Akka
Actor Model

Uangel 미디어플랫폼팀
허강무

Outline

Two Models of Concurrency

공유 메모리를 사용하는 방식

- Lock / Mutex 사용
- Thread
- ex) 멀티 스레드 프로세스

메시지 전달 방식

- 공유 메모리 사용 X / Message 통신
- Process
- ex) Actor Model / CSP

Uangel 미디어플랫폼팀
허강무

Why Actor Model

공유 메모리 사용 시 문제점

1. 스레드가 늘어나면 늘어날수록 실수하기 쉬워지며 예측 불가능한 상황이 많이 발생
2. 예측 불가능한 상황을 방지하기 위해 lock을 사용한다.
3. lock을 걸면 그만큼 성능의 저하가 일어나며 병목현상의 원인이 된다.
4. 스레드가 아무리 늘어나도 성능 향상의 한계점이 생긴다.

Actor Model은 공유하는 자원을 없애고, 메시지로만 통신하도록 제한하여
동시성 환경에서의 문제점을 타개할 방법으로 고안된 모델

Uangel 미디어플랫폼팀
허강무

Actor Model

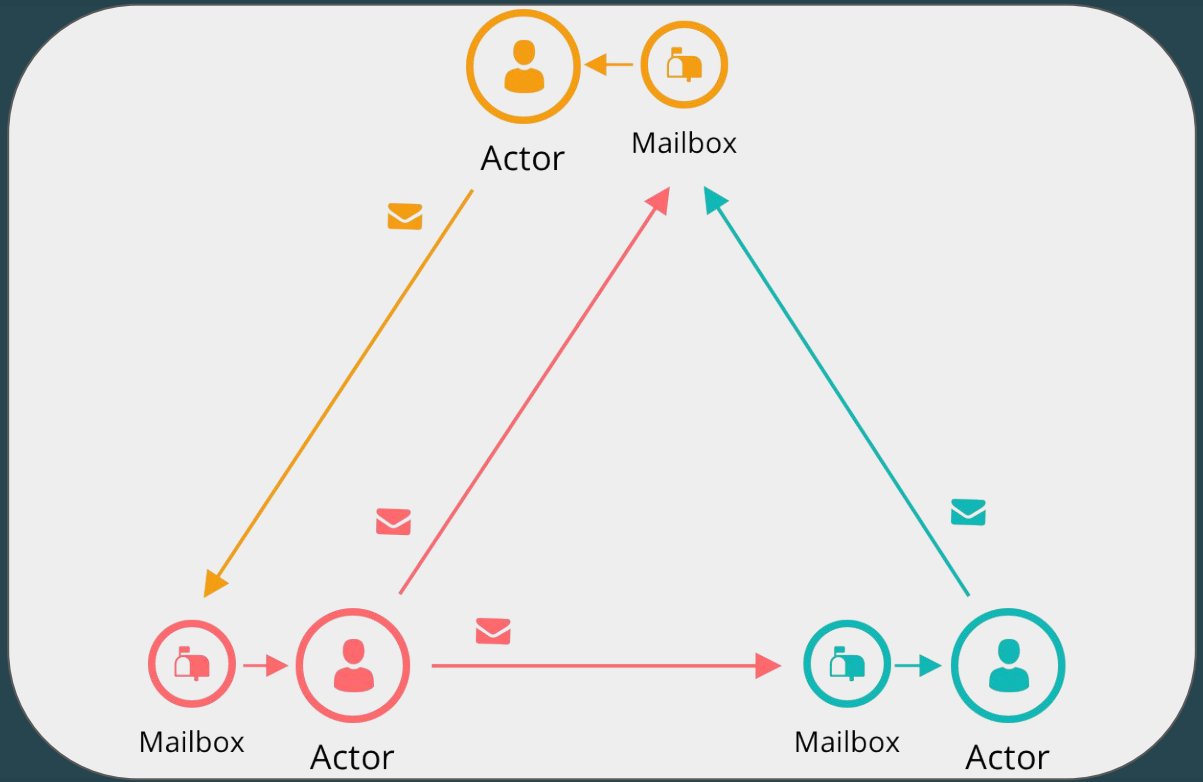
Actor Model

"모든 것은 액터다(Everything is an actor)"

- 1973년 발표된 모델로, 병행 연산을 위한 수학적 모델의 일종
- 병행 디지털 계산의 범용적 기본 요소로 "액터"라는 개념을 도입
- 액터는 자원을 공유하지 않고, 메시지로 통신한다.
    - Actor Model은 모든 것이 Actor로 구성되어있는 모델
- 병행 모델들의 고질적인 문제점인 교착 상태, 경쟁 상태 등의 발생 가능성이 낮다.
- Erlang, Scala 등의 언어들이 액터 모델에 기초하여 병행성 기능 제공

