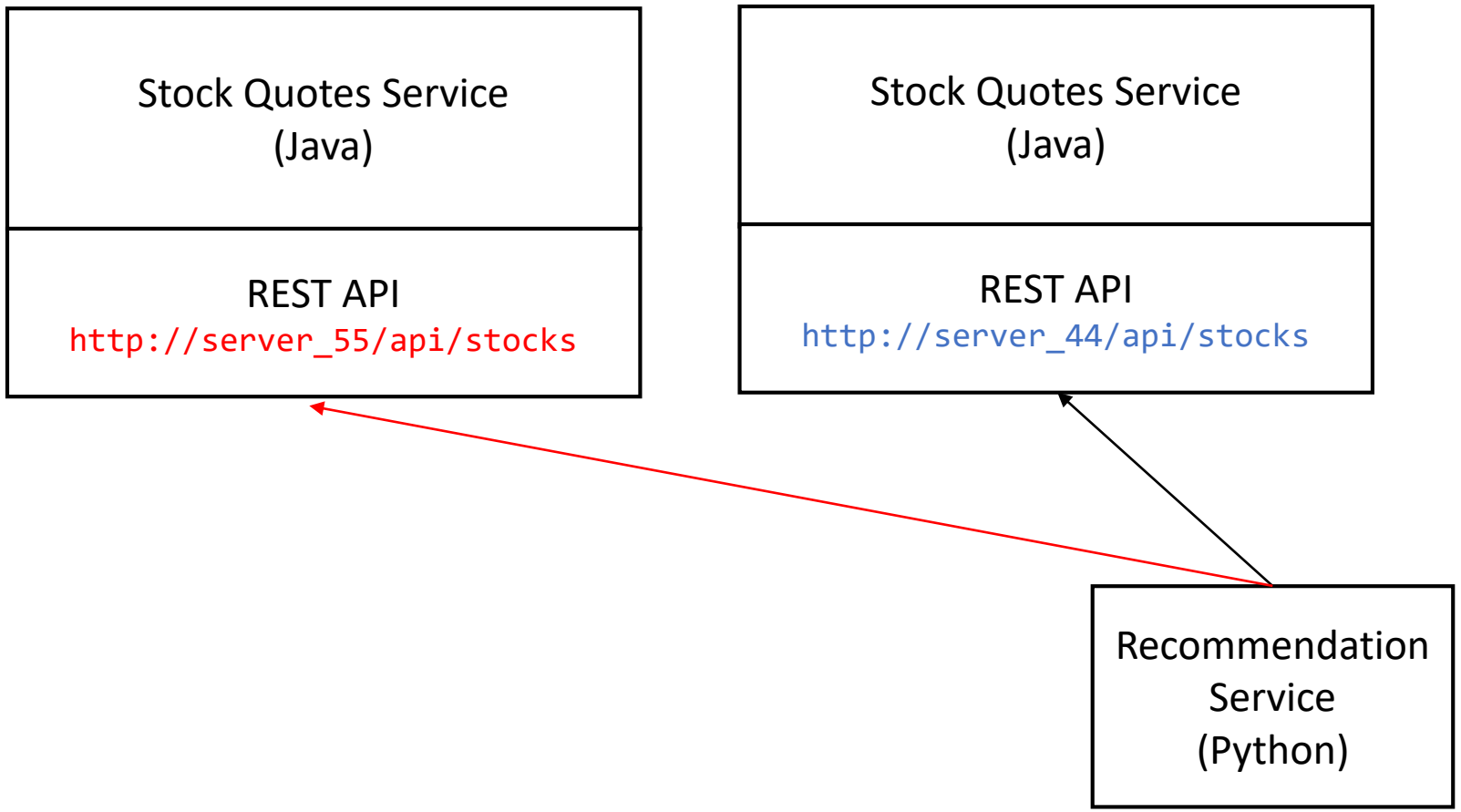
Loose Coupling in Services

- Prevents platform coupling
- Prevents URL coupling

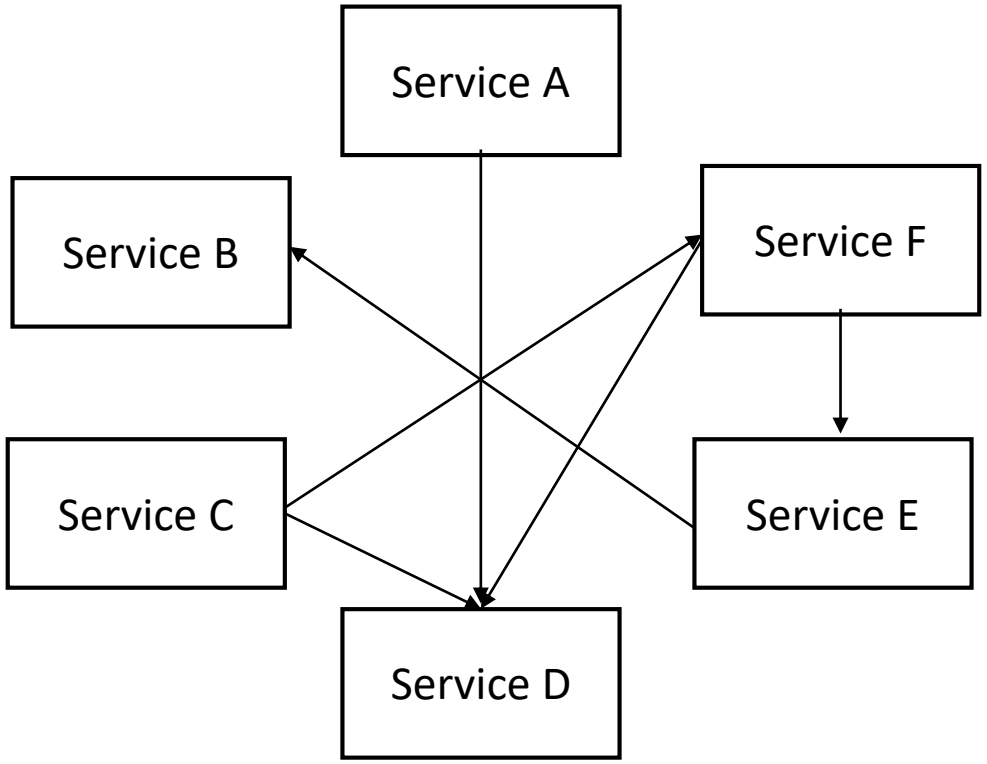
Loose Coupling in Services



Loose Coupling in Services

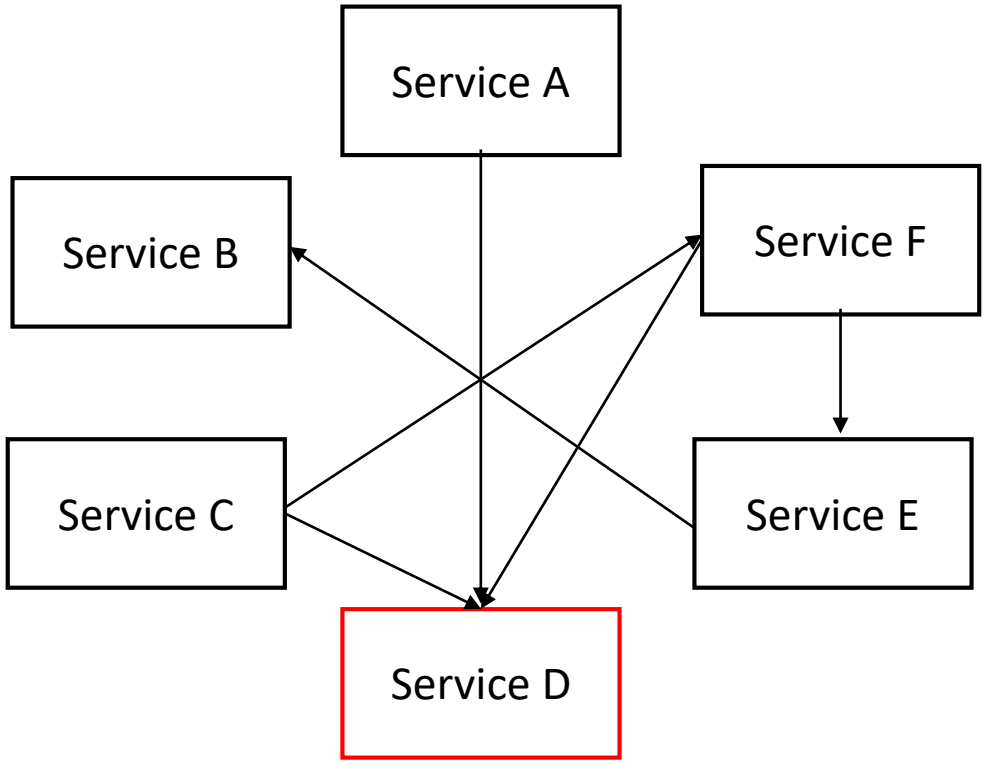


Loose Coupling



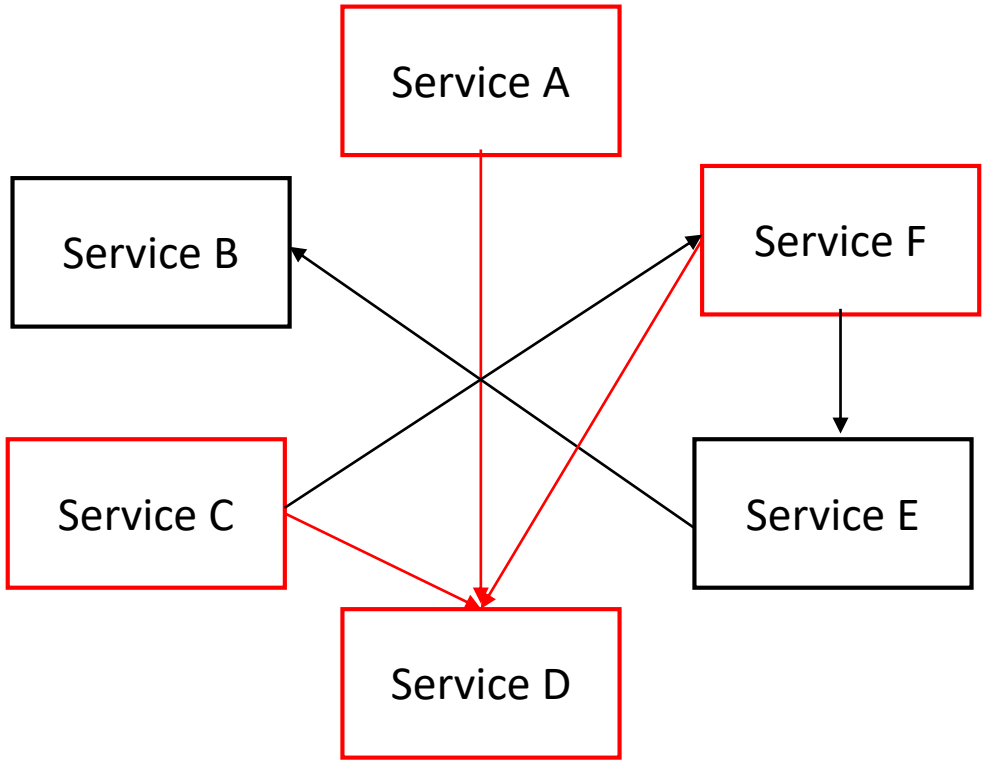
