AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Akka

http://home.agh.edu.pl/~fmal/akka

Filip Malawski

fmal@agh.edu.pl

The Reactive Manifesto

- http://www.reactivemanifesto.org/
- Reactive Systems
  - Responsive – niskie czasy odpowiedzi
  - Resilient – system pozostaje responsywny po awarii
  - Elastic – system jest odporny na zmianę obciążenia
  - Message Driven - luźne powiązania, izolacja, transparentność lokalizacji

Akka

- Platforma do tworzenia aplikacji zgodnie z Reactive Manifesto
- Najważniejsze cechy:
  - Aktorzy – wątki komunikujące się przez wiadomości
  - Obsługa błędów – wysokopoziomowe mechanizmy obsługi błędów
  - Transparentność lokalizacji – aktorzy mogą być zdalni
  - Skalowalność: Scale up – równoległość; Scale out – rozproszenie

https://doc.akka.io/docs/akka/current/index-actors.html

Aktorzy

- Wszystkie działania wykonywane są przez aktorów
- Aktor posiada:
  - stan oraz zachowanie (podobnie jak obiekt)
  - własny wątek w którym jest wykonywany
  - wątki aktorów są bardzo lekkie
- Komunikacja z Aktorem następuje tylko poprzez wiadomości
  - każdy aktor posiada swoją skrzynkę pocztową
  - skrzynka to kolejka wiadomości

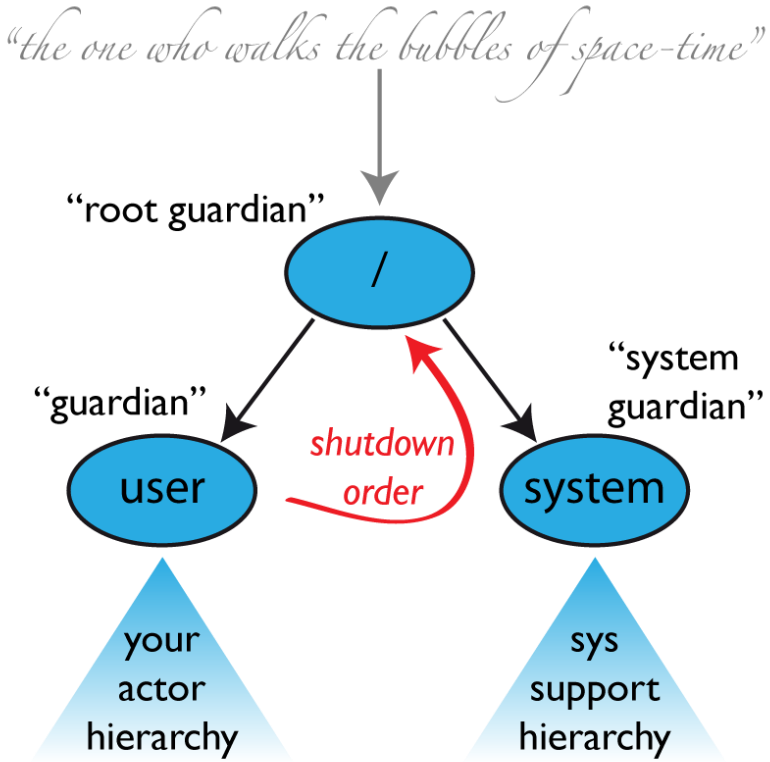
Aktorzy

- Aktorzy ułożeni są w hierarchię
  - Struktura drzewa
  - Aktor może tworzyć nowych Aktorów, którymi zarządza
  - Każdy Aktor ma dokładnie jednego rodzica (supervisora)
  - Supervisor decyduje o tym jak zostaną obsłużone błędy u jego podwładnych

https://doc.akka.io/docs/akka/current/actors.html

Struktura Aktorów

- Aktorzy top-level tworzeni przy starcie:
  - root, user, system



Obsługa błędów

- Jeśli Aktor zgłosi błąd (wyjątek), wtedy jego supervisor postępuje zgodnie z jedną ze strategii:
  - Resume – wznawia działanie podwładnego, zachowując jego stan
  - Restart – restartuje podwładnego, kasując jego stan
  - Stop – zatrzymuje działanie podwładnego
  - Escalate – zgłasza wyjątek do swojego supervisora

Obsługa błędów c.d.

