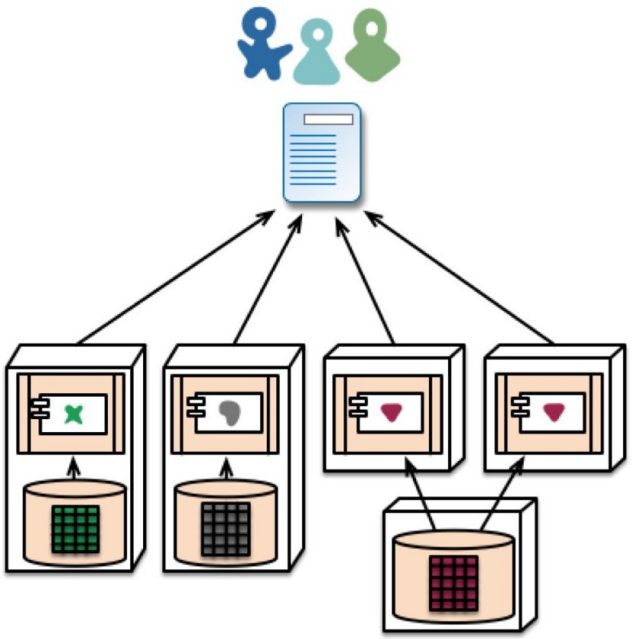
Microservices external API and data management patterns

Nikola Zivkov
@nikolazivkov
Seavus

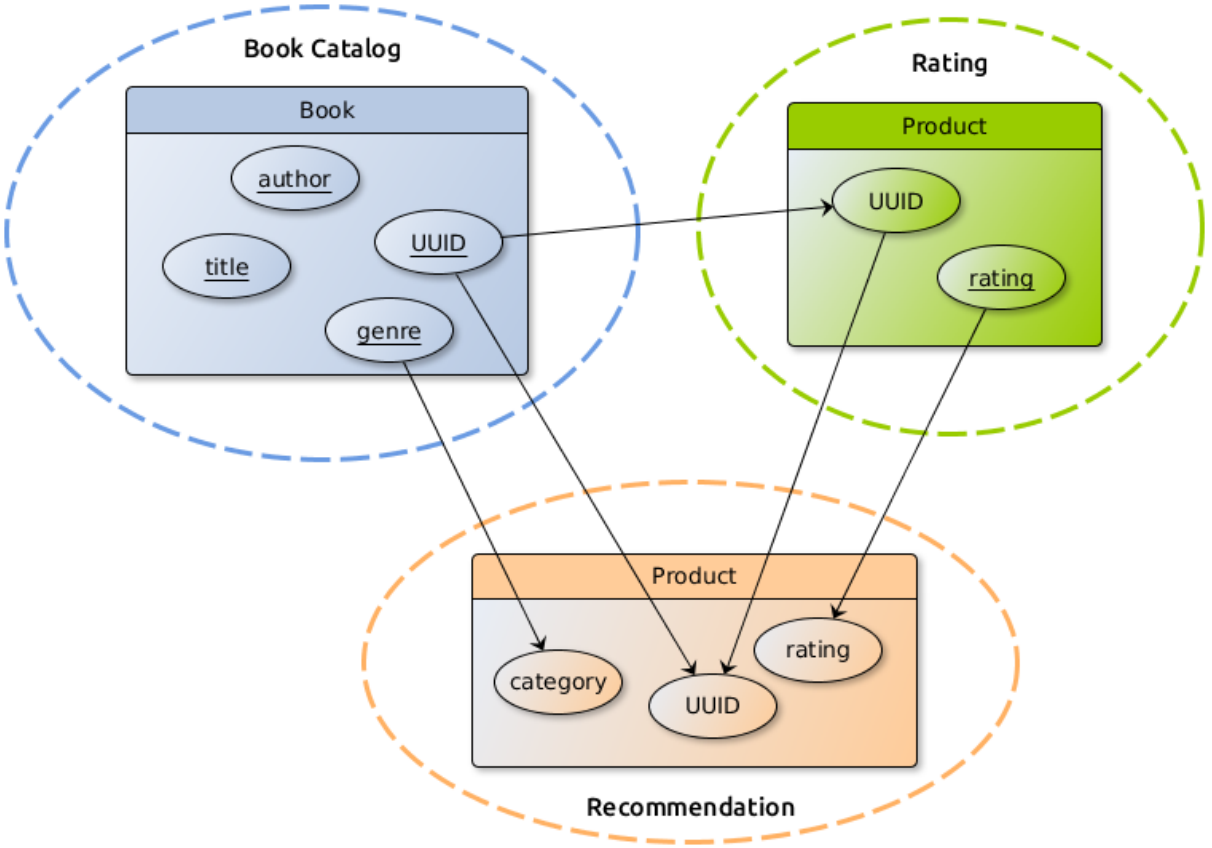
Setting up the context

- Microservices
  - Small (solves one problem)
  - Running in own process
  - Individually deployable
  - Individually scalable
  - Database per service
  - Etc.