The Spiderweb

Loose Coupling



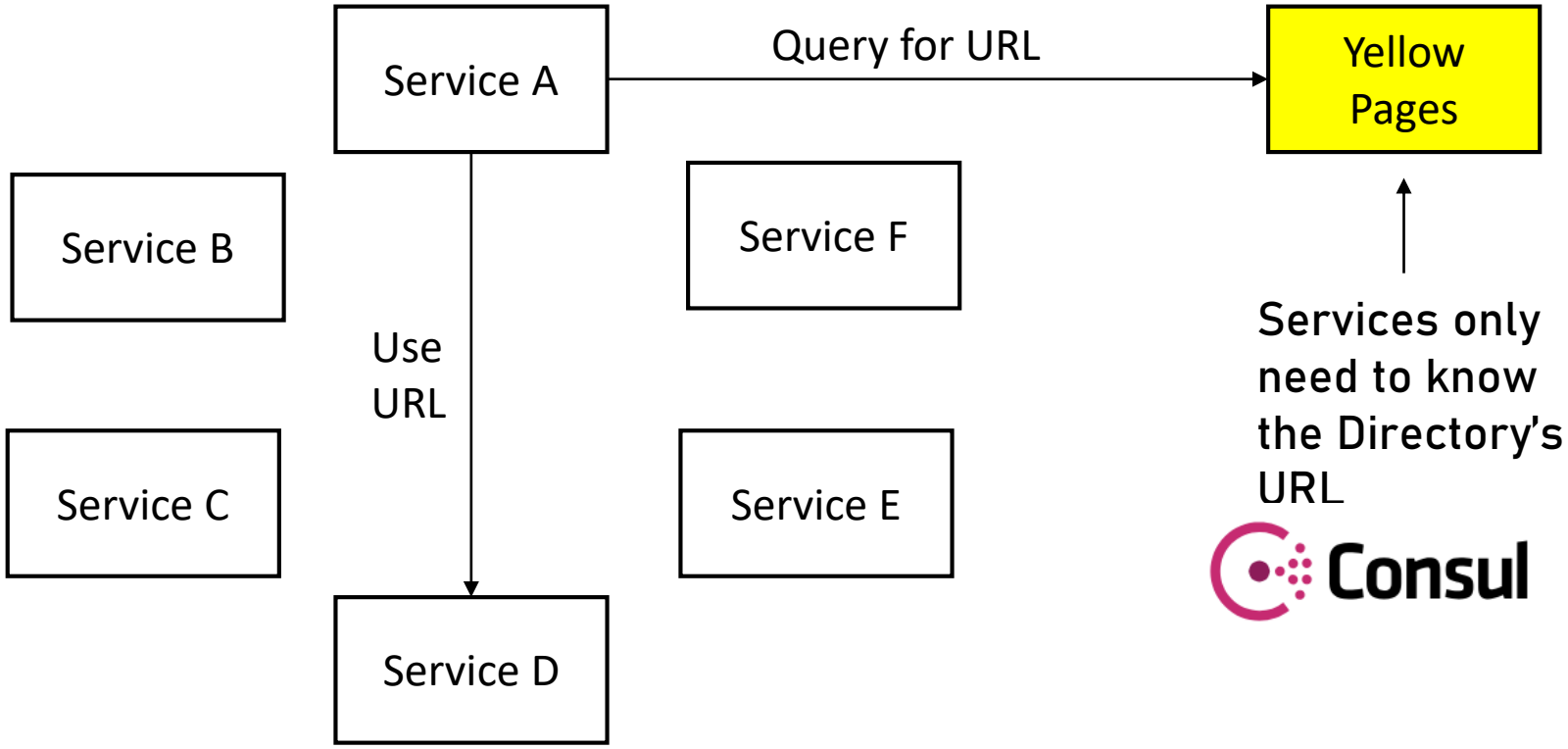
The Spiderweb

Loose Coupling

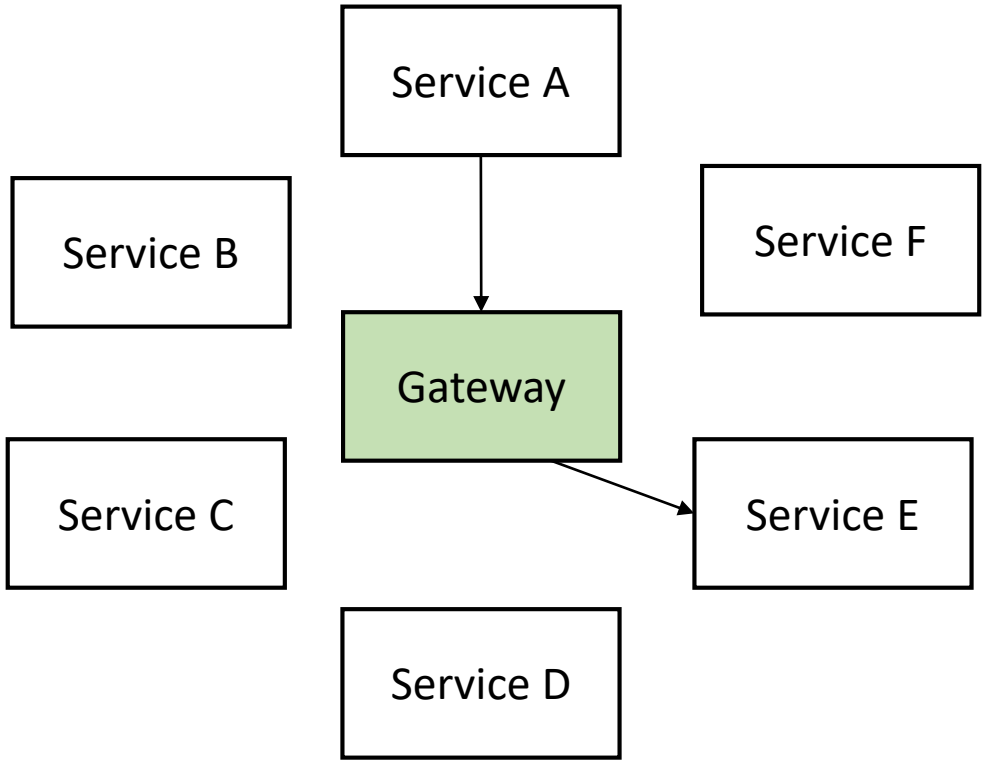


The Spiderweb

Loose Coupling - Directory



Loose Coupling – Gateway



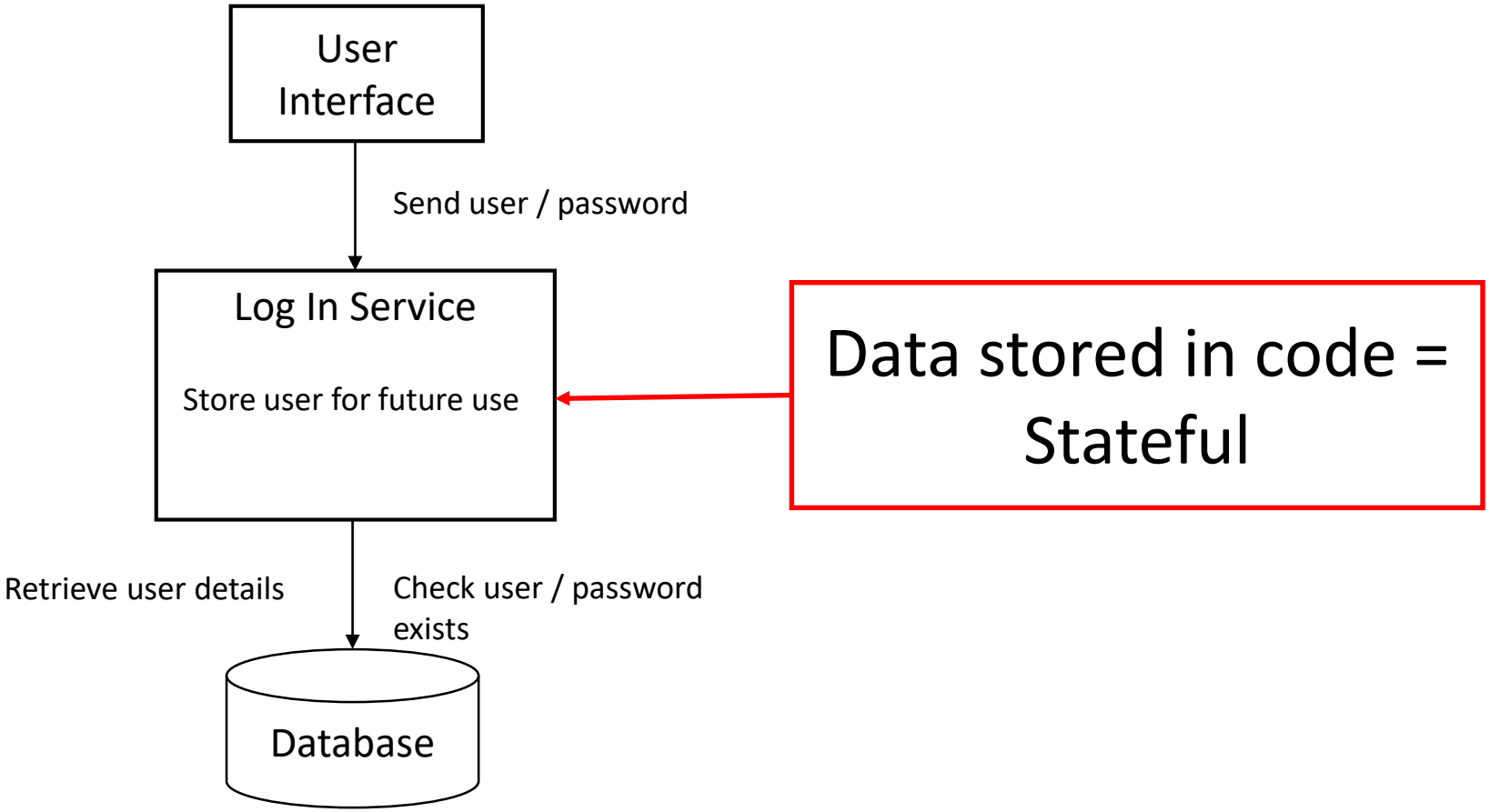
Services only need to know the Gateway's URL

