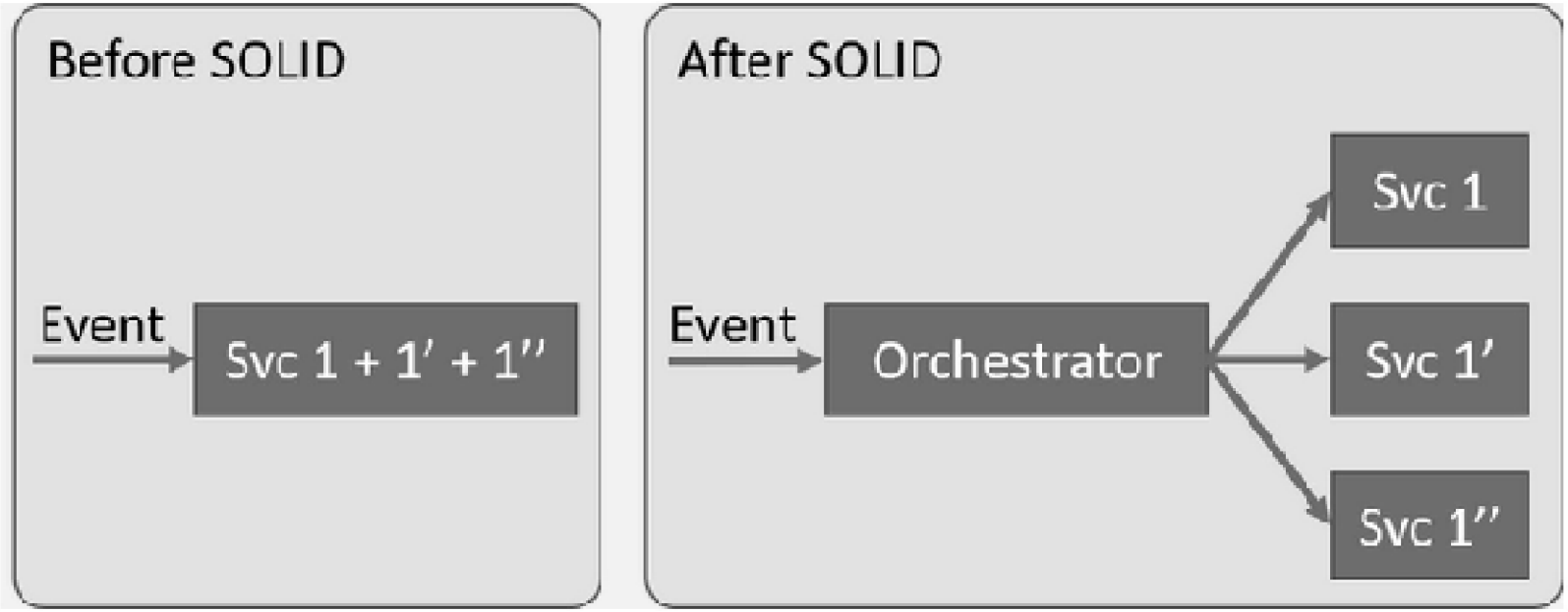
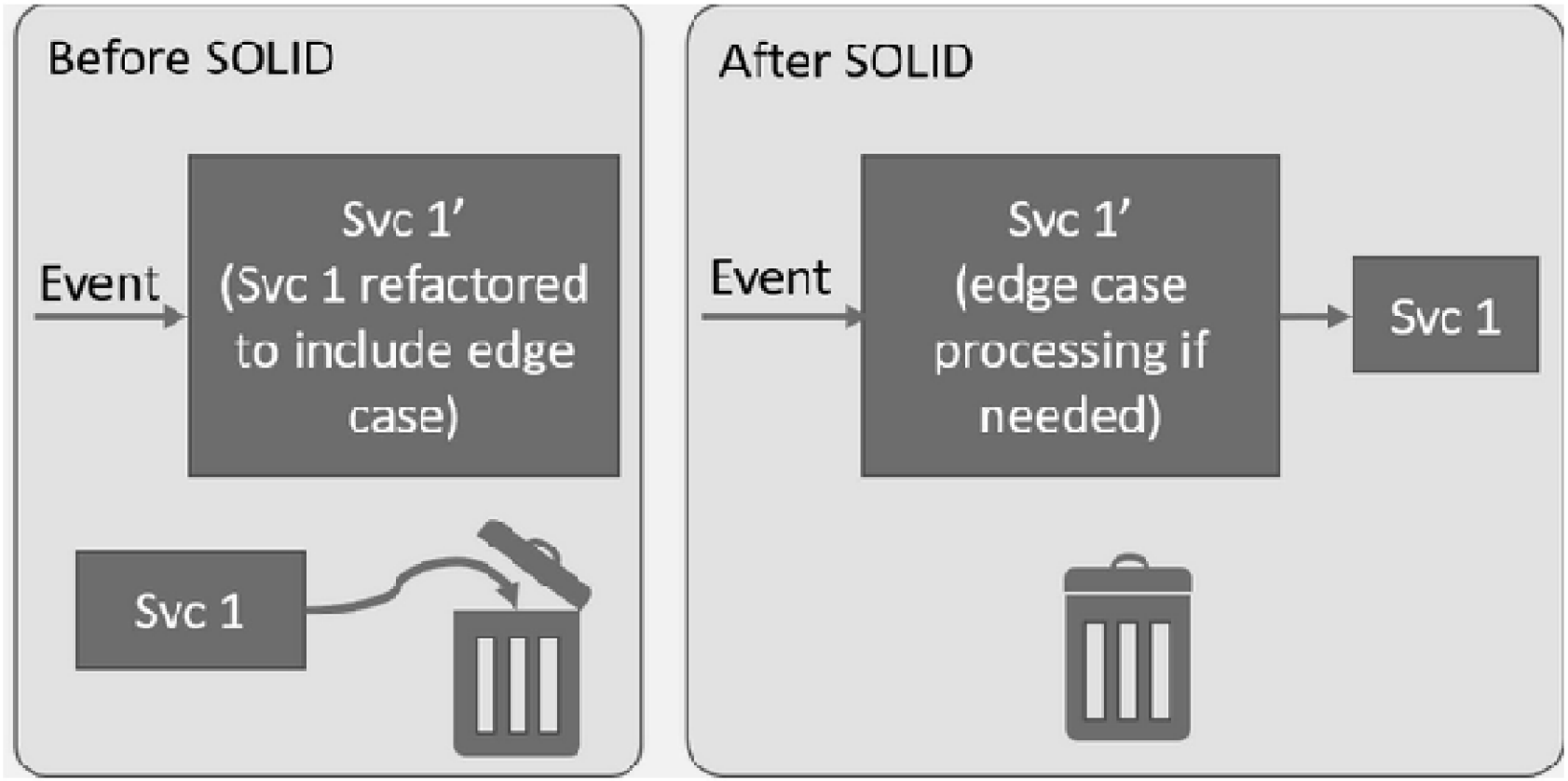
Sisteme Distribuite

Cursul 7

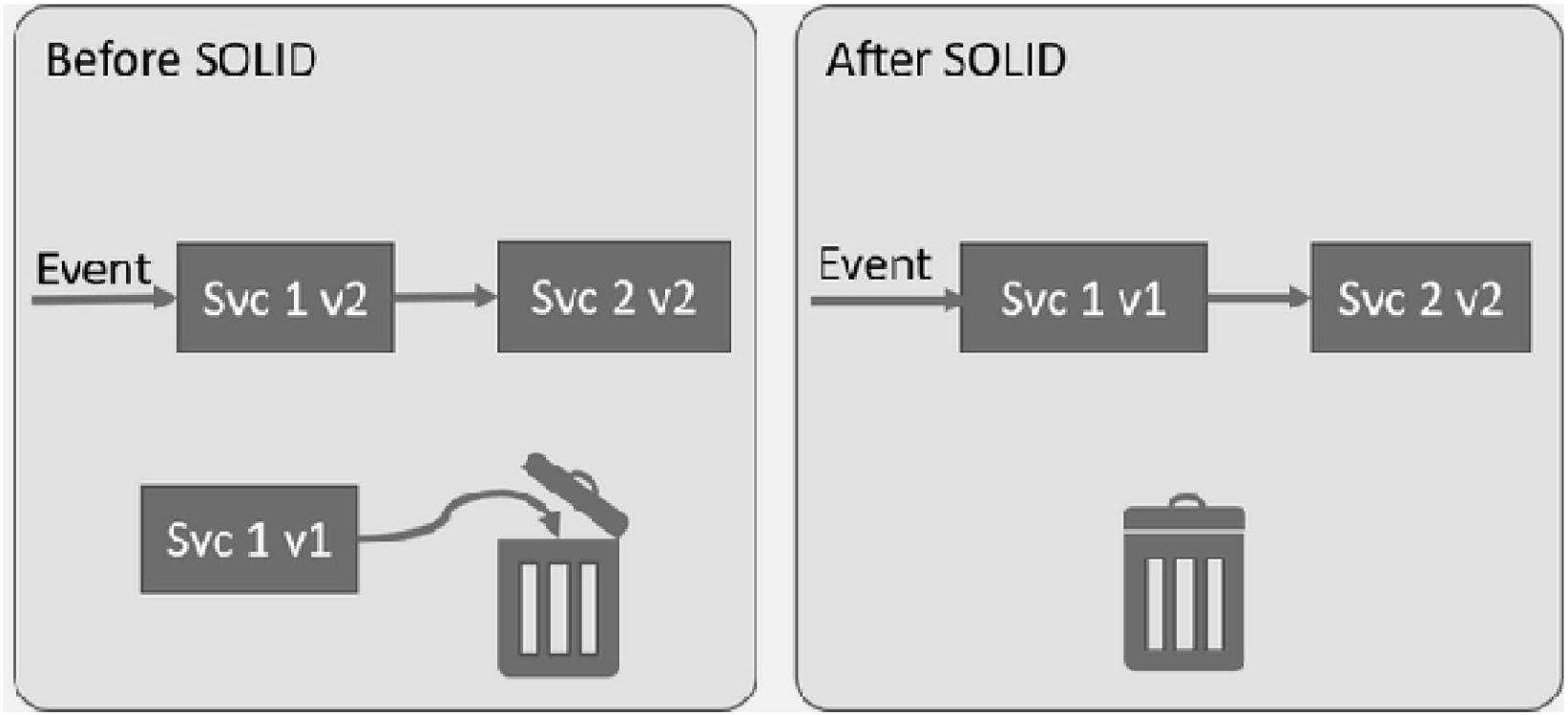
Principul responsabilității unice



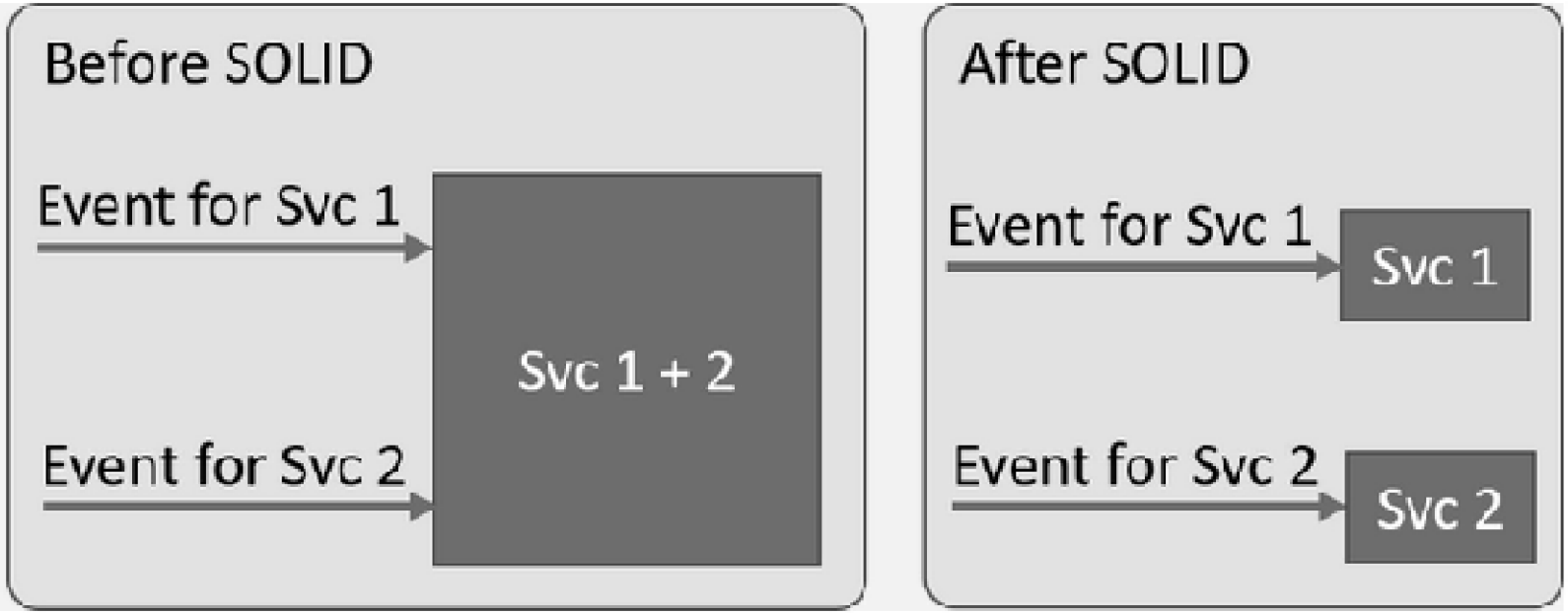
Deschis pentru extindere închis pentru modificare