Uangel 미디어플랫폼팀
허강무

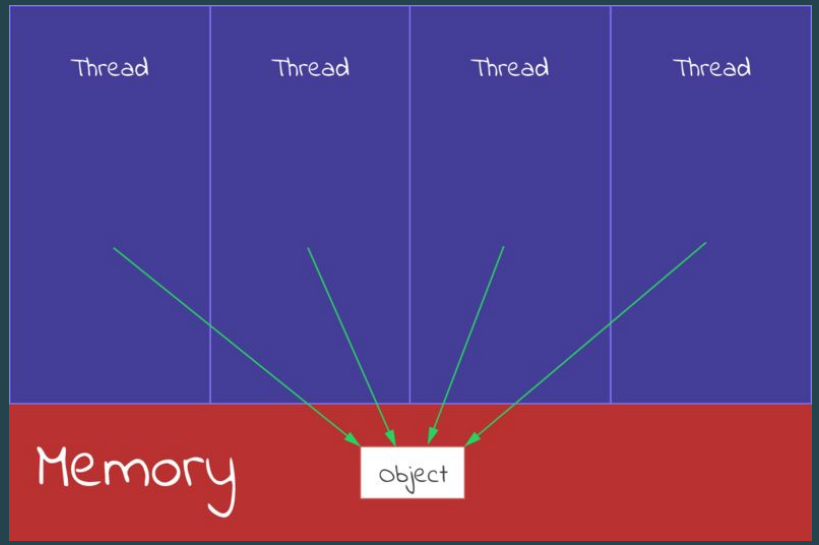
Actor Model



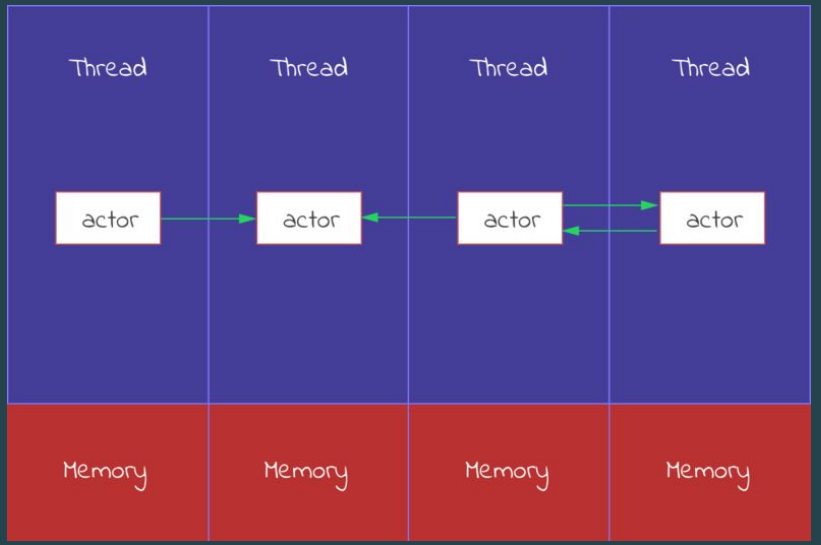
Uangel 미디어플랫폼팀
허강무

Difference

기존 방식



Actor Model



Uangel 미디어플랫폼팀
허강무

Difference

기존 방식으로 세션 생성 과정

1. taskInfo = new TaskInfo()
2. callInfo = new CallInfo()
3. taskInfo.addCall(callInfo)
4. SessionManager.add(taskInfo)
5. SessionManager.add(callInfo)

Uangel 미디어플랫폼팀
허강무

Difference

기존 방식으로 세션 생성 과정

1. taskInfo = new TaskInfo()
2. callInfo = new CallInfo()
3. taskInfo.addCall(callInfo)
4. SessionManager.add(taskInfo)
5. SessionManager.add(callInfo)