Stateless

The application's state is stored in only two places – the data store and the user interface

State = Application's Data

Stateless Example



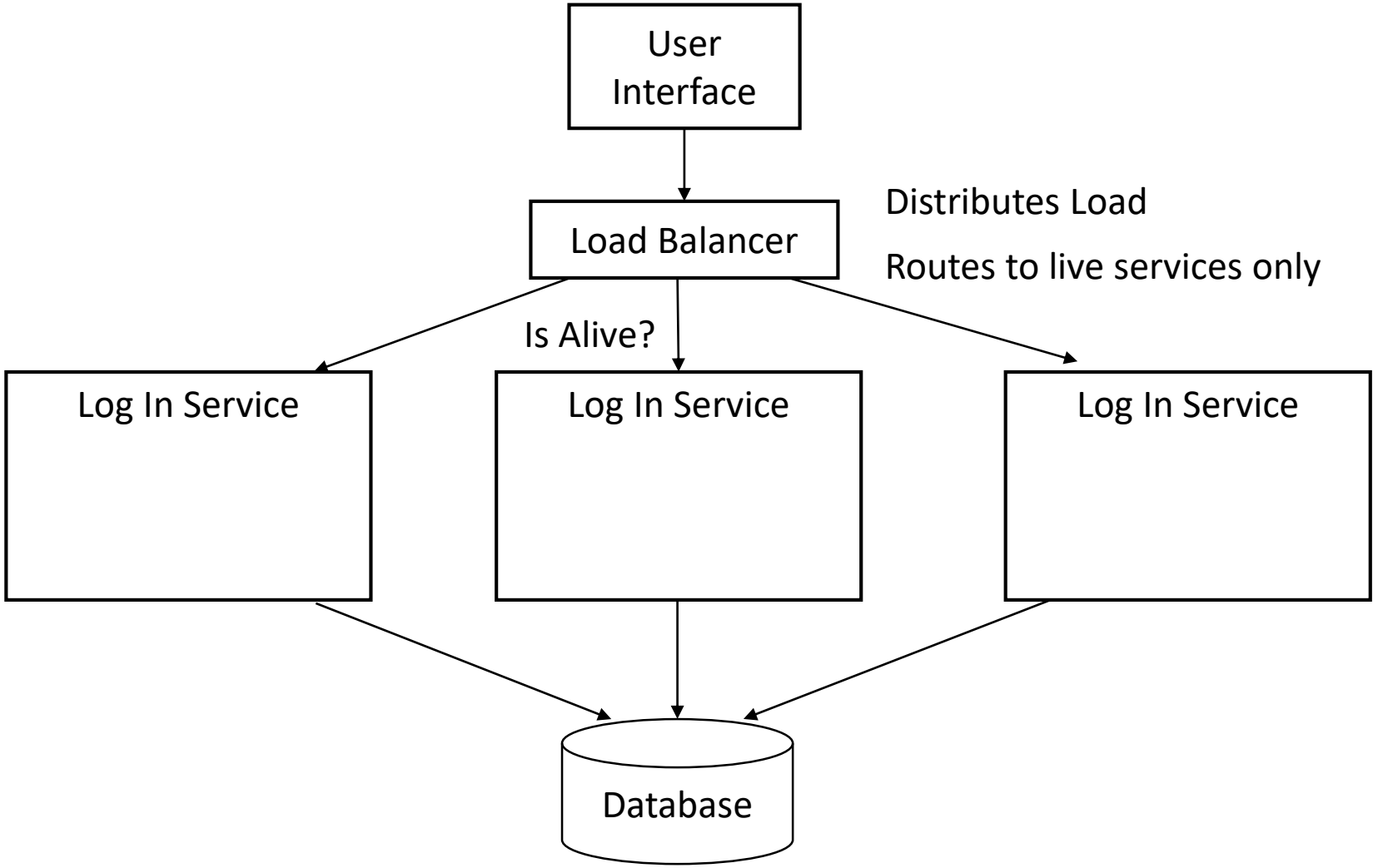
Scalability - A Reminder

- Grow and shrink as needed
- Scale Up vs Scale Out
- Scale Out is usually preferred

Redundancy - A Reminder

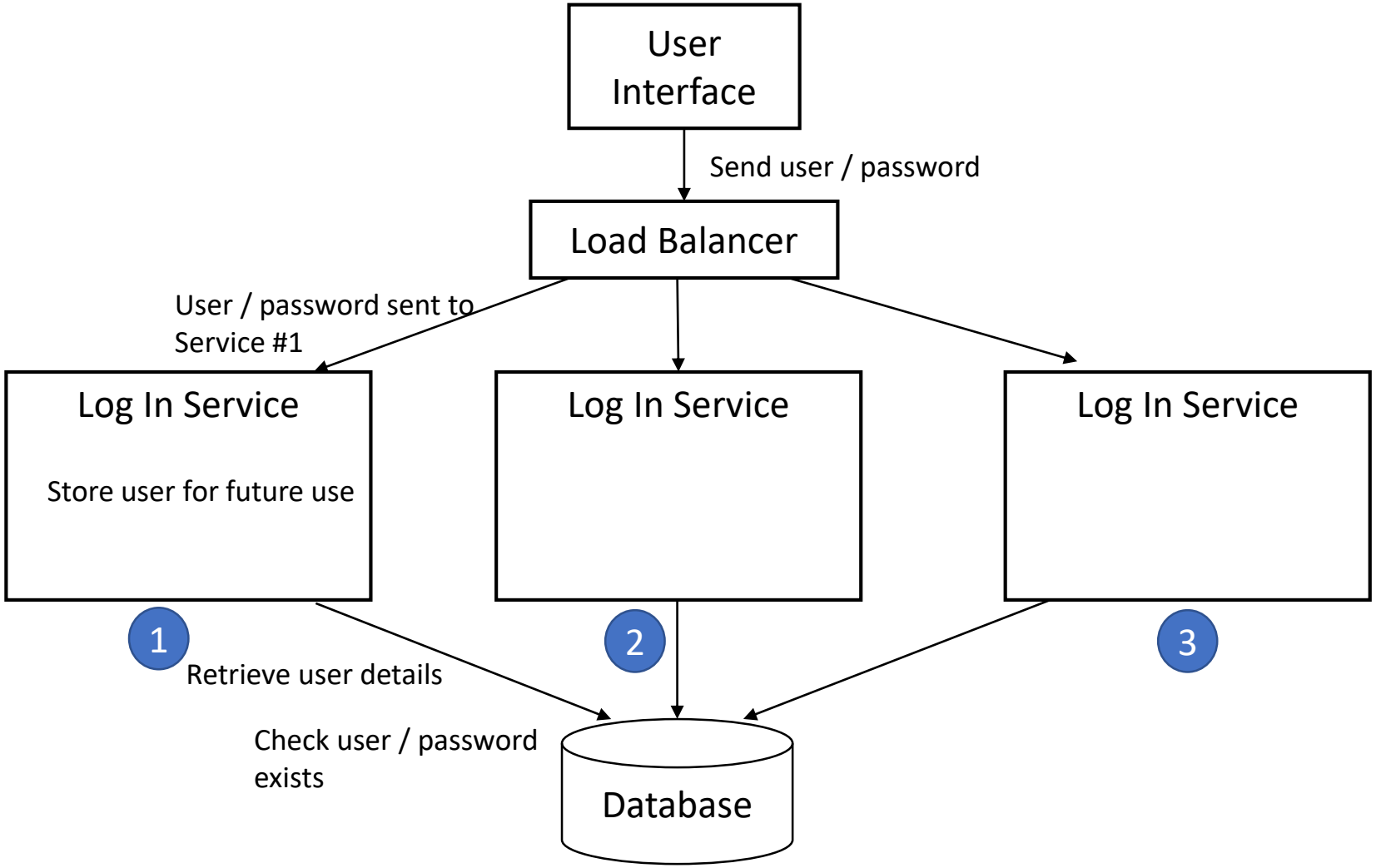
- Allows the system to function properly when resource is not working
- Example:
  - A system with more than one server
  - When a server goes down, the other continue working