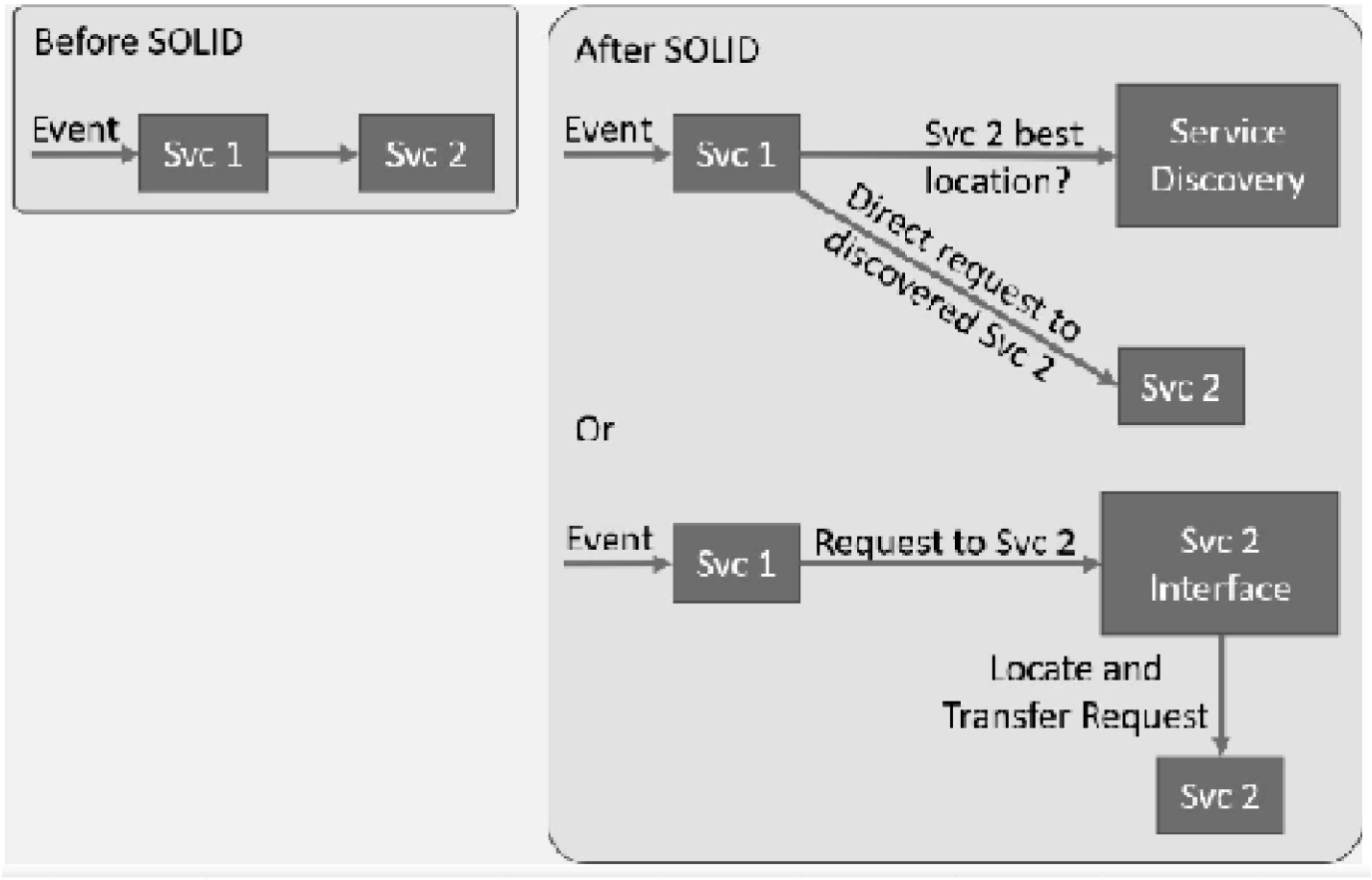
Substituția Liskov



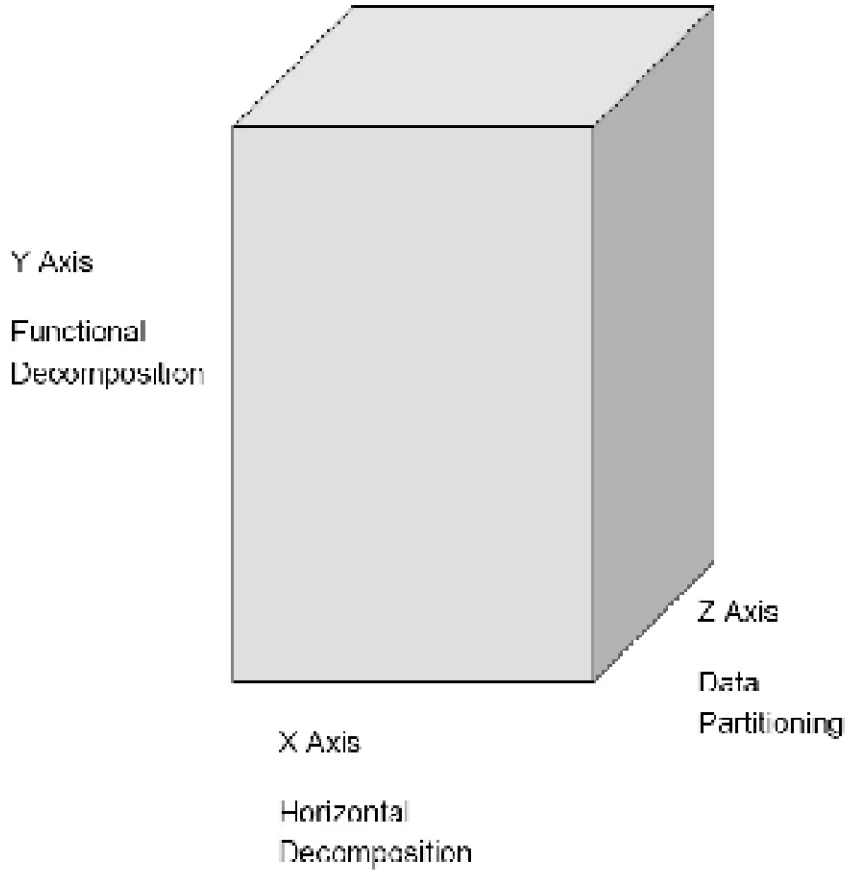
Separarea interfețelor



Controlul invers

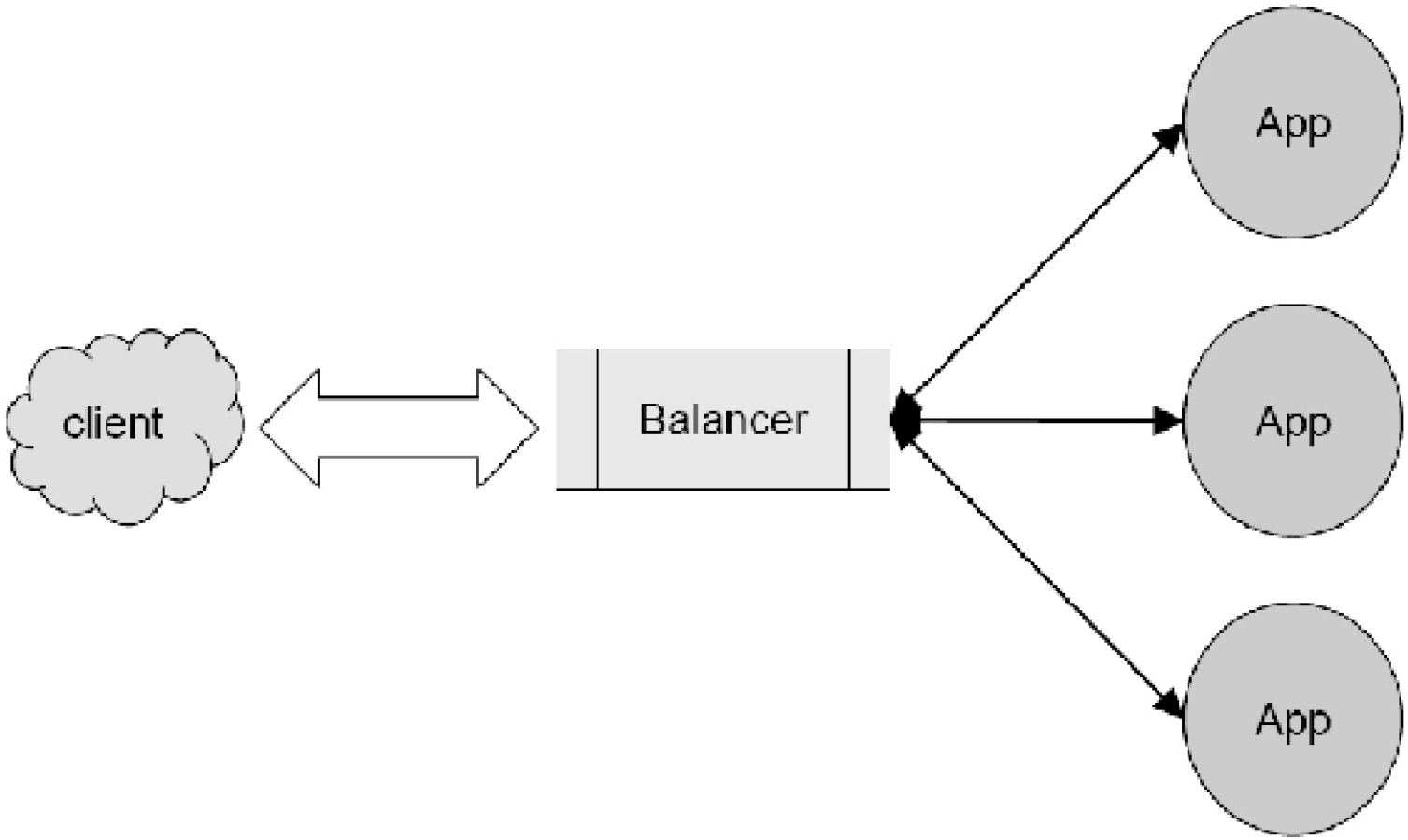


Principii de scalare a microserviciilor

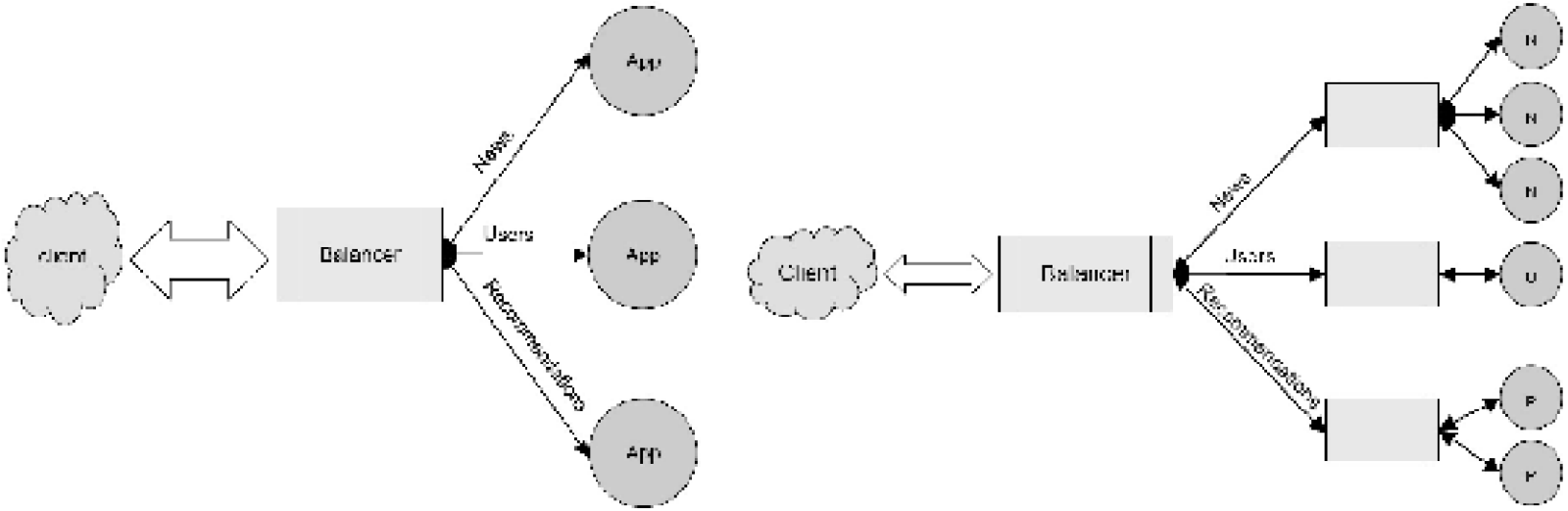


The Scale Cube

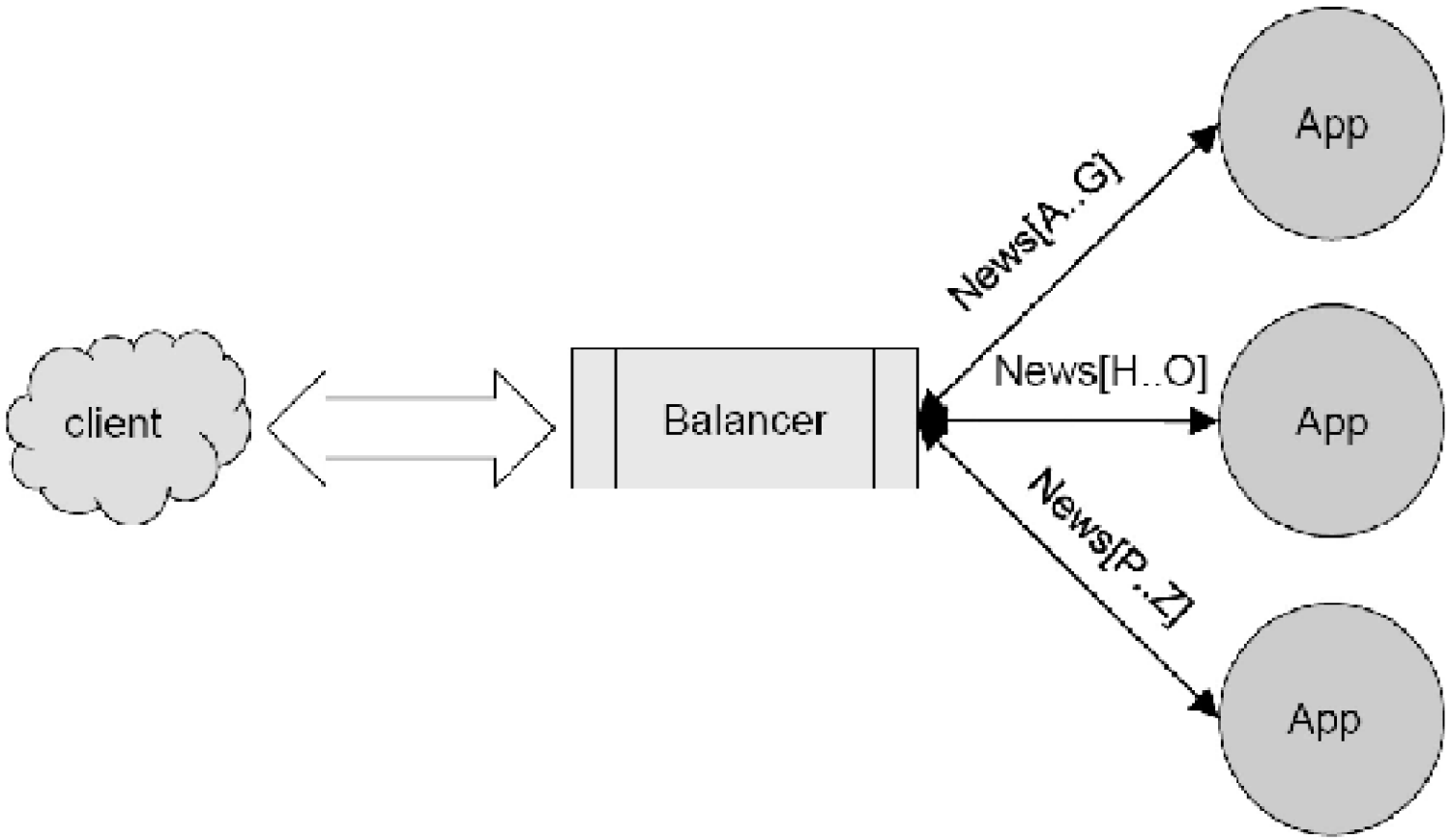
Cubul scalării - axa X



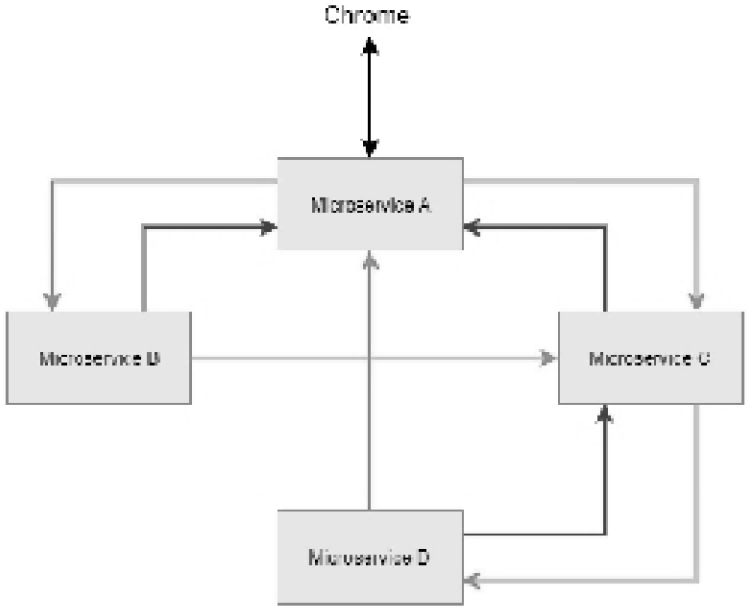
Cubul scalării - axa Y

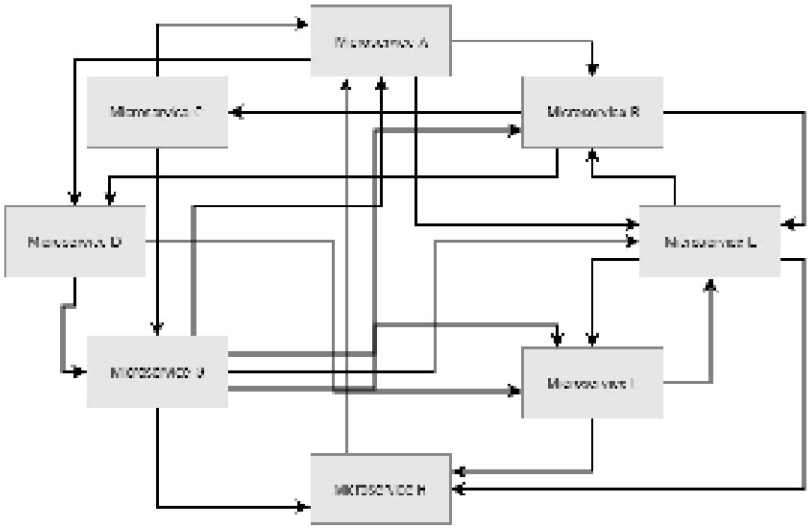


Cubul scalării - axa Z



Steaua Morții



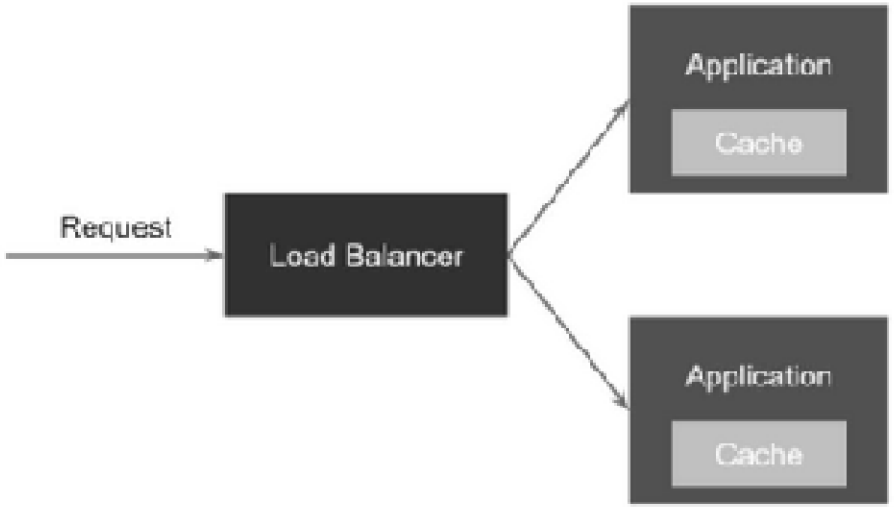


