Testing and Model Checking Concurrent Software

Gregory Beale[1]; Ariel Virgulto[2]; Dr. Steven Lauterburg[3]
[1]Randolph-Macon College, [2]Univeristy of Connecticut, [3]Salisbury University

Abstract

Despite advances in software, development environments, and hardware, programs continue to have errors. Testing software is important to ensure programs run efficiently and predictably. Concurrent programming, specifically message passing, can complicate system and model states through asynchronous message passing. JPF-actor (Basset) is a tool that allows users to systematically test actor programs in both Java and Scala. However, as new versions of Scala, Java, and JPF have been released, JPF-actor has not been updated, rendering it obsolete. We seek to rewrite and restore the compatibility of the Basset toolkit with the new Akka actor protocols and libraries instantiated within concurrent Scala and Java programs.

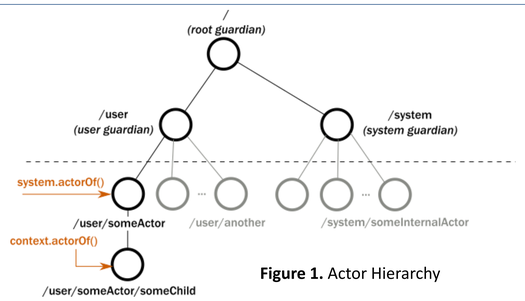

Figure 1. Actor Hierarchy

Introduction

Advances in programming languages, development environments, and hardware have necessitated software testing to ensure error-free models and programs. Multiple, faster processors in computing cores allow for quicker builds and faster run times. As a result of smaller wait times, the debugging process can be more extensive and quicker. Message passing, as a form of communication between cores and systems, is used in both parallel and distributed applications, which have become increasingly popular since the speed of single processors and transistors has plateaued in the last decade. In parallel computing, all processors have access to shared memory, while in distributed computing, each processor has private memory and passes messages between each other in order to communicate. Message passing communication is becoming more prevalent as distributed systems grow in use. Java Path Finder (JPF) is a program used to verify concurrent programs, written mostly in Java and Scala. JPF's main goal is to find and alert the user to a number of defects or pitfalls that could affect the performance of concurrent programs. JPF uses model-checking method that systematically explores all relevant message schedules in actor programs. However, JPF's actor module, designed to specifically test concurrent software that uses actors, is out-of-date and needs to be updated for compatibility Akka and Scala's actor library. Developing a basic understanding of the protocols and design behind Akka's actor implementation is the first step to restoring JPF's functionality in a world growing increasingly favorable of actor-oriented distributed computing.

Problems

Actor Messaging Programming

Actors communicate solely by passing messages back and forth. These messages are stored in each other's mailbox, however, there is no guarantee that messages are delivered in the order in which they are sent. [3] JPF-actor looks to see if this will cause problems by exploring all the "relevant" message schedules between actors. Fig. 3 illustrates a simple state exploration diagram. [3] From this mailbox, actors dequeue and process each message one at a time. Actor programs have non-determinism as a result of the order in which messages are delivered, not the ordering of shared memory access. [3] The lack of method calls between calls allows for encapsulation without resorting to blocking or locks in a hierarchical structure as illustrated in Fig. 1. [1] An illustration of message passing between actors is shown in Fig. 2. [1]

JPF-actor Functionality

Compatibility between software is the main problem we are attempting to correct. The JPF-actor software was created in 2010 to work with older versions of JPF, Java, ActorFoundry, and, Scala which have all since been updated, except JPF-actor. [2] ActorFoundry is no longer used, and Scala has been updated to use Akka's message passing protocols, libraries, and foundations. Basset will currently not work with any of these languages and tools. Thus, JPF cannot test any software that implements actors for concurrent programming, leaving a large hole in its functionality. Updating JPF-actor will require an overhaul of the testing features in accordance to Akka's message passing libraries and protocols.

