RESH und D-RESH: Fehlertoleranz mit aktueller Virtualisierungstechnik

Hans P. Reiser[1], Franz J. Hauck[1] und Rüdiger Kapitza[2]

[1] Distributed Systems Lab, Faculty of Computer Science, University of Ulm, Germany
Email: {reiserh,fjh}@acm.org
[2] Department of Distributed Systems and Operating Systems, University of Erlangen-Nürnberg, Germany
Email: rrkapitz@cs.fau.de

Bei vielen Fehlertoleranz-Infrastrukturen wird ein „fail-stop"-Fehlerverhalten vorausgesetzt. Dies ist allerdings nicht immer eine realistische Annahme. Zum einen kann zufälliges Fehlverhalten von Systemkomponenten zu beliebig unerwünschten Systemzuständen führen. Zum anderen sind Softwarefehler häufig der Auslöser von Problemen; die Komplexität von heutigen Systemen macht es oft praktisch unmöglich, die Fehlerfreiheit eines Systems zu verifizieren.

In den letzten Jahren haben Virtualisierungstechniken wieder große Beliebtheit erreicht (z. B. VMware, Xen). Zum Teil werden diese Techniken auch eingesetzt, um die Verfügbarkeit von Systemen zu erhöhen. Die Vereinfachung des „Deployments", der Migration von Diensten sowie der Ressourcenverwaltung leistet Unterstützung, um einen ausgefallenen Dienst auf einem neuen Rechner anzubieten. Eine Kapselung von Systemeinheiten (wie z.B. Gerätetreibern) in virtuellen Maschinen verhindert die gegenseitige Beeinflussung bei Fehlerhaftigkeit einzelner Komponenten. Zudem kann Virtualisierung zum externen Monitoring von Betriebssystemen mit dem Ziel der Erkennung von Angriffen eingesetzt werden. In unserem sich in Vorbereitung befindenden Projekt untersuchen wir neue, innovative Konzepte, um mit Hilfe von Virtualisierungstechnik hochverfügbare Dienste zu realisieren.

Unsere RESH-Architektur (*Redundant Execution on a Single Host*) erlaubt es, Dienste redundant in mehreren virtuellen Maschinen eines einzelnen Rechners auszuführen. RESH geht von einer Virtualisierungsarchitektur aus, wie sie von Xen bereitgestellt wird, mit einem minimalen Hypervisor unterhalb der Betriebssysteminstanzen (Domänen). Gewöhnlich bekommt dabei das Betriebssystem einer *Domain-0* direkten Zugriff auf die Hardware, während andere Domänen virtuelle Gerätetreiber verwenden, die mit Domain-0 interagieren. In RESH wird die *Domain-NV* (network/voting) in einer separaten virtuellen Maschine von Domain-0 abgespalten. Die Domain-NV enthält ein minimales Betriebssystem, das lediglich aus Gerätetreibern und Protokollinstanzen für den Netzzugriff sowie aus einem *Voter* besteht, der die redundante Ausführung kontrolliert. Die Architektur geht dabei davon aus, dass externe Clients mittels TCP/IP mit dem lokal replizierten Dienst interagieren. Diese Kommunikation wird von Domain-NV transparent abgefangen und an alle Replikate verteilt. RESH lässt sich zum einen für *N-Version-Programming* einsetzen. Unterschiedliche Betriebssysteme, Middleware-Infrastrukturen und Dienstimplementierungen können für einen Dienst in jeweils separaten virtuellen Maschinen ausgeführt werden. Der Voter sorgt dafür, dass sich Softwarefehler in einzelnen Instanzen tolerieren lassen. Zum anderen kann die gleiche Software redundant ausgeführt werden, um die Auswirkungen zufällig auftretender Fehler (z.B. nicht erkannte Bitfehler im Speicher) durch den Voter zu erkennen und nach außen zu verbergen.

Mit D-RESH (*Distributed RESH*) wird die RESH-Architektur um Replikation über mehrere physische Rechner erweitert. Die Domain-NV wird in D-RESH um eine hostübergreifende Gruppenkommunikations-Funktionalität ergänzt. Korrektes Verhalten des Hypervisor und der RESH-Instanz vorausgesetzt, führt die RESH-Architektur dazu, dass einzelne Rechner nach außen ein fail-stop-Ausfallverhalten zeigen. D-RESH sorgt dafür, dass solche Ausfälle vor Clienten verborgen werden.

Besonders geeignet für die RESH-Architektur ist moderne Standard-Hardware, die über mehrere CPUs (bzw. Mehr-Kern-CPUs) verfügt. Insgesamt wird durch RESH und D-RESH eine Infrastruktur bereitgestellt, mit der sich hochverfügbare Dienste effizient und basierend auf Standardhardware realisieren lassen.