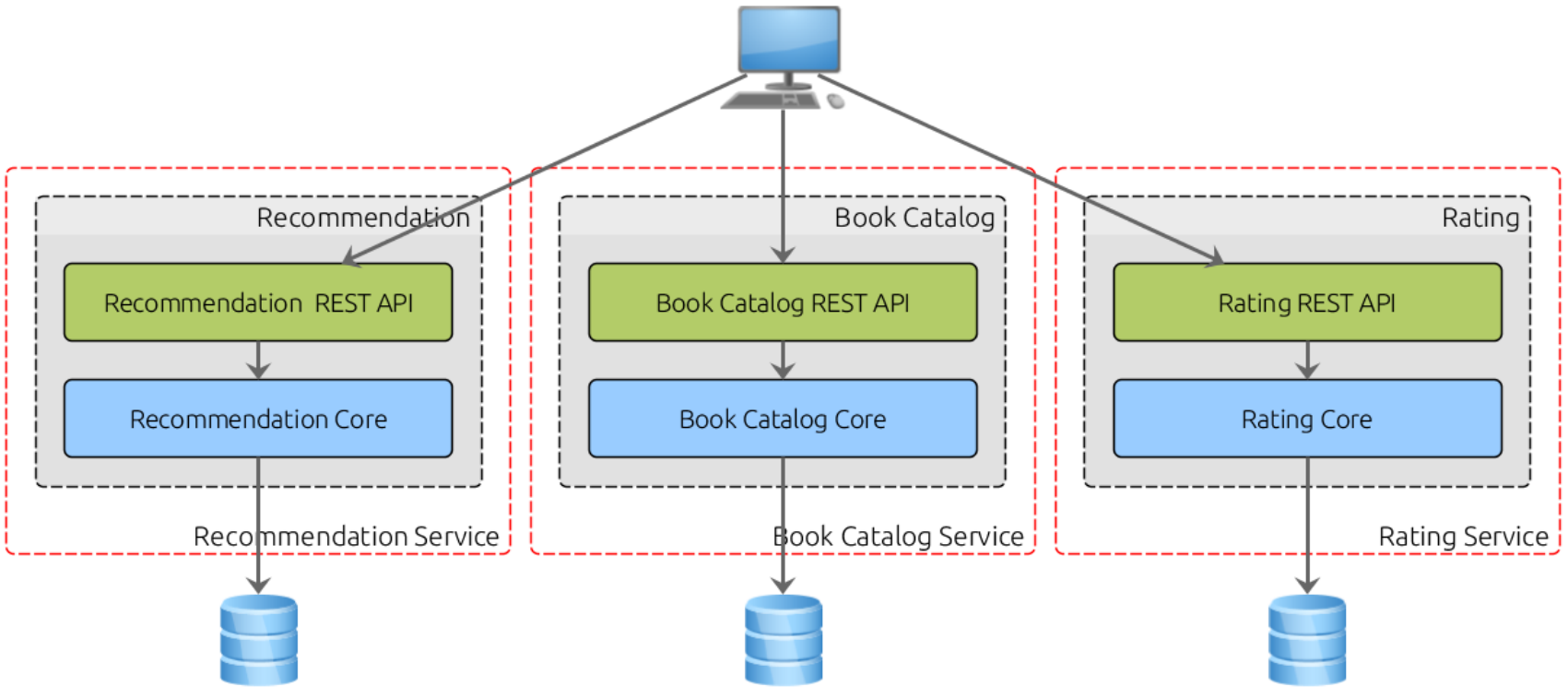


Bookstore application

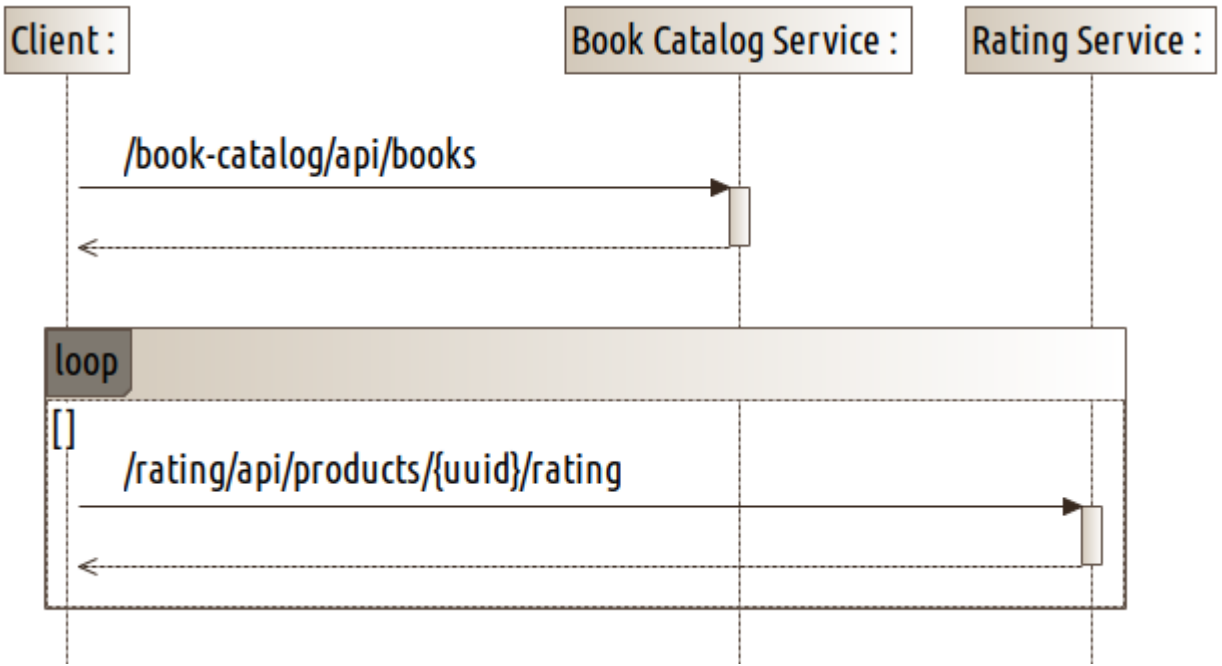
Bookstore context map



Direct connection



Direct connection communication flow



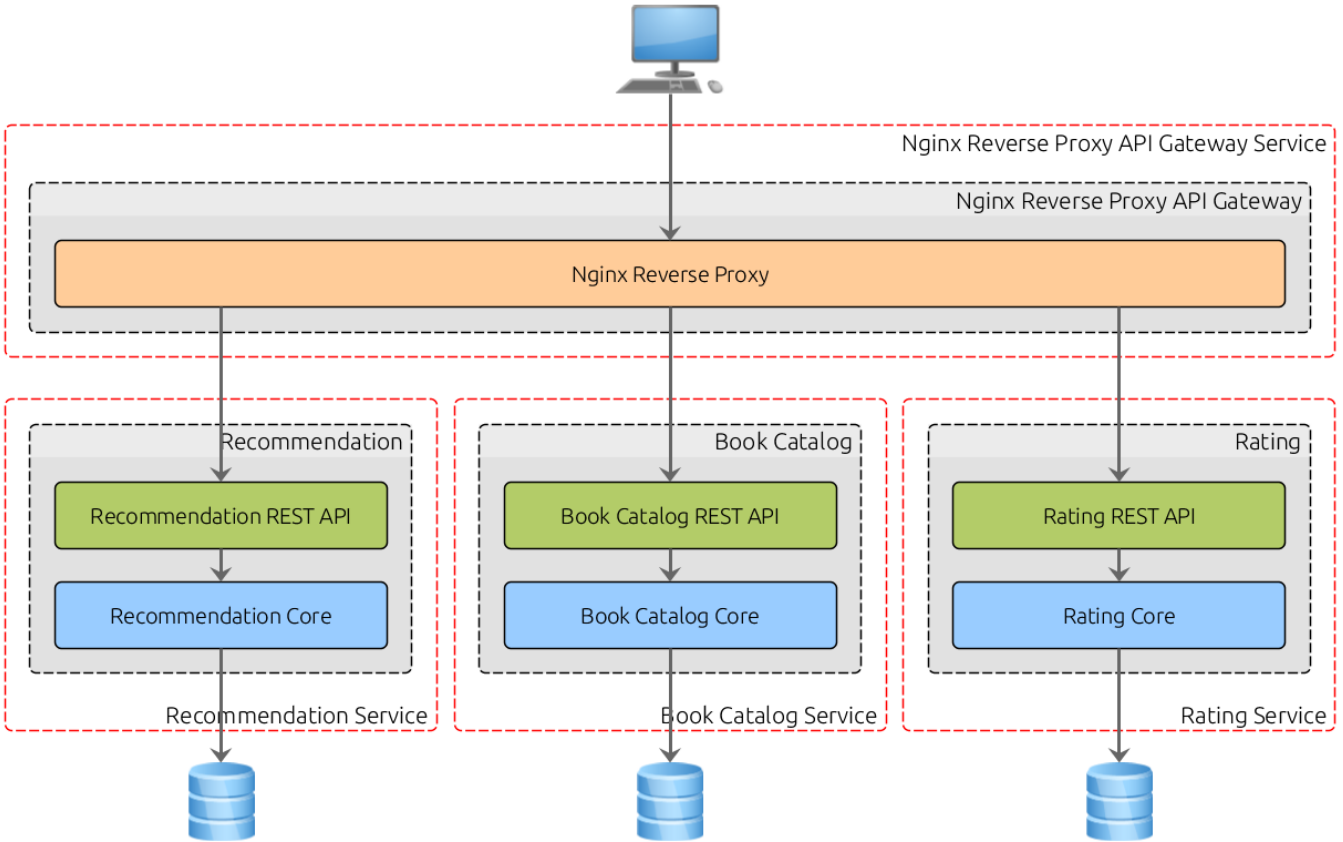
Challenges (part 1)

- The number of service instances and their locations (host+port) changes dynamically.
- Partitioning into services can change over time and should be hidden from clients.

