Distributed Peer-to-Peer Control for Harness

Christian Engelmann,
Oak Ridge National Laboratory

Überblick

- **Was ist Harness?**
- **Aufgabenstellung**
- **System Analyse**
- **System Entwurf**
- **Implementation/Test**
- **Zusammenfassung**

Was ist Harness?

- ***H*eterogeneous *A*daptable *R*econfigurable *N*etworked *S*ystem*S***
- **Nachfolger von PVM (Parallel Virtual Machine)**
- **Konzipiert als DVM (Distributed Virtual Machine)**
  - Dynamische Systemstruktur
  - Fehler-Toleranz (Hochverfügbarkeit)
  - Plug-In Mechanismus
- **Befindet sich in der Forschungs- und Entwicklungsphase**
- **Einsatzgebiet im verteilten Höchstleistungsrechnen**
- **Harness-Team:**
  - Oak Ridge National Laboratory (Oak Ridge, USA)
  - University of Tennessee (Knoxville, USA)
  - Emory University (Atlanta, USA)

Aufgabenstellung

- **Entwicklung eines verteilten Steuermechanismus (Distributed Control)**
  - Steuerung des verteilten Systems
  - Realisierung der Fehler-Toleranz
- **Hauptziel**
  - Entwicklung, Realisierung und Test eines Prototypen mit den Eigenschaften:
    - Korrektheit
    - Fehler-Toleranz
    - lineare Skalierbarkeit
    - Heterogenität
    - Effizienz

System Analyse

- **Hochverfügbarkeit in verteilten Systemen**
  - Redundanz von Hard- und Software
  - Servergruppe mit Hochverfügbarkeit von Diensten

- **Hochverfügbarkeit von Diensten**
  - hot-standby Redundanz durch aktive Replikation der Dienste
    - konsistente Replikation der Zustände der Dienste
    - kontinuierliche Fortführung von Diensten bei Ausfällen
  - warm-standby Redundanz durch passive Replikation der Dienste
    - konsistente Backups der Zustände der Dienste
    - Rückfall in den Backupzustand von Diensten bei Ausfällen

System Analyse

- **Hochverfügbarkeit einer verteilten virtuellen Maschine**
  - Hochverfügbarkeit eines verteilten Dienstes
  - Jeder Server ist Teil des Zustands des verteilten Dienstes.
  - Jeder Server kann den Zustand des verteilten Dienstes ändern.

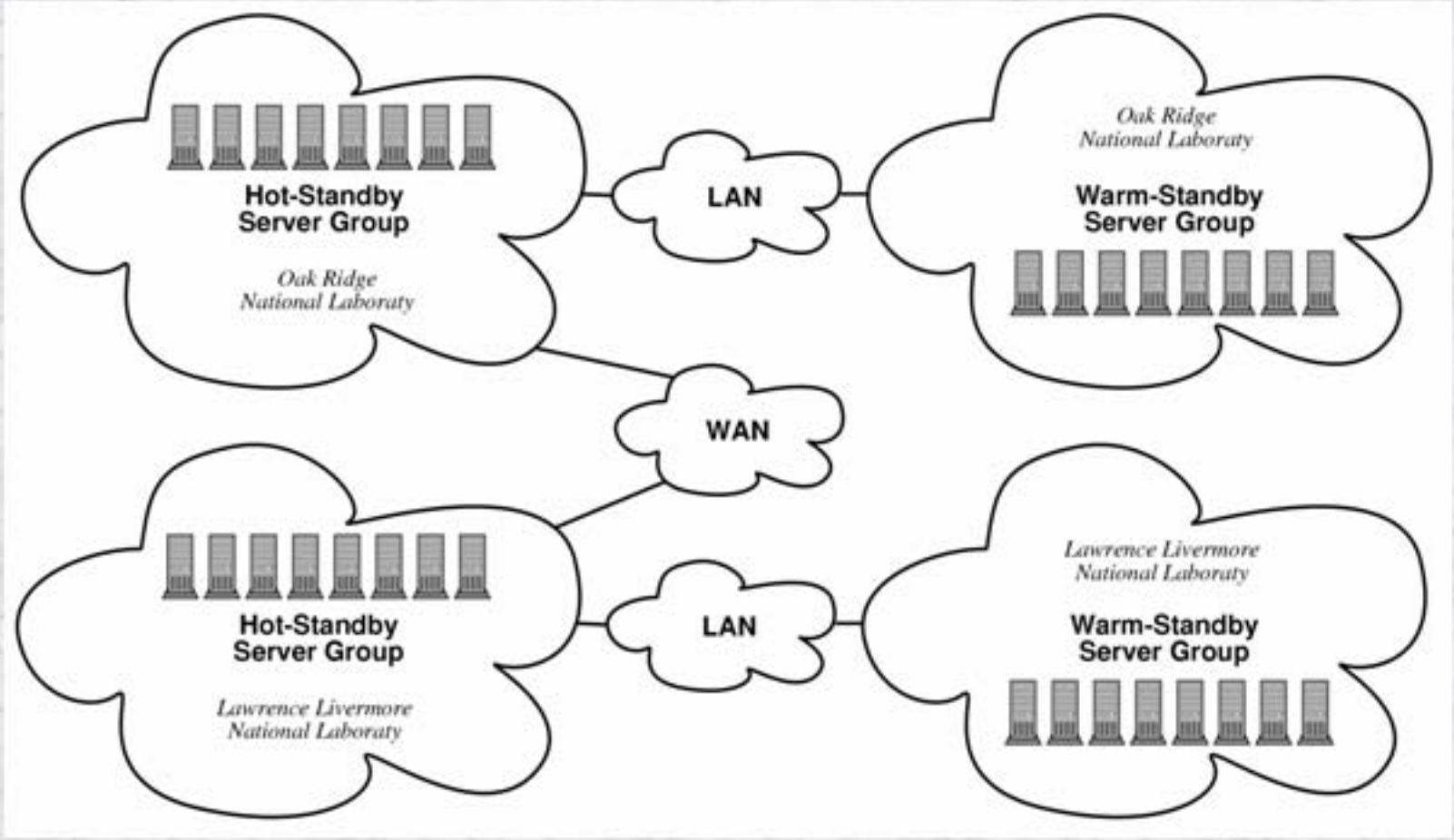
- **Notwendigkeit eines verteilten Steuermechanismus**
  - Replikation eines verteilten Dienstes
  - Zustandsänderungen eines verteilten Dienstes
  - Management der Servergruppe
  - Ausfallerkennung und -erholung

