SOSCON

삼성 오픈소스 컨퍼런스

SAMSUNG OPEN SOURCE CONFERENCE

OPEN YOUR UNIVERSE WITH SOSCON

리액티브 시스템과 아카

adMarketplace

임백준

2015년 10월 28일

임백준

- 現 adMarketplace 디렉터
- 모건스탠리, 바클리스, 도이치은행, 벨연구소, 삼성 SDS 등 근무
- 팟캐스트 〈나는 프로그래머다〉 진행자
- 지디넷 칼럼니스트
- 한빛미디어와 함께 쓴 책
    - 〈아카 시작하기〉, 2015
    - 〈나는 프로그래머다〉, 2015
    - 〈폴리글랏 프로그래밍〉, 2014
    - 〈누워서 읽는 퍼즐북〉, 2010
    - 〈프로그래밍은 상상이다〉, 2008
    - 〈뉴욕의 프로그래머〉, 2007
    - 〈소프트웨어 산책〉, 2005
    - 〈나는 프로그래머다〉, 2004
    - 〈누워서 읽는 알고리즘〉, 2003
    - 〈행복한 프로그래밍〉, 2003

리액티브 시스템이란

외부 사건에 반응하는 시스템

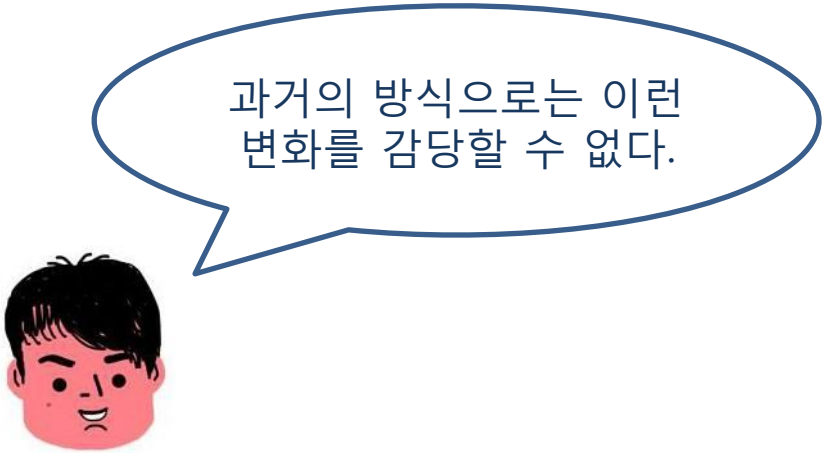
- A reactive system is a system that responds (reacts) to external events.

