Raport Systemy Rozproszone
Lab 4 Akka
Radosław Kopeć 305333

Zadanie 1

Po implementacji MathActorDivide i zakomentowaniu zbędnych printów:
Testy dla sekwencji operacji:

```
system.tell(new MathActor.MathCommandMultiply(5, 3, null));
system.tell(new MathActor.MathCommandMultiply(5, 2, null));
system.tell(new MathActor.MathCommandDivide(15, 3, null));
system.tell(new MathActor.MathCommandDivide(15, 5, null));
system.tell(new MathActor.MathCommandDivide(15, 0, null));
Thread.sleep(2000);
system.tell(new MathActor.MathCommandMultiply(5, 3, null));
system.tell(new MathActor.MathCommandMultiply(5, 2, null));
system.tell(new MathActor.MathCommandDivide(15, 3, null));
system.tell(new MathActor.MathCommandDivide(15, 5, null));
```

Strategia Domyślna Stop:

Możemy zauważyć, że aktor divide zatrzymał się i już nie przetwarza wiadomości. Aktor Multiply działa dalej.

Strategia Resume:

```
C:\Users\radek\.jdks\openjdk-16\bin\java.exe ...
Z1 main: started
[actorMath-akka.actor.default-dispatcher-5] INFO akka.event.slf4j.Slf4jLogger - Slf4jLogger started
Z1 main: actor system ready
actorMultiply: I made 1 operations
ActorDivide: I made 1 operations
actorMultiply: I made 2 operations
ActorDivide: I made 2 operations
[actorMath-akka.actor.default-dispatcher-3] ERROR akka.actor.typed.Behavior$ - Supervisor ResumeSupervisor saw failure: / by zero
java.lang.ArithmeticException Create breakpoint : / by zero
	at Z1.MathActorDivide.onMathCommandDivide(MathActorDivide.java:31)
	at akka.actor.typed.javadsl.BuiltReceive.receive(ReceiveBuilder.scala:213)
	at akka.actor.typed.javadsl.BuiltReceive.receiveMessage(ReceiveBuilder.scala:204)
	at akka.actor.typed.javadsl.Receive.receive(Receive.scala:53)
	at akka.actor.typed.javadsl.AbstractBehavior.receive(AbstractBehavior.scala:64)
	at akka.actor.typed.Behavior$.interpret(Behavior.scala:274)
	at akka.actor.typed.Behavior$.interpretMessage(Behavior.scala:230)
	at akka.actor.typed.internal.InterceptorImpl$$anon$2.apply(InterceptorImpl.scala:57)
	at akka.actor.typed.internal.SimpleSupervisor.aroundReceive(Supervision.scala:124)
	at akka.actor.typed.internal.InterceptorImpl.receive(InterceptorImpl.scala:85)
	at akka.actor.typed.Behavior$.interpret(Behavior.scala:274)
	at akka.actor.typed.Behavior$.interpretMessage(Behavior.scala:230)
	at akka.actor.typed.internal.adapter.ActorAdapter.handleMessage(ActorAdapter.scala:131)
	at akka.actor.typed.internal.adapter.ActorAdapter.aroundReceive(ActorAdapter.scala:107)
	at akka.actor.ActorCell.receiveMessage(ActorCell.scala:579)
	at akka.actor.ActorCell.invoke(ActorCell.scala:547)
	at akka.dispatch.Mailbox.processMailbox(Mailbox.scala:270)
	at akka.dispatch.Mailbox.run(Mailbox.scala:231)
	at akka.dispatch.Mailbox.exec(Mailbox.scala:243) <5 internal calls>
Z1 main: sending second package of messages
Z1 main: messages send
```

```
	at akka.actor.typed.Behavior$.interpret(Behavior.scala:274)
	at akka.actor.typed.Behavior$.interpretMessage(Behavior.scala:230)
	at akka.actor.typed.internal.adapter.ActorAdapter.handleMessage(ActorAdapter.scala:131)
	at akka.actor.typed.internal.adapter.ActorAdapter.aroundReceive(ActorAdapter.scala:107)
	at akka.actor.ActorCell.receiveMessage(ActorCell.scala:579)
	at akka.actor.ActorCell.invoke(ActorCell.scala:547)
	at akka.dispatch.Mailbox.processMailbox(Mailbox.scala:270)
	at akka.dispatch.Mailbox.run(Mailbox.scala:231)
	at akka.dispatch.Mailbox.exec(Mailbox.scala:243) <5 internal calls>
Z1 main: sending second package of messages
Z1 main: messages send
actorMultiply: I made 3 operations
actorMultiply: I made 4 operations
ActorDivide: I made 3 operations
ActorDivide: I made 4 operations
|
```

Aktor Multiply działał dobrze, niezależnie od aktora Divide.
Aktor Divide po nieudanej operacji wznowił działanie, jego licznik
się nie zmienił i policzył tylko udane operacje.

Strategia Restart:

Aktor Multiply działał dobrze, niezależnie od aktora Divide.
Aktor Divide po nieudanej operacji wznowił działanie, ale jego
licznik operacji wyzerował się.

Wnioski:
- Strategia STOP - zatrzymuje aktora uniemożliwiając dalsze
  przetwarzanie wiadomości.
- Strategia RESUME- po prostu ignoruje błąd, nie zmienia stanu
  aktora.

- Strategia RESTART - restartuje aktora, inicjalizuje jego stan
  na nowo.

Zadanie 2

a)

Po zmodyfikowaniu ActorTextService:

b)

Z2_NodeB

Z2_NodeA

Wnioski:
Komunikacja w klastrze przebiegła pomyślnie. Do jej przeprowadzenia potrzebny był Recepcjonista który posiada informacje o wszystkich dołączonych do niego aktorach.