Überblick

Virtualisierungsbasierte Fehlertoleranz

- Widersprüchliche Entwicklung
  - (Geschäfts-)Kritische Dienste werden zunehmend in die Cloud verlagert
  - Hard- und Software-Ausfälle sind nicht die Ausnahme, sondern die Regel
    - Abstürze einzelner Prozesse
    - Ausfälle ganzer Rechner und sogar Datenzentren

  → Generische Fehlertoleranzmechanismen für Anwendungen erforderlich
- Vorteile einer virtuellen Maschine (VM) gegenüber einer physischen
  - Kurze Startzeit
  - Einfache Migrierbarkeit (auch während der Ausführung)
  - Paralleler Betrieb mehrerer Maschinen auf einem Rechner möglich
- Herausforderungen
  - Wie lassen sich die speziellen Eigenschaften von virtuellen Maschinen für die Bereitstellung von Fehlertoleranzmechanismen ausnutzen?
  - Wie kann Fehlertoleranz als Dienst des Cloud-Anbieters realisiert werden?

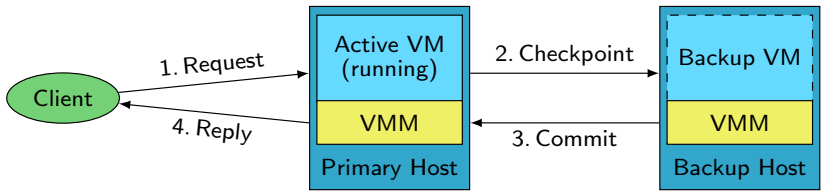
Remus

- Anforderungen
  - Anwendungs- und Hardware-unabhängiger Ansatz
  - Keine Modifikationen im Anwendungs- bzw. Betriebssystemquellcode
  - Kein externalisierter Zustand darf verloren gehen
- Remus
  - Behandlung einer virtuellen Maschine als Black-Box
  - Passiv replizierter Ansatz basierend auf häufigen Sicherungspunkten
  - Ausnutzung bereits existierender VM-Migrationsmechanismen
  - Steigerung der Effizienz durch spekulative Ausführung
  - Aufrechterhaltung von Netzwerkverbindungen im Fehlerfall
- Literatur

Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley et al.
Remus: High availability via asynchronous virtual machine replication
Proceedings of the 5th Symposium on Networked Systems Design and Implementation (NSDI '08), S. 161–174, 2008.

Architektur

- *Primary*
  - Ausführung der zu sichernden *aktiven* virtuellen Maschine
  - Periodisches Erzeugen von Sicherungspunkten der aktiven VM [z. B. alle 25 ms]
  - Puffern des ausgehenden Netzwerkverkehrs bis der Backup den Erhalt des korrespondierenden Sicherungspunkts bestätigt hat
- *Backup*
  - Auf letztem Sicherungspunkt basierendes VM-Image im Hauptspeicher
  - Senden von Bestätigungen für Sicherungspunkte
  - Ein Backup für mehrere Primaries denkbar



Migration von virtuellen Maschinen — Einschub

- Anforderungen
  - Möglichst keine bzw. kurze Dienstunterbrechung während der Migration
  - Möglichst geringe Dauer des Migrationsvorgangs
- Naiver Ansatz: *„Stop the World“*
  - Unterbrechung der laufenden VM auf dem Ausgangsrechner
  - Kopieren sämtlicher für die VM relevanter Daten auf den Zielrechner
  - Fortsetzung der VM auf dem Zielrechner
- Zu migrierende Ressourcen
  - Daten im Arbeitsspeicher
  - Daten auf der Festplatte
  - Netzwerkverbindungen
- Literatur

Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen et al.
Live migration of virtual machines
Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI '05), S. 273–286, 2005.

Migration des Arbeitsspeicherinhalts — Einschub

- Mechanismus zur Erkennung veränderter Seiten
  - Markierung aller Speicherseiten der Schattenseitentabelle als *rein-lesend*
  - Schreibzugriff auf Speicherseite löst einen Trap aus
  - VMM markiert entsprechende Speicherseite als *„dirty“*
- *Pre-Copy*-Ansatz
  - *Push*-Phase
    - Rundenbasiertes Kopieren von Speicherseiten auf den Zielrechner
    - Erste Runde: Duplizieren aller Speicherseiten
    - Runde n: Kopieren aller in Runde $n-1$ veränderten Speicherseiten
    - Mögliche Kriterien für das Ende der Phase (Beispiele)
      - Keine veränderten Speicherseiten seit der letzten Runde
      - Über mehrere Runden konstante Anzahl an veränderten Speicherseiten
      - Vordefinierte maximale Anzahl n_{max} an Runden
  - *Stop-and-Copy*-Phase
    - Stoppen der VM auf dem Ausgangsrechner
    - Kopieren der in der letzten Runde der Push-Phase veränderten Speicherseiten
    - Fortsetzen der VM auf dem Zielrechner

