Département mathématique Informatique

Traitement parallèle en Big Data

C

|  |  |  |
| --- | --- | --- |
|  | **ECOLE NORMALE SUPERIEURE**  **DE L’ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA** |  |

**Examen**

**Professeur :**

**Mr. MOHAMED YOUSSFI**

3ème année II-BDCC

Filière d’ingénieur : Ingénieur informatique, Big Data et Cloud Computing

C

**Rapport**

**2023/2024**

**Réalisé par :**

**Ayoub ETOULLALI**

**SOMMAIRE**

[Introduction 2](#_Toc154398551)

[Travail à faire 3](#_Toc154398552)

[A. Établir une architecture technique du projet 3](#_Toc154398553)

[B. Créer un Projet Maven incluant les micro-services suivants : keynote-service, conference-service, gateway-service, discovery-service, config-service et angular-front-app 4](#_Toc154398554)

[*Conference-service* 4](#_Toc154398555)

[*Keynote-service* 5](#_Toc154398556)

[*gateway-service* 7](#_Toc154398557)

[*discovery-service* 7](#_Toc154398558)

[*config-service* 7](#_Toc154398559)

[C. Développer et tester les micro-services discovery-service et gateway-service et config-service 9](#_Toc154398560)

[*gateway-service* 9](#_Toc154398561)

[*config-service* 9](#_Toc154398562)

[*discovery-service* 9](#_Toc154398563)

[Teste 10](#_Toc154398564)

[D. Développer et tester le micro-service Keynote-service (Entities, DAO, service, DTO, Mapper, RestController) 13](#_Toc154398565)

[*keynote-service* 13](#_Toc154398566)

[Teste 14](#_Toc154398567)

[E. Développer et tester le micro-service conférence-service (Entities, DAO, service, DTO, Mapper, RestController, Client Rest Open Feign) 15](#_Toc154398568)

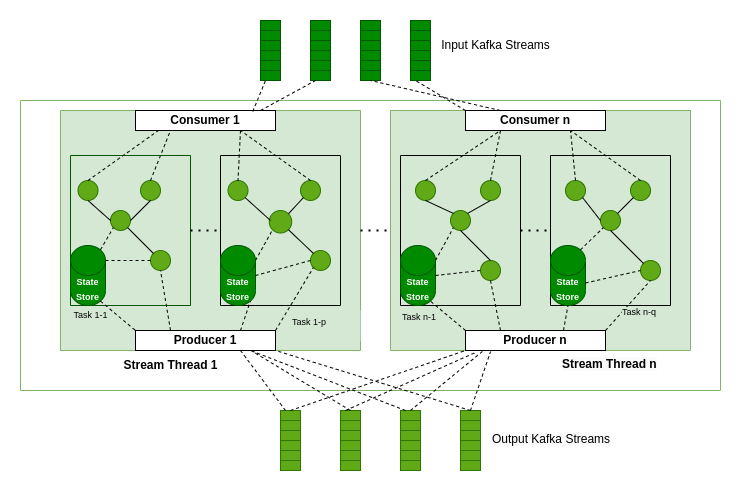
[F. Développer un simple frontend web pour l’application 19](#_Toc154398569)

[G. Sécuriser l’application avec une authentification Keycloak 25](#_Toc154398570)

[H. Déployer l’application avec Docker et Docker compose 34](#_Toc154398571)

[Conclusion 35](#_Toc154398572)

Introduction



**Le traitement parallèle occupe une place centrale dans le domaine du Big Data, où le volume, la variété et la vélocité des données posent des défis uniques. Face à cette réalité, les approches traditionnelles de traitement des données se révèlent souvent insuffisantes. C'est dans ce contexte que le traitement parallèle émerge comme une solution incontournable, permettant la distribution et l'exécution simultanée de tâches complexes sur des ressources informatiques multiples. Cette approche révolutionnaire transcende les limitations imposées par la capacité de traitement d'une seule machine, offrant une scalabilité et une efficacité sans précédent pour le traitement des ensembles de données massifs. Dans cette exploration, nous plongerons dans les fondements du traitement parallèle en Big Data, examinant les concepts clés, les modèles de programmation et les technologies qui propulsent cette révolution dans le domaine de l'informatique distribuée.**