Scalable & Redundant Architecture

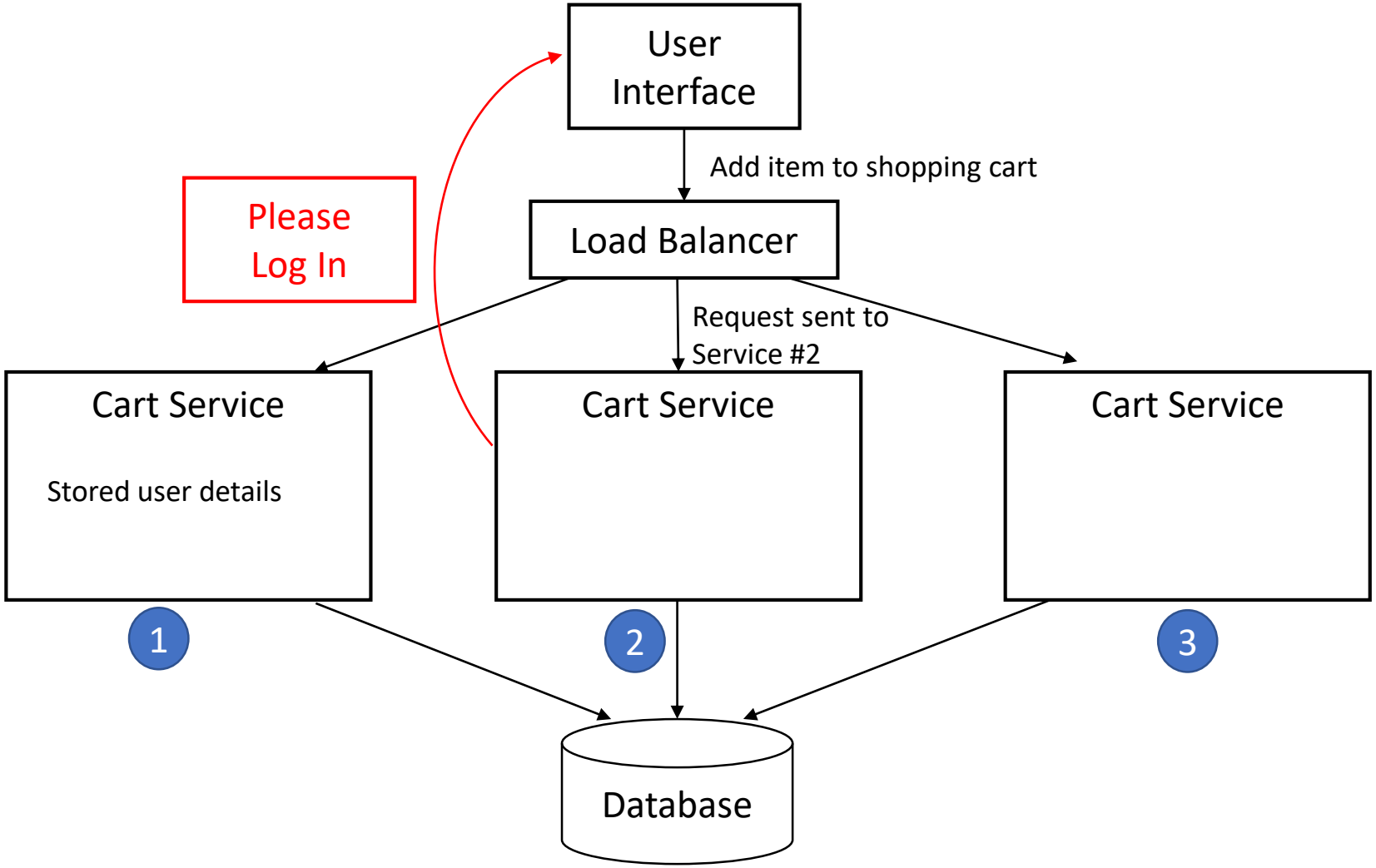


- Scalability
- Redundancy

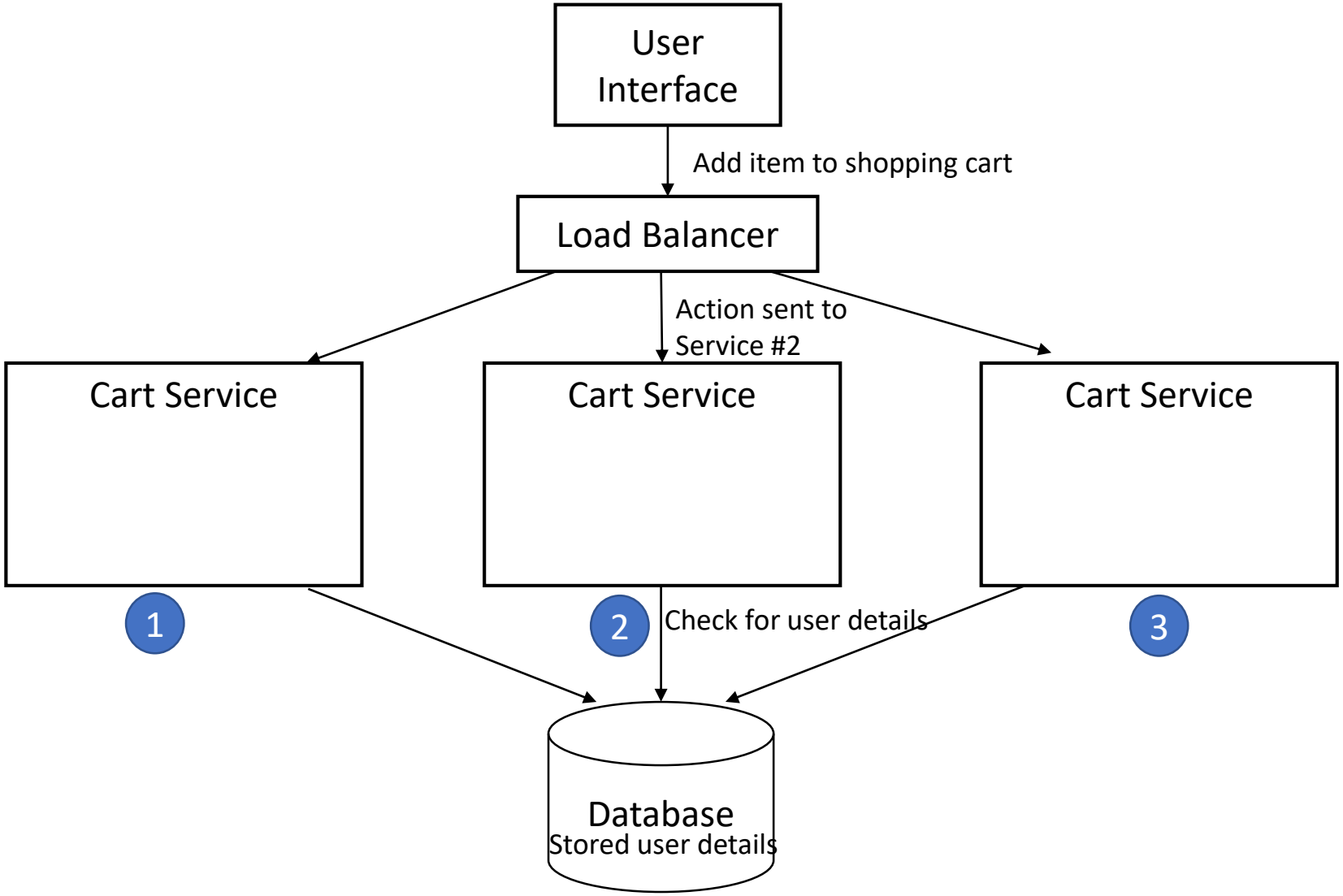
Stateful Example



Stateful Example



Stateless Example



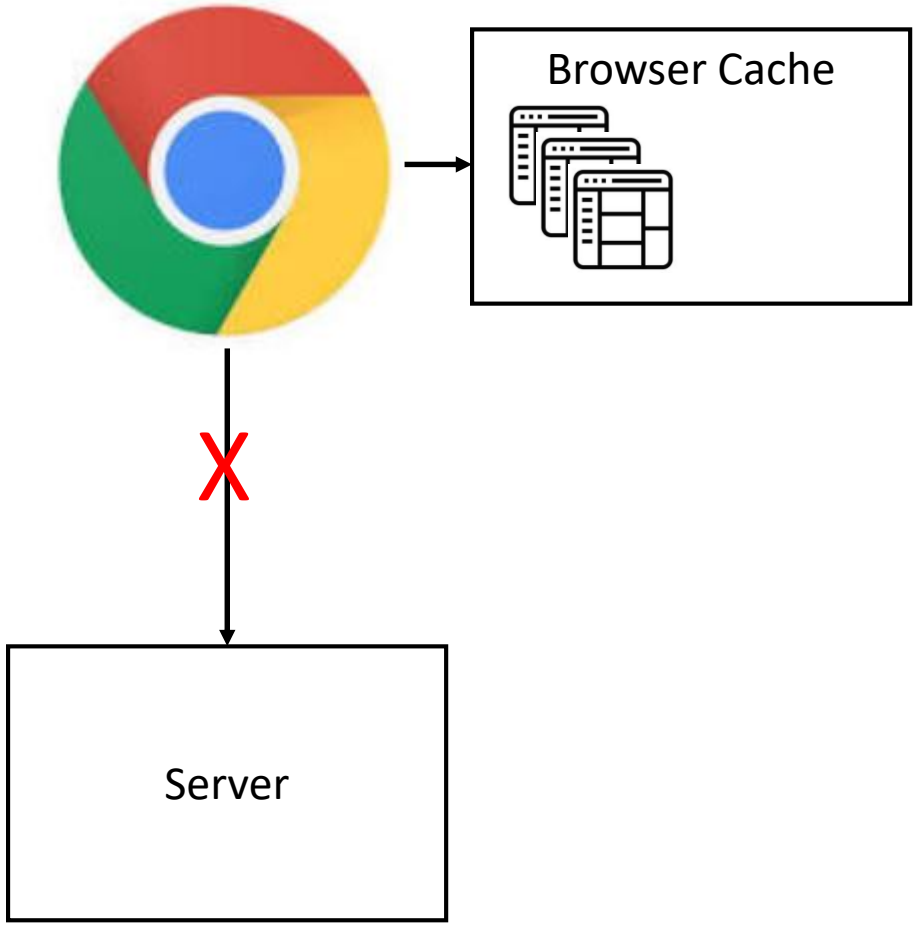
Stateless

- Always use stateless architecture
- Supports Scalability and Redundancy

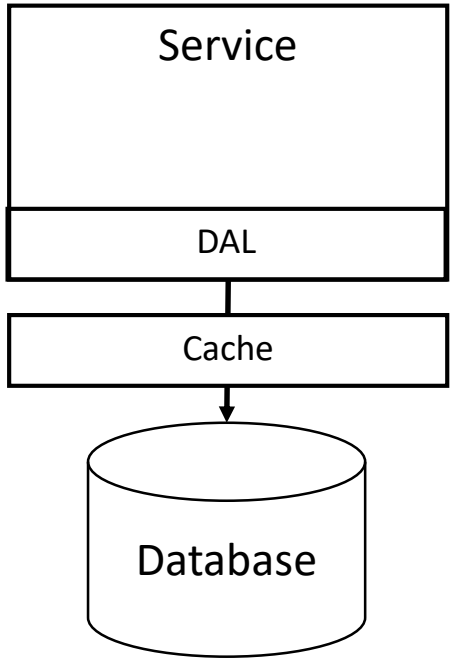
Caching

Bring data closer to its consumer so that its retrieval will be faster

Browser Cache



Service Cache



- Serialize to Objects

- Compile the SQL
- Retrieve the data

Cache Tradeoff

Single Source of Truth

	Reliability	Performance
Database	High Data is saved to disk	Good Data is retrieved from disk, then serialized
Cache	Poor Data is stored in memory	Excellent Data is retrieved from memory

What to Cache?