Lock 필요

Uangel 미디어플랫폼팀
허강무

Difference

Actor Model 방식으로 세션 생성 과정

Uangel 미디어플랫폼팀
허강무

Actor

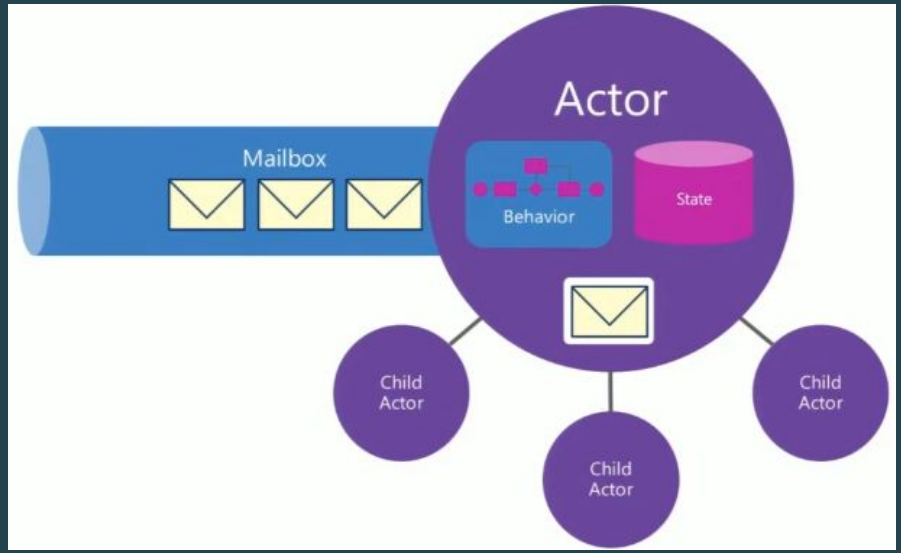
1. Actor는 서로간에 **공유하는 자원이 없고** 서로간의 상태/자원을 건드릴 수도 없다.
2. 하나의 Actor에서 동작하는 **스레드는 1개로 제한**한다.
3. Actor는 스레드 생성하지 않는다
4. Actor는 lock을 사용하는 외부 시스템의 자원을 사용하지 않는다.
5. Actor는 오직 **message만을 이용해서 정보를 전달**한다.