Travail à faire

1. Établir une architecture technique du projet



Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

1. Créer un Projet Maven incluant les micro-services suivants : keynote-service, conference-service, gateway-service, discovery-service, config-service et angular-front-app

# *Conference-service*

<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-oauth2-client</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.thymeleaf.extras</groupId>  
 <artifactId>thymeleaf-extras-springsecurity6</artifactId>  
 <version>3.1.0.M1</version>  
 </dependency>  
  
 <!-- https://mvnrepository.com/artifact/org.webjars/bootstrap -->  
 <dependency>  
 <groupId>org.webjars</groupId>  
 <artifactId>bootstrap</artifactId>  
 <version>5.3.2</version>  
 </dependency>  
  
 <dependency>  
 <groupId>nz.net.ultraq.thymeleaf</groupId>  
 <artifactId>thymeleaf-layout-dialect</artifactId>  
 <version>3.2.1</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <!-- -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-rest</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-config -->  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-config</artifactId>  
 <version>4.1.0</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-consul-discovery -->  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-consul-discovery</artifactId>  
 <version>4.1.0</version>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springdoc</groupId>  
 <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>  
 <version>2.0.3</version>  
 </dependency>  
 </dependencies>

# *Keynote-service*

<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
<!-- <dependency>-->  
<!-- <groupId>org.springframework.boot</groupId>-->  
<!-- <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>-->  
<!-- </dependency>-->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- -->  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-rest</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-config -->  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-config</artifactId>  
 <version>4.1.0</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-consul-discovery -->  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-consul-discovery</artifactId>  
 <version>4.1.0</version>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springdoc</groupId>  
 <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>  
 <version>2.0.3</version>  
 </dependency>  
  
 </dependencies>

# *gateway-service*

<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-consul-discovery</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-gateway</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-config</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
</dependencies>

# *discovery-service*

<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-consul-config</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-consul-discovery</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-vault-config</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
</dependencies>

# *config-service*

<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-config-server</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-consul-discovery</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
</dependencies>

1. Développer et tester les micro-services discovery-service et gateway-service et config-service

# *gateway-service*

**Application**

@SpringBootApplication  
public class GatewayServiceApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(GatewayServiceApplication.class, args);  
 }  
  
 @Bean  
 DiscoveryClientRouteDefinitionLocator dynamicRoutes(ReactiveDiscoveryClient rdc,  
 DiscoveryLocatorProperties dlp){  
 return new DiscoveryClientRouteDefinitionLocator(rdc,dlp);  
 }  
  
}

**Application.properties**

server.port=9999  
spring.application.name=gateway-service  
spring.config.import=optional:configserver:http://localhost:8888

# *config-service*

**Application**

@SpringBootApplication  
@EnableConfigServer  
@EnableDiscoveryClient  
public class Application {  
 public static void main(String[] args) {  
 SpringApplication.*run*(Application.class, args);  
 }  
  
}

**Application.properties**

server.port=8888  
spring.application.name=config-service  
spring.cloud.config.server.git.uri=file:///C:/Tools/config-repo

# *discovery-service*

Une image contenant texte, capture d’écran, Police, multimédia

Description générée automatiquement

**consul-application**

@Bean  
CommandLineRunner commandLineRunner(){  
 return args -> {  
 //Ajouter un secret <keypair>  
 vaultTemplate.opsForVersionedKeyValue("secret")  
 .put("keypair",  
 Map.*of*("privKey","54321",  
 "pubKey","8999"));  
 };  
};

**Application.properties**

server.port=8084  
spring.application.name=config-consul  
  
management.endpoints.web.exposure.include=\*  
  
#by default  
spring.config.import=optional:consul:, vault://  
  
spring.cloud.vault.token=hvs.yn9wJcMXw3vDilCUCqtHaMTX  
spring.cloud.vault.scheme=http  
spring.cloud.vault.kv.enabled=true

# Teste

**Démarrer consul**

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Une image contenant texte, logiciel, Icône d’ordinateur, Logiciel multimédia

Description générée automatiquement

**Démarrer vault**

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant capture d’écran, texte, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, affichage, Police