Hochfrequente Sicherungspunkte

- Arbeitsspeicher
  - Implementierung auf Basis des Stop-and-Copy-Mechanismus
    - Periodisches Suspendieren der aktiven VM
    - Kopieren modifizierter Speicherseiten in einen lokalen Puffer
    - Fortsetzen der zu sichernden aktiven VM auf demselben Rechner
  - Zur Ausführung asynchrone Übertragung des Pufferinhalts zum Backup
- Festplatte
  - Primary
    - Aufzeichnen aller Schreiboperationen auf dem Primary
    - Asynchrones Weiterleiten der Schreiboperationen an den Backup
  - Backup
    - Zwischenspeichern der Schreiboperationen im Arbeitsspeicher
    - Ausführen der Schreiboperationen, sobald Sicherungspunkt bestätigt wurde
- Backup: Bestätigung des Sicherungspunkts erst, wenn sowohl die Arbeitsspeicher- als auch die Festplattenaktualisierungen vorliegen

Konsistenz und Transparenz bei Fehlersituationen

- Netzwerkkonfiguration auf dem Primary
  - Eingehender Netzwerkverkehr wird direkt an die aktive VM weitergeleitet
  - Ausgehender Netzwerkverkehr wird gepuffert
    - Solange der korrespondierende Sicherungspunkt vom Backup nicht bestätigt wurde, reflektieren die gepufferten Nachrichten *spekulativen* Zustand
    - Bei Eintreffen einer Bestätigung: Versenden des gesicherten Teils des Puffers
- Umschalten auf den Backup im Fehlerfall
  - Fehlererkennung: Timeout nach Ausbleiben neuer Sicherungspunkte
  - Laden des letzten vollständigen Sicherungspunkts
  - Fortsetzen der virtuellen Maschine
  - Verlust eventuell auf dem Primary vorhandenen spekulativen Zustands
- Aufrechterhaltung von Netzwerkverbindungen
  - Ausnutzung zuverlässiger Übertragungsprotokolle (z. B. TCP)
  - Umschalten erscheint als temporärer Fehler, der toleriert werden kann
  - Erneutes Senden von Nachrichten erforderlich [Für die Anwendung transparent.]

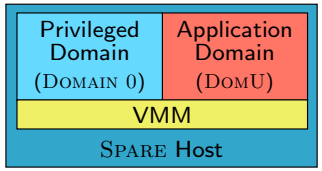
SPARE

- Anforderungen
  - Tolerierung beliebigen Fehlverhaltens der Cloud-Anwendung
    - Anwendungsprozess kann abstürzen
    - Dienstinstanz kann falsche Antworten liefern
  - Ausfalltoleranz für restliche Systemkomponenten
  - Tolerierung von maximal f gleichzeitig fehlerhaften Komponenten
  - Minimierung der Dienstunterbrechung im Fehlerfall
- SPARE
  - Nutzung eines hybriden Fehlermodells
  - Tolerierung von Fehlern durch aktive Replikation
  - Steigerung der Ressourceneffizienz durch Einsatz passiver Replikate
- Literatur

Tobias Distler, Rüdiger Kapitza, Ivan Popov, Hans P. Reiser et al.
SPARE: Replicas on hold
Proceedings of the 18th Network and Distributed System Security Symposium (NDSS '11), S. 407–420, 2011.

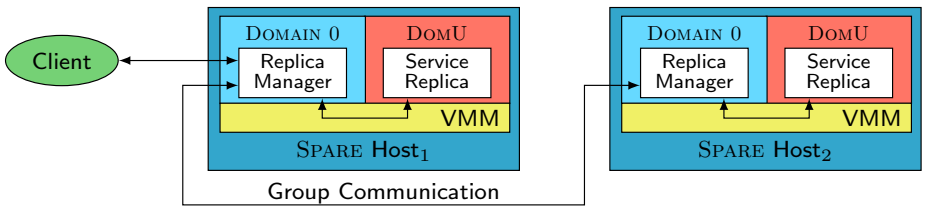
Aufbau eines SPARE-Replikats

- Hybrides Fehlermodell
  - Anwendungen von Cloud-Nutzern
    - Dem Cloud-Anbieter unbekannter, eventuell ungesicherter Code
    - Ausführung beliebiger Dienste
    - → Tolerierung **byzantinischer Fehler** in der Anwendung
  - Cloud-Infrastruktur
    - Betrieb und Wartung durch den Cloud-Anbieter
    - Bereitstellung einer begrenzten Anzahl an Diensten
    - → Tolerierung von **Ausfällen** im Rest des Systems
- Gleichberechtigte Replikate (Beispiel: Xen)
  - Isolation der Anwendung in einer virtuellen Maschine (DOMU)
  - Ausführung der Replikationslogik in der DOMAIN 0