출처 http://www.wikipedia.org/

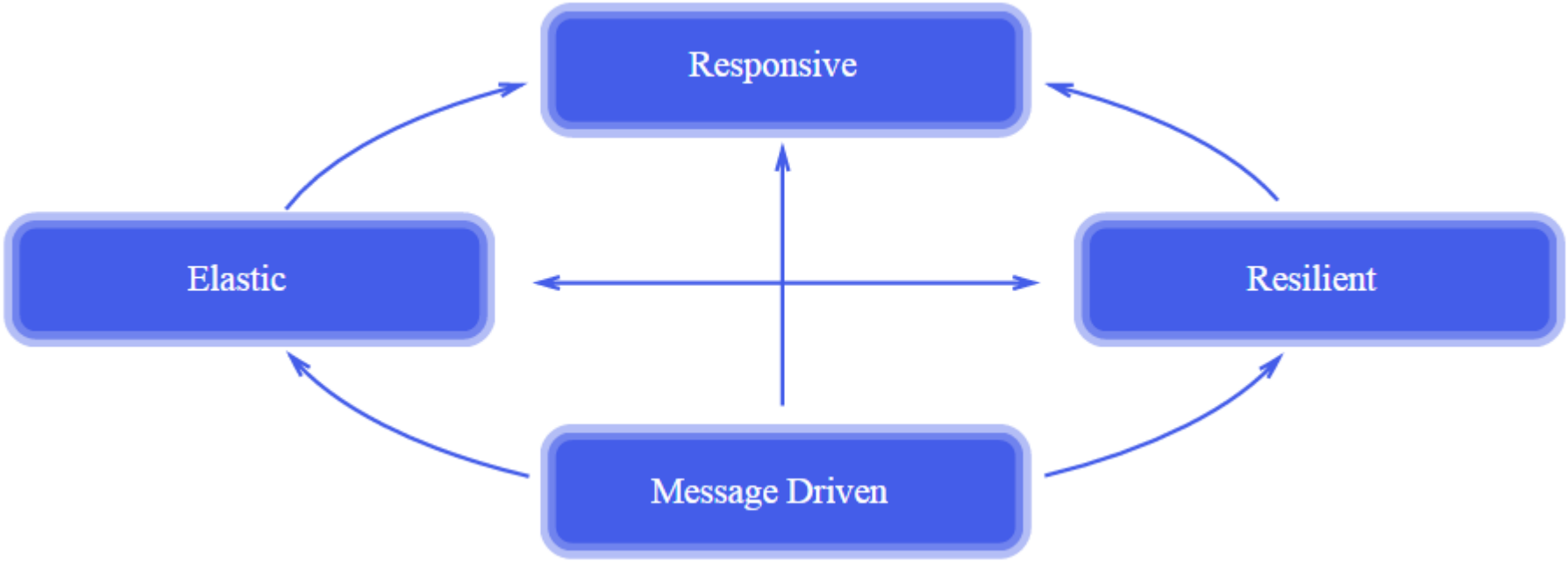
왜 리액티브 시스템인가

컴퓨팅 환경의 변화

- 수십 대의 서버 → 수백, 수천 대의 서버
- 수십, 수백 개의 코어 → 수천, 수만 개의 코어
- 초(second) 단위 반응 속도 → 밀리초(millisecond) 단위 반응 속도
- 연간 몇 시간의 다운 시간 → 0 다운 시간
- 기가바이트 데이터 → 페타바이트 데이터
- 소프트웨어 설계의 복잡성 - 동시성, 장애처리, 확장성



리액티브 시스템의 속성



- Responsive **반응성**
- Elastic **탄력성**
- Resilient **유연성**
- Message Driven **메시지 중심**

출처 http://www.reactivemanifesto.org/

메시지 중심이란

- 비동기적 메시지 전달 방식 (편지)
- 메시지 vs 이벤트

메시지 중심의 장점

- 컴포넌트 사이의 느슨한 결합
- 장소 투명성
- 메시지로서의 에러 처리
- 로드 관리
- 메시지 모니터링
- back-pressure

탄력성이란

- 유입되는 트래픽 량에 따라서 자원을 늘리거나 줄임
- Scale up, scale down, scale out, scale in

탄력성의 장점

- 효율적인 자원 할당
- 라이브 성능 측정을 통한 동적인 스케일링

유연성이란

- 고장이 일어나도 시스템 전체는 계속 반응하는 것
- 중복 replication - high availability
- 감금 containment - 컴포넌트 하나의 고장이 다른 컴포넌트에 영향을 주지 못하도록
- 분리 isolation - 감금의 결과
- 대리 delegation - 고장 처리는 외부의 전문 컴포넌트에게 맡김

유연성의 장점

- Let It Crash

반응성이란

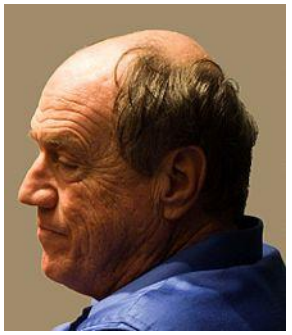
- 소프트웨어의 품질을 나타내는 시금석
- 고장이 빠르게 감지되고 효과적으로 처리되고 있음을 뜻함
- 빠르고 효과적인 응답 시간을 보장
- 최종 사용자의 확신을 구축

리액티브 프로그래밍의 원리

- 느슨한 결합 Loosely coupled
- 난블로킹 Non-blocking
- 비동기적 Asynchronous
- 경합 최소화 Minimize contention
- 지역성 극대화 Maximize locality
- 아무 것도 공유하지 말라 Share nothing
- 끌어당기기는 좋고, 밀기는 나쁘다 Pull, not Push
- 역류 Back-pressure
- 게으름 혹은 리액티브 Lazy or reactive

액터모델 **Actor Model**

- 칼 휴이트가 1973에 제안한 수학적 모델
- 액터는 3가지 일을 수행한다.
    1. 메시지를 전달 받는다.
    2. 메시지를 전송한다.
    3. 다른 액터를 생성한다.



Carl Hewitt

- 1986년 얼랭(Erlang)에 의해서 구현
- 스칼라로 작성된 4개의 액터 모델
    - ❖ 스칼라지
    - ❖ 리프트
    - ❖ 스칼라 표준 라이브러리 (Scala 2.1.7, 2006)
    - ❖ 아카 (Akka 0.5, 2010)

아카

- 스칼라 언어로 구현된 액터 모델 라이브러리
- 동시성 프로그래밍과 분산 컴퓨팅의 통합
- 비동기적 메시지 전달 중심
- Let It Crash 철학
- 지역 투명성을 통한 분산 컴퓨팅 지원
- 자바 개발자를 위한 자바 API
- Akka.NET

Jonas Bonér

아카의 사용 사례

- 아파치 스파크
- 블리자드 엔터테인먼트 Blizzard Entertainment
- 월스트리트 - 크레딧 스위스, 모건스탠리 ☺
- AppNexus, Tapad, adMarketplace 등 adTech 업계
- 가디언, BBC, 허핑턴포스트 등 디지털 미디어
- 아마존, 이베이, 그루폰, 월마트 등 전자상거래 업체
- 인텔, 시스코 등 IT 업체