Description générée automatiquement

**Config**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

1. Développer et tester le micro-service Keynote-service (Entities, DAO, service, DTO, Mapper, RestController)

# *keynote-service*

**entities**

@Entity  
@Data @NoArgsConstructor @AllArgsConstructor @Builder  
public class Keynote {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String firstName;  
 private String lastName;  
 private String email;  
 private String jobTitle;  
}

**Web RestController**

@RestController  
@RefreshScope  
public class Controller {  
 @Value("${global.params.p1}")  
 private String p1;  
 @Value("${global.params.p2}")  
 private String p2;  
 @Value("${inventory.params.x}")  
 private String x;  
 @Value("${inventory.params.y}")  
 private String y;  
 @GetMapping("/params")  
 public Map<String, String> params(){  
 return Map.*of*("p1",p1,"p2",p2,"x",x,"y",y);  
 }  
}

@RestController  
public class KeynoteRestController {  
 private KeynoteRepository keynoteRepository;  
  
 public KeynoteRestController(KeynoteRepository keynoteRepository) {  
 this.keynoteRepository = keynoteRepository;  
 }  
 @GetMapping("/keynotes")  
// @PreAuthorize("hasAuthority('ADMIN')")  
 public List<Keynote> products(){  
 return keynoteRepository.findAll();  
 }  
}

**Application**

@Bean  
CommandLineRunner start(KeynoteRepository keynoteRepository){  
 return args -> {  
 Random random=new Random();  
 for (int i = 1; i <10 ; i++) {  
 keynoteRepository.saveAll(List.*of*(  
 Keynote.*builder*()  
 .firstName("user "+i)  
 .email("email" + 1)  
 .jobTitle("jobTitle" + 1)  
 .build()  
 ));  
 }  
  
 };  
}

**application.properties**

server.port=8082  
spring.application.name=keynote-service  
spring.config.import=optional:configserver:http://localhost:8888  
  
#demande Actuator au démarrage, expose tout (refresh, slash..)  
management.endpoints.web.exposure.include=\*

# Teste

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

1. Développer et tester le micro-service conférence-service (Entities, DAO, service, DTO, Mapper, RestController, Client Rest Open Feign)

**entities**

@Entity  
@NoArgsConstructor @AllArgsConstructor @Getter @Setter @ToString @Builder  
public class Conference {  
 private Long id;  
 private String titre;  
 private ConferenceType type; // académique ou commerciale  
 private Date date;  
 private int duree; // en minutes  
 private int nombreInscrits;  
 private int score;  
  
 // Relation avec les reviews  
 @OneToMany(mappedBy = "conference", cascade = CascadeType.*ALL*)  
 private List<Review> reviews;  
}

**Enum**

public enum ConferenceType {  
 *ACADEMIC*,  
 *COMMERCIAL*}

**model**

@NoArgsConstructor @AllArgsConstructor @Getter @Setter @Builder  
public class Keynote {  
 private Long id;  
 private String firstName;  
 private String lastName;  
 private String email;  
 private String jobTitle;  
}

**Web RestController**

@RestController  
@RefreshScope  
public class Controller {  
 @Value("${global.params.p1}")  
 private String p1;  
 @Value("${global.params.p2}")  
 private String p2;  
 @Value("${conference.params.x}")  
 private String x;  
 @Value("${conference.params.y}")  
 private String y;  
 @GetMapping("/params")  
 public Map<String, String> params(){  
 return Map.*of*("p1",p1,"p2",p2,"x",x,"y",y);  
 }  
}

@Controller  
public class ConferenceController {  
 private ConferenceRepository conferenceRepository;  
 private ClientRegistrationRepository clientRegistrationRepository;  
  
 public ConferenceController(ConferenceRepository conferenceRepository, ClientRegistrationRepository clientRegistrationRepository) {  
 this.conferenceRepository = conferenceRepository;  
 this.clientRegistrationRepository = clientRegistrationRepository;  
 }  
  
 @GetMapping("/index")  
 @PreAuthorize("hasAuthority('USER')")  
 public String index() {  
 return "index";  
 }  
  
 @GetMapping("/")  
 public String index1() {  
 return "index";  
 }  
  
 @GetMapping("/conferences")  
 @PreAuthorize("hasAuthority('ADMIN')")  
 public String conferences(Model model) {  
 List<Conference> conferenceList = conferenceRepository.findAll();  
 model.addAttribute("conferences", conferenceList);  
 return "conferences";  
 }  
  
