Verteilte hochverfügbare Anwendungen mit Erlang

Nachdem viele Jahre das Mooresche-Gesetz Gültigkeit besaß, kommt die Weiterentwicklung von Prozessoren an physikalische Grenzen. Demgegenüber wird der Traffic und die Anforderungen an Anwendungen immer größer. Die logische Konsequenz ist die horizontale Skalierung der Anwendung. Hierbei muss nicht nur an den Betrieb von Anwendungen auf mehreren Kernen einer CPU, sondern auch an den Betrieb einer Anwendung auf mehreren Servern gedacht werden.

Threads und das Problem mit „shared memory“

In Anwendung in Java, C#, Go, etc. kommunizieren Threads über einen gemeinsamen Speicherbereich („shared memory“) miteinander. Die sichere Kommunikation über „shared memory“ muss aufwändig via Mutex, Locks, Futures und anderen Techniken sichergestellt werden. Besonders problematisch wird es wenn ein Thread aufgrund eines unvorhergesehenen Ereignisses abstürzt. In diesem Fall kann der Zustand des „shared memory“ undefiniert sein. Es lässt sich nicht bestimmen, ob der Thread gerade in den „shared memory“ geschrieben hat und damit fertig war. Daher muss angenommen werden, dass der „shared memory“ korrupt ist. Somit sind nach einem Fehler in einem Thread auch alle anderen Threads betroffen.

Prozesse und Message Passing

Verwendet man hingegen Prozesse mit isoliertem Speicher, so lässt sich die Auswirkung eines Fehler auf einen Prozess beschränken. Alle anderen Prozesse können ohne Beeinflussung weiterlaufen. Zur Kommunikation zwischen den Prozessen kommen Nachrichten (Messages) zum Einsatz, welche von Prozess zu Prozess kopiert werden und die Isolation des Speichers nicht verletzen. Prozesse in diesem Modell lassen sich zudem auf unterschiedlichen Rechner ausführen, da ihre Speicherbereich getrennt sind und die Nachrichten auch zwischen Rechner ausgetauscht werden können.

Erlang

Eine der wenigen Sprachen die das Konzept der isolierten Prozesse und des Message Passing unterstützt ist Erlang. Erlang geht noch einen Schritt weiter und bildet die Kommunikation zwischen lokalen Prozessen und entfernten Prozessen (auf anderen Nodes) für den Entwickler vollkommen transparent ab. Hierdurch kann der Entwickler sich auf die Fachlichkeit der Anwendung konzentrieren.

Die Standard-Library „OTP“ ermöglicht zudem den Aufbau von Supervisor-Trees in denen Prozesse überwacht werden. In einer Fehlersituation werden sie vom Supervisor neu gestartet. Eine Fehler bleibt auf den betroffenen Bereich beschränkt. Ein kleiner Fehler hat somit nur eine kleine Auswirkung. Andere Teilsysteme der Anwendung laufen ohne Beeinflussung weiter. Durch die Orchestrierung der Anwendung in Supervisor-Trees erlaubt Erlang die Erstellung hochverfügbare Anwendungen.

Ein ausgereiftes Pattern Matching, Code-Austausch während der Laufzeit und die funktionale Natur runden das Gesamtbild von Erlang ab.

Einsatzbeispiele

Erlang wird an vielen Stellen eingesetzt und hat bewiesen, dass es sich außerordentlich gut für hochverfügbare verteilte Systeme eignet. Bei Ericsson lief die Telefonvermittlungsanlage AXD301 mit einer Verfügbarkeit von unglaublichen 99,9999999% (das sind neun Neunen!).

Das WhatsApp-Backend ist komplett in Erlang implementiert und läuft auf mehreren hunderten Erlang-Nodes auf mehreren Servern. Ein Node bedient hierbei eine Millionen Benutzer.

Mögliche Einsatzbereiche

Damit eignet sich Erlang hervorragend für Systeme mit hohen Zugriffszahlen und hohen Anforderungen an die Verfügbarkeit. (Web-)Server mit direkten Kundenkontakt bieten ein solches Szenario. Kein Kunde wird für einfache Anfragen lange Wartezeiten oder gar einen Ausfall über einen längeren Zeitraum akzeptieren.

Links:
* https://pragprog.com/articles/erlang
* https://t3n.de/magazin/erlang-247504/
* https://ratopi.de/erlang/Anwendungen-mit-Erlang--Pietsch.pdf

Dr. Ralf Th. Pietsch, April 2019