Cache should hold data that is
frequently accessed
and
rarely modified

What to Cache?

Cache should hold data that is

frequently accessed

and

rarely modified

Retrieval should be fast and easy for:

- Optimal user experience
- Minimum load

Using cache:

- Retrieval is fast (in-memory)
- UX is optimal

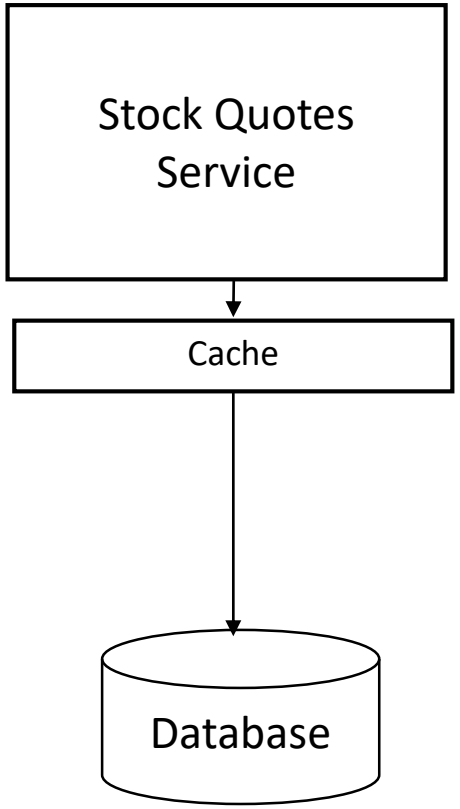
What to Cache?

Cache should hold data that is frequently accessed and

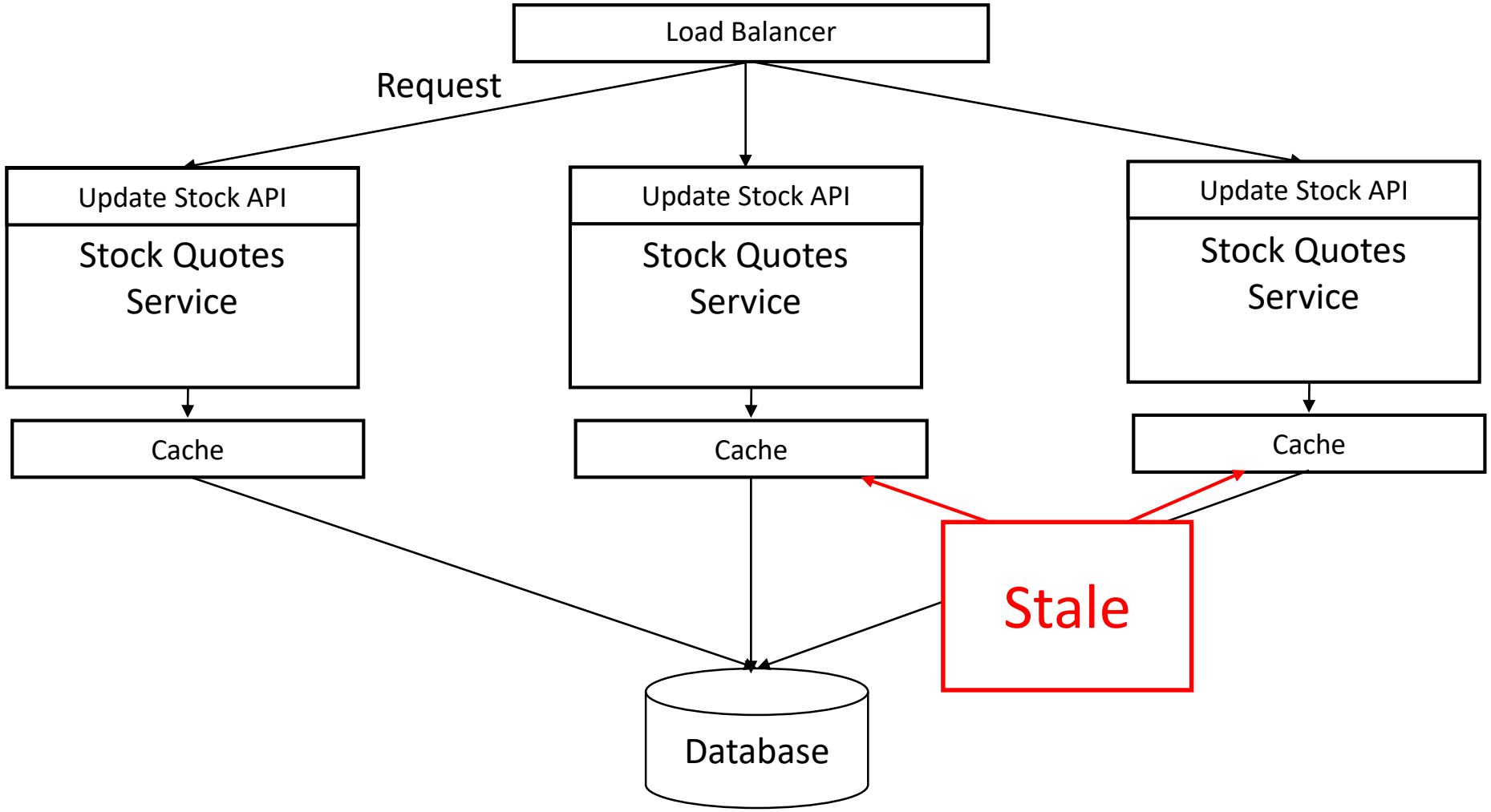
rarely modified

- Syncing cache and DB is a challenge
- When not in sync, leads to data corruption and bad user experience

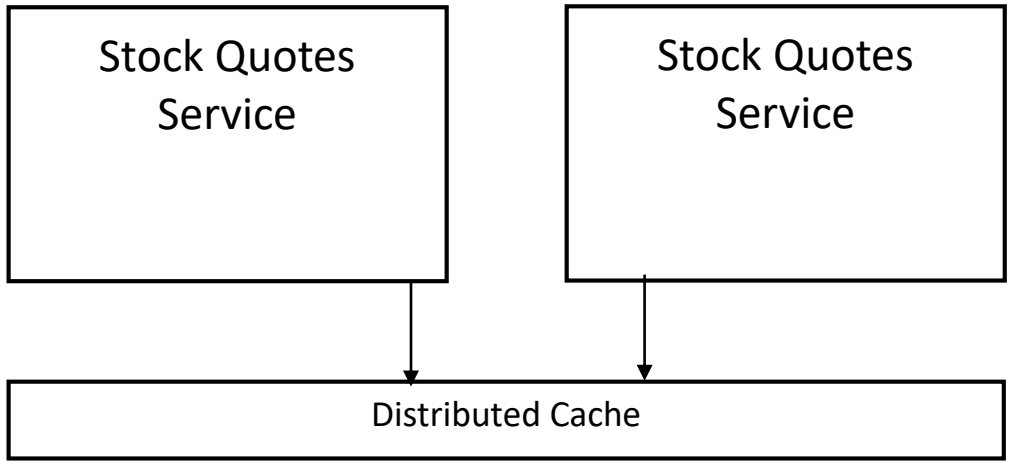
Example



Example


Load Balancer
Request
Update Stock API
Stock Quotes Service
Cache
Update Stock API
Stock Quotes Service
Cache
Update Stock API
Stock Quotes Service
Cache
Stale
Database

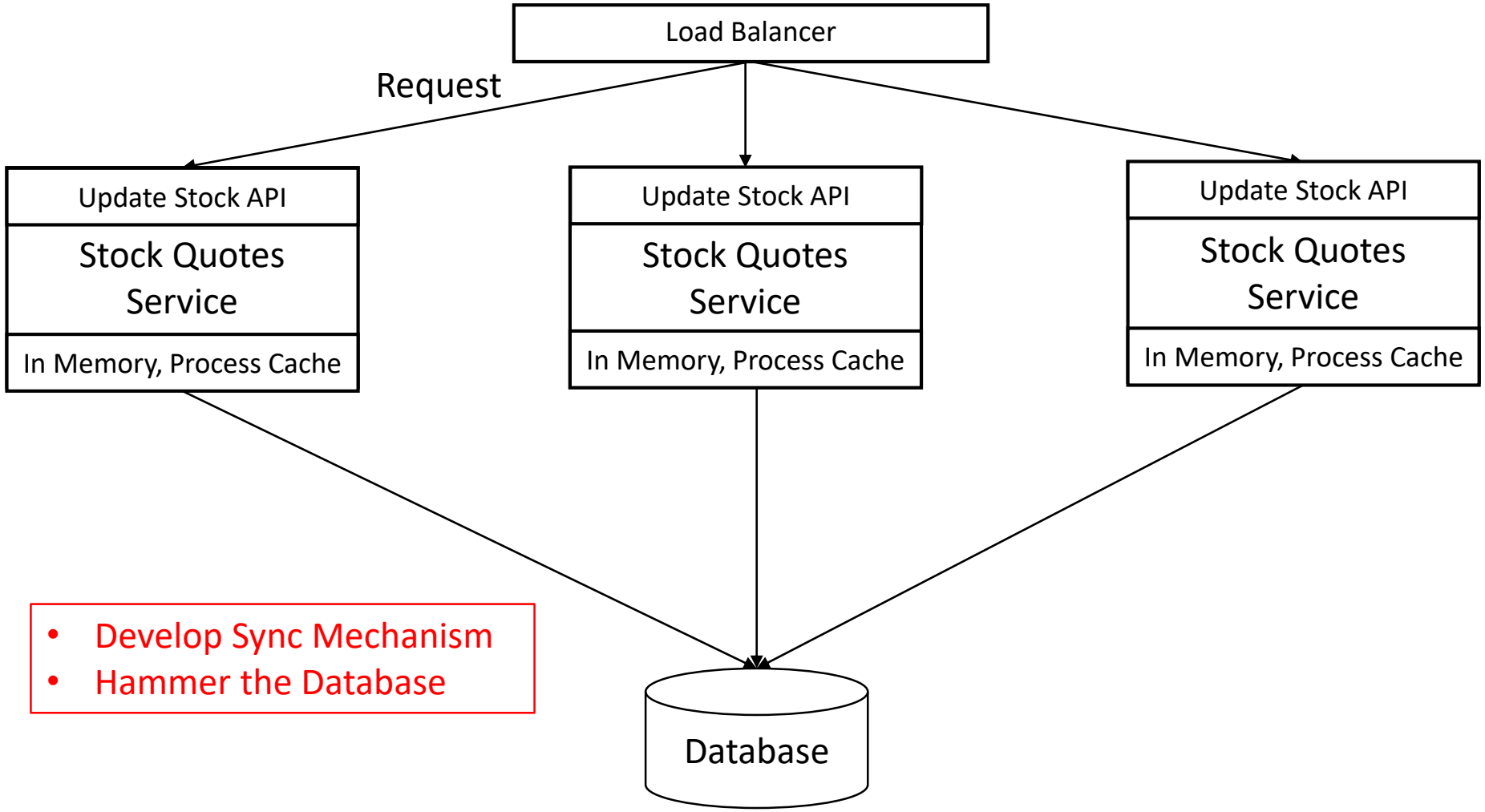
Cache Types



- Existing libraries
- Can be easily implemented using static concurrent collection
- Great performance
- Size is limited to the process's memory