Mecanisme de caching

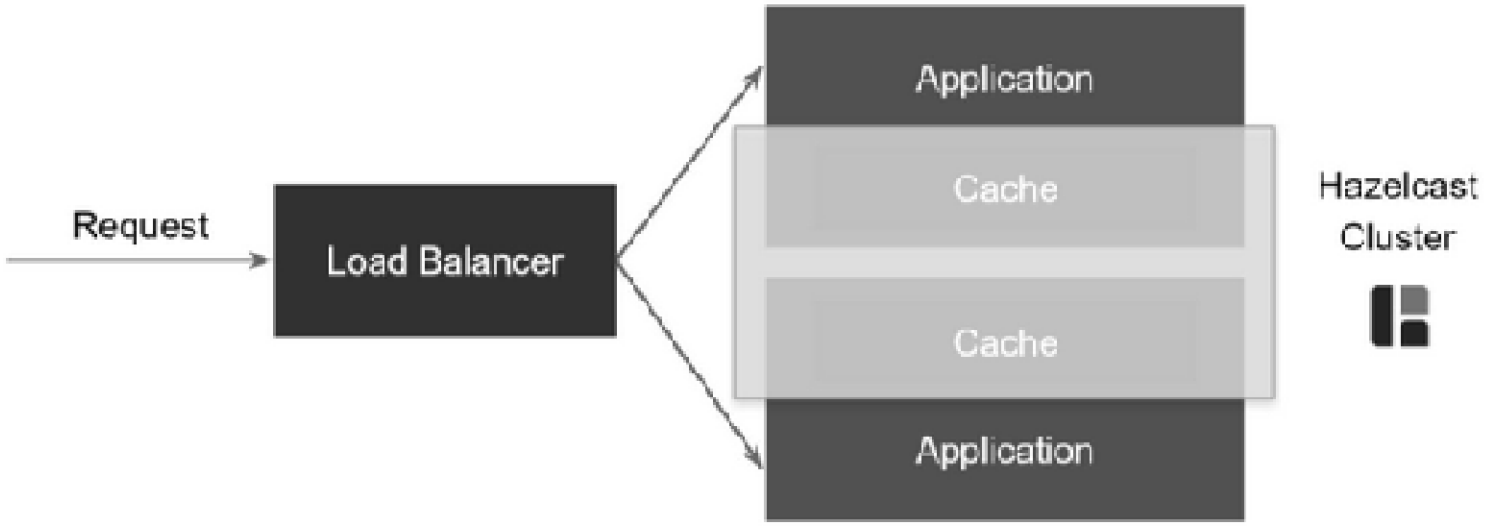
- caching?
- când?
- hand made
- Red Hat JBoss Data Grid

Modelul de proiectare - cache încapsulat (embeded)

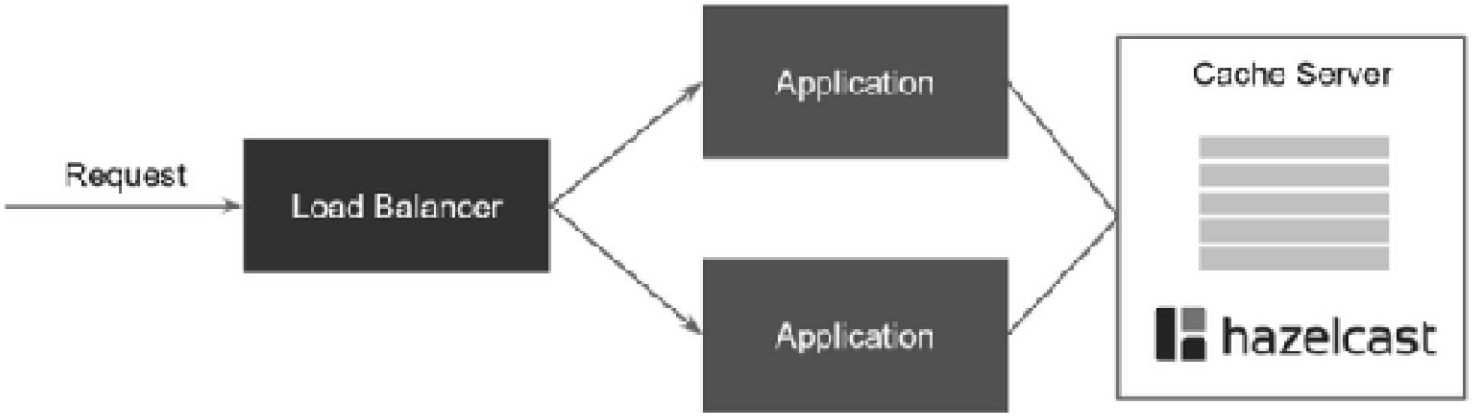
1. Cererile vin
2. modul lb transfera cererea
3. Apoi serviciul verifică duplicate
    1. Daca da valoarea
    2. altfel calcul



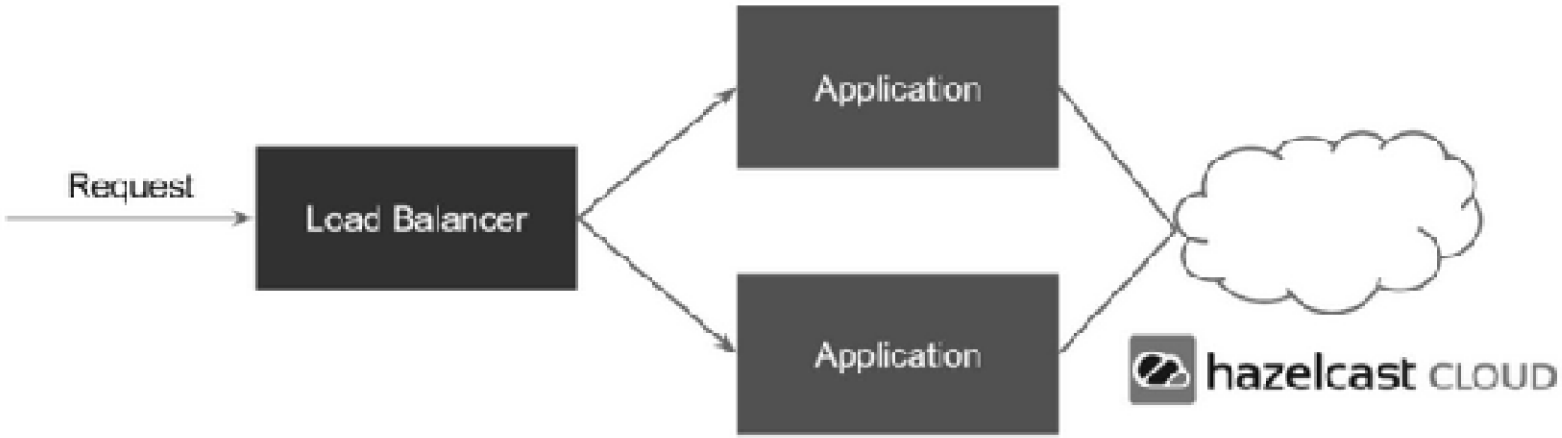
Modelul de proiectare - cache încapsulat distribuit (cache comun)



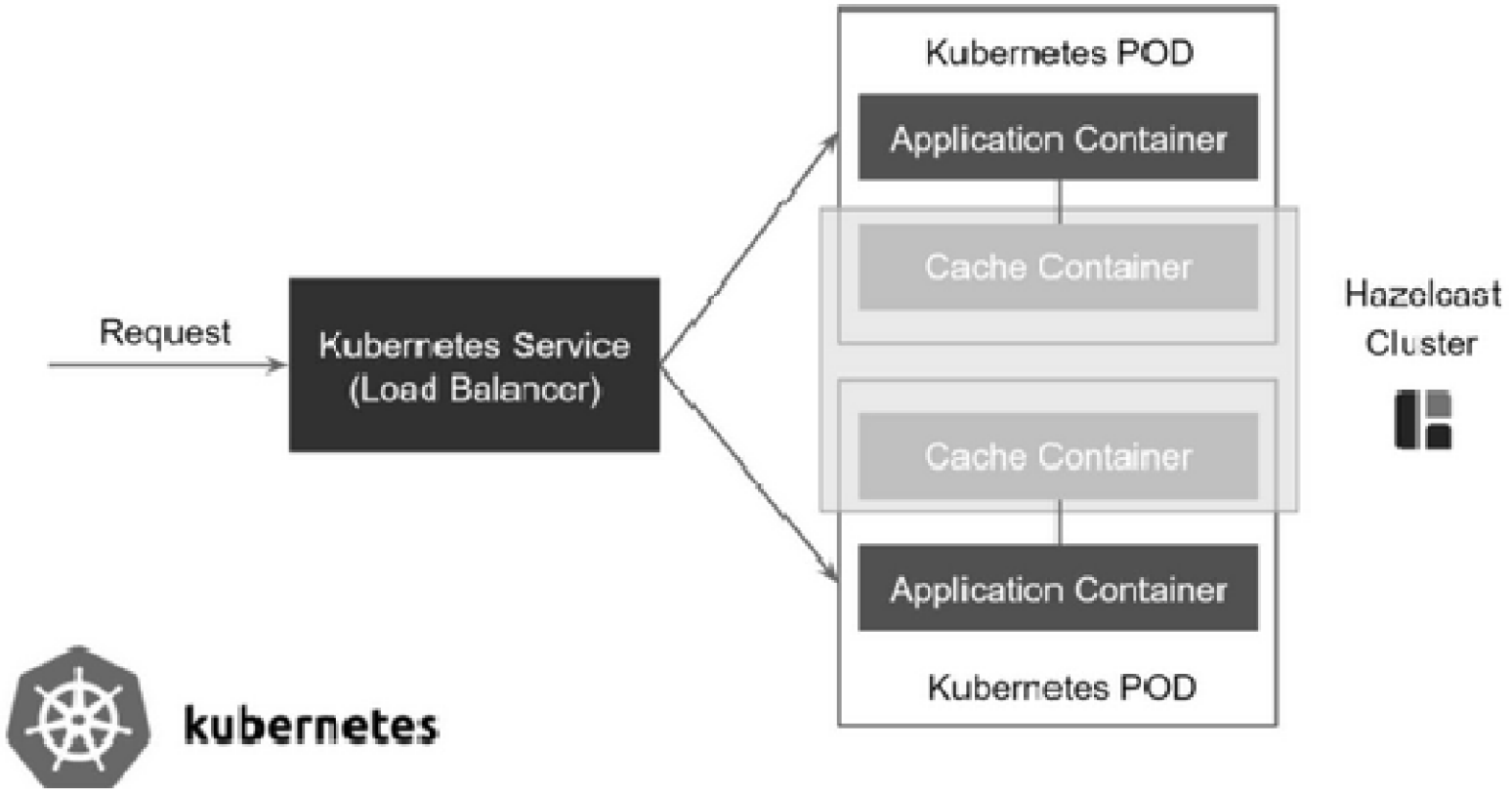
Modelul de proiectare cache client server