 @GetMapping("/keynotes")  
 public String keynotes(Model model) {  
 SecurityContext context = SecurityContextHolder.*getContext*();  
 Authentication authentication = context.getAuthentication();  
 OAuth2AuthenticationToken oAuth2AuthenticationToken= (OAuth2AuthenticationToken) authentication;  
 DefaultOidcUser oidcUser = (DefaultOidcUser) oAuth2AuthenticationToken.getPrincipal();  
 String jwtTokenValue=oidcUser.getIdToken().getTokenValue();  
 RestClient restClient = RestClient.*create*("http://localhost:8082");  
 List<Keynote> keynotes = restClient.get()  
 .uri("/keynotes")  
 .headers(httpHeaders -> httpHeaders.set(HttpHeaders.*AUTHORIZATION*, "Bearer "+jwtTokenValue))  
 .retrieve()  
 .body(new ParameterizedTypeReference<>(){});  
 model.addAttribute("keynotes", keynotes);  
 return "keynotes";  
 }  
  
 @GetMapping("/auth")  
 @ResponseBody  
 public Authentication authentication(Authentication authentication) {  
 return authentication;  
 }  
  
 @GetMapping("/notAutorized")  
 public String notAutorized() {  
 return "notAuthorized";  
 }  
  
 @GetMapping("/oauth2Login")  
 public String oauth2Login(Model model) {  
 String authorizationRequestBaseUri = "oauth2/authorization";  
 Map<String, String> oauth2AuthenticationUrls = new HashMap();  
 Iterable<ClientRegistration> clientRegistrations = (Iterable<ClientRegistration>) clientRegistrationRepository;;  
 clientRegistrations.forEach(registration ->{  
 oauth2AuthenticationUrls.put(registration.getClientName(),  
 authorizationRequestBaseUri + "/" + registration.getRegistrationId());  
 });  
 model.addAttribute("urls", oauth2AuthenticationUrls);  
 return "oauth2Login";  
 }  
}

**Application**

@SpringBootApplication  
public class ConferenceFrontThymeleafAppApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(ConferenceFrontThymeleafAppApplication.class, args);  
 }  
  
 @Bean  
 CommandLineRunner commandLineRunner(ConferenceRepository conferenceRepository){  
 return args -> {  
 conferenceRepository.save(Conference.*builder*().titre("conference 1").type(ConferenceType.*ACADEMIC*).duree(60).build());  
 conferenceRepository.save(Conference.*builder*().titre("conference 2").type(ConferenceType.*ACADEMIC*).duree(120).build());  
 conferenceRepository.save(Conference.*builder*().titre("conference 3").type(ConferenceType.*COMMERCIAL*).duree(60).build());  
 conferenceRepository.save(Conference.*builder*().titre("conference 4").type(ConferenceType.*COMMERCIAL*).duree(30).build());  
   
 };  
 }  
}

**application.properties**

spring.datasource.url=jdbc:h2:mem:Conference-db  
spring.h2.console.enabled=true  
  
  
spring.security.oauth2.client.registration.google.clientId=260777653322-4oagm6rn1qtnrm0ubp98o2jia5cnbb89.apps.googleusercontent.com  
spring.security.oauth2.client.registration.google.client-secret=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  
spring.security.oauth2.client.provider.google.user-name-attribute=email  
  
spring.security.oauth2.client.registration.github.clientId=edb49a0a8c36f1e84464  
spring.security.oauth2.client.registration.github.client-secret=2bb2ad89fb0440cfb5257aa9740ac7fe8f6be4af   
spring.security.oauth2.client.provider.github.user-name-attribute=login  
  
  
spring.security.oauth2.client.registration.keycloak.client-name=keycloak  
spring.security.oauth2.client.registration.keycloak.client-id=enset-Conference-client  
spring.security.oauth2.client.registration.keycloak.client-secret=sRrSOVPk2EuOdeMzkFTZhuoCv36ArVwk  
spring.security.oauth2.client.registration.keycloak.scope=openid,profile,email,offline\_access  
spring.security.oauth2.client.registration.keycloak.authorization-grant-type=authorization\_code  
spring.security.oauth2.client.registration.keycloak.redirect-uri=http://localhost:8081/login/oauth2/code/enset-Conference-client  
spring.security.oauth2.client.provider.keycloak.issuer-uri=http://localhost:8080/realms/enset-realm  
  
spring.security.oauth2.client.provider.keycloak.user-name-attribute=preferred\_username  
#spring.security.oauth2.resourceserver.jwt.issuer-uri=http://localhost:8080/realms/ebank-realm  
  