Uangel 미디어플랫폼팀
허강무

Actor

Actor가 할 수 있는 일

1. Message 송신
   다른 Actor 또는 자기 자신에게
2. Message 수신 & 처리
3. 새로운 Actor 생성



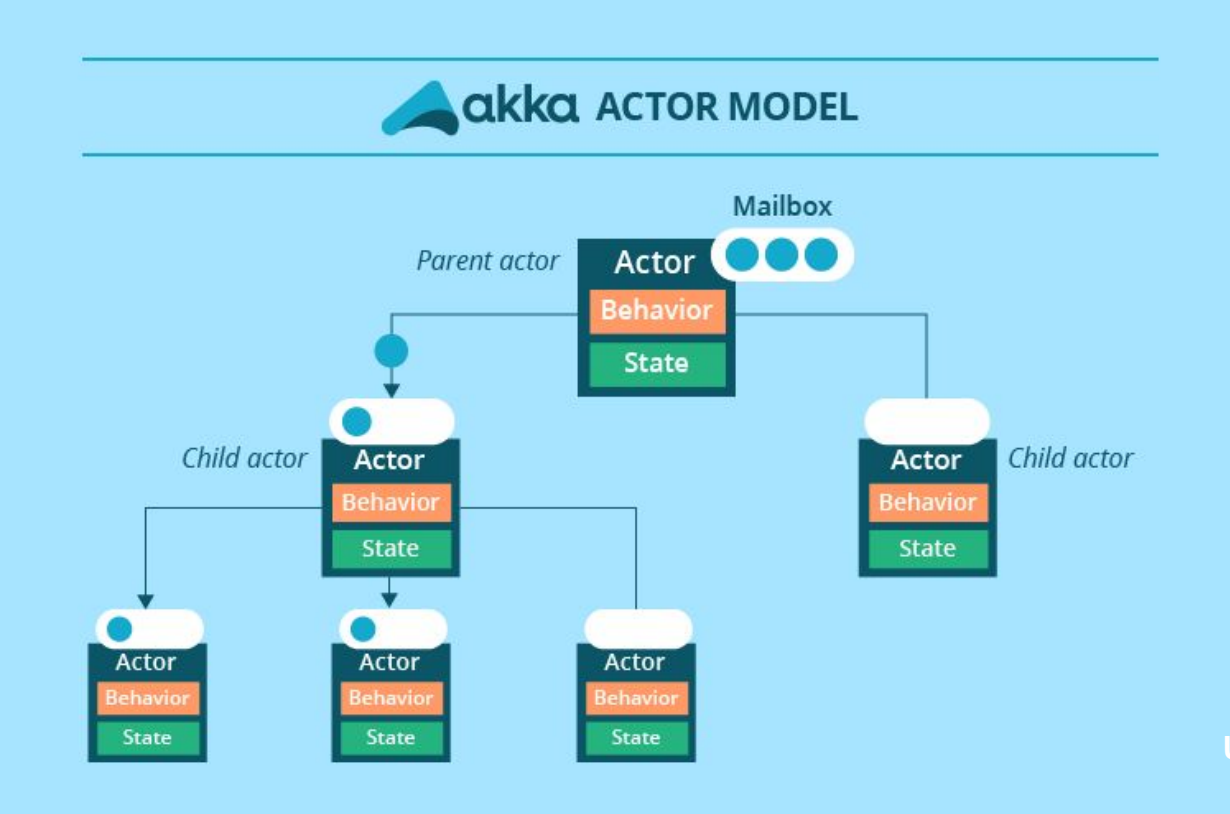
Uangel 미디어플랫폼팀
허강무

What is Akka

- 분산처리를 위해 액터 모델을 이용한 오픈소스 프레임워크
- JVM위에서 병행, 분산 애플리케이션 구축을 간편하게 할 수 있도록 도와준다
  (개발자가 메시지 처리에 집중하게 만들어준다.)
- 병행성을 위한 다중 프로그래밍 모델 지원한다.
  (Java환경및 .net Framework에서 이용가능)
- 작성언어: Scala

Uangel 미디어플랫폼팀
허강무

Akka Actor Model



Uangel 미디어플랫폼팀
허강무

Actor System

Actor System

- 액터들을 담기 위한 컨테이너
- 액터 자체는 가볍지만 액터 시스템은 무거워서 보통 액터 시스템은 어플리케이션에 하나만 생성

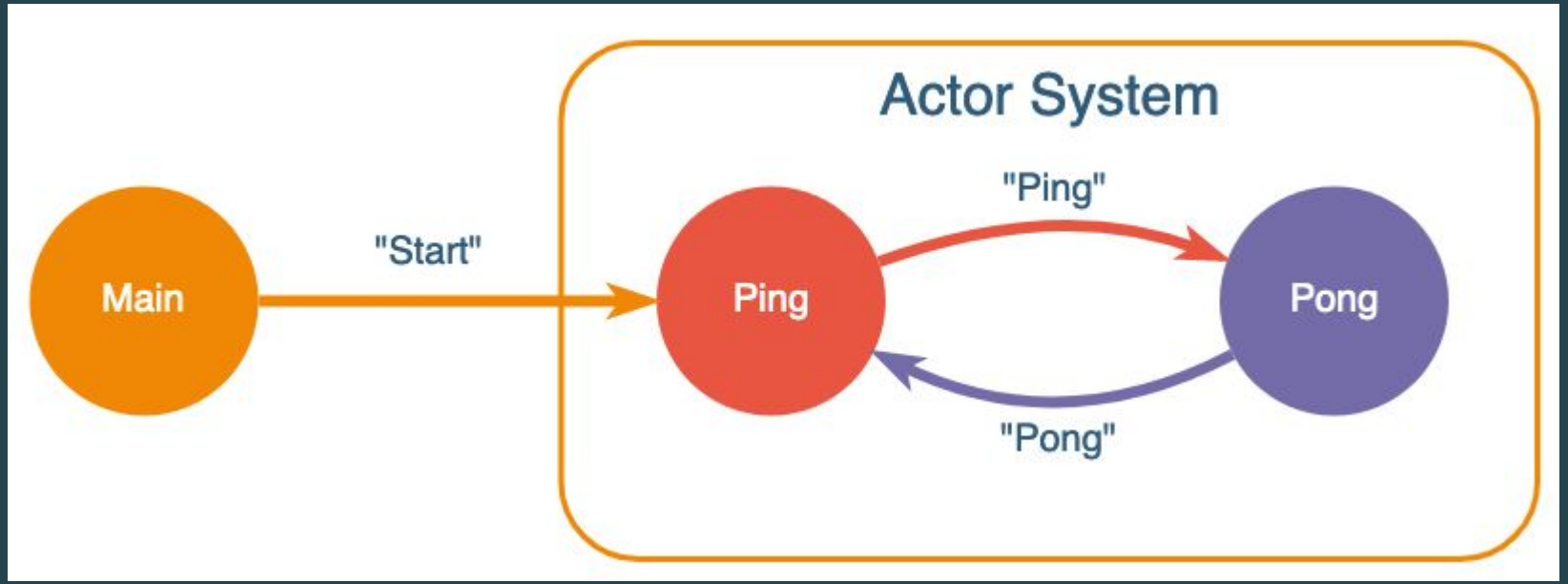
Key Point

- 액터 내부에서 일어나는 일은 어느 누구와도 '공유'되지 않는다
- 액터가 메시지를 처리할 때 두 개 이상의 스레드가 동시에 동작하지 않는다
- 액터는 코드 내에서 다른 스레드 생성이나, Lock을 사용하는 코드에 접근하면 안된다
- 위의 내용은 액터 시스템에서 동시성 보장해주지만, 액터 시스템 내의 코드 자체에서 위의 내용을 위반한 행동을 할 수 있으니 의식하고 코드를 작성해야 한다
  (액터 내부에서 스레드 생성 or 다른 객체에 임의 접근 등..)