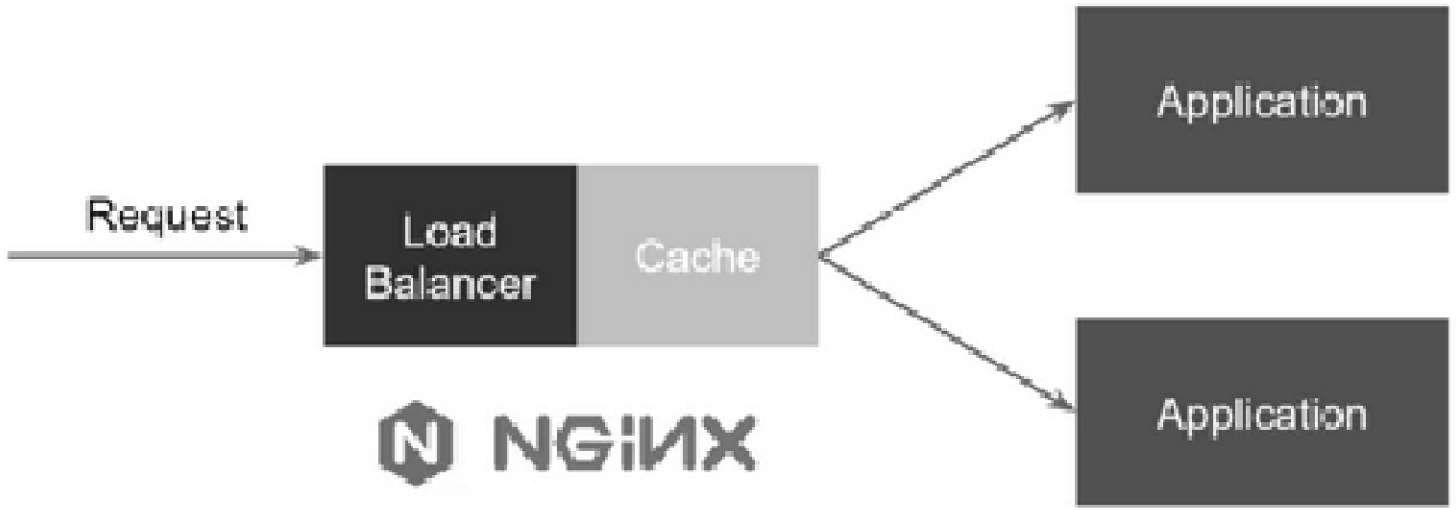
Modelul de proiectare cache în nor



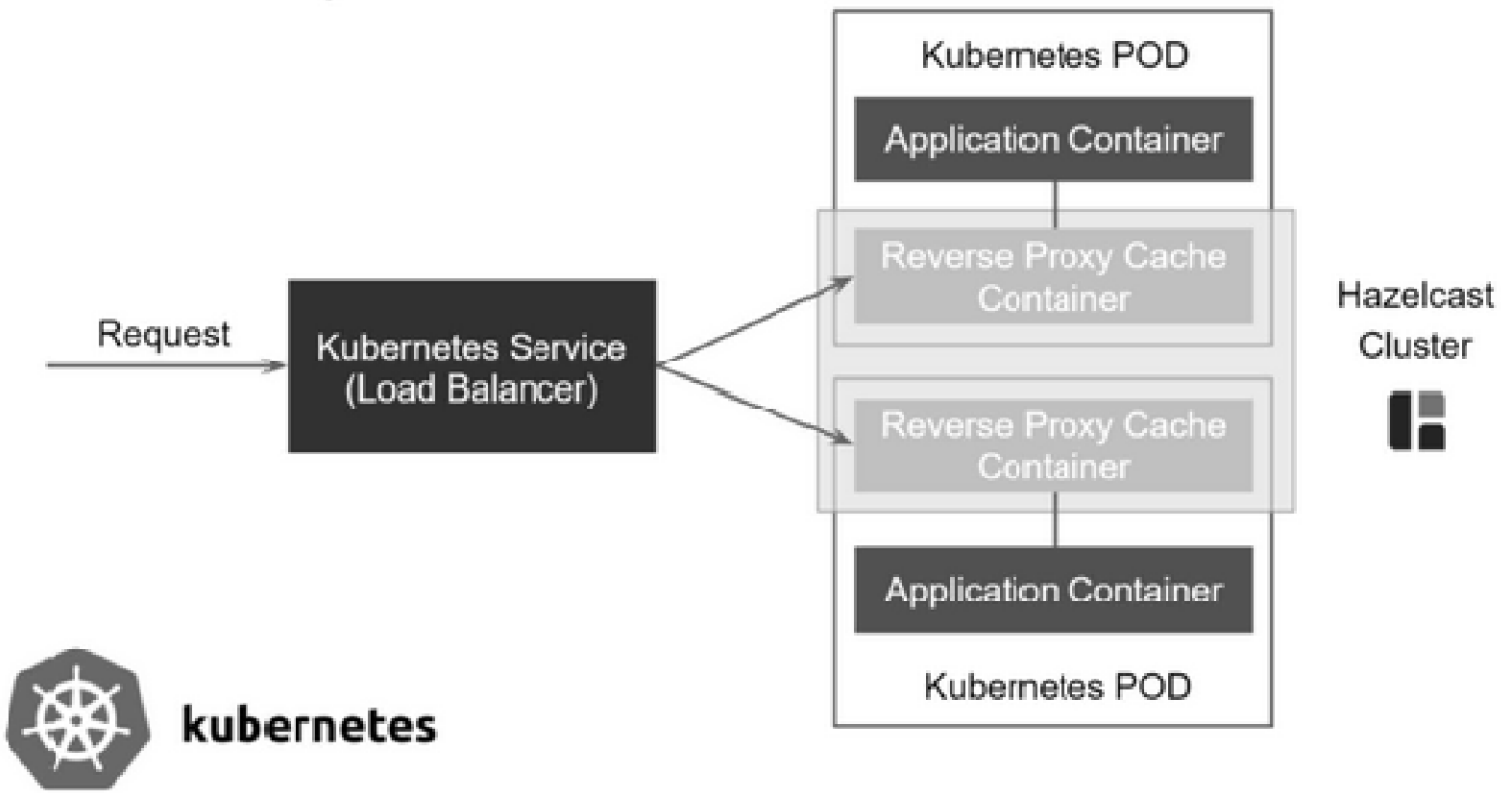
Modelul atașamentului (sidecar)



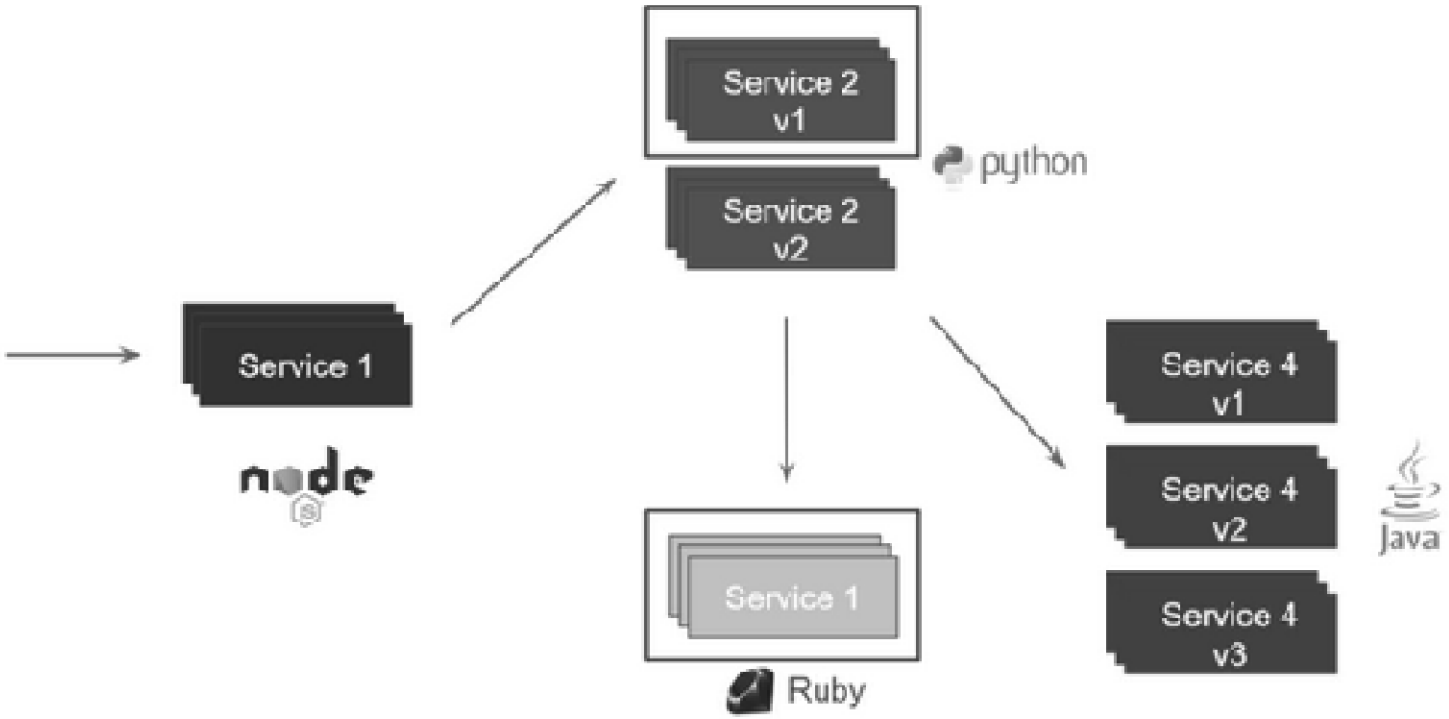
Modelul de proiectare cu intermediar invers (inverse proxy)