###  
  
server.port=8081  
spring.application.name=conference-service  
spring.config.import=optional:configserver:http://localhost:8888  
  
#demande Actuator au démarrage, expose tout (refresh, slash..)  
management.endpoints.web.exposure.include=\*

**Test**

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

1. Développer un simple frontend web pour l’application

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

**Conferences**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 layout:decorate="template"  
>  
<head>  
 <meta charset="UTF-8">  
 <title>Conference</title>  
 <link rel="stylesheet" href="/webjars/bootstrap/5.3.2/css/bootstrap.min.css">  
</head>  
<body>  
 <div layout:fragment="content">  
 <div class="ps-3">  
 <div class="card">  
 <div class="card-body">  
 <table class="table">  
 <thead>  
 <tr>  
 <th>ID</th><th>Titre</th><th>Type</th><th>Date</th><th>Durée</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="c:${conferences}">  
 <td th:text="${c.id}"></td>  
 <td th:text="${c.titre}"></td>  
 <td th:text="${c.type}"></td>  
 <td th:text="${c.date}"></td>  
 <td th:text="${c.duree}"></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 </div>  
 </div>  
 </div>  
  
</body>  
</html>

**Keyno**tes

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 layout:decorate="template"  
>  
<head>  
 <meta charset="UTF-8">  
 <title>Keynote</title>  
 <link rel="stylesheet" href="/webjars/bootstrap/5.3.2/css/bootstrap.min.css">  
</head>  
<body>  
 <div layout:fragment="content">  
 <div class="ps-3">  
 <div class="card">  
 <div class="card-body">  
 <table class="table">  
 <thead>  
 <tr>  
 <th>ID</th><th>First Name</th><th>Email</th><th>Job Title</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="p:${keynotes}">  
 <td th:text="${p.id}"></td>  
 <td th:text="${p.firstName}"></td>  
 <td th:text="${p.email}"></td>  
 <td th:text="${p.jobTitle}"></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 </div>  
 </div>  
 </div>  
  
</body>  
</html>

**Index**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 layout:decorate="template"  
>  
<head>  
 <meta charset="UTF-8">  
 <title>Customers</title>  
 <link rel="stylesheet" href="/webjars/bootstrap/5.3.2/css/bootstrap.min.css">  
</head>  
<body>  
<div layout:fragment="content">  
 <div class="ps-3">  
 <div class="card">  
 <div class="card-body">  
 <div>  
 <h1>Welcome to the dashboard </h1>  
 <p>....</p>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
</body>  
</html>

**Template**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">  
<head>  
 <meta charset="UTF-8">  
 <title>Template Conference</title>  
 <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/5.3.2/css/bootstrap.min.css">  
 <script src="webjars/bootstrap/5.3.2/js/bootstrap.bundle.js"></script>  
</head>  
<body>  
<nav class="navbar navbar-expand-lg bg-dark navbar-dark">  
 <div class="container-fluid">  
 <a class="navbar-brand" href="https://ayoub-etoullali.netlify.app/">  
 <img src="data:image/jpeg;base64,"  
 alt="Avatar Logo" style="width:40px;" class="rounded-pill">  
 </a>  
 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">  
 <span class="navbar-toggler-icon"></span>  
 </button>  
 <div style="color: white" class="collapse navbar-collapse" id="navbarSupportedContent">  
 <ul class="navbar-nav me-auto mb-2 mb-lg-0">  
 <li class="nav-item">  
 <a class="nav-link active" aria-current="page" th:href="@{/index}">Home</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link" th:href="@{/conferences}">Conferences</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link" th:href="@{/keynotes}">Keynotes</a>  
 </li>  
  