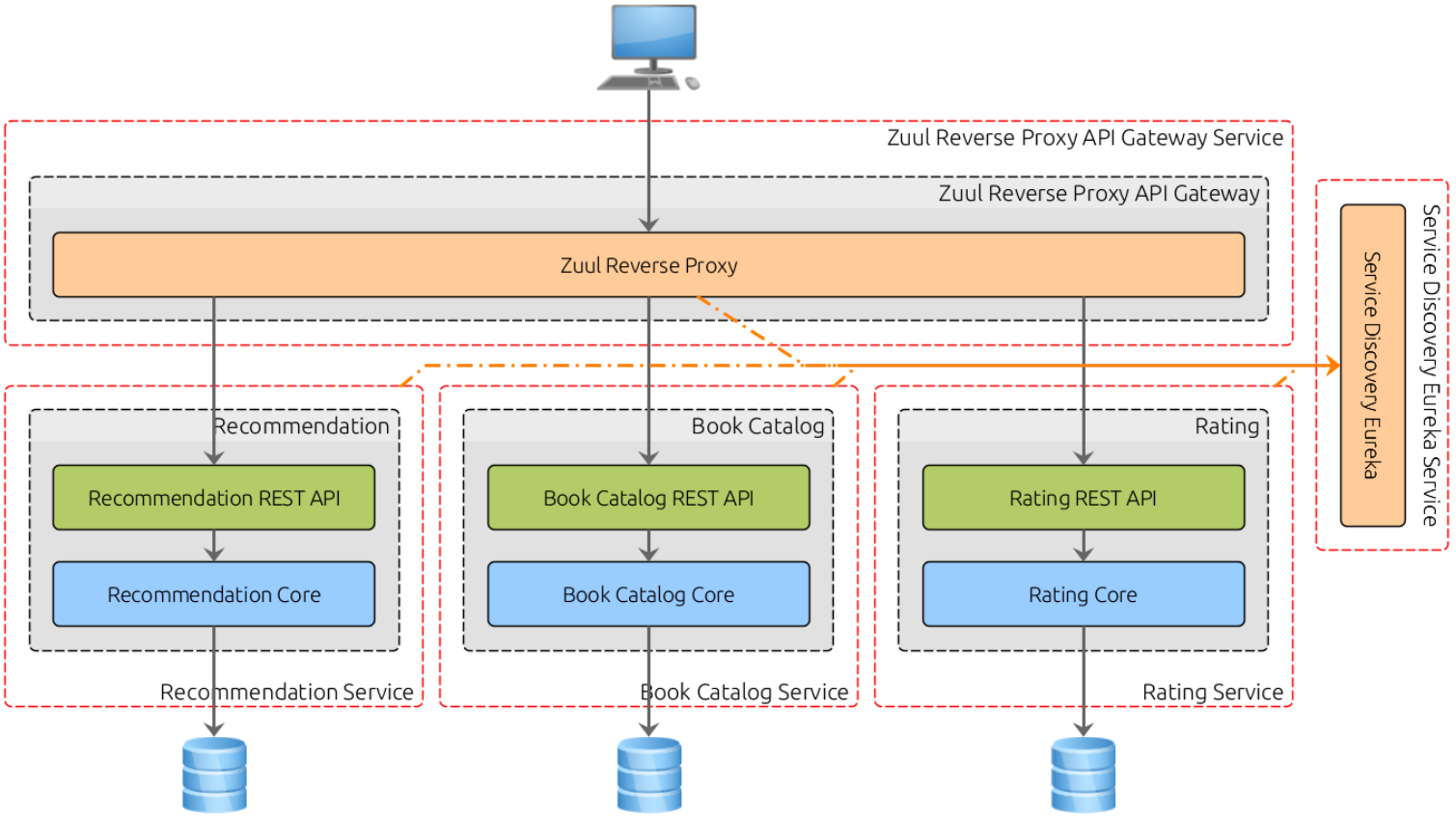
Solution

Reverse proxy API gateway

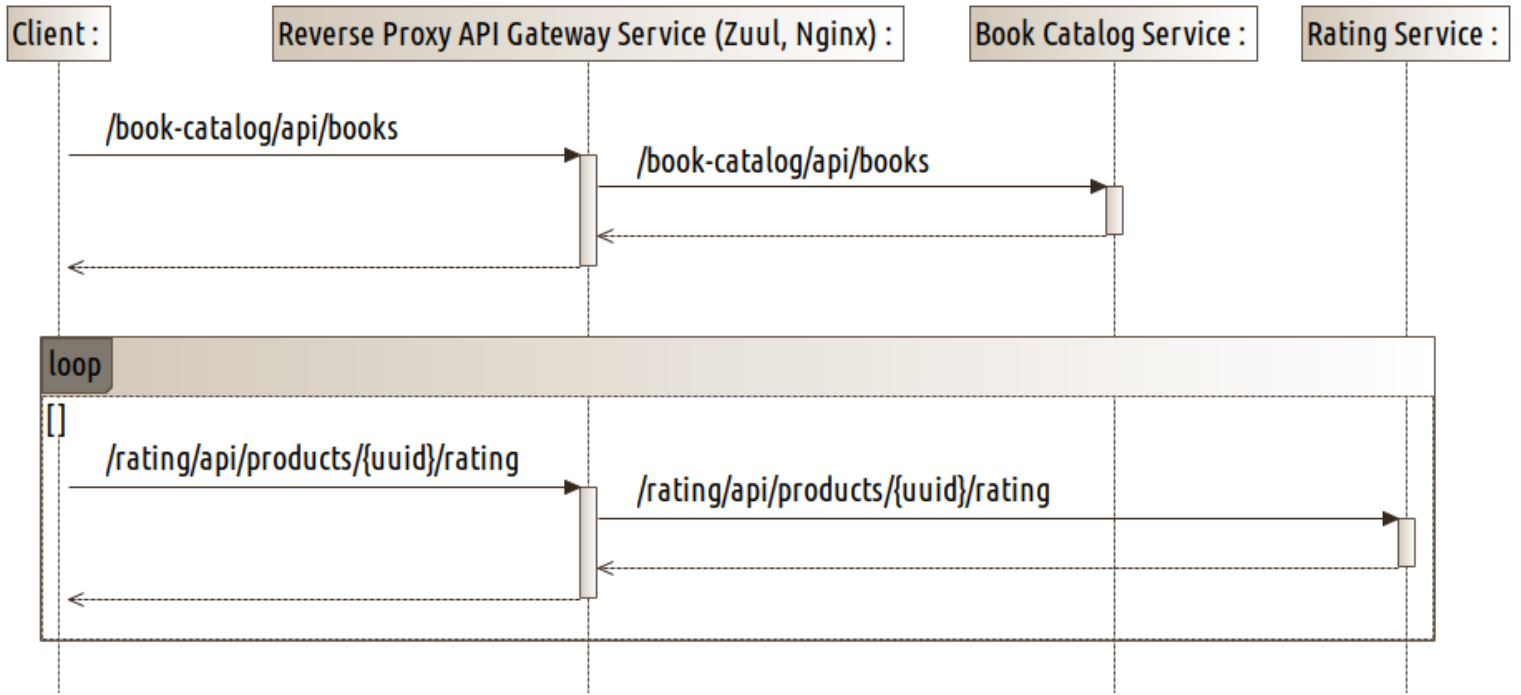
Nginx reverse proxy API gateway



Zuul reverse proxy API gateway



Reverse proxy API gateway communication flow



Reverse proxy API gateway summary

- Single point of entry for clients.
- Multiple service instances are handled by load balancing through service discovery (Spring Cloud Ribbon).