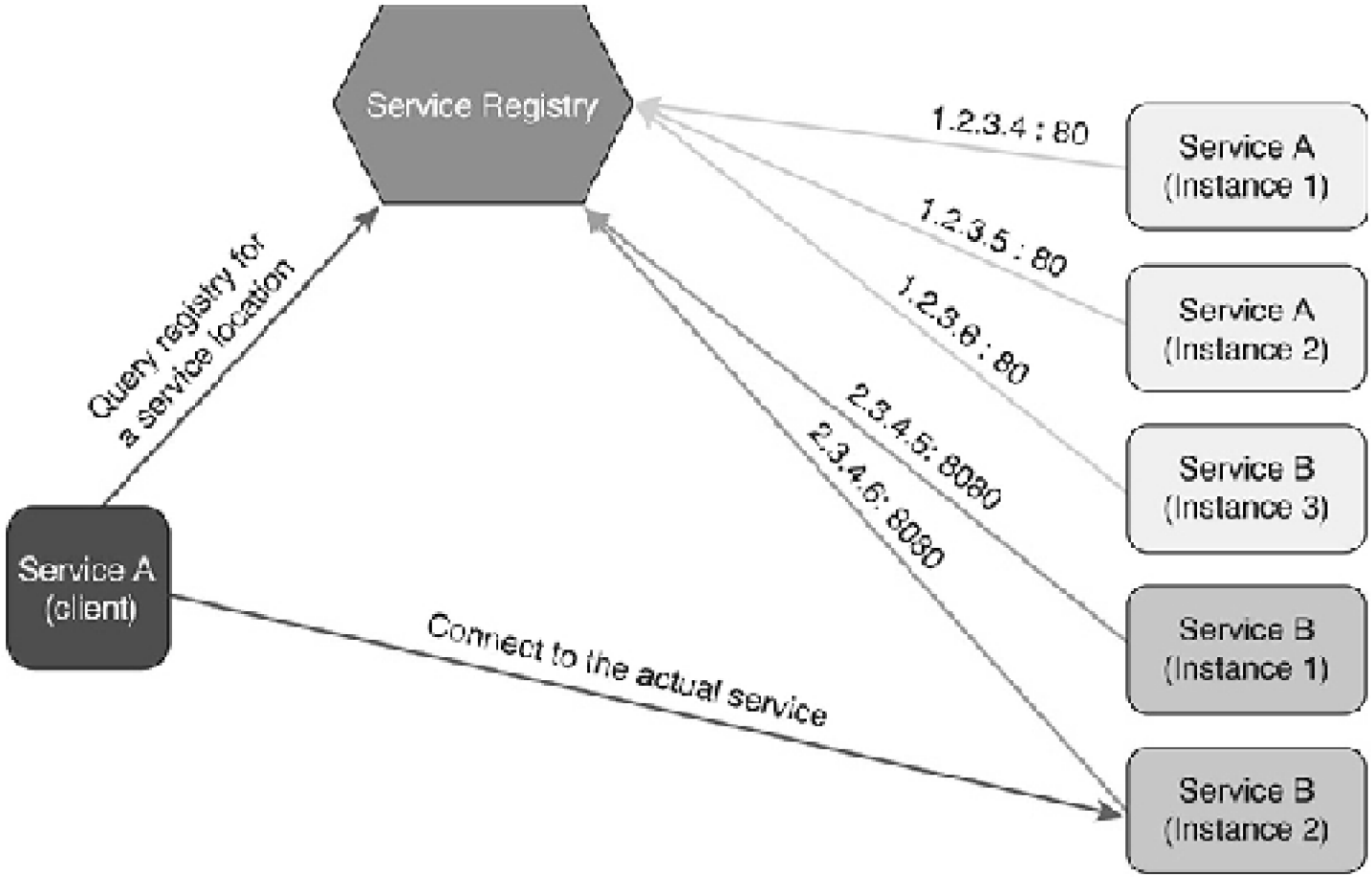
Intermediarul invers & atașamentul



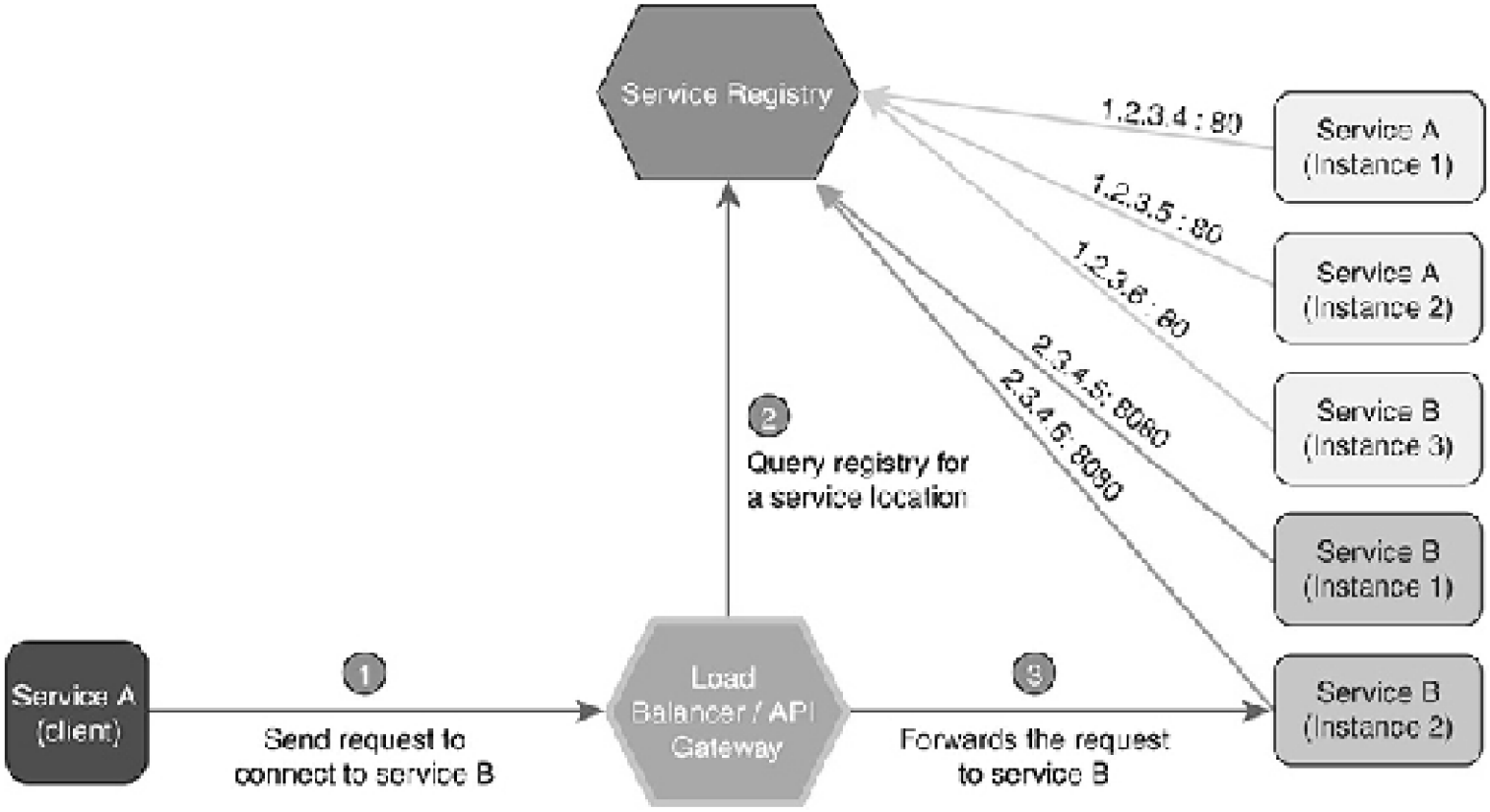
Intermediarul invers & atașamentul



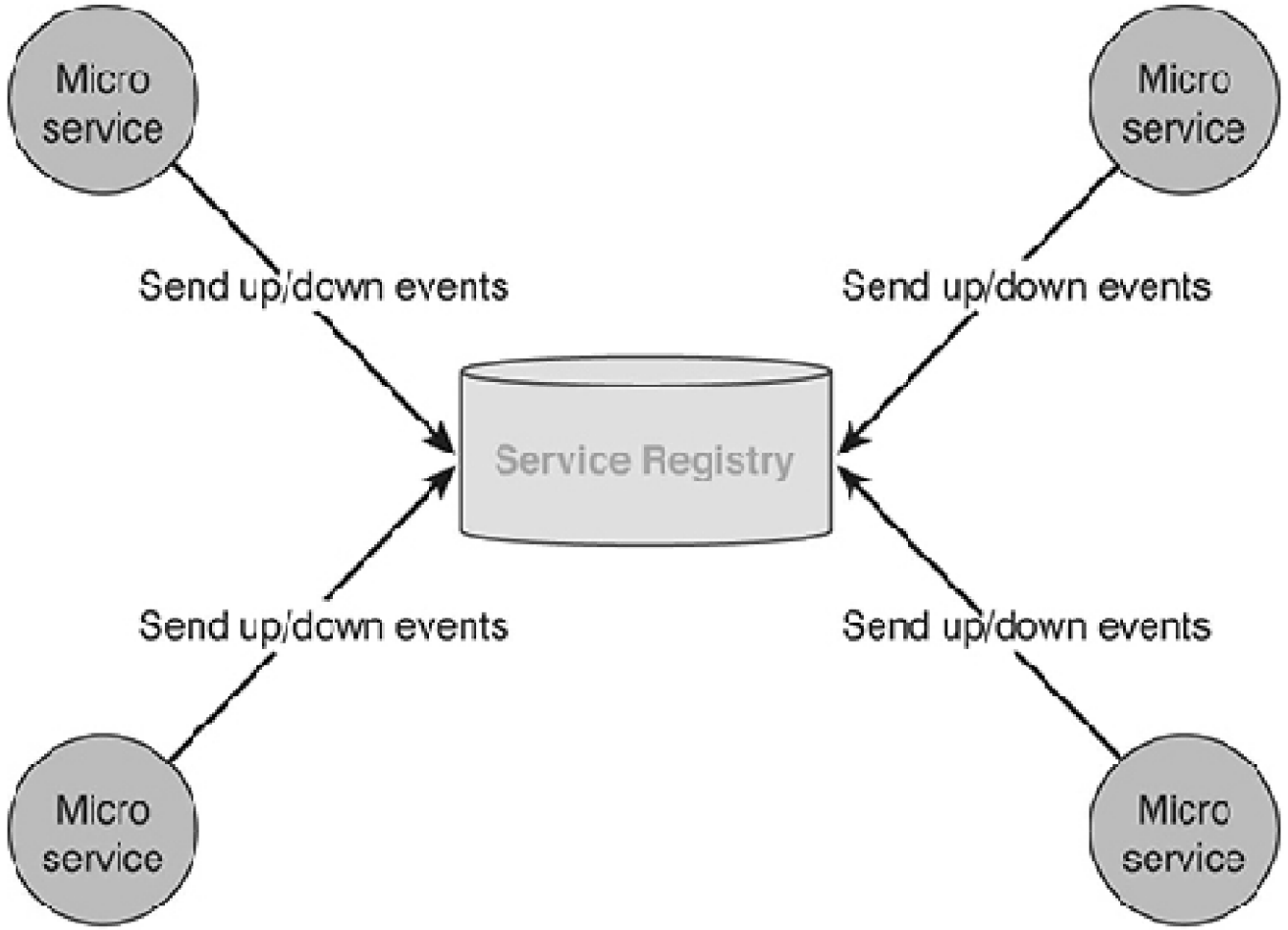
Descoperirea la nivel de client



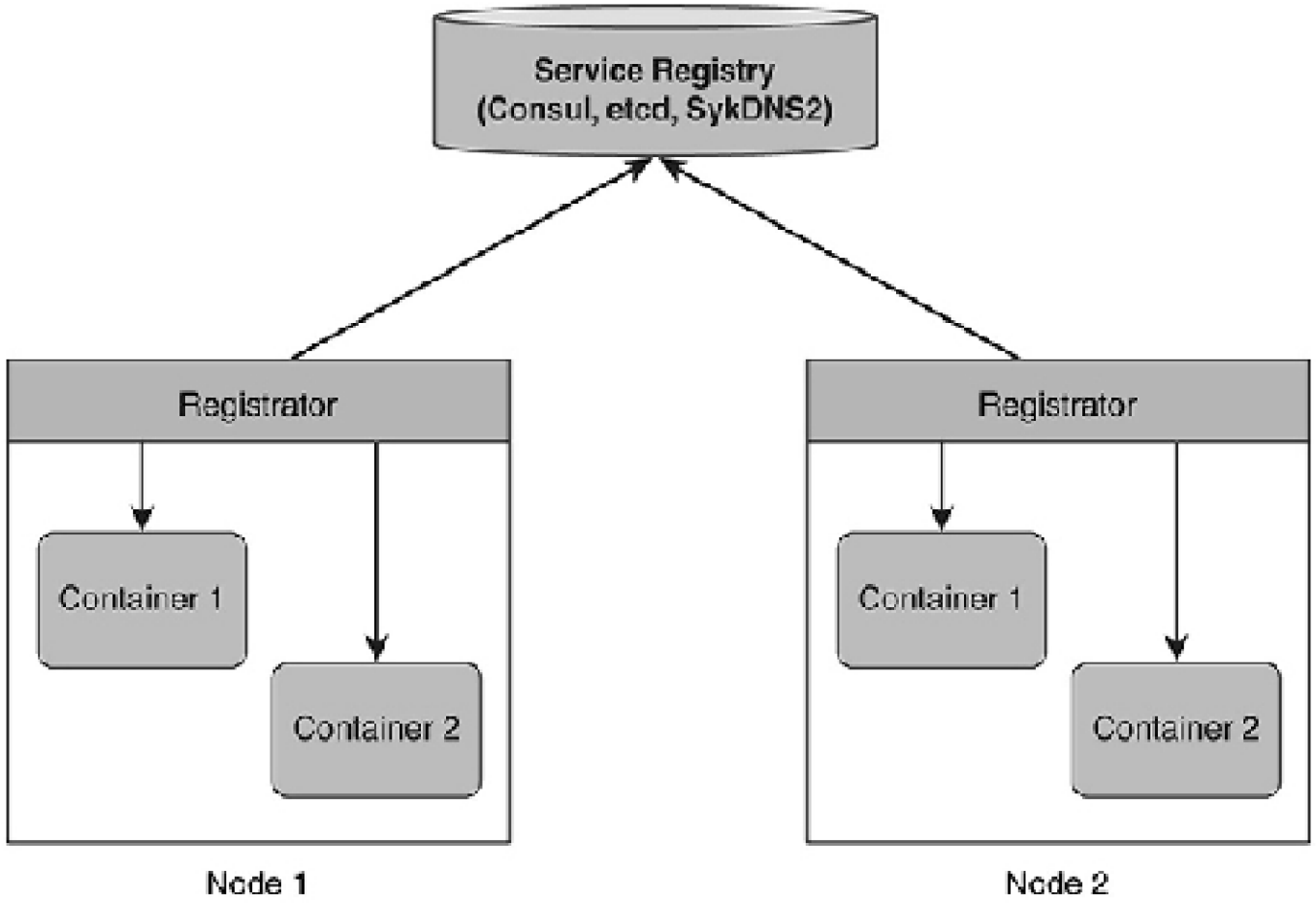
Descoperirea la nivel de server



Registre de servicii - Autoînregistrare



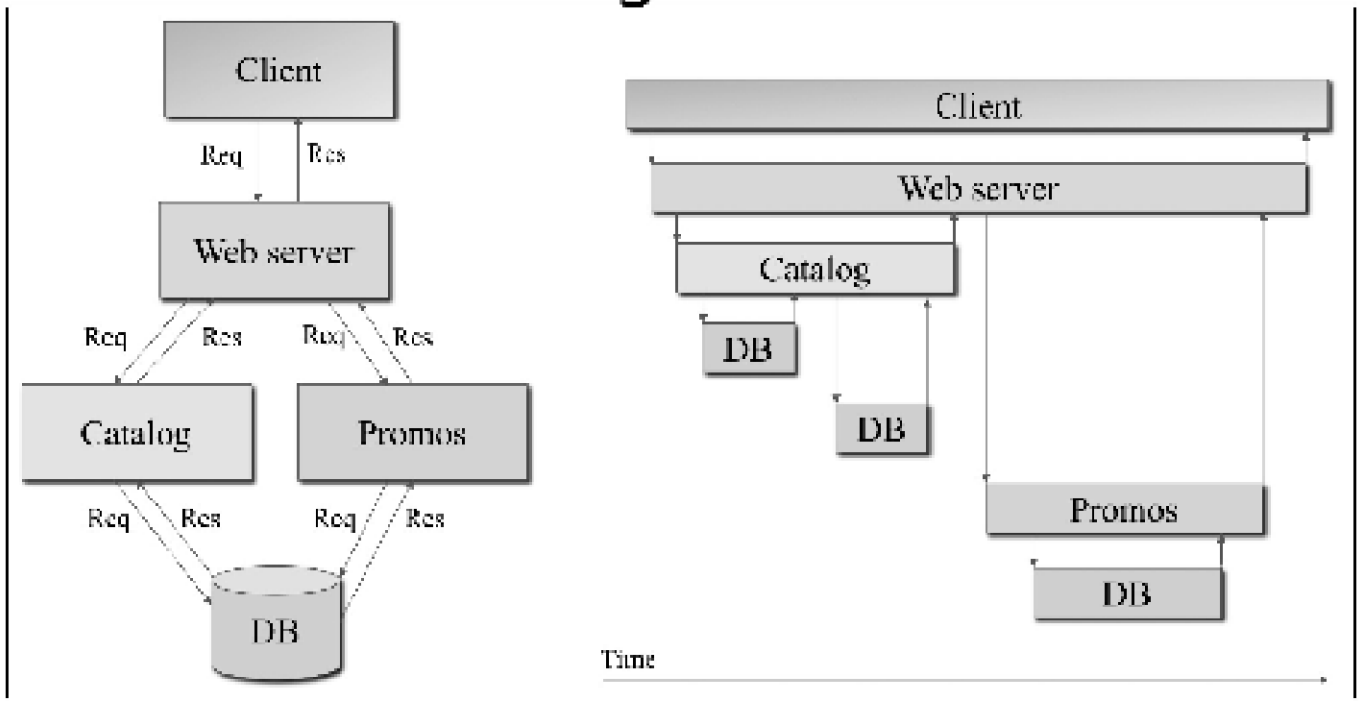
Registre de servicii - Instrumente Externe