 <li class="nav-item">  
 <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>  
 </li>  
 </ul>  
 <ul class="navbar">  
 <li class="nav-item dropdown">  
 <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">  
 <span th:text="${#authentication.name}"></span>  
 </a>  
 <ul class="dropdown-menu" aria-labelledby="navbarDropdown">  
 <li><a class="dropdown-item" th:href="@{/login}">Login</a></li>  
 <li>  
 <form th:action="@{/logout}" method="post">  
 <button class="btn btn-success">Logout</button>  
 </form>  
 </li>  
 <li><hr class="dropdown-divider"></li>  
 <li><a class="dropdown-item" href="#">Profile</a></li>  
 </ul>  
 </li>  
 </ul>  
 </div>  
 </div>  
</nav>  
<div layout:fragment="content">  
  
</div>  
</body>  
</html>

**Oauth2login**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 layout:decorate="template"  
>  
<head>  
 <meta charset="UTF-8">  
 <title>Customers</title>  
 <link rel="stylesheet" href="/webjars/bootstrap/5.3.2/css/bootstrap.min.css">  
</head>  
<body>  
 <div layout:fragment="content">  
 <div class="ps-3">  
 <div class="card">  
 <div class="card-body">  
 <ul class="list-group">  
 <li class="list-group-item" th:each="url:${urls}">  
 <div class="d-grid">  
 <a class="btn btn-outline-info" th:text="${url.key}" th:href="${url.value}"></a>  
 </div>  
 </li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
  
</body>  
</html>

n*otAuthorized*

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 layout:decorate="template"  
>  
<head>  
 <meta charset="UTF-8">  
 <title>Customers</title>  
 <link rel="stylesheet" href="/webjars/bootstrap/5.3.2/css/bootstrap.min.css">  
</head>  
<body>  
 <div layout:fragment="content">  
 <div class="ps-3">  
 <div class="card">  
 <div class="card-body">  
 <h2 class="text-danger">Not authorized</h2>  
 </div>  
 </div>  
 </div>  
 </div>  
  
</body>  
</html>

1. Sécuriser l’application avec une authentification Keycloak

**Coference**

package ma.enset.conferencefrontthymeleafapp.sec;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.Customizer;  
import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.authority.mapping.GrantedAuthoritiesMapper;  
import org.springframework.security.oauth2.client.oidc.web.logout.OidcClientInitiatedLogoutSuccessHandler;  
import org.springframework.security.oauth2.client.registration.ClientRegistrationRepository;  
import org.springframework.security.oauth2.core.oidc.user.OidcUserAuthority;  
import org.springframework.security.oauth2.core.user.OAuth2UserAuthority;  
import org.springframework.security.web.SecurityFilterChain;  
  
import java.util.\*;  
  
@Configuration  
@EnableWebSecurity  
@EnableMethodSecurity(prePostEnabled = true)  
public class SecurityConfig {  
 private ClientRegistrationRepository clientRegistrationRepository;  
  
 public SecurityConfig(ClientRegistrationRepository clientRegistrationRepository) {  
 this.clientRegistrationRepository = clientRegistrationRepository;  
 }  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 return http  
 .csrf(Customizer.*withDefaults*())  
 .authorizeHttpRequests(ar->ar.requestMatchers("/","/oauth2Login/\*\*","/webjars/\*\*","/h2-console/\*\*").permitAll())  
 .authorizeHttpRequests(ar->ar.anyRequest().authenticated())  
 .headers(h->h.frameOptions(fo->fo.disable()))  
 .csrf(csrf->csrf.ignoringRequestMatchers("/h2-console/\*\*"))  
 .oauth2Login(al->  
 al.loginPage("/oauth2Login")  
 .defaultSuccessUrl("/")  
 )  
 .logout((logout) -> logout  
 .logoutSuccessHandler(oidcLogoutSuccessHandler())  
 .logoutSuccessUrl("/").permitAll()  
 .clearAuthentication(true)  
 .deleteCookies("JSESSIONID"))  
 .exceptionHandling(eh->eh.accessDeniedPage("/notAutorized"))  
 .build();  
 }  
 private OidcClientInitiatedLogoutSuccessHandler oidcLogoutSuccessHandler() {  
 final OidcClientInitiatedLogoutSuccessHandler oidcLogoutSuccessHandler =  
 new OidcClientInitiatedLogoutSuccessHandler(this.clientRegistrationRepository);  
 oidcLogoutSuccessHandler.setPostLogoutRedirectUri("{baseUrl}?logoutsuccess=true");  
 return oidcLogoutSuccessHandler;  
 }  
  