- Other implementations: AWS API Gateway, etc.

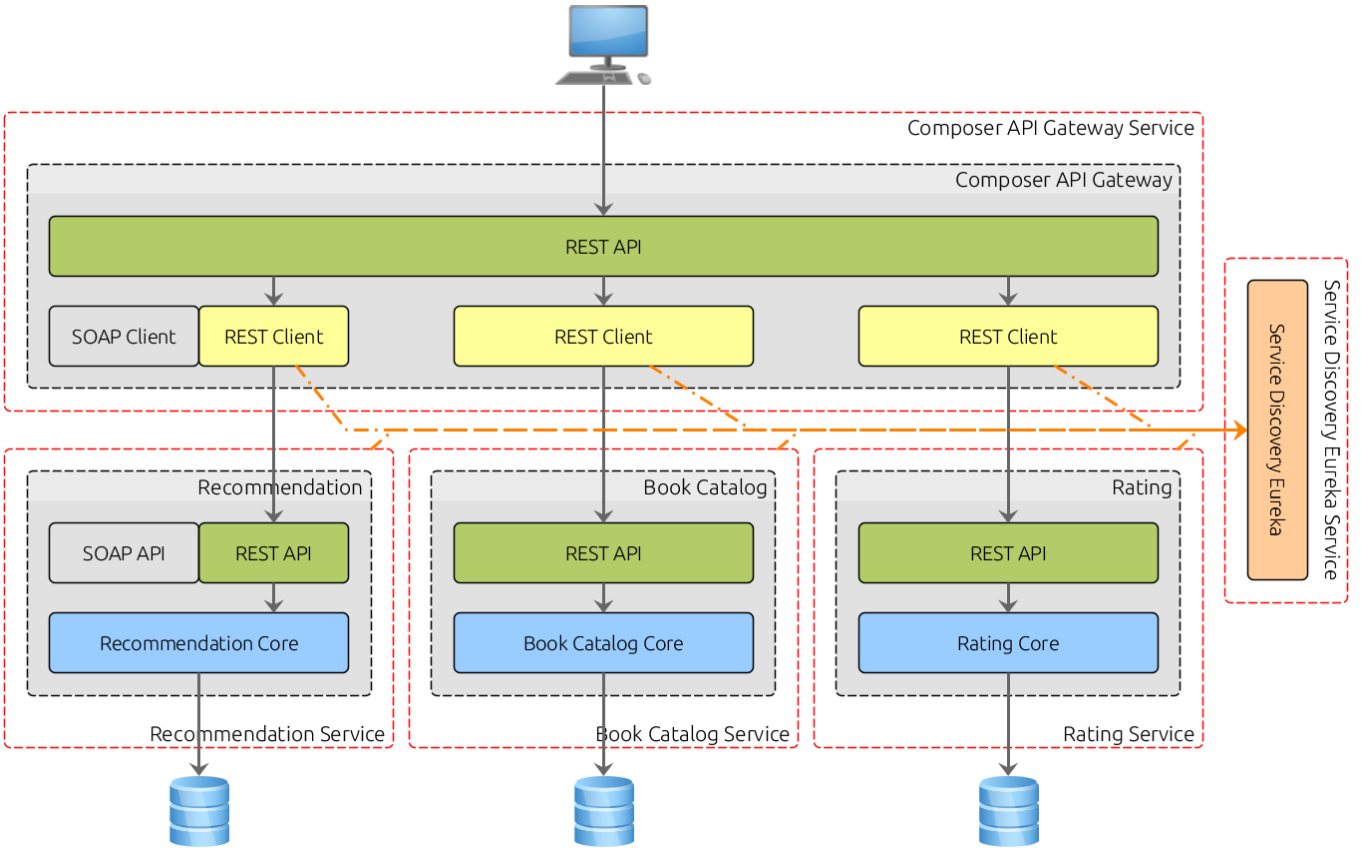
Challenges (part 2)

- Microservices typically offer fine-grained APIs resulting in multiple calls between the client and the server (chatty APIs). This is especially bad when network calls are expensive such as the case with mobile clients.
- Services might be using diverse set of protocols, some of which might not be web friendly (e.g. SOAP).