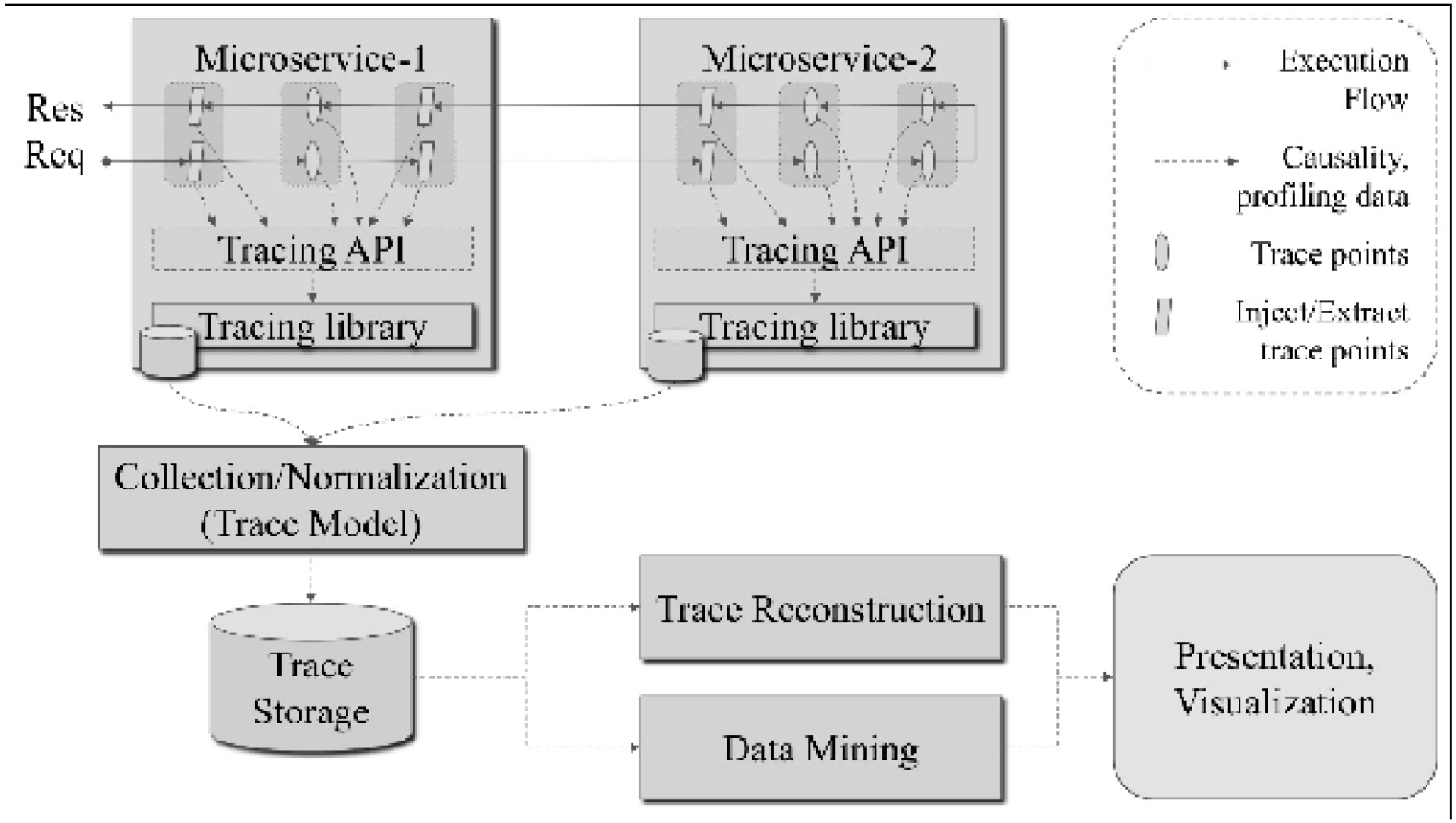
Registrator (https://github.com/gliderlabs/registrator)

Urmarirea distribuită

- distributed tracing sau
- end-to-end tracing sau
- workflow-centric tracing



Urmarirea distribuită



Ordonarea temporală - Lamport

- Pentru a realiza sincronizarea ceasurilor logice Lamport a definit o funcţie de tipul "s-a întâmplat înainte" notată cu "$\rightarrow$".
- De exemplu a $\rightarrow$ b, a s-a întâmplat înainte de b.
- Această relaţie se poate observa în două cazuri:
    - Dacă avem a şi b evenimente aparţinând aceluiaşi proces şi a apare înainte de b atunci a$\rightarrow$b este adevărată.
    - Dacă a este un eveniment al unui mesaj transmis de un proces şi b este un eveniment al unui mesaj recepţionat de alt proces de asemenea a$\rightarrow$b este adevărată.

Ordonarea temporală - Lamport

- Totuşi faptul că sincronizarea este realizată numai funcţie de sosiri sau plecări permite folosirea operatorului Lamport şi garantează astfel controlul. Utilizând această metodă evenimentele într-un sistem distribuit se pot asigna la timpi ca mai jos:

1. Dacă a apare după b în acelaşi proces:

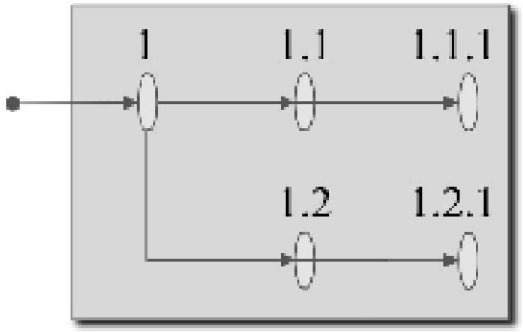
$$T(a) > T(b);$$

2. Dacă a şi b reprezintă emiterea şi recepţia unui mesaj $T(a) < T(b)$;

3. Pentru toate evenimentele a şi b

$$T(a) \neq T(b).$$

Urmarirea distribuită - cauzalitate



Execution ID	Event ID	Parent ID	Causality
X	1	-	
X	1.1	1	1 → 1.1
X	1.1.1	1.1	1.1 → 1.1.1
X	1.2	1	1 → 1.2
X	1.2.1	1.2	1.2 → 1.2.1

Stabilirea de relațiii cauzale utilizând metadate dinamice cu lungime variabilă

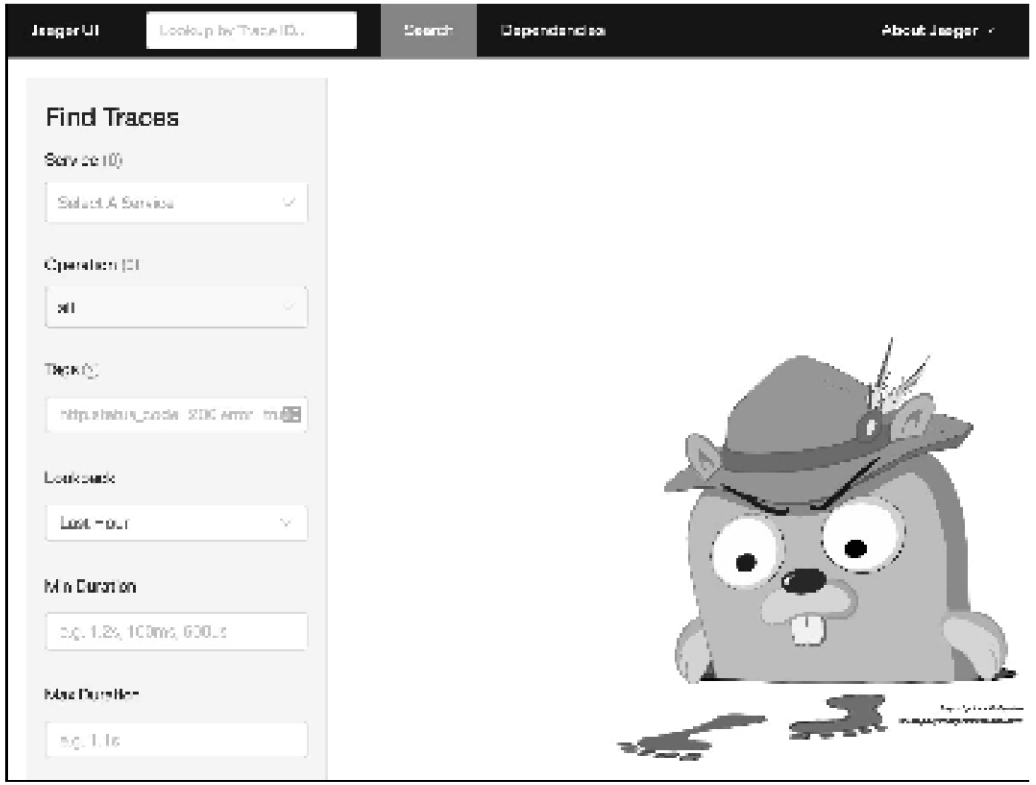
Urmarirea distribuită a aplicațiilor asincrone

- **În Spring**
- *opentracing-spring-cloud-starter*

```
<dependency>
   <groupId>io.opentracing.contrib</groupId>
   <artifactId>opentracing-spring-cloud-
starter</artifactId>
   <version>0.1.13</version>
</dependency>
```

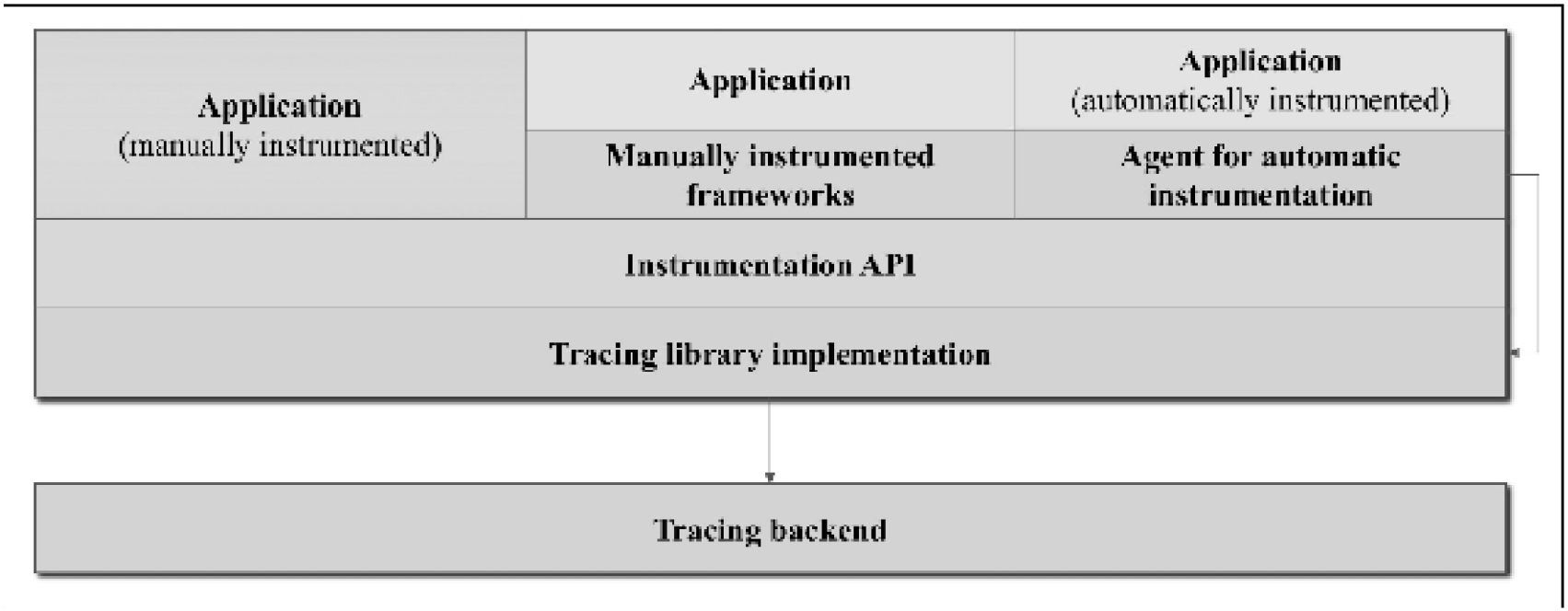
Urmarire in Docker - Jaeger

```
docker run -d --name jaeger \
  -e COLLECTOR_ZIPKIN_HTTP_PORT=9411 \
  -p 5775:5775/udp \
  -p 6831:6831/udp \
  -p 6832:6832/udp \
  -p 5778:5778 \
  -p 16686:16686 \
  -p 14268:14268 \
  -p 9411:9411 \
  jaegertracing/all-in-one:1.6
```



- Jaeger UI la http://localhost:16686

Instrumentație bazată pe agenți



Instrumente și terminologie

- Zipkin (fost Big Brother Bird sau B3) de Twitter
- Stackdriver de Goagl'
- X-Ray de Amazoane
- Ben Sigelman la KubeCon EU 2018 - multă confuzie

Terminologie - urmărire

Observabilitatea într-o plasă de servicii

- abordare tip intermediare
- aplicația ca cutie neagra
- RED
- surse eterogene de servicii
- nu garanteaza tot timpul informații complete

Stiluri de achiție a stării (sampling)

- Head-based consistent sampling sau
- upfront sampling

Stiluri de achiție a stării (sampling)

Probabilistic sampling

```
class ProbabilisticSampler(probability: Double) {
  def isSampled: Boolean = {
    if (Math.random() < probability) {
      return true
    } else {      return false    }
  }
}
```