 @Bean  
 public GrantedAuthoritiesMapper userAuthoritiesMapper() {  
 return (authorities) -> {  
 final Set<GrantedAuthority> mappedAuthorities = new HashSet<>();  
 authorities.forEach((authority) -> {  
 if (authority instanceof OidcUserAuthority oidcAuth) {  
 mappedAuthorities.addAll(mapAuthorities(oidcAuth.getIdToken().getClaims()));  
 System.*out*.println(oidcAuth.getAttributes());  
 } else if (authority instanceof OAuth2UserAuthority oauth2Auth) {  
 mappedAuthorities.addAll(mapAuthorities(oauth2Auth.getAttributes()));  
 }  
 });  
 return mappedAuthorities;  
 };  
 }  
 private List<SimpleGrantedAuthority> mapAuthorities(final Map<String, Object> attributes) {  
 final Map<String, Object> realmAccess = ((Map<String, Object>)attributes.getOrDefault("realm\_access", Collections.*emptyMap*()));  
 final Collection<String> roles = ((Collection<String>)realmAccess.getOrDefault("roles", Collections.*emptyList*()));  
 return roles.stream()  
 .map((role) -> new SimpleGrantedAuthority(role))  
 .toList();  
 }  
}

**keyknote**

JwtAuthConverter

@Component  
public class JwtAuthConverter implements Converter<Jwt, AbstractAuthenticationToken> {  
 private final JwtGrantedAuthoritiesConverter jwtGrantedAuthoritiesConverter=new JwtGrantedAuthoritiesConverter();  
  
  
 @Override  
 public AbstractAuthenticationToken convert(Jwt jwt) {  
 Collection<GrantedAuthority> authorities = Stream.*concat*(  
 jwtGrantedAuthoritiesConverter.convert(jwt).stream(),  
 extractResourceRoles(jwt).stream()  
 ).collect(Collectors.*toSet*());  
 return new JwtAuthenticationToken(jwt, authorities,jwt.getClaim("preferred\_username"));  
 }  
  
 private Collection<GrantedAuthority> extractResourceRoles(Jwt jwt) {  
 Map<String , Object> realmAccess;  
 Collection<String> roles;  
 if(jwt.getClaim("realm\_access")==null){  
 return Set.*of*();  
 }  
 realmAccess = jwt.getClaim("realm\_access");  
 roles = (Collection<String>) realmAccess.get("roles");  
 return roles.stream().map(role->new SimpleGrantedAuthority(role)).collect(Collectors.*toSet*());  
 }

SecurityConfig

@Configuration  
@EnableWebSecurity  
@EnableMethodSecurity(prePostEnabled = true)  
public class SecurityConfig {  
 private JwtAuthConverter jwtAuthConverter;  
  
 public SecurityConfig(JwtAuthConverter jwtAuthConverter) {  
 this.jwtAuthConverter = jwtAuthConverter;  
 }  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 return http  
 .authorizeHttpRequests(ar->ar.anyRequest().authenticated())  
 .oauth2ResourceServer(o2->o2.jwt(jwt->jwt.jwtAuthenticationConverter(jwtAuthConverter)))  
 .headers(h->h.frameOptions(fo->fo.disable()))  
 .csrf(csrf->csrf.ignoringRequestMatchers("/h2-console/\*\*"))  
 .build();  
 }  
}

**Interfaces**

Conference : USER, ADMIN

Keynote : ADMIN

Google

Une image contenant texte, logiciel, nombre, capture d’écran

Description générée automatiquement

Github

Une image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

Description générée automatiquement

Keycloak

Une image contenant texte, logiciel, Icône d’ordinateur, Logiciel multimédia

Description générée automatiquement

Auth

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Postman

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Jwt

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Theamleaf

Home

Une image contenant texte, capture d’écran

Description générée automatiquement

authentification

Une image contenant texte, logiciel, nombre, Page web

Description générée automatiquement

Google (USER)

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Une image contenant texte, logiciel, capture d’écran

Description générée automatiquement

Github (USER)