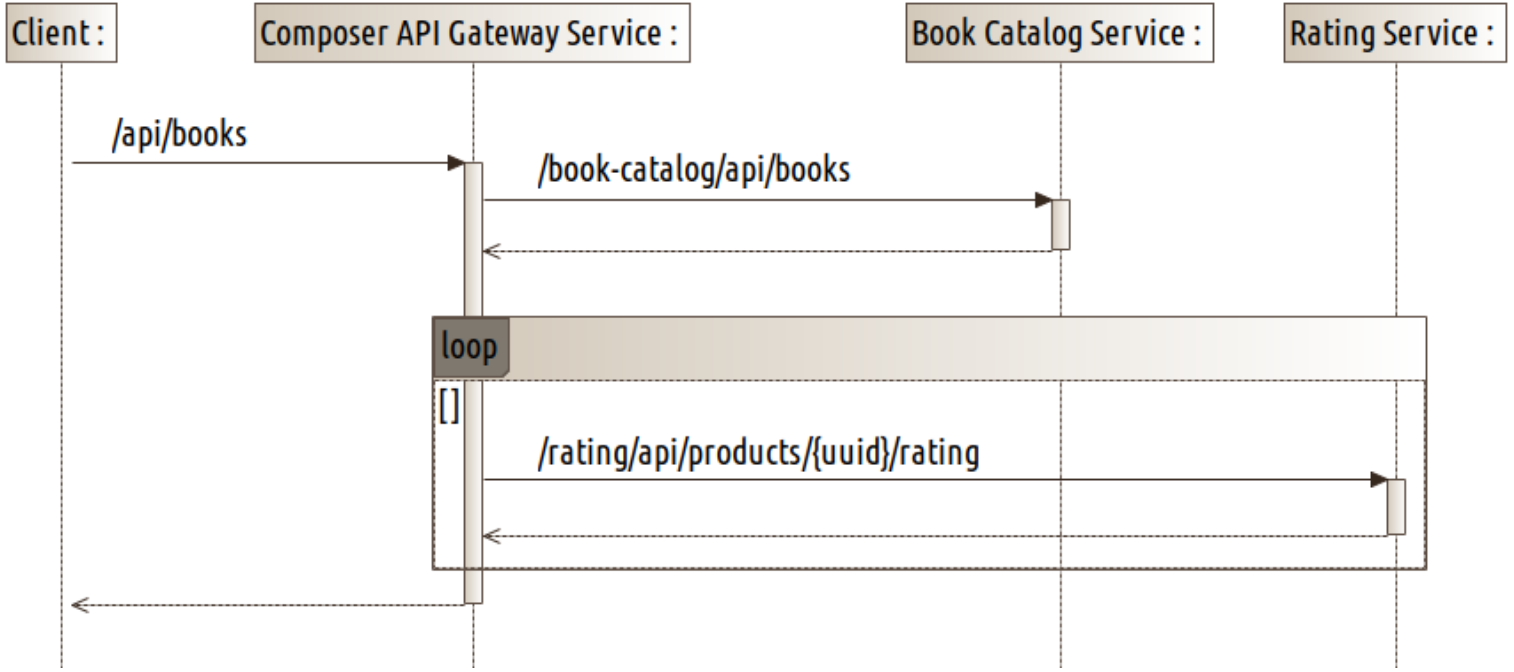
Solution

Composer API gateway

Composer API gateway



Composer API gateway communication flow



Composer API gateway summary

- Single client request is fanned out to multiple microservices.
- Responses are joined in memory before returned to the client.

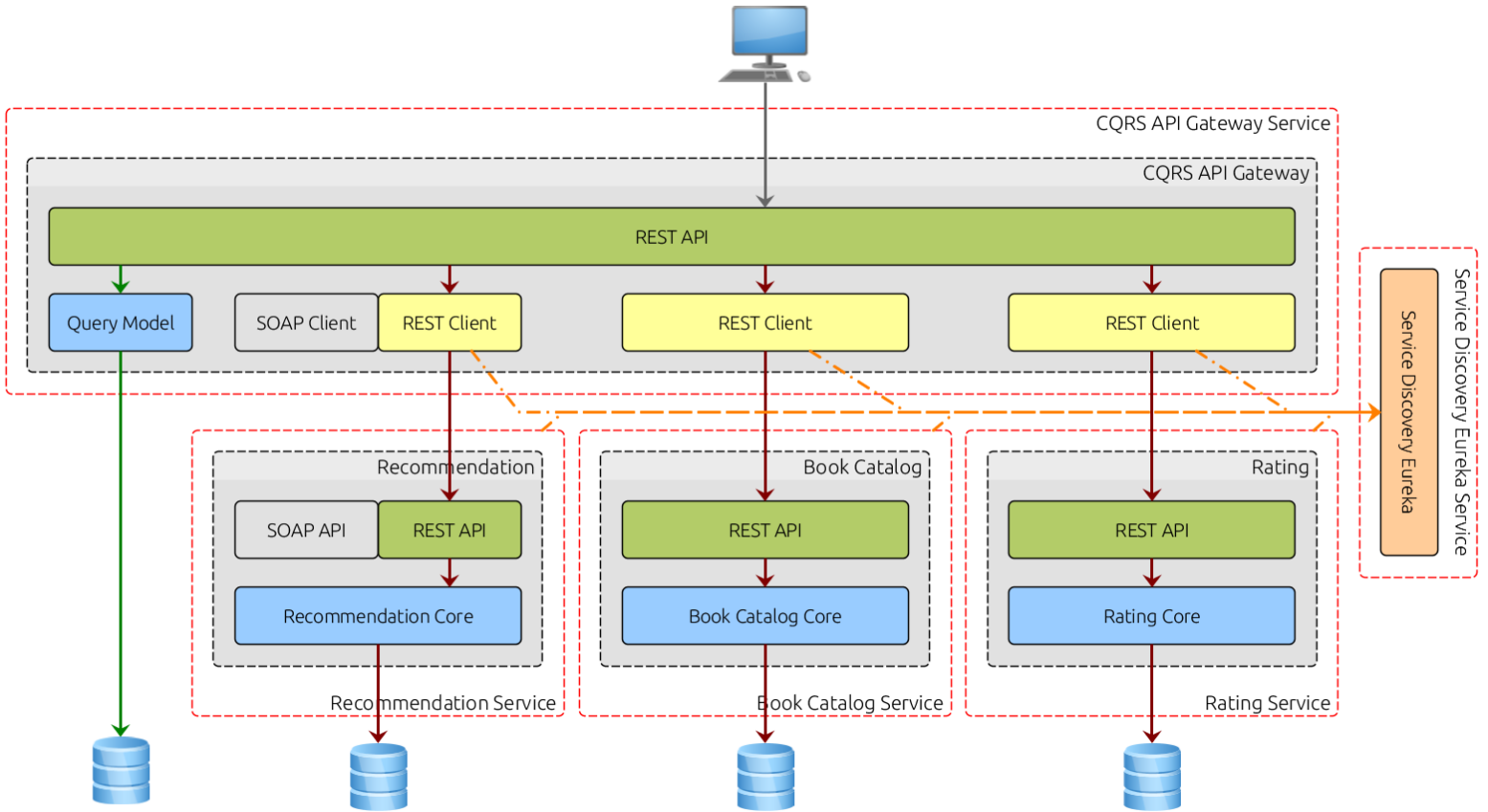
Challenges (part 3)