Stiluri de achiție a stării (sampling)

```
class RateLimiter(creditsPerSecond: Double, maxBalance: Double) {
 val creditsPerNanosecond = creditsPerSecond / 1e9
  var balance: Double = 0
 var lastTick = System.nanoTime()
  def withdraw(amount: Double): Boolean = {
  val currentTime = System.nanoTime()
  val elapsedTime = currentTime - lastTick
  lastTick = currentTime
  balance += elapsedTime *creditsPerNanosecond
  if (balance > maxBalance) {
   balance = maxBalance   }
  if (balance >= amount) {
   balance -= amount
   return true   }
  return false  }  }
```

Achiziție probabilistică cu ieșire garantată

```
class GuaranteedThroughputSampler(
  probability: Double,
  minTracesPerSecond: Double) {
  val probabilistic = new ProbabilisticSampler(probability)
  val lowerBound = new RateLimitingSampler(minTracesPerSecond)

  def isSampled: Boolean = {
    val prob: Boolean = probabilistic.isSampled()
    val rate: Boolean = lowerBound.isSampled()
    if (prob) {
      return prob}
    return rate  }
}
```

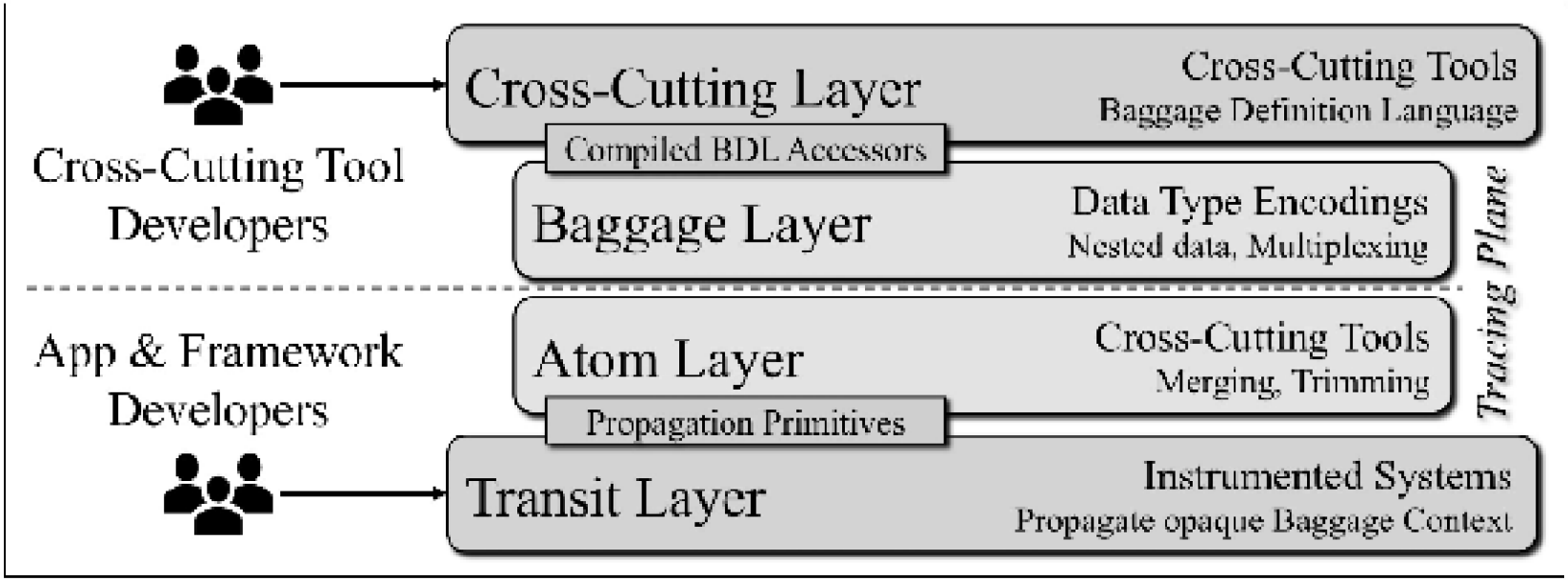
Achiziție adaptivă

- Achizitie locală adaptivă
- Achizitie global adaptivă

Cei trei piloni ai observabilității

- metrics
- logs
- distributed tracing

Propagarea contextului distribuit



Planul Brown de urmărire

Planul Brown de urmărire

Baggage Definition Language (BDL)

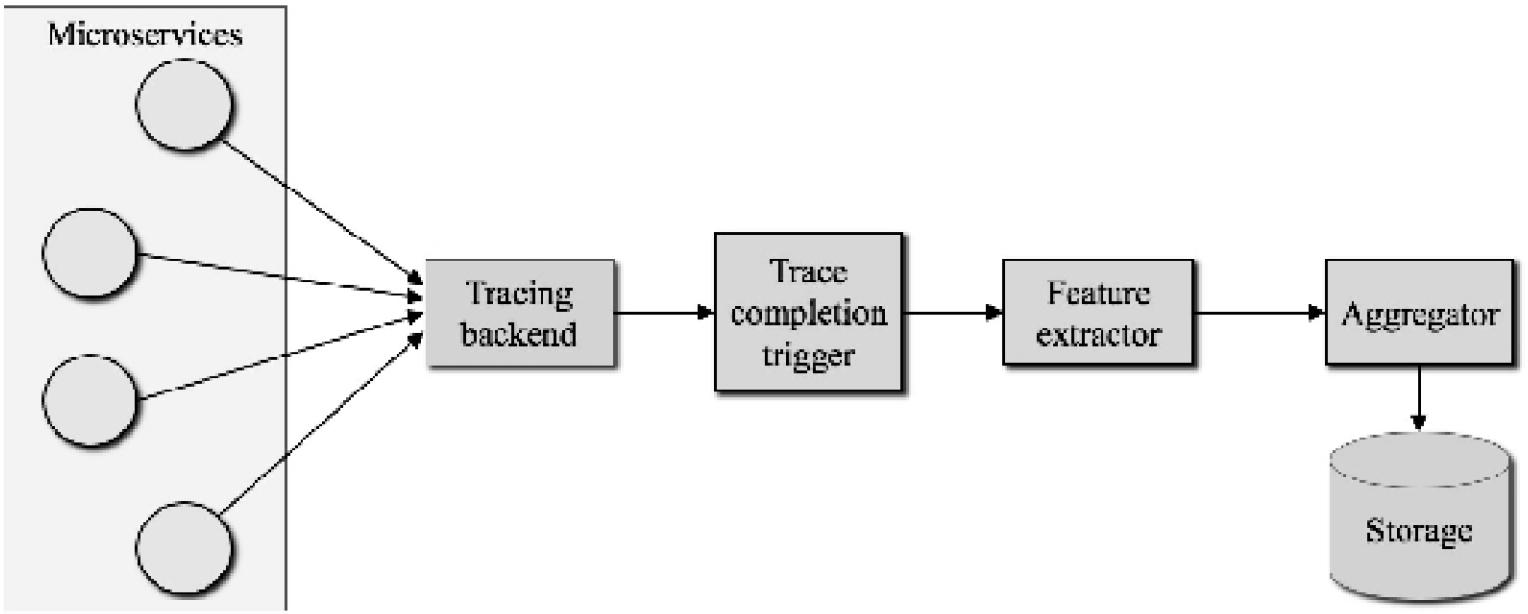
```
bag TracingTool {
   int64 traceID = 0;
   int64 spanID  = 1;
   bool  sampled = 2;
}
```

- decuplare instrumentație de propagare metadate

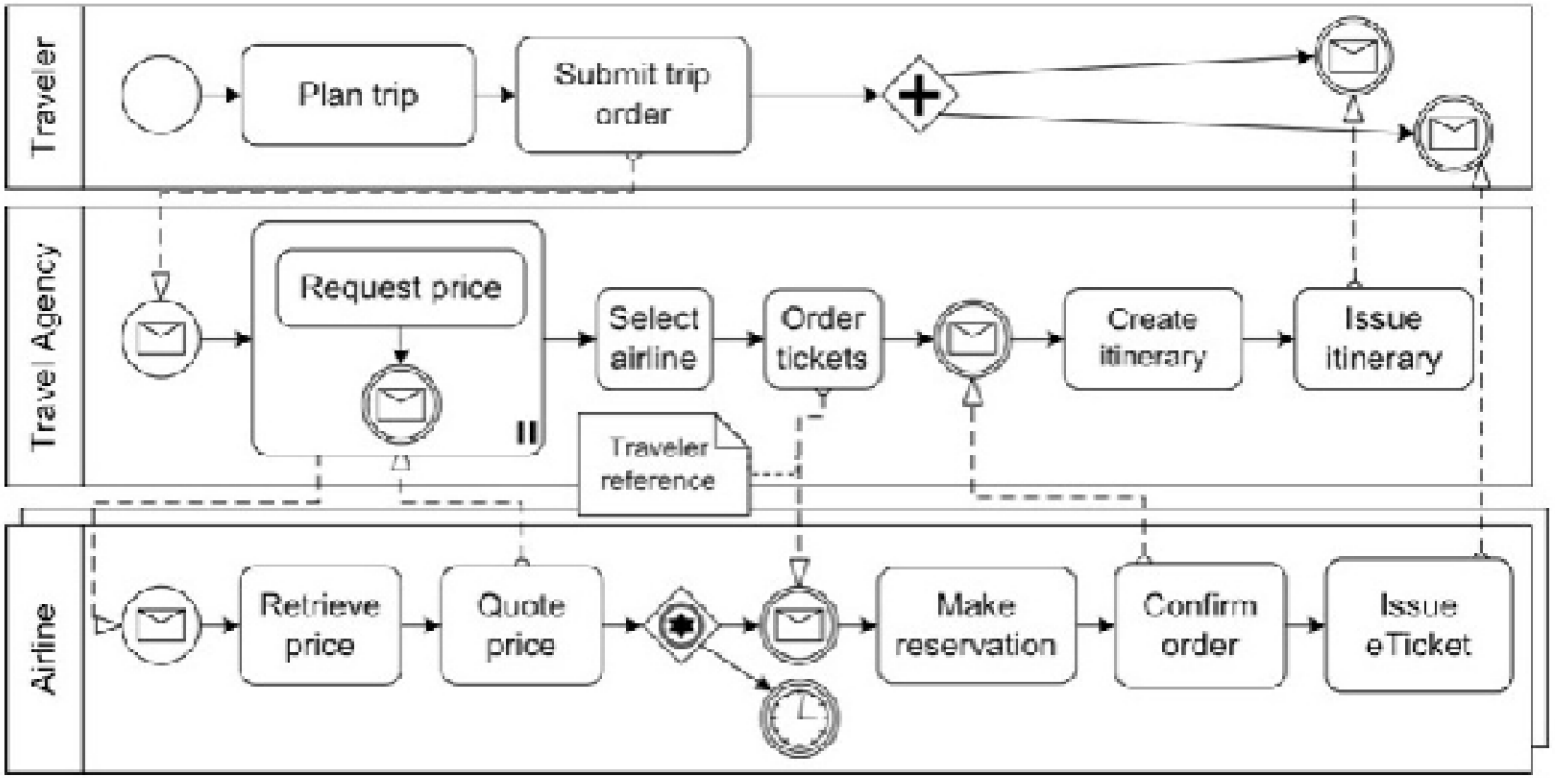
Planul Brown de urmărire

- nivelul bagaj
  - multiplexare in cadrul contextului
  - deasupra strat de urmarire
  - abordare crosscutting
- nivelul atom
  - distribuie metadatele in fromat binar
  - operatii
    - Serialize(BaggageContext): Bytes
    - Deserialize(Bytes): BaggageContext
    - Branch(BaggageContext): BaggageContext
    - Merge(BaggageContext, BaggageContext): BaggageContext
    - Trim(BaggageContext): BaggageContext
- nivelul tranzit

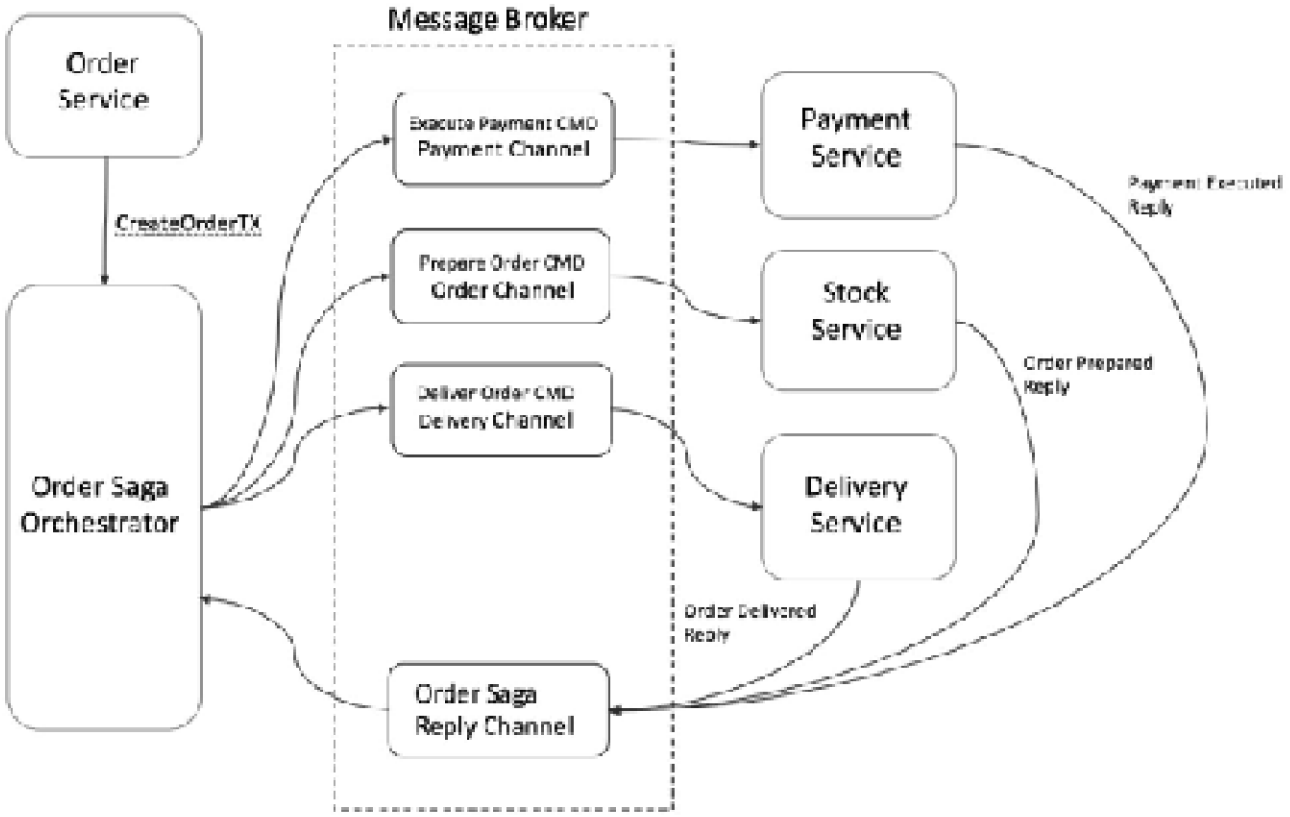
Componentele unui pipe line cu data mining