Une image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

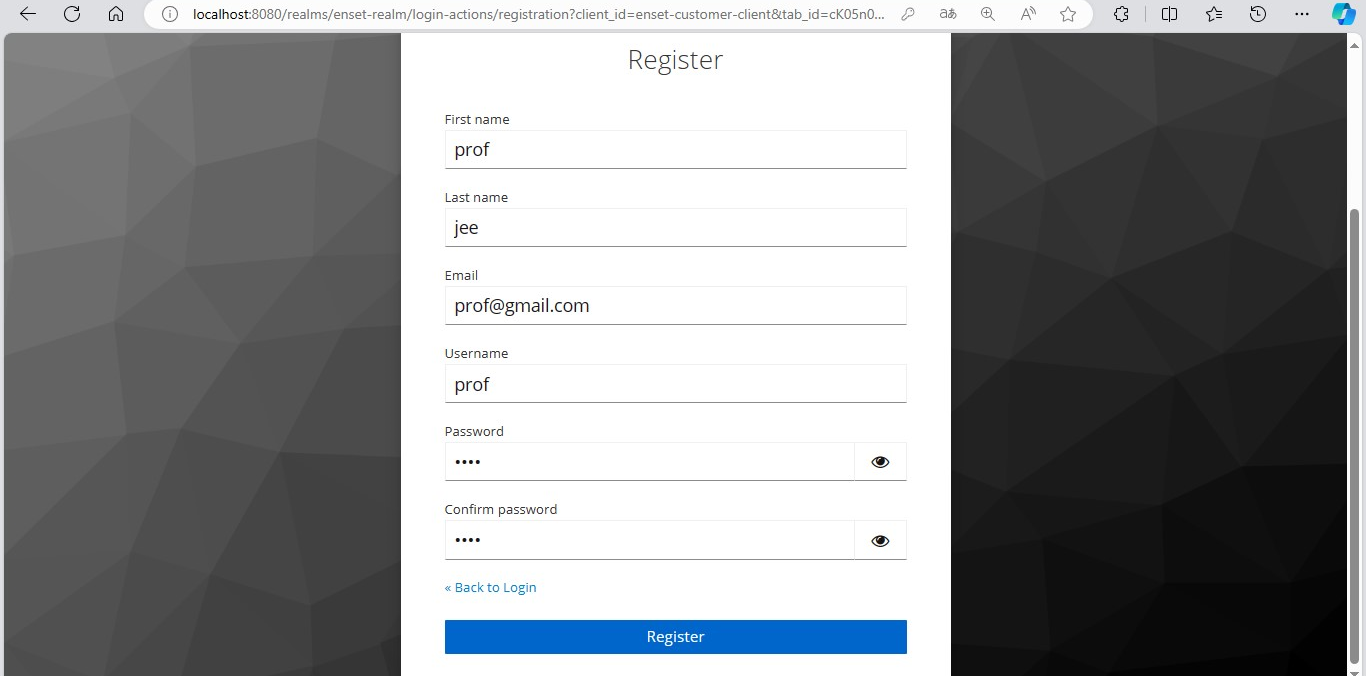
Description générée automatiquement

keycloak

Une image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

Description générée automatiquement

Ajouter user



Keycloak

Prof (USER)



Une image contenant texte, logiciel, nombre, Police

Description générée automatiquement

Une image contenant texte, Police, logiciel, capture d’écran

Description générée automatiquement

Ayoub (ADMIN)

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

1. Déployer l’application avec Docker et Docker compose

**En cour**

Conclusion

**En conclusion, le traitement parallèle se dresse comme un pilier indispensable pour débloquer le potentiel du Big Data. Les défis posés par la gestion des données massives sont vastes, mais le traitement parallèle offre une réponse robuste, permettant de tirer parti de la puissance collective de multiples ressources. Des modèles de programmation tels que MapReduce ont ouvert la voie, tandis que des frameworks modernes tels que Spark et Flink ont apporté des améliorations significatives en termes de vitesse et de flexibilité. Les bases de données parallèles, les systèmes de fichiers distribués et l'utilisation judicieuse des GPU enrichissent encore davantage cette approche. À l'aube d'une ère où les données continuent de croître de manière exponentielle, le traitement parallèle en Big Data demeure une clé maîtresse, ouvrant la porte à une compréhension plus approfondie, à des insights plus rapides et à des découvertes qui façonnent l'avenir de l'informatique et de l'analyse des données.**