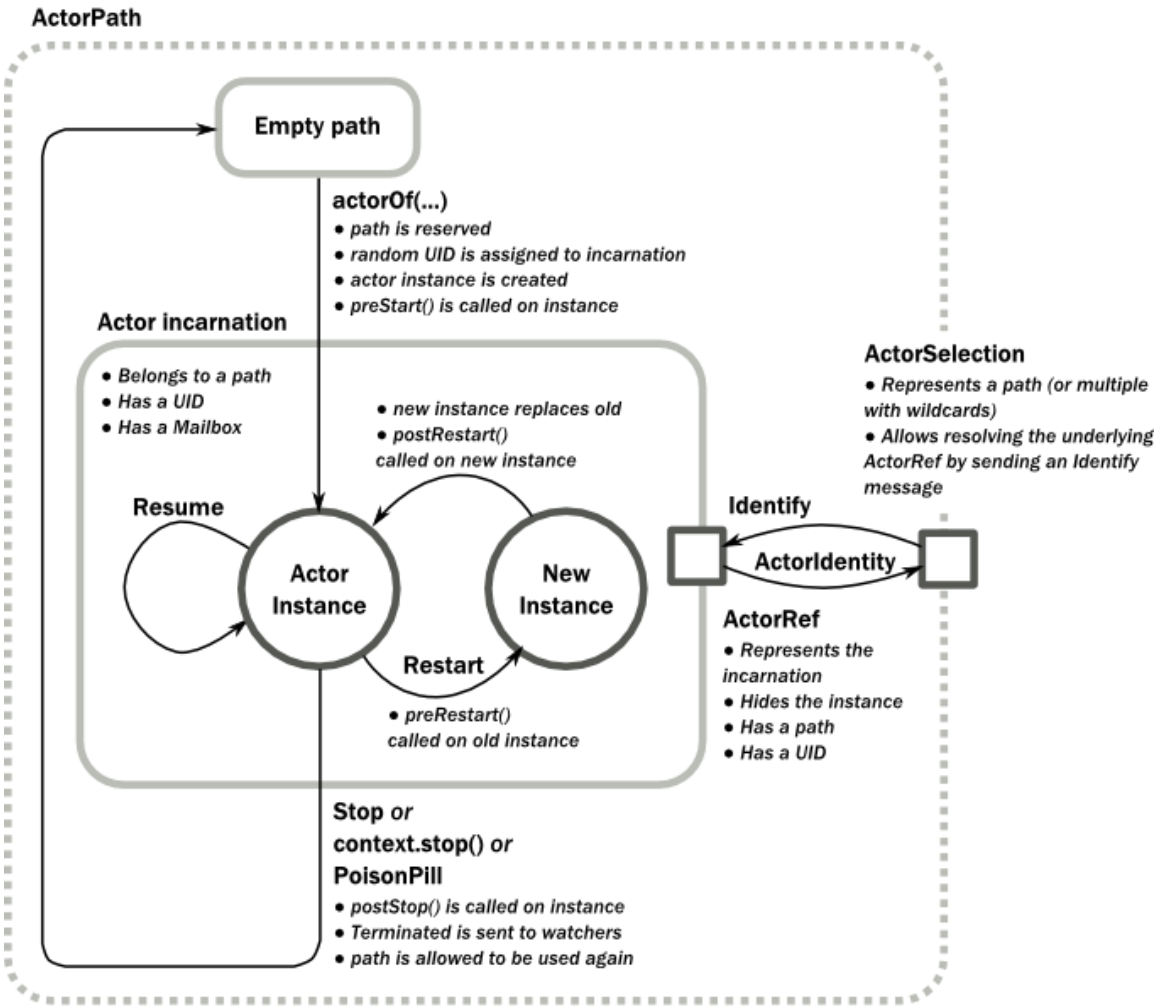
- Jeśli root zgłosi wyjątek, system jest zatrzymywany
- Strategia obsługi wyjątku u aktora wpływa też na jego wszystkich podwładnych
- Strategia może być:
  - OneForOne – uruchamiana tylko dla podwładnego, który zgłosił błąd
  - AllForOne – uruchamiana dla wszystkich podwładnych (mimo, że błąd zgłosił tylko jeden)

https://doc.akka.io/docs/akka/current/fault-tolerance.html

Referencje oraz ścieżki

- Aktor dostępny jest przez ActorRef
  - referencja do Aktora
  - nie mamy dostępu bezpośrednio do Aktora, więc może się on znajdować w różnych lokalizacjach (lokalnie lub zdalnie)
- Aktor znajduje się w drzewie
  - Można podać jego ścieżkę
  - Ścieżka wskazuje na inkarnację Aktora
  - Jeśli zrobimy restart to jest nowa instancja, ale ta sama inkarnacja
  - Ścieżki logiczne oraz fizyczne – mogą być różne, jeśli Aktor jest zdalny