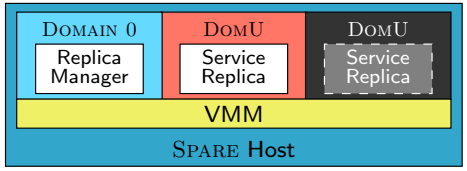
Basisarchitektur

- $f + 1$ physische Rechner
  - *Replikat-Manager*: Zentrale Komponente eines SPARE-Replikats
  - *Dienstreplikat*: Instanz der Anwendung
  - Gruppenkommunikation: Festlegung einer Reihenfolge für Client-Anfragen
- Bearbeitung einer Client-Anfrage
  1. Client sendet Anfrage an eines der Replikate
  2. Replikat-Manager R verteilt die Anfrage per Gruppenkommunikation
  3. Dienstreplikate führen die Anfrage aus und senden ihre Antwort an R
  4. R leitet die Antwort an den Client weiter, sobald sie *verifiziert* ist
     - Kriterium für erfolgreiche Verifikation: $f + 1$ identische Antworten
     - $f + 1$ identische Antworten: Beweis für Korrektheit bei maximal f Fehlern



Tolerierung von Fehlersituationen

- Replikat-Manager: Erkennung bzw. Vermutung von Fehlersituationen
  - Empfang unterschiedlicher Antworten [→ Mindestens ein Replikat ist fehlerhaft.]
  - Empfang von weniger als $f + 1$ Antworten [→ Vermutung eines oder mehrerer Ausfälle.]
  - → Zusätzliche Antworten für Verifikation erforderlich
- Einsatz eines zusätzlichen *Schattenreplikat*s pro physischem Rechner
  - Normalzustand: Pausierte virtuelle Maschine [→ Minimierung des Ressourcenverbrauchs]
  - Periodisches Aufwecken zum Einspielen von Zustandsaktualisierungen
  - Längerfristige Aktivierung im (vermuteten) Fehlerfall
- Tolerierung von Fehlern
  - Zusätzliche Bearbeitung der Anfrage durch aktivierte Schattenreplikate
  - Mehrheitsentscheid über alle vorliegenden Antworten



Unterstützung langlaufender Anwendungen

- Problem
  - Obergrenze von max. f Fehlern wird bei langlaufenden Diensten erreicht
  - Mehrheitsentscheid kann bei mehr als f Fehlern falsches Ergebnis liefern

  → Behebung von Fehlern erforderlich
- Lösungsansatz: *Proaktive Wiederherstellung* von Replikaten
  - Periodisches Erzeugen neuer VMs auf Basis verifizierter Zustände
  - Vorgehensweise
    1. Speicherung des Schattenreplikatzustands in einem Sicherungspunkt
    2. Anlegen eines neuen Schattenreplikats unter Nutzung des Sicherungspunkts
    3. Aktivierung des alten Schattenreplikats
    4. Umschalten der Dienstausführung auf das ehemalige Schattenreplikat
    5. Verwerfen der alten aktiven virtuellen Maschine
  - Vorteile
    - Fehlerobergrenze bezieht sich auf Fenster zwischen zwei Wiederherstellungen
    - Beseitigung nicht oder nur sehr schwer zu detektierender Fehler
- Zusätzlich: Manuelle Beseitigung schwerwiegender Hardware-Ausfälle

Zusammenfassung

- Remus
  - Transparente passive Replikation virtueller Maschinen
    - Primary-Replikat: Ausführung der zu sichernden VM
    - Backup-Replikat: Speicherung von Sicherungspunkten
  - Hochfrequente VM-Sicherungspunkte
    - Asynchrone Übertragung von Sicherungspunkten auf ein Backup-Replikat
    - Puffern des ausgehenden Netzwerkverkehrs bis zur Bestätigung des Backup
  - Umschalten auf das Backup-Replikat im Fehlerfall
- SPARE
  - Einsatz eines hybriden Fehlermodells
    - Beliebiges Fehlverhalten der Cloud-Anwendung
    - Ausfälle im restlichen System
  - Einsparung von Ressourcen mittels passiver Schattenreplikate
    - Periodisches Einspielen verifizierter Zustandsaktualisierungen
    - Bereitstellung weiterer Antworten für Fehlertoleranz per Mehrheitsentscheid
  - Proaktive Wiederherstellung von Anwendungsreplikaten