System Entwurf

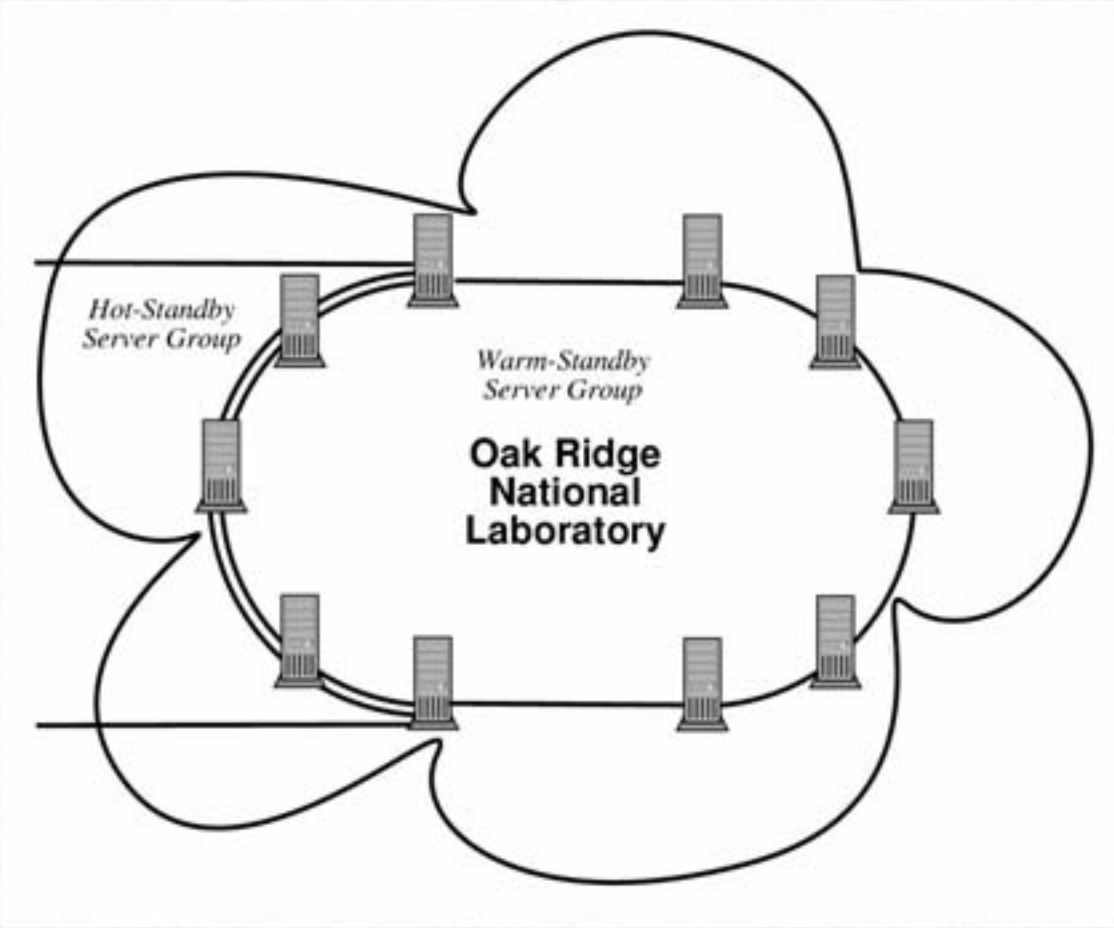
- **logische Architektur**
  - hot-standby Servergruppe mit aktiver Replikation
  - warm-standby Servergruppe mit passiver Replikation
  - regulierbare Hochverfügbarkeit
    (1 < Größe der hot-standby Servergruppe < Anzahl der Server)

