IT Resources for Nonprofits

Markus Dale, medale@gmail.com

May 2025

Introduction

Building Complex Systems

- Complex problems (e.g. distributed command & control)
- Modern software development - Layers of complexity (OS, libraries...)
    - Vertical scaling - Multi-core CPU with caches
    - Horizontal scaling - distributed system

Software Development Cost - Maintenance 60/60 Rule

Build for Maintenance

- Modularity: Decompose functionality to encapsulate logic and state (Parnas '72)
- Can use Object Oriented programming for encapsulation (single thread)
- How to handle multiple users, concurrent events?
- How to take advantage of multi-core/multithreaded - shared object access?

Abstractions - Programming Paradigms

- Object-oriented programming - decompose large problem domain
- Functional programming - immutable data structures, algorithms
- Actor model - multi-core/multi-machine

(Traditional) Explicit Concurrency Handling

- Race conditions on shared resources
- Threads & locks
  - Starvation
  - Slow
- For distributed systems - distributed locks (even slower)
- How to deal with failures?

Carl Hewitt, 1973

Figure 1: Carl Hewitt, 1973

Actor Model Implementations

- Ericsson - Erlang
- Akka - Scala/Java (open source Apache Pekko)
- C++ Actor Framework (CAF), Microsoft Orleans (.NET), Actix (Rust)...

Why Akka (open source Apache Pekko)?

- Mature (started in 2009) and widely used actor toolkit
  - Tesla PowerWall battery management
  - Fortnite (Epic Games - 50 million players per day)
  - Tubi, Disney Streaming, iHeartRadio
  - Renault Factory management
  - PayPal, Verizon, CapitalOne etc.

Tesla Powerwall - Digital Twins

Pekko Actors

Dispatchers

- Threadpools - control concurrency of different parts
- Dedicated threadpool for blocking operations

Pekko Supervision for Failure Handling

Discovery

Routers, Schedulers

- Create child actors with same implementation
- Routing strategy: broadcast, round-robin, random, consistent-hashing
- Scheduler: recurring, one-time

Mailboxes

Pub/Sub Topics

Testing

Pekko Cluster

- Cluster membership/discovery
- Cluster singleton
- Cluster sharding
- Distributed data
- Multi-DC

Persistence

- Event sourcing - Journal/checkpoints/replay
- Durable state

Utilities: Streams, HTTP, gRPC, retry and CircuitBreaker

When to Use?

- Resilience, high concurrency, or distributed system
- Local actor system - scale to distributed
- Multiple “microservices” in one cluster - can scale separately
- Use with Kubernetes

Not Everything's An Actor

- Learning curve - Rock the JVM Akka/Pekko classes
- Single Responsibility Principle
- Combine with OOP/functional programming
- BlueHalo expertise - Crossfire production system

Questions?