- Not straightforward to implement queries that join data from multiple services.

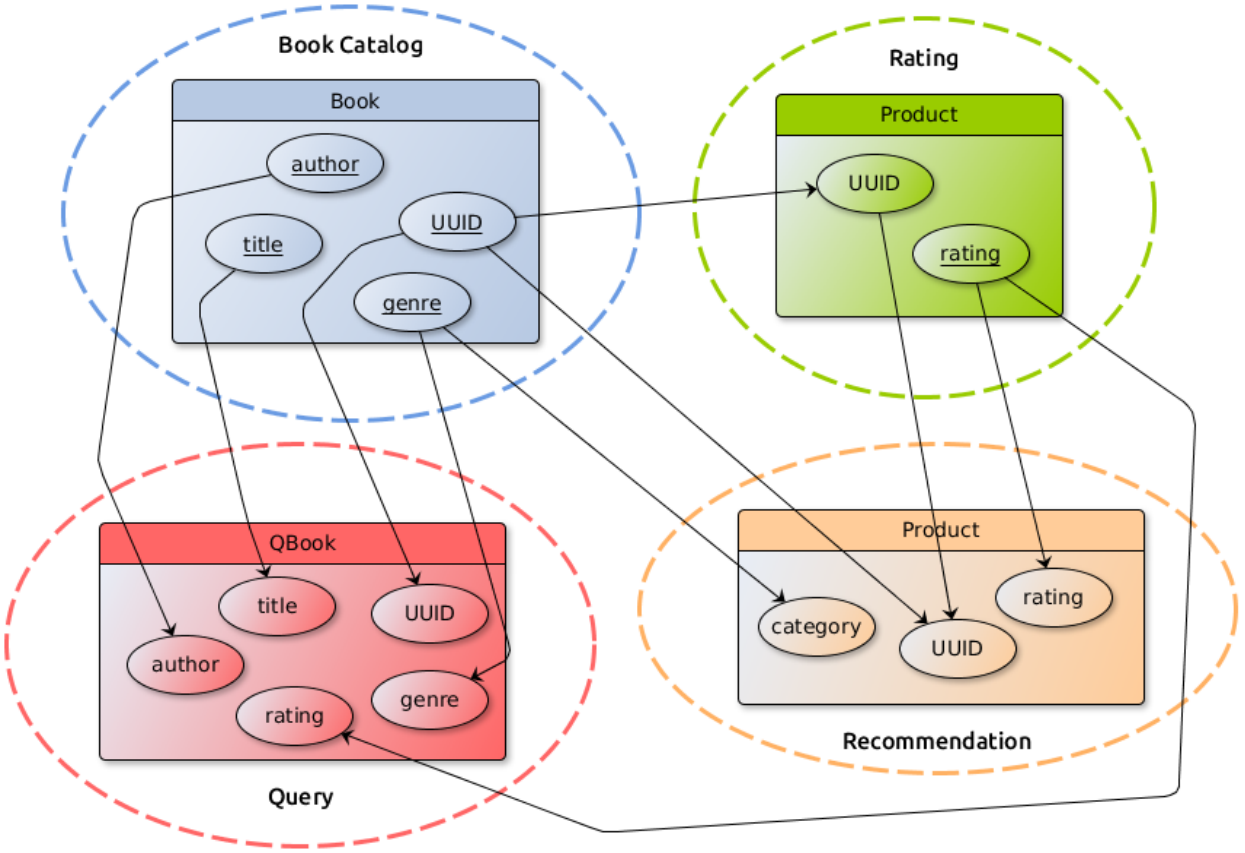
Solution

Command Query Responsibility Segregation (CQRS)