Uangel 미디어플랫폼팀
허강무

Akka Example Code

Ping Pong Example



Akka Example Code

Main

```
public class PingPongMain {
   public static void main(String[] args) {
      ActorSystem actorSystem = ActorSystem.create("TestSystem");
      ActorRef ping = actorSystem.actorOf(Props.create(PingActor.class), "pingActor_MAIN");
      ping.tell("Start", ActorRef.noSender());
   }
}
```

Uangel 미디어플랫폼팀
허강무

Akka Example Code

PingActor

```
public class PingActor extends UntypedAbstractActor {
   private static final Logger logger = getLogger(PingActor.class);
   private ActorRef pong;
   private int count = 0;

   @Override
   public void preStart(){
      this.pong = context().actorOf(Props.create(PongActor.class, getSelf()), "pongActor");
   }

   @Override
   public void onReceive(Object message) {
      if(message instanceof String){
         String msg = (String)message;
         logger.info("Ping Received {}_{}", msg, count++);
         pong.tell("Ping", getSelf());
      }
   }
}
```

Uangel 미디어플랫폼팀
허강무

Akka Example Code

PongActor

```
public class PongActor extends UntypedAbstractActor {
   private static final Logger logger = getLogger(PongActor.class);
   private ActorRef ping;
   private int count = 0;

   public PongActor(ActorRef ping){
      this.ping = ping;
   }

   @Override
   public void onReceive(Object message) throws Throwable {
      if(message instanceof String) {
         String msg = (String) message;
         logger.info("Pong Received {}_{}", msg, count++);
         ping.tell("Pong", getSelf());
         Thread.sleep(1000);
      }
   }
}
```

Akka Example Code

Result

```
Ping Received start_0
Pong Received ping_0
Ping Received pong_1
Pong Received ping 1
Ping Received pong_2
Pong Received ping_2
Ping Received pong_3
Pong Received ping_3
Ping Received pong 4
Pong Received ping_4
Ping Received pong_5

...
```

Uangel 미디어플랫폼팀
허강무

Pros and Cons

장점

- 동시성, 이벤트 기반 및 분산 시스템을 구축하기 쉬움
- 효율적인 스레드 사용 가능. 이론상 수백만개의 액터가 돌아가도 문제가 없음
- Akka의 경우 원격 배포기능 제공하므로, URL을 이용한 다른 프로세스 및 다른 서버에 있는 액터와도 통신 가능.
- 액터 관리자가 존재하여 장애 발생 시 액터를 재시작하는 방법으로 장애에 대응 가능
- 마이크로서비스에 적합하며, 손쉬운 서버 Scale - out 가능

Uangel 미디어플랫폼팀
허강무

Pros and Cons

단점

- 복잡한 분산 트랜젝션을 고려한 액터 시스템을 구상하는 것이 굉장히 어려움
- 메시지 손실 가능성이 존재
- 테스트 및 디버깅이 어렵고, 코드리딩이 힘들어짐
- 메모리에 직접 접근하는 방식이 아닌 메시지 전달 방식이므로 속도 저하 발생
- 액터 사이의 통신을 할 경우 유용한 타입 시스템의 사용이 어려움

  (액터는 기본적으로 어떤 유형의 메시지든 받을 수 있다)

In conclusion

결론적으로

제대로 구현이 된다는 전제 하에 액터 모델의 장점을 고스란히 가질 수 있다.

(고부하 환경에 대한 내성, 손쉬운 확장성, 강력한 장애 저항력, 효율적인 리소스 사용)

하지만 제대로 구현하기까지 많은 노력과 경험이 필요하며,

역량이 부족할 시 프로젝트의 실패를 초래할 수 있으므로 도입에 신중해야 한다.

Uangel 미디어플랫폼팀
허강무

Reference

- https://akka.io/
- https://ko.wikipedia.org/wiki/%ED%96%89%EC%9C%84%EC%9E%90_%EB%AA%A8%EB%8D%B8
- https://www.scnsoft.com/blog/akka-actors-for-microservices
- https://arild.github.io/csp-presentation/#1
- https://github.com/petabridge/akka-bootcamp
- https://www.slideshare.net/krivachy/the-dark-side-of-akka-and-the-remedy