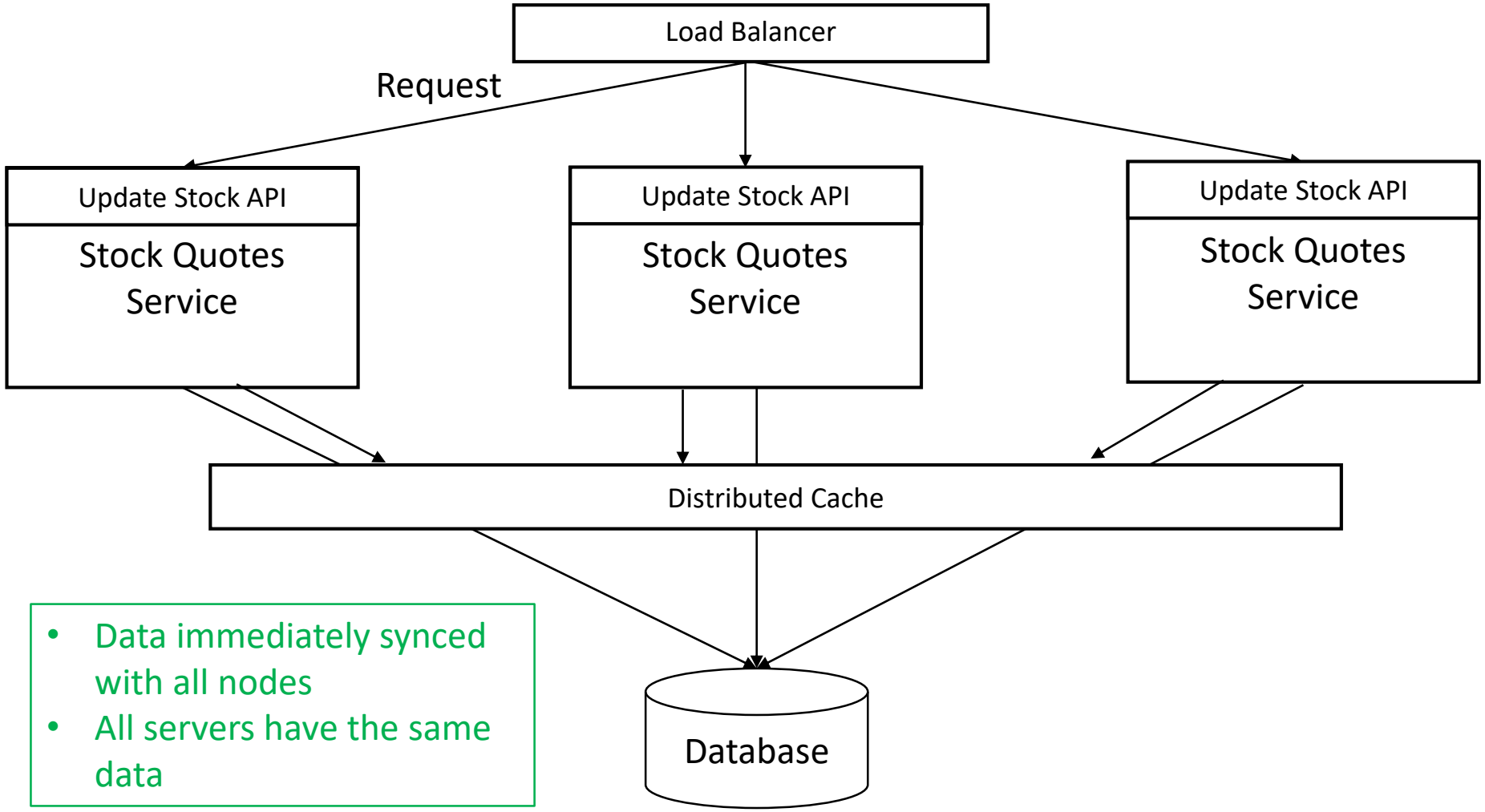
- External product
- Data is stored in separate process
- Provides interface for accessing the data
- Size virtually unlimited
- Auto nodes syncing
- Not the best performance
- Stores only primitive types

Example



- Develop Sync Mechanism
- Hammer the Database

Example



- Data immediately synced with all nodes
- All servers have the same data

Choosing Cache Type

Distributed Cache
Distribution among servers
Failover capabilities
Large Cache storage

Choosing Cache Type

Distributed Cache	In-Memory, In-Process Cache
Distribution among servers	Best performance possible
Failover capabilities	Store complex objects
Large Cache storage	

Choosing Cache Type

Distributed Cache	In-Memory, In-Process Cache
Distribution among servers	Best performance possible
Failover capabilities	Store complex objects
Large Cache storage	Very easy to use
Requires training and setup	

Messaging

Means of communication between the various services

Messaging

- Not just REST API
- Not Exclusive

Messaging Criteria

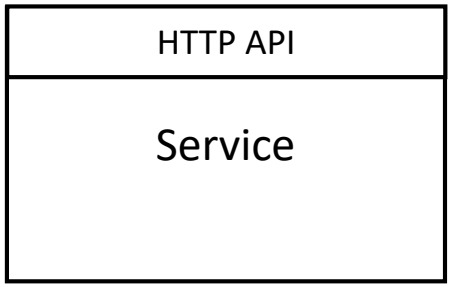
- Performance
- Message Size
- Execution Model
- Feedback & Reliability
- Complexity

REST API

- De-Facto Standard for HTTP-based systems

```
GET http://server/api/orders/17

POST http://server/api/orders
        {order data…}
```



REST API

Performance	Very Fast

REST API

Performance	Very Fast
Message Size	Same as HTTP protocol limitations (Usually Get -> 8KB, POST & PUT -> dozens MB)

REST API

Performance	Very Fast
Message Size	Same as HTTP protocol limitations (Usually Get -> 8KB, POST & PUT -> dozens MB)
Execution Model	Request / Response Great for quick, short actions, not suitable for long processes

REST API

Performance	Very Fast
Message Size	Same as HTTP protocol limitations (Usually Get -> 8KB, POST & PUT -> dozens MB)
Execution Model	Request / Response Great for quick, short actions, not suitable for long processes
Feedback & Reliability	Immediate feedback via Response Codes

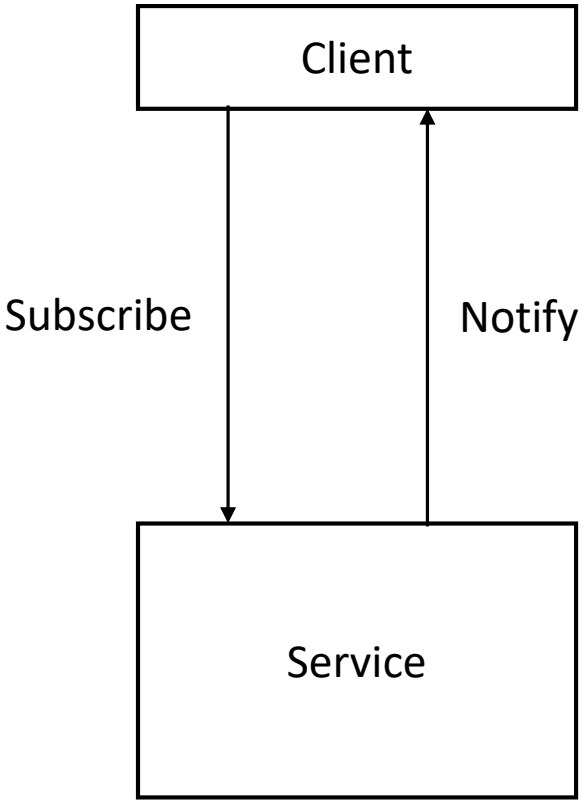
REST API

Performance	Very Fast
Message Size	Same as HTTP protocol limitations (Usually Get -> 8KB, POST & PUT -> dozens MB)
Execution Model	Request / Response Great for quick, short actions, not suitable for long processes
Feedback & Reliability	Immediate feedback via Response Codes
Complexity	Extremely easy to implement

REST API

Performance	Very Fast
Message Size	Same as HTTP protocol limitations (Usually Get -> 8KB, POST & PUT -> dozens MB)
Execution Model	Request / Response Great for quick, short actions, not suitable for long processes
Feedback & Reliability	Immediate feedback via Response Codes
Complexity	Extremely easy to implement
Useful For	Traditional Web Apps

HTTP Push Notifications



Real-Time Communication:

SignalR

socket.io

HTTP Push Notifications

- Uses advanced web techniques (ie. Web Sockets)
- Very popular in chats

HTTP Push Notification

Performance	Excellent

HTTP Push Notification

Performance	Excellent
Message Size	Limited Usually no more than a few KB

HTTP Push Notification

Performance	Excellent
Message Size	Limited Usually no more than a few KB
Execution Model	Web Socket connection / Long Polling

HTTP Push Notification

Performance	Excellent
Message Size	Limited Usually no more than a few KB
Execution Model	Web Socket connection / Long Polling
Feedback & Reliability	None (Fire & Forget) (Can be implemented, quite complex)