아카 구조물

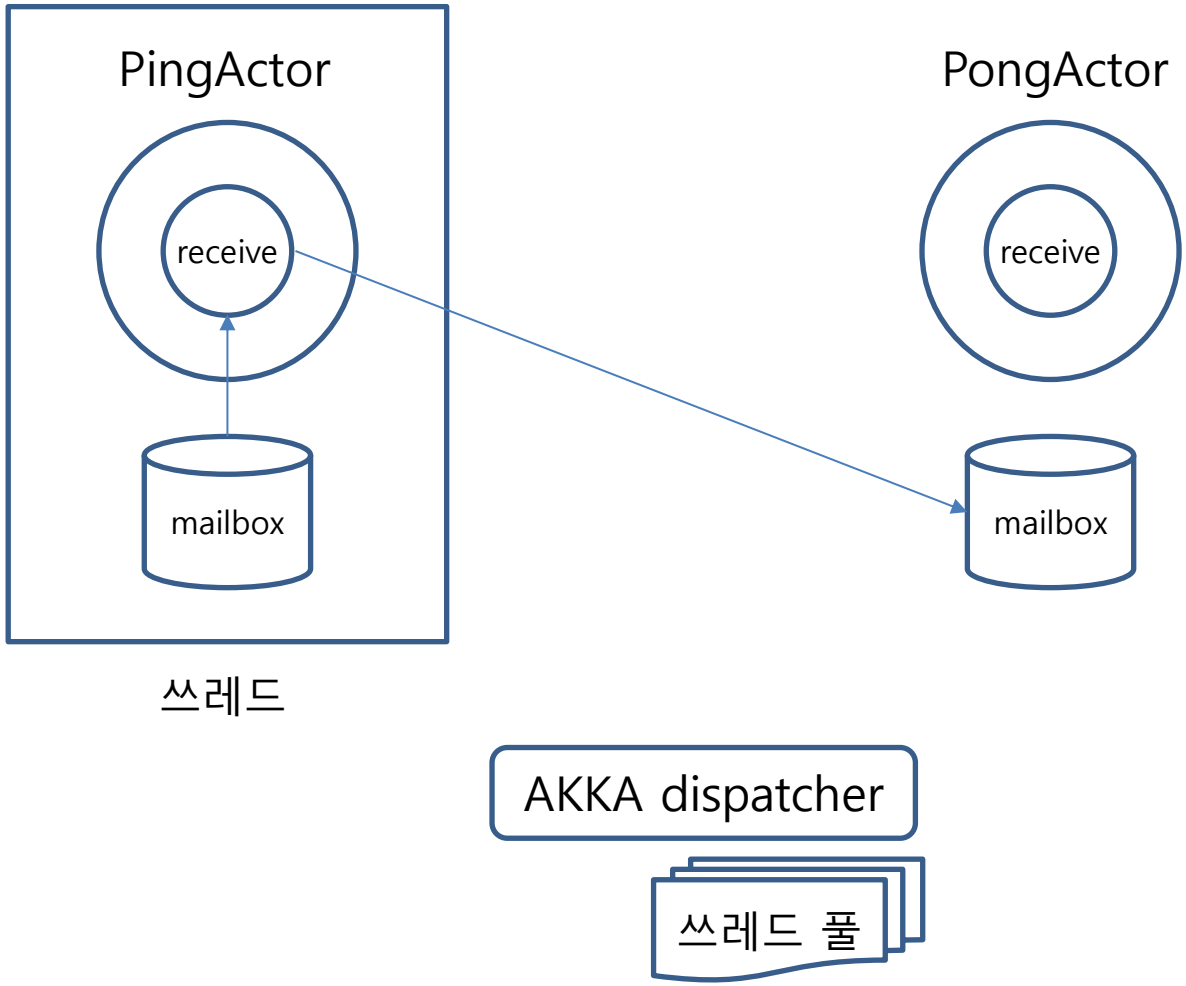
- 액터 이외에 다양한 구조물을 지원 (아카 > 액터)
  - ❖ 라우터 Router
  - ❖ 퓨처 Future
  - ❖ 에이전트 Agent
  - ❖ 상태기계 State Machine
  - ❖ 클러스터 Cluster
  - ❖ 이벤트 버스 Event Bus
  - ❖ 캐멀 통합 Camel Integration
  - ❖ Akka-http
  - ❖ Akka-stream
  - ❖ Akka-extension

액터 코드

```
class PingActor extends Actor {
    def receive = {
        case “Pong” => sender ! “Ping”
    }
}

class PongActor extends Actor {
    def receive = {
        case “Ping” => sender ! “Pong”
    }
}
```

액터와 메일박스


PingActor
receive
mailbox
PongActor
receive
mailbox
쓰레드
AKKA dispatcher
쓰레드 풀

참고자료

- 아카 홈페이지 http://akka.io
- 리액티브 선언 http://www.reactivemanifesto.org
- 타입세이프 홈페이지 http://www.typesafe.com
- 아카 시작하기, 임백준, 한빛미디어, 2015년 10월
- Akka In Action, Raymond Roestenburg, Manning, 2015년 12월
- Effective Akka, Jamie Allen, O'Reilly, 2013년 9월
- Akka Concurrency, Derek Wyatt, Artima, 2013년 5월


SOSCON
삼성 오픈소스 컨퍼런스
SAMSUNG OPEN SOURCE CONFERENCE
OPEN YOUR UNIVERSE WITH SOSCON

THANK YOU!