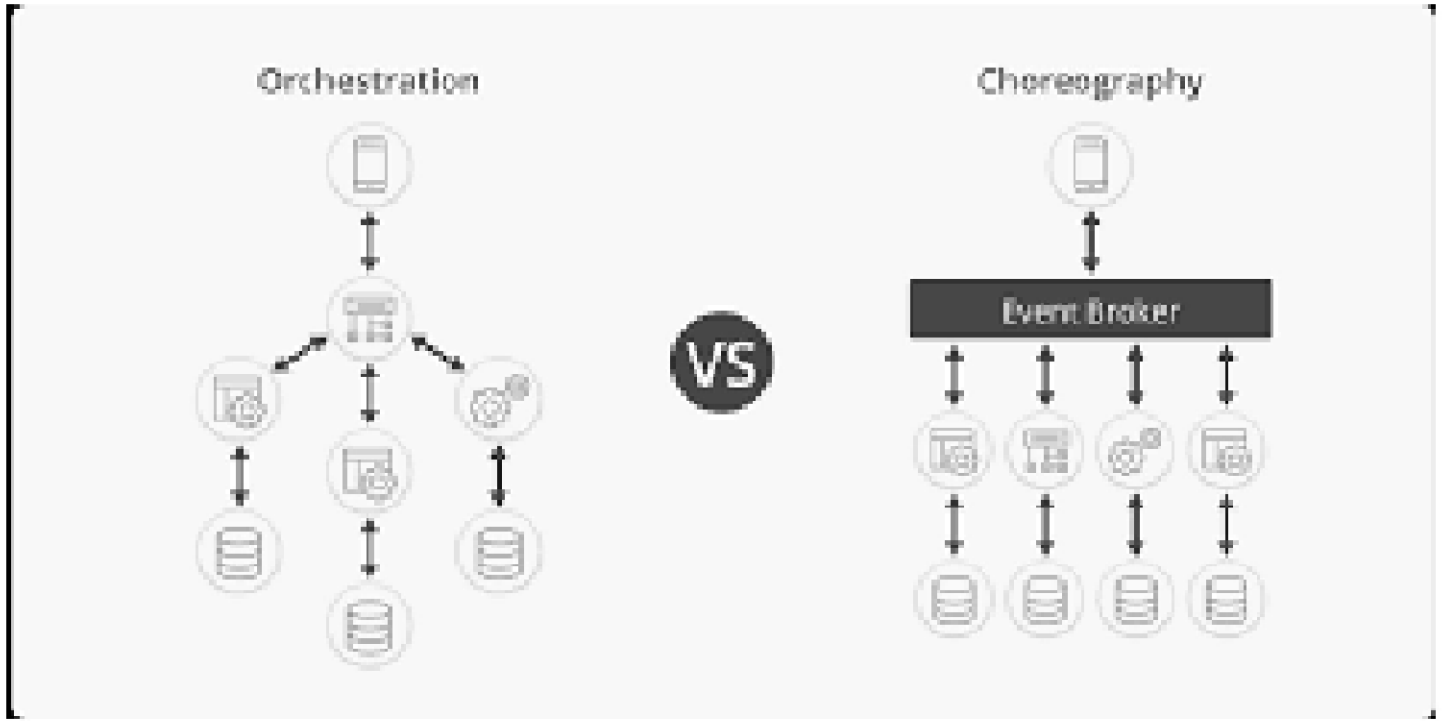
Coordonare distribuită - coregrafie



Coordonare centralizată - orchestrație



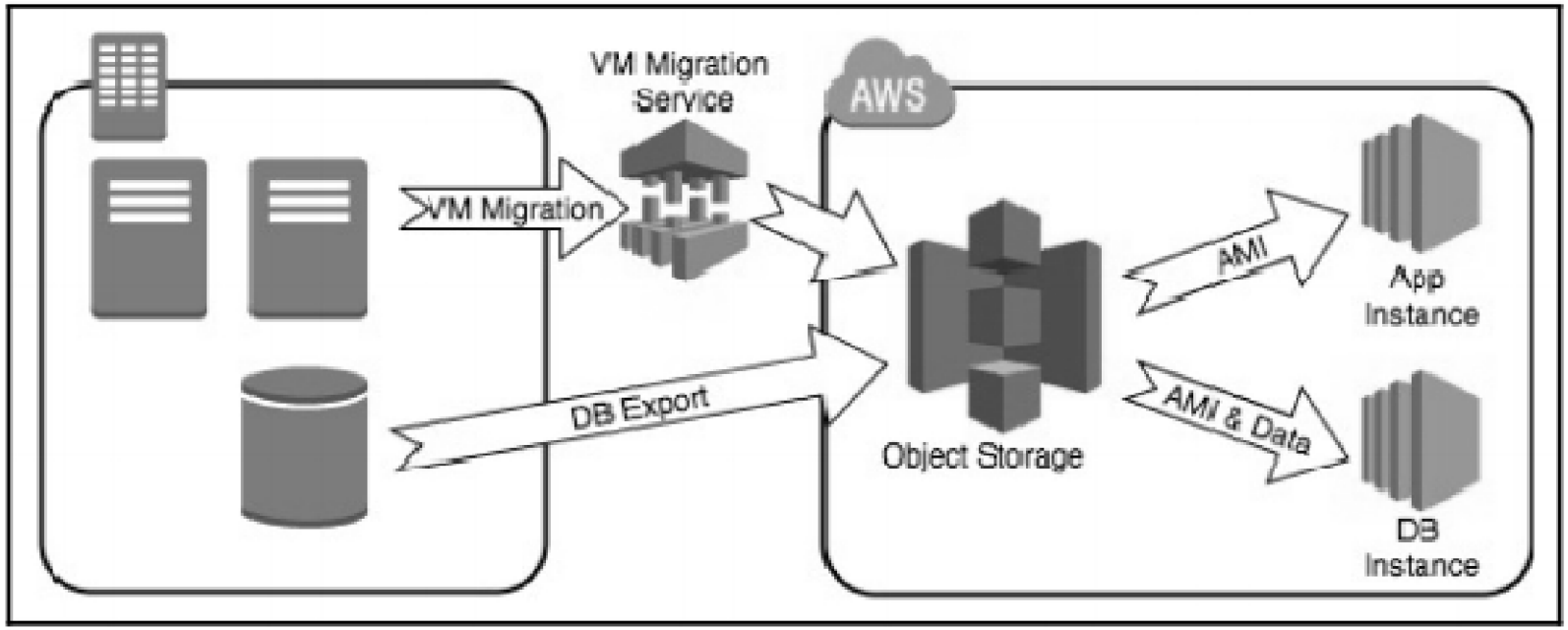
Discuție ....



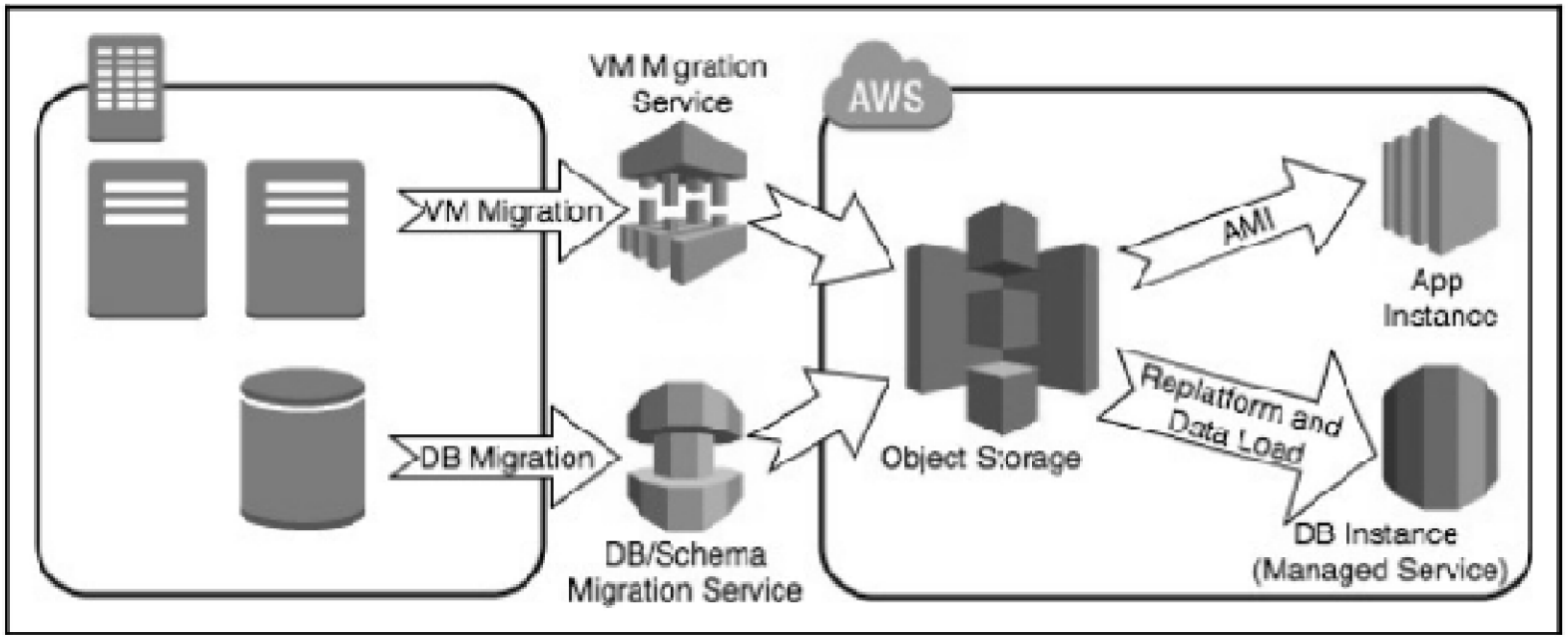
Modele de migrare a aplicațiilor

- Rehost
- Replatform
- Repurchase
- Refactor
- Retire
- Retain

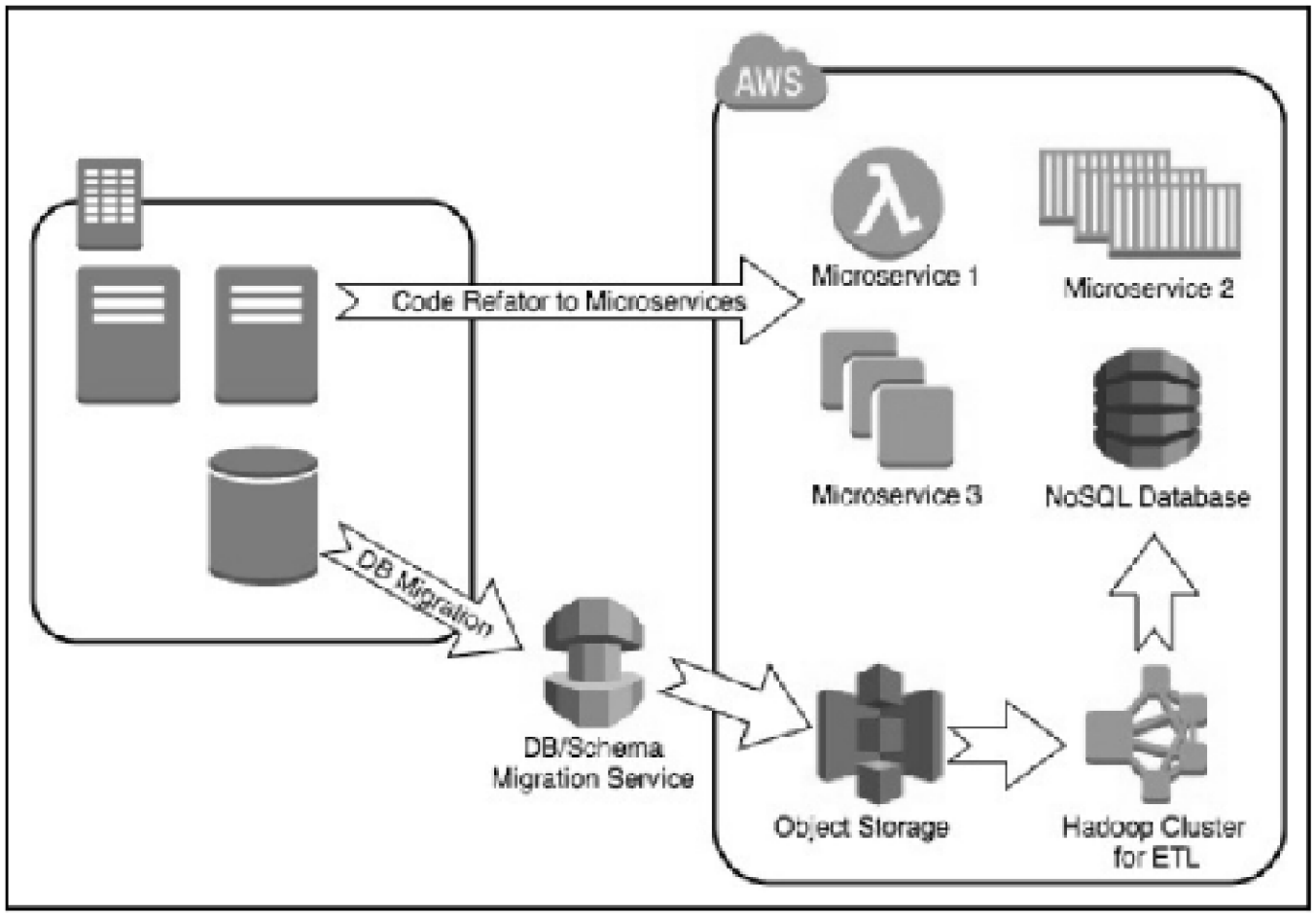
Rehost sau lift-and-shift



Replatform


VM Migration Service
AWS
VM Migration
AMI
App Instance
DB Migration
Replatform and Data Load
Object Storage
DB/Schema Migration Service
DB Instance (Managed Service)

Refactor


AWS
Code Refator to Microservices
DB Migration
Microservice 1
Microservice 2
Microservice 3
NoSQL Database
DB/Schema
Migration Service
Object Storage
Hadoop Cluster
for ETL

Mici observații privind securitate MSA

- REST?
- Oauth2
- SAML
- token