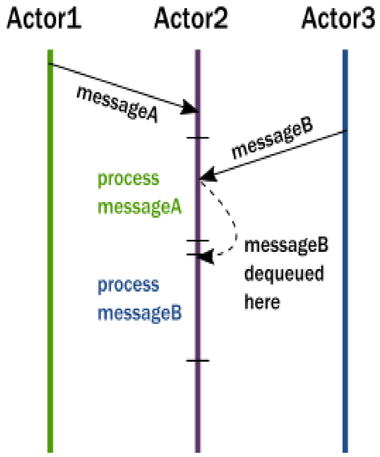

Figure 2. Message Passing Behavior between Actors

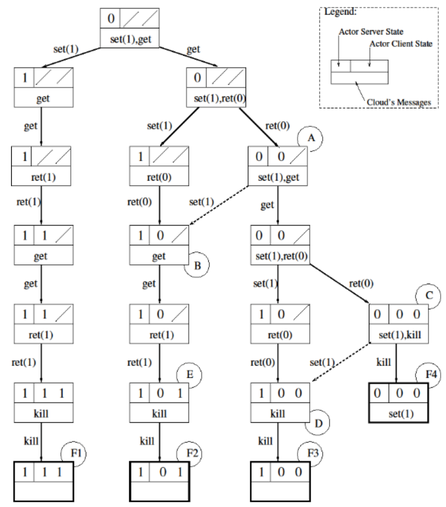

Figure 3. JPF-actor exploring different message schedules

Method and Preliminary Results

The first step to restoring compatibility for JPF-actor is to understand the new message passing protocols that are present in Akka's actor libraries. With online resources from Akka, we developed a number of Java and Scala implementations of old actor programs written in ActorFoundry. These included a Pi Calculation, Fibonacci Number Calculator, QuickSort, MergeSort, and Chameneos, all using the modern Akka library for actors. Implementing these programs offers a way to explore the hierarchy and the power of actor programming. The Pi Calculator illustrates the basic reduction in overhead run time that is achieved by using concurrent software. The Fibonacci program results in the top down creation, then termination, of thousands of actors to calculate a certain Fibonacci number, showing the scalability of any actor program. The out-of-order-message delivery is illustrated and utilized in the concurrency game Chameneos. JPF-actor can build from and utilize these concurrent programs, which demonstrate key aspects of the Akka language for actors, in order to retain compatibility with Java and Scala.

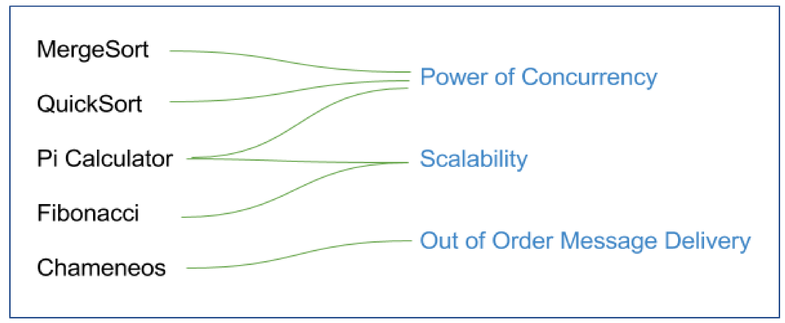

Figure 4. Actor Model Trait Exemplified

Conclusion and Future Work

Software-testing and model-checking is necessary to ensure the accuracy and sustainability of concurrent programs, especially because JPF was developed overall as a test kit by NASA. Compatibility is also important to keep up as software is constantly being improved. As a long-term project, we are at the beginning of exploring the new message-passing protocol implemented in Scala through Akka. These programs represent the beginning of Dr. Lauterburg's effort to update JPF-actor to accommodate Java 8 and the updated Akka and Scala languages. Future work on JPF-actor will build the groundwork of the basic Akka programs we have developed. Once JPF-actor is updated, it will have the functionality to test different Akka programs to make sure they are defect- and error-free.

Contact

<Dr. Steven Lauterburg>
<Salisbury University>
Email: stlauterburg@Salisbury.edu
Website: http://faculty.salisbury.edu/~stlauterburg/

References

1. "Documentation." *Documentation | Akka.* N.p., 2011-2017. Web. 20 July 2017. http://akka.io/docs/.
2. NASA."JPF–Actor."2010. https://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/jpf-actor.
3. Lauterburg, Steve. "Systematic Testing for Concurrent Software." 7 July 2017, PowerPoint File.