Cykl życia Aktora



Przykład – Z1

```
// create actor system & actors
final ActorSystem system = ActorSystem.create("local_system");
final ActorRef actor = system.actorOf(Props.create(Z1_MathActor.class), "math");
System.out.println("Started. Commands: 'hi', 'm [nb1] [nb2]', 'q'");

// read line & send to actor
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
while (true) {
  String line = br.readLine();
  if (line.equals("q")) {
       break;
  }
  actor.tell(line, null);      // send message to actor
}

// finish
system.terminate();
```

Przykład – Z1_MathActor

```
@Override
public AbstractActor.Receive createReceive() {
return receiveBuilder()
    .match(String.class, s -> {
         if (s.equals("hi")) {
                 System.out.println("hello");
         } else if (s.startsWith("m")) {
                  context().child("multiplyWorker").get().tell(s, getSelf());
         } else if (s.startsWith("result")) {
                  System.out.println(s);
         }
     })
     .matchAny(o -> log.info("received unknown message"))
     .build();
}
```

Przykład – Z1_MathActor

```
@Override
public void preStart() throws Exception {
   context().actorOf(Props.create(Z1_MultiplyWorker.class), "multiplyWorker");
}

private static SupervisorStrategy strategy
   = new OneForOneStrategy(10, Duration.create("1 minute"), DeciderBuilder.
        // todo: match arithmetic exception
        matchAny(o -> restart()).
        build());

@Override
public SupervisorStrategy supervisorStrategy() {
   return strategy;
}
```

Przykład – Z1_MultiplyWorker

```
@Override
public AbstractActor.Receive createReceive() {
    return receiveBuilder()
          .match(String.class, s -> {
                  String result = Multiply(s);
                   getSender().tell("result: " + result, getSelf());
          })
          .matchAny(o -> log.info("received unknown message"))
          .build();
}

private String Multiply(String s){
    String[] split = s.split(" ");
    int a = Integer.parseInt(split[1]);
    int b = Integer.parseInt(split[2]);
    return (a*b) + "";
}
```

Zadanie 1

- Zadanie 1a: dodatkowy aktor do dzielenia
- Zadanie 1b: strategie obsługi błędów

Zadanie 1a (1 pkt)

- Dodać kolejnego aktora, służącego do dzielenia (analogicznie jak mnożenie)
  - Stworzyć nową klasę Z1_DivideWorker
  - Aktor z tą klasą powinien wykonywać dzielenie przy komendzie ‘d [nb1] [nb2]’
  - Dodać utworzenie aktora typu Z1_DivideWorker w aktorze Z1_MathActor
  - Dodać obsługę wiadomości ‘d [nb1] [nb2]’

Zadanie 1b (1 pkt)

- Strategie obsługi błędów
  - Dodać zliczanie ilości wykonanych operacji u workerów – każdy osobno!
  - Dodać informację o ilości wykonanych operacji przy zwracaniu wyniku działania

    np. `m 2 2 => 4 (operation count: 2)`
  - W aktorze Z1_MathActor zastosować strategię, która:
    - wznawia (resume) podwładnego, jeśli rzucił wyjątek arytmetyczny (np. `d 2 0`)
    - restartuje podwładnego, jeśli rzucił inny wyjątek (np. `d aaa`)
  - Sprawdzić działanie AllForOneStrategy

Zdalni Aktorzy

- Aktorzy dostępni są przez ActorRef, co zapewnia transparentność lokalizacji
- Komunikacja ze zdalnym systemem aktorów odbywa się analogicznie jak z lokalnym
- Należy jednak pamiętać, że wiadomości wysyłane przez sieć mają większe ograniczenia
- Wymagane uruchomienie Remoting
- Znajdowanie zdalnego aktora przez ścieżkę

Zdalni Aktorzy – Z2

- Dwie aplikacje: Z2_AppLocal, Z2_AppRemote, każda startuje na innym porcie
- Każda aplikacja ma jednego aktora: Z2_LocalActor, Z2_RemoteActor

```
// config
File configFile = new File("remote_app.conf");
Config config = ConfigFactory.parseFile(configFile);

final ActorSystem system = ActorSystem.create("local_system", config);
```

Zadanie 2 (1 pkt)

- Komunikacja pomiędzy zdalnymi aktorami
  - Zaimplementować lokalnego aktora, który wysyła wiadomości z konsoli (String) do zdalnego aktora (znajdowanego po ścieżce) oraz wypisuje na konsolę otrzymaną odpowiedź
  - Zaimplementować zdalnego aktora, który otrzymuje wiadomości typu String i zwraca ten sam tekst napisany wielkimi literami (toUpperCase)
  - Komunikacja przez localhost