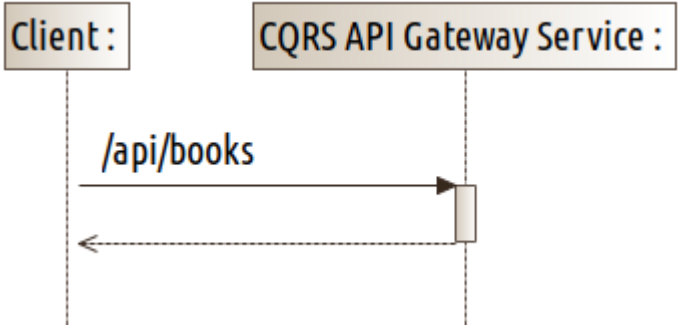
CQRS API gateway



CQRS API gateway Bookstore context map



CQRS API gateway communication flow



CQRS API gateway summary

- Queries are executed against one or more materialized views that are kept up to date by subscribing to streams of events emitted from microservices when data changes occur.
- Data is materialized views is eventually consistent.

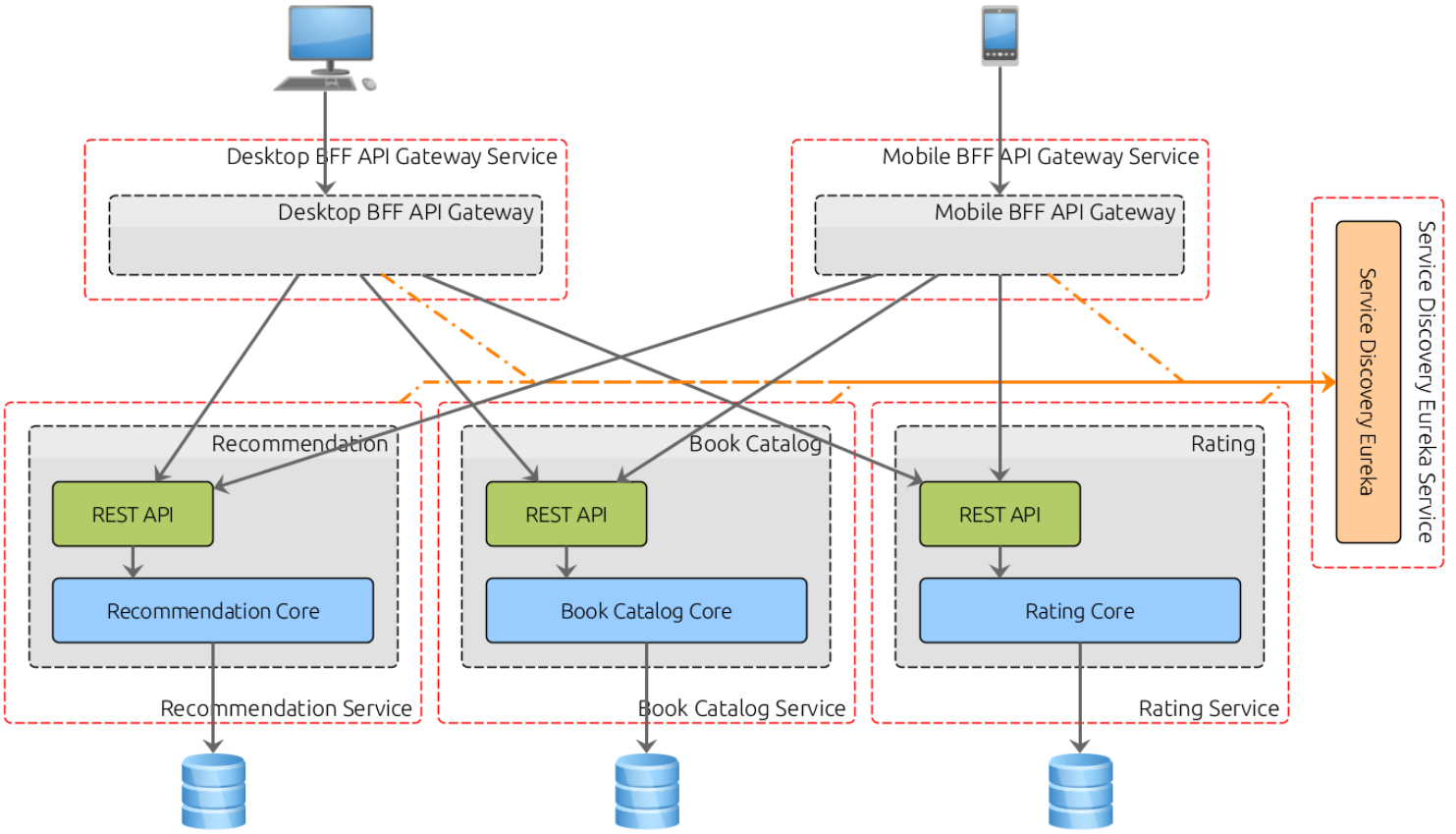
Challenges (part 4)

- Different clients need different data. For example, the desktop browser version of a product details page is typically more elaborate than the mobile version.

Solution

Backend for Frontend
(BFF)

Backend for Frontend (BFF)



Backend for Frontend summary

- Multiple API gateways to satisfy the needs of different clients.

The patterns

- API gateway and BFF
  http://microservices.io/patterns/apigateway.html
- API composition
  http://microservices.io/patterns/data/api-composition.html
- CQRS
  http://microservices.io/patterns/data/cqrs.html
- And many others at...
  http://microservices.io/

Thank You