

## Virtualisierungsbasierte Fehlertoleranz

Motivation

Remus

## Remus

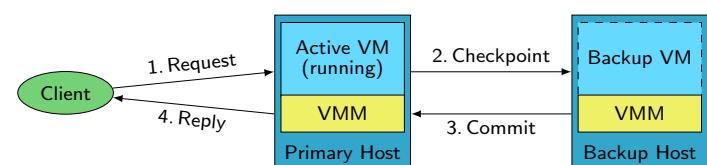
- Anforderungen
  - Anwendungs- und Hardware-unabhängiger Ansatz
  - Keine Modifikationen im Anwendungs- bzw. Betriebssystemquellcode
  - **Kein externalisierter Zustand darf verloren gehen**
- Remus
  - Behandlung einer virtuellen Maschine als Black-Box
  - **Passiv replizierter Ansatz** basierend auf häufigen Sicherungspunkten
  - Ausnutzung bereits existierender VM-Migrationsmechanismen
  - Steigerung der Effizienz durch **spekulativer Ausführung**
  - Aufrechterhaltung von Netzwerkverbindungen im Fehlerfall
- Literatur
  - **Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley, Norm Hutchinson, Andrew Warfield**  
**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.

## 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?

## 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



- Anforderungen
  - Möglichst **geringe Dauer des Migrationsvorgangs**
  - Möglichst keine bzw. kurze Dienstunterbrechung während der Migration
- **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, Eric Jul, Christian Limpach et al. **Live migration of virtual machines**  
*Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI '05)*, S. 273–286, 2005.

## 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

- 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

## 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.]