M Ű E G Y E T E M   1 7 8 2

Budapest University of Technology and Economics
Department of Measurement and Information Systems
Fault Tolerant Systems Research Group

System Integration and Supervision Laboratory Exercises
2014/2015 spring semester

Using the Actor Model for Concurrent Applications: Akka

Syllabus
v1.0

Gábor Szárnyas
(szarnyas@mit.bme.hu)

April 1, 2015

1   Introduction

During this laboratory course, the student get familiar with practical applications and tools associated with system integration and supervision. The course presents the major implementation steps and supervision methods of a distributed application, using industrial middleware technologies and supervision technologies. Each laboratory is built around a certain technology:

1. Implementing Workflows in Java

2. System Supervision with Complex Event Processing

3. Reliable Messaging over IBM WebSpehere MQ

4. JMS- and JMX-based Communication

5. Developing OSGi Services

6. Using the Actor Model for Concurrent Applications: Akka

7. Visual Data Analysis of Supervisory Data

2   Reactive Programming

Reactive programming is a trending programming paradigm. In contrast to most traditional programming paradigms, reactive programming is based around data flows and the propagation of change [4]. To gain insight to reactive programming, **please read the Reactive Manifesto** [3], a short document discussing the ideas behind reactive programming.

2.1   Motivation

The main motivation behind for using novel programming paradigms (or often rediscovering existing ones) is the fact that while Moore's law still holds true, the single-core speed of current microprocossers has come to a plateau. To overcome this limitation, they provide more processing cores (Figure 1).

2.2   The Actor Model

The Actor model originally appeared in 1973 [6]. The main idea of the actor model is to process information using separate actors, each with its own state. The actors communicate with immutable messages.

3   Akka

Akka is a toolkit and runtime for building highly concurrent, distributed, and fault tolerant event-driven applications on the JVM [1].
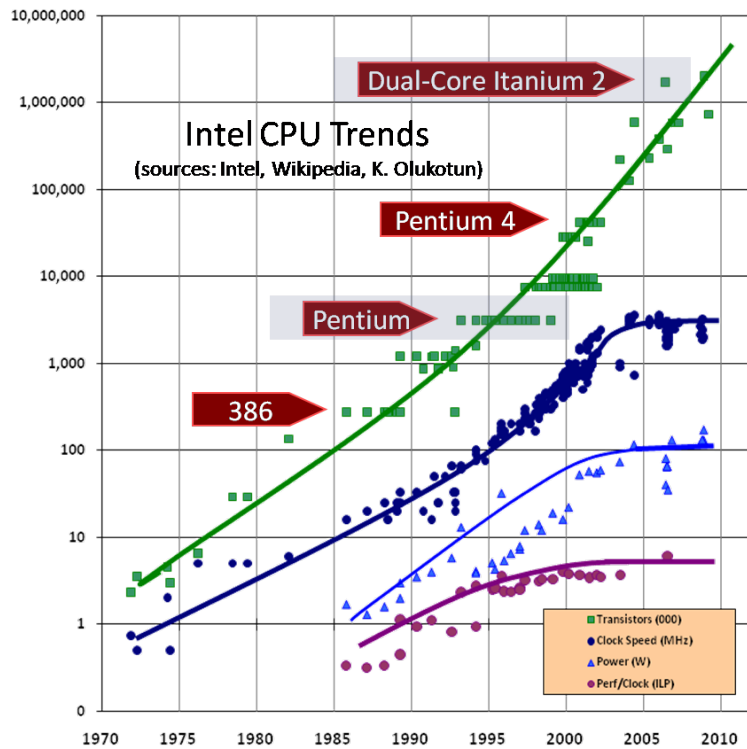
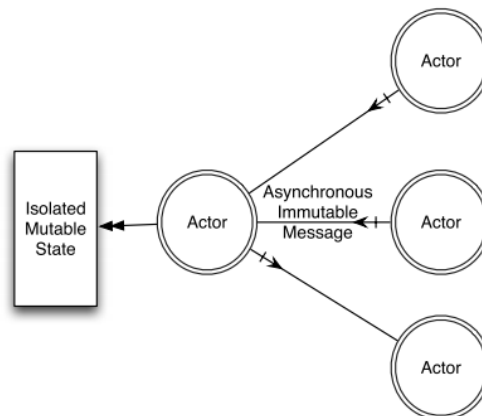Figure 1: Intel CPU introductions (graph updated August 2009) [5].

Figure 2: The Actor Model [7].

Figure 3: The Logo of the Scala Programming Language.

3.1 The Scala Language

Like Java, Scala (from *scalable* and *language*) is a strongly object-oriented and statically typed language. However, Scala also uses elements from functional programming. Scala natively supports building concurrent applications. The logo of the Scala language is shown on Figure 3.

3.2 Akka API

Akka provides and API for both Scala- and Java-based applications [2]. For this laboratory, we recommend the Java API of Akka 2.3.1.

4 Questions

On this laboratory, all questions will be selected from the list below. The questions may be answered in either English or Hungarian. **The answers to the questions are provided in the Akka documentation.**[1].

1. What is the difference between *concurrency* and *parallelism?*

2. What is the main idea behind the actor model?

3. What is the difference between a *typed* and an *untyped* actor?

4. Describe the semantics of the `tell` and ask methods for sending messages.

5. What is the role of *futures*?

References

[1] Akka. `http://akka.io/`.

[2] Akka Documentation. `http://akka.io/docs/`.

[3] The Reactive Manifesto. `http://www.reactivemanifesto.org/`.

[4] Wikipedia: Reactive programming. `http://en.wikipedia.org/wiki/Reactive_programming`.

[5] Herb Sutter. The Free Lunch Is Over – A Fundamental Turn Toward Concurrency in Software. `http://www.gotw.ca/publications/concurrency-ddj.htm`, March 2005.

[6] Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, pages 235–245, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc.

[7] Venkat Subramaniam. Programming Concurrency on the JVM. `http://ljcbookclub.wordpress.com/2012/02/15/programming-concurrency-on-the-jvm/`, February 2012.

---

[1]Also available in a single document: `http://doc.akka.io/docs/akka/2.3.1/AkkaJava.pdf`