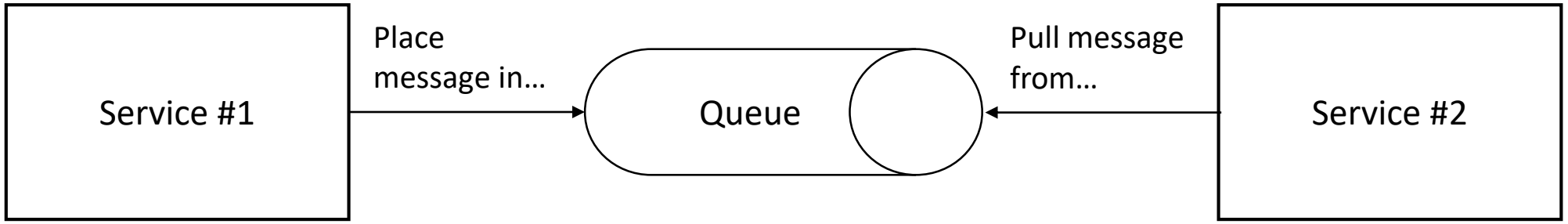
HTTP Push Notification

Performance	Excellent
Message Size	Limited Usually no more than a few KB
Execution Model	Web Socket connection / Long Polling
Feedback & Reliability	None (Fire & Forget) (Can be implemented, quite complex)
Complexity	Extremely easy to implement

HTTP Push Notification

Performance	Excellent
Message Size	Limited Usually no more than a few KB
Execution Model	Web Socket connection / Long Polling
Feedback & Reliability	None (Fire & Forget) (Can be implemented, quite complex)
Complexity	Extremely easy to implement
Useful For	Chat, Monitoring

Queue



- Messages will be handled once and only once
- Messages will be handled in order

Queue

Performance	Not so good (Push / Poll, DB Persistence)

Queue

Performance	Not so good (Push / Poll, DB Persistence)
Message Size	Technically almost not limited But use small messages

Queue

Performance	Not so good (Push / Poll, DB Persistence)
Message Size	Technically almost not limited But use small messages
Execution Model	Polling

Queue

Performance	Not so good (Push / Poll, DB Persistence)
Message Size	Technically almost not limited But use small messages
Execution Model	Polling
Feedback & Reliability	Very reliable

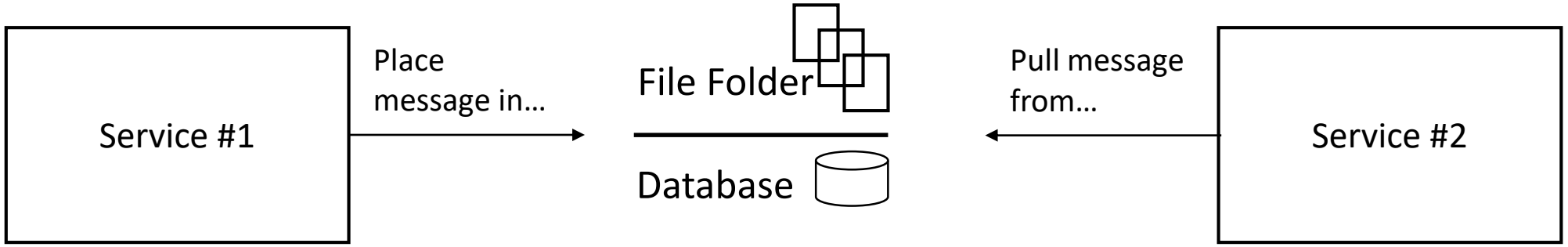
Queue

Performance	Not so good (Push / Poll, DB Persistence)
Message Size	Technically almost not limited But use small messages
Execution Model	Polling
Feedback & Reliability	Very reliable
Complexity	Requires training and setup

Queue

Performance	Not so good (Push / Poll, DB Persistence)
Message Size	Technically almost unlimited But use small messages
Execution Model	Polling
Feedback & Reliability	Very reliable
Complexity	Requires training and setup
Useful For	Complex system with lots of data, when order and reliability are top priority

File-based & Database-based



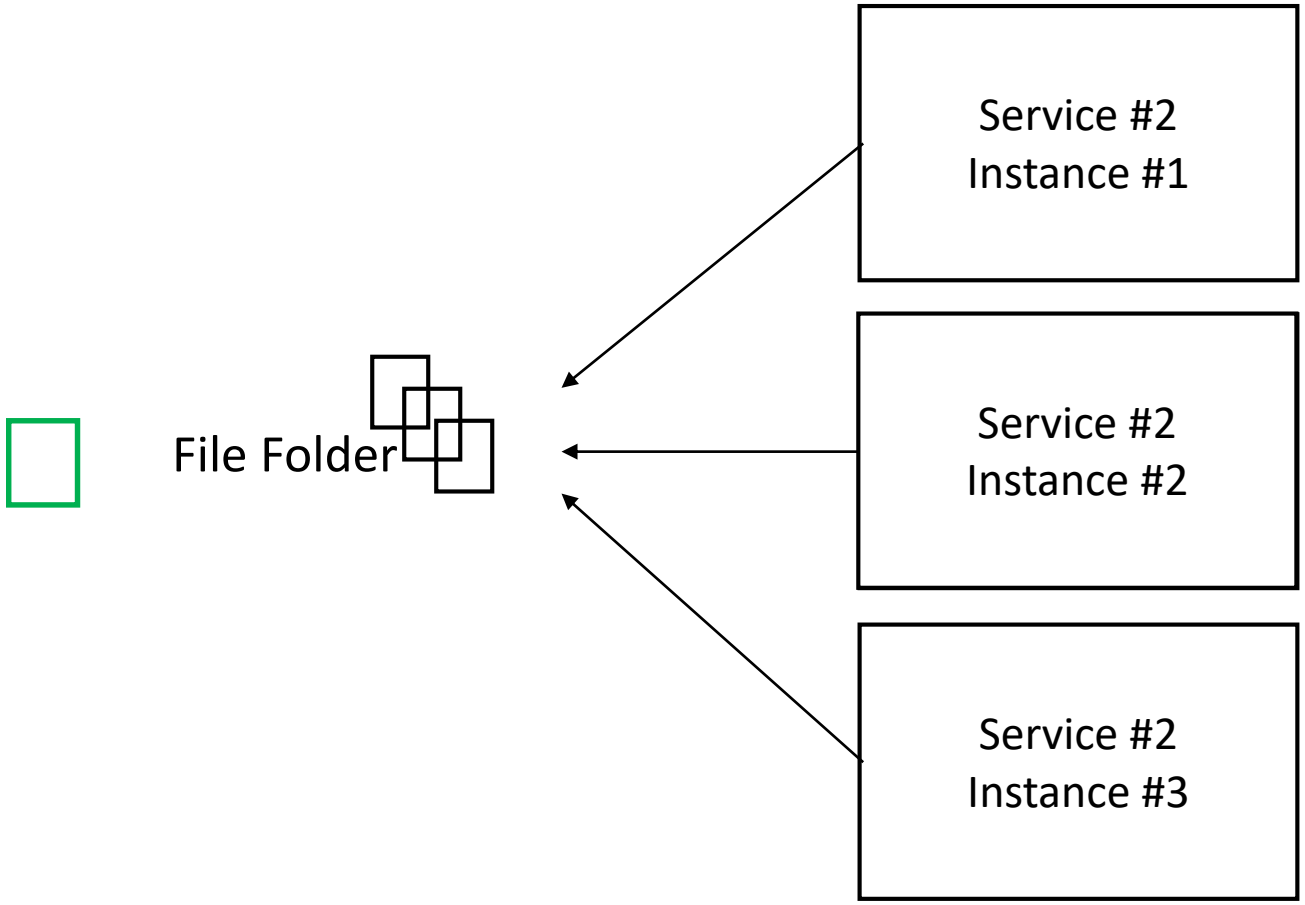
File-based & Database-based

Performance	Not so good (Push / Poll, DB Persistence)
Message Size	Unlimited
Execution Model	Polling
Feedback & Reliability	Very reliable
Complexity	Requires training and setup
Useful For	Complex system with lots of data. Better use queues

File-based & Database-based

Problems:

1. File locked
2. Duplicate processing

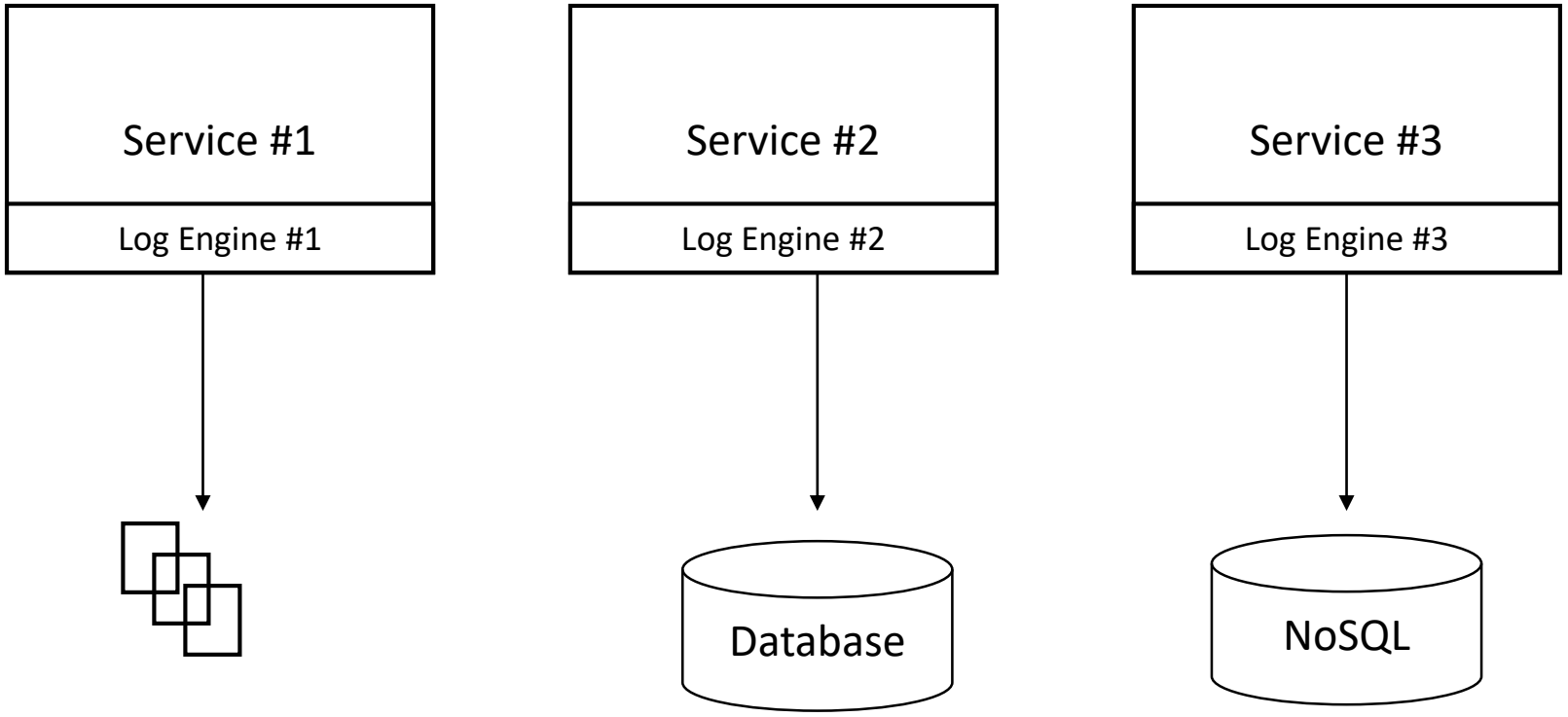


Messaging Summary

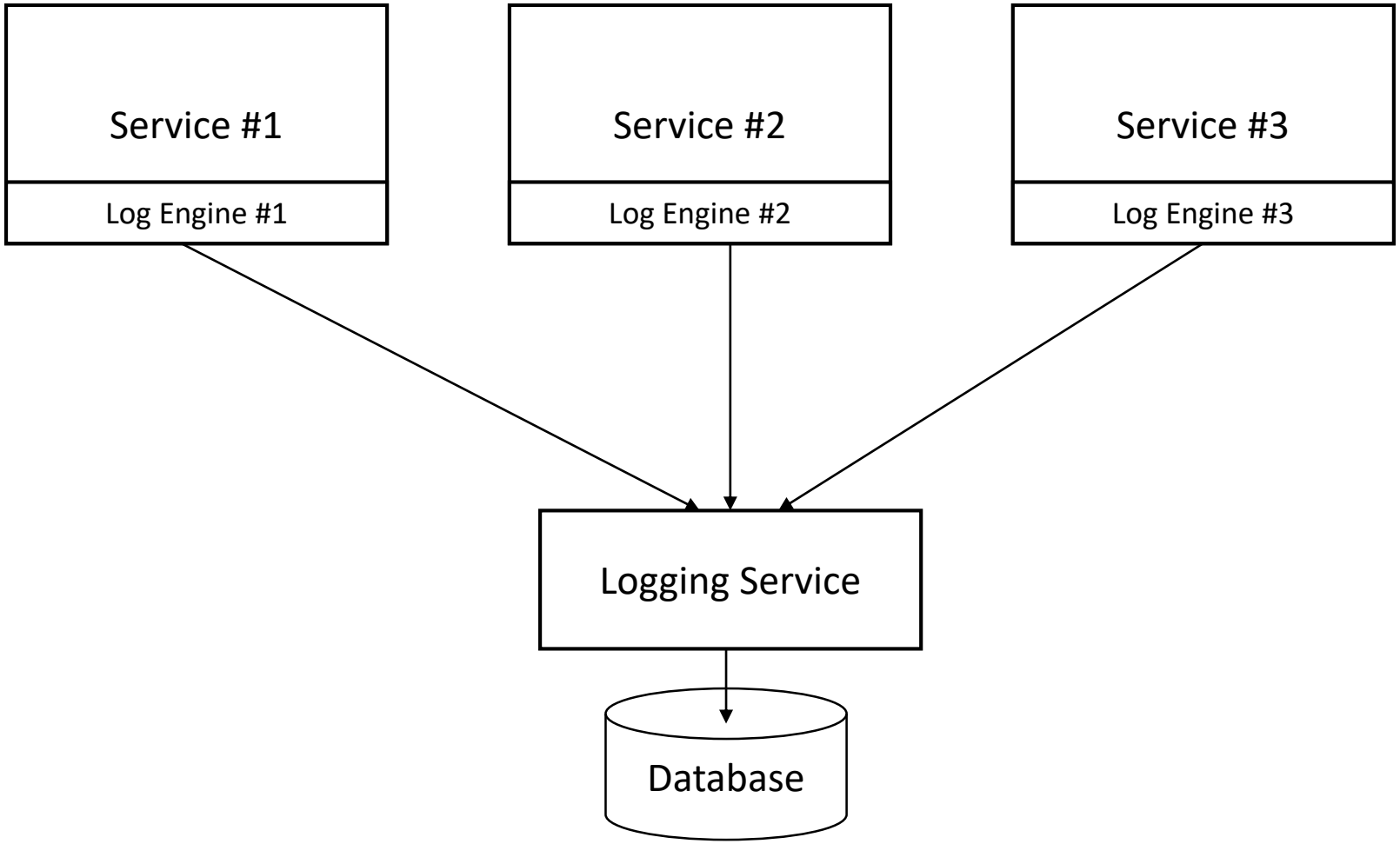
	REST API	HTTP Push	Queue	File- & DB-based
Performance	Very fast	Excellent	Not so good	Not so good
Message Size	Same as HTTP limitations	Limited	Technically unlimited	Unlimited
Execution Model	Request / Response	Web Socket / Long Polling	Polling	Polling
Feedback & Reliability	Immediate feedback	None	Very reliable	Very reliable
Complexity	Extremely easy	Extremely easy	Requires training and setup	Requires training and setup
Useful For	Traditional web apps	Chat, Monitoring	Complex system with lots of data, where order and reliability are top priority	Complex system with lots of data. Better use queues

Logging & Monitoring

Central Logging Service



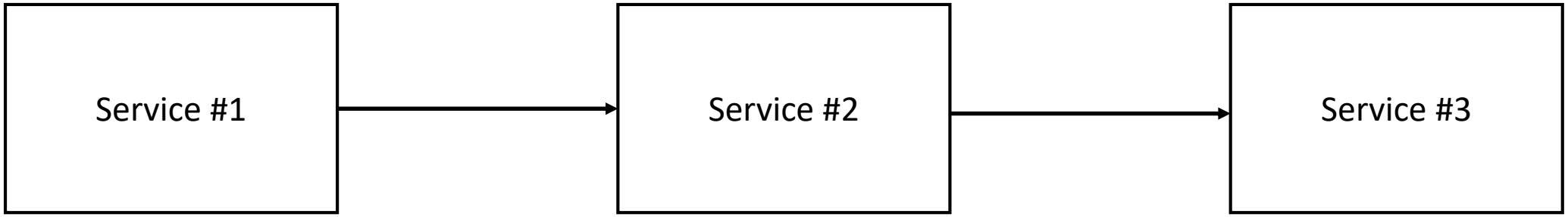
Central Logging Service



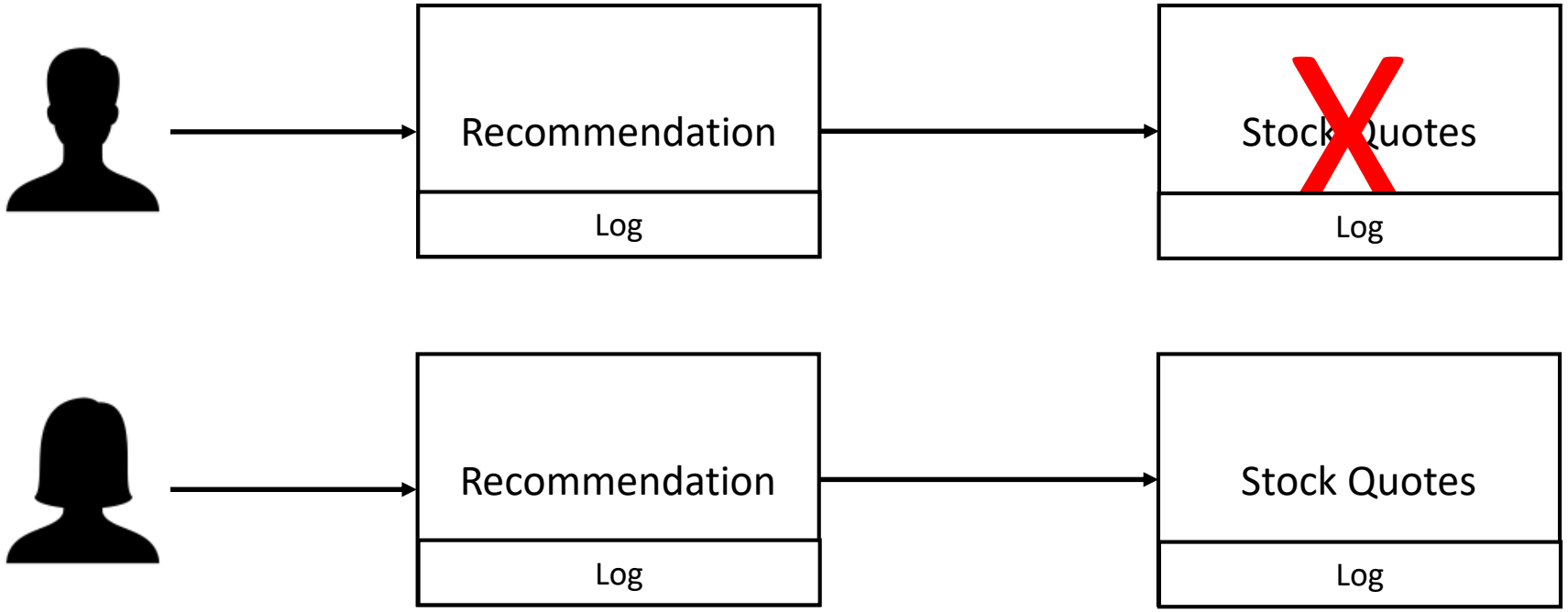
Central Logging Service

- Implementation:
  - API
  - Watch folders

Correlation ID



Correlation ID



CorID 17

CorID 18

CorID 17

CorID 18

System Architecture- Summary

- Use these concepts to design a fast ,secure, reliable and easy to maintain system
- Make the choice as early as possible
- They are not exclusive…
- But they are the most important ones