- **physikalische Architektur**
  - unterschiedliche geographische Standorte von zusammenarbeitenden Rechenzentren
  - linear skalierbare und heterogenene Netzwerkarchitektur (TCP/IP peer-to-peer Ring)

System Entwurf



System Entwurf



System Entwurf

- **verläßlicher Broadcast (Reliable Broadcast)**
  - Zustandsänderungen werden allen Servern verläßlich mitgeteilt.
- **atomarer Broadcast (Atomic Broadcast)**
  - Zustandsänderungen werden allen Servern verläßlich und in der selben Reihenfolge mitgeteilt.
- **verteilte Abstimmung (Distributed Agreement)**
  - Zustandsänderungen werden nur mit Zustimmung der Servergruppe ausgeführt.
- **Transaktionssteuerung (Transaction Control)**
  - Zustandsänderungen werden auf allen Servern in der selben Reihenfolge ausgeführt.
- **Mitgliedschaft (Membership)**
  - Management der Gruppe

System Entwurf

- **Kommunikation in einem unidirektionalen Ring**

Algorithmus	vollständiges Netz	unidirektionaler Ring
verläßlicher Broadcast	$O = 2, \quad C = n^2$	$O = 2n, C = 2n$
atomarer Broadcast	$O = 2, \quad C = n^2$	$O = 2n, C = 2n$
verteilte Abstimmung	$O = n^2, \quad C = n^3$	$O = 3n, C = 3n$
Transaktionssteuerung	$O = 2n^2, C = n^3 + n^2$	$O = 4n, C = 4n$

- **Kommunikation in einem bidirektionalen Ring**

Algorithmus	vollständiges Netz	bidirektionaler Ring
verläßlicher Broadcast	$O = 2, \quad C = n^2$	$O = n, \quad C = 2n$
atomarer Broadcast	$O = 2, \quad C = n^2$	$O = n, \quad C = 2n$
verteilte Abstimmung	$O = n^2, \quad C = n^3$	$O = n + n/2, C = 3n$
Transaktionssteuerung	$O = 2n^2, C = n^3 + n^2$	$O = 2n, \quad C = 4n$

System Entwurf

- **Ausfallerkennung**
  - Jeder Server überwacht seine Ringnachbarn.
  - Jeder TCP/IP Fehler startet die Ausfallerholung.
- **Ausfallerholung**
  - Die Ringnachbarn eines Ausfalls regenerieren die Ringarchitektur.
  - Alle nicht bestätigten Nachrichten werden erneut gesendet.
  - Alle nicht verlorengegangenen Nachrichten werden gefiltert.

Implementation/Test

- **Prototyp**
  - bidirektionaler Ring
  - Transaktionssteuerung und Mitgliedschaft
  - Ausfallerkennung und -erholung
  - C++ unter Linux
- **Test**
  - *fast korrekt (0,01%) und fast fehler-tolerant*
  - Probleme mit der Ringsynchronization und der Ausfallerholung
  - heterogen, effizient und linear skalierbar
  - kein Verhungern von Servern

Zusammenfassung

- **Gelöste Probleme**
  - heterogene und linear skalierbare verteilte Systemarchitektur
  - effiziente und linear skalierbare Kommunikationsalgorithmen
  - symmetrische Algorithmen (keine Steuerung durch Token)
  - kein Verhungern von Servern
- **Ungelöste Probleme**
  - Ringsynchronization
  - Ausfallerholung
- **Zukünftige Arbeit**
  - komplette Überarbeitung/System Analyse (Review)
  - Implementation in ANSI-C (Optimierung)

Distributed Peer-to-Peer Control for Harness

Christian Engelmann,
Oak Ridge National Laboratory