Zadanie 2

- Wskazówki:
  - getContext().actorSelection(path).tell(…)
  - getSelf().path()
- Zdalna ścieżka - format:

```
akka.tcp://systemName@127.0.0.1:3552/user/actorName
```

Strumienie

- Akka Streams API
- Implementacja koncepcji

http://www.reactive-streams.org/

  - Asynchroniczne strumienie
  - Non-blocking back pressure (sterowanie przepływem)

https://doc.akka.io/docs/akka/current/stream/

Strumienie

- Budowa strumienia:
  - Source – źródło danych
  - Flow – transformacja danych
  - Sink – odbiór danych
- Struktura oraz modularność:
  - Każdy element strumienia może być użyty ponownie
  - Definicja elementu strumienia nie oznacza jego utworzenia
  - Można budować dowolne grafy przepływu

Strumienie – Z3

```
final ActorSystem system = ActorSystem.create("stream_system");
final Materializer materializer = ActorMaterializer.create(system);
//final ActorRef actor = system.actorOf(Props.create(Z3_SinkActor.class), "sink");

final Source<Integer, NotUsed> source = Source.range(1, 10);
final Flow flow = Flow.of(Integer.class).map(val -> val * 2);

final Sink<Integer, CompletionStage<Done>> sinkPrint =
   Sink.foreach(i -> System.out.println(i));

source.runWith(sinkPrint, materializer);
```

Zadanie 3 (1 pkt)

- Zmodyfikować przykład Z3 tak, aby:
  - Zamiast mnożenia x2 wyliczana była silnia (metoda scan we Flow)
  - Strumień wysyłany był do aktora, który wypisze go na konsolę (Sink.actorRef)

Zadanie domowe

- Obsługujemy księgarnię internetową
- Umożliwia ona 3 typy operacji:
  - Wyszukiwanie pozycji (zwraca cenę lub informację o braku pozycji)
  - Zamówienie pozycji (zwraca potwierdzenie zamówienia)
  - Strumieniowanie tekstu książki – z prędkością jednej linijki (lub zdania) na sekundę

Zadanie domowe

- Założenia:
  - Klient posiada aplikację (konsolową) opartą o platformę Akka
  - Serwer to pojedyncza maszyna, z dużą ilością zasobów (ale nie nieskończoną)
  - Chcemy być w stanie obsłużyć jak najwięcej klientów równolegle (na jednym serwerze)
  - Chcemy zminimalizować czasy odpowiedzi systemu
  - Chcemy zminimalizować ilość danych przesyłanych przez sieć (należy unikać przesyłania wszystkich wiadomości jako String)

Zadanie domowe

- Wyszukiwanie pozycji:
  - Należy przeszukać dwie bazy danych
  - Każda baza danych ma postać pliku tekstowego (jedna linia -> jeden tytuł + cena)
  - Zakładamy, że jedna lub obie bazy mogą być czasowo niedostępne
  - Zakładamy, że wyszukiwanie w bazie może być czasochłonne (należy przeszukiwać obie bazy równolegle)
  - Jeśli jakaś pozycja jest w obu bazach to ma tę samą cenę

Zadanie domowe

- Zamówienie pozycji
  - Następuje przez:
    - zapisanie nowej linii z tytułem (bez ceny) do pliku orders.txt, który stanowi bazę zamówień
    - oraz wysłanie potwierdzenia do klienta
  - Należy zwrócić uwagę na synchronizację dostępu do pliku (bazy)

Zadanie domowe

- Strumieniowanie tekstu
  - Klient wysyła tytuł, serwer odpowiada strumieniem linii lub zdań z pliku o tej nazwie
  - Jedna linia(lub zdanie) na sekundę (throttle)

Zadanie domowe

- Obsługa błędów
  - Należy zastosować odpowiednie strategie obsługi błędów
- Schemat
  - Należy przygotować schemat w wersji elektronicznej
  - Schemat powinien zawierać zaproponowaną strukturę aktorów wraz ze wskazaniem ich cyklu życia (stały/tymczasowy, strategia obsługi błędów)

Zadanie domowe

- Punktacja
  - Schemat 2 pkt
  - Wyszukiwanie 4 pkt
  - Zamówienia 2 pkt
  - Strumieniowanie tekstu 2 pkt
- Język
  - Dowolny obsługiwany przez Akka

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Dziękuję