Distributed Peer-to-Peer Control for Harness

by
Christian Engelmann,
Oak Ridge National Laboratory

Overview

- Introduction
- Objectives
- System Design
- Implementation
- Conclusion

What is Harness?

- *H*eterogeneous *A*daptable *R*econfigurable *N*etworked *S*ystem*S*
- Successor of PVM (Parallel Virtual Machine)
- Concieved as DVM (Distributed Virtual Machine)
- Enables Collaborative Computing
- Introduces Fault-Tolerance in Distributed Computing
- Provides Distributed Plug-In Mechanism
- Still in the Stage of Research and Development
- Collaborative Effort between:
  - Oak Ridge National Laboratory (Oak Ridge, USA)
  - University of Tennessee (Knoxville, USA)
  - Emory University (Atlanta, USA)

Objectives

- **Development of a Distributed Control**
  - to provide Fault-Tolerance in Harness
  - to avoid Single Point (or Set of Points) of Failure
    by automatic Detection of and Recovery from Faults
    (including Multiple and Cascaded Faults)

- **Main Goal**
  - Development, Implementation and Test of a Prototype which meets the Criteria of Correctness, Fault-Tolerance, Scalability, Heterogenity and Efficiency

System Design

- **High-Available Distributed System**
  - **Redundancy of Hard- and Software Components using a Server Group with a high availability of Services**
- **High-Available Service**
  - **Continuous Service (Hot-Standby) by** *consistent replication* **of the Service State to all Servers in a Server Group**
  - **Roll-Back and Restart of Service (Warm-Standby) by** *consistent backup* **of the Service State to all Servers in a Server Group**

System Design

- **High-Available Distributed Virtual Machine**
  - **High Availability of a *Distributed* Service**
  - **Every Member is part of the Service State**
  - **Every Member can change the Service State**

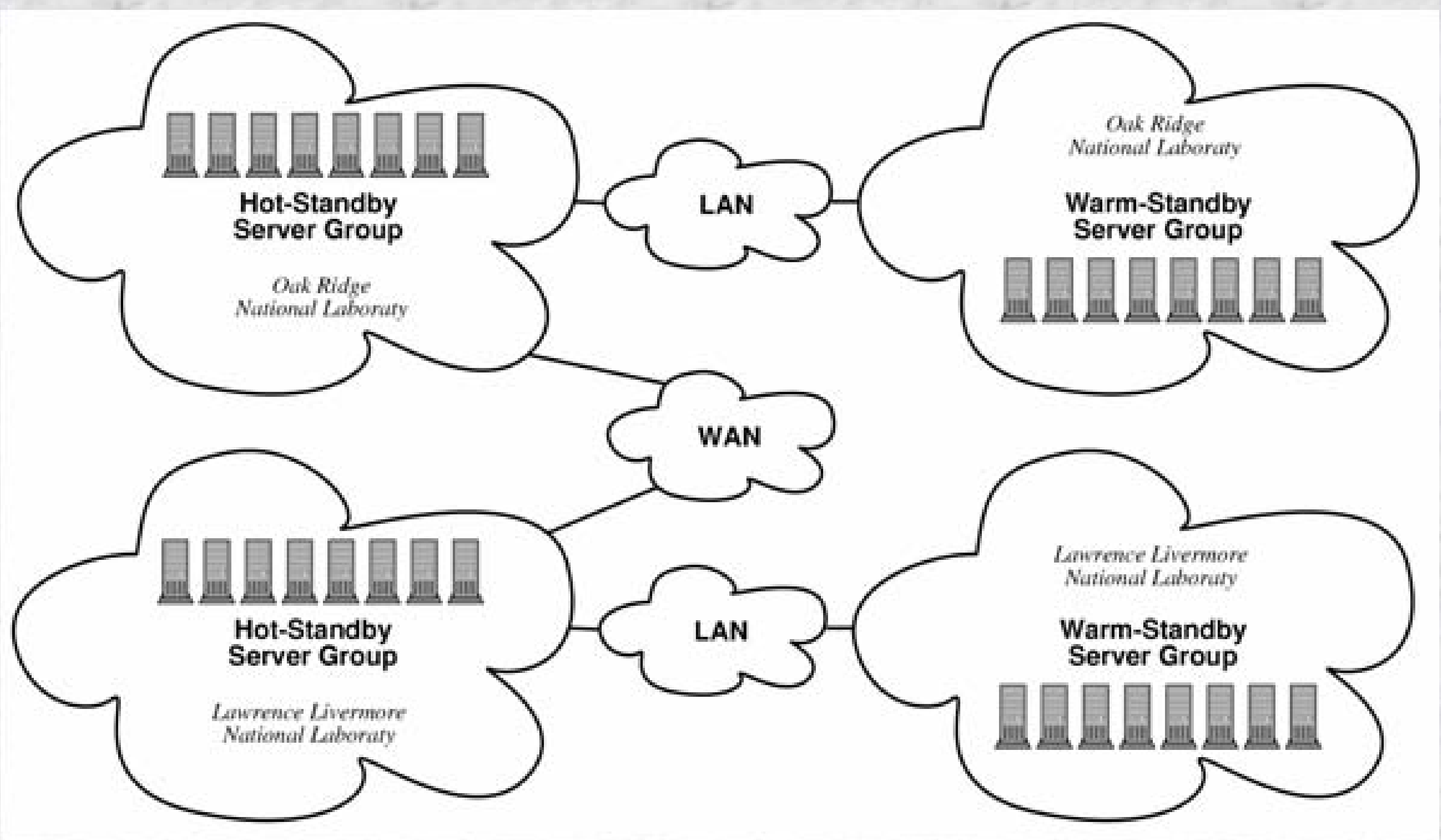
- **A Distributed Control is needed to manage:**
  - **State-Replication**
  - **State Changes**
  - **Membership**
  - **Fault Detection and Recovery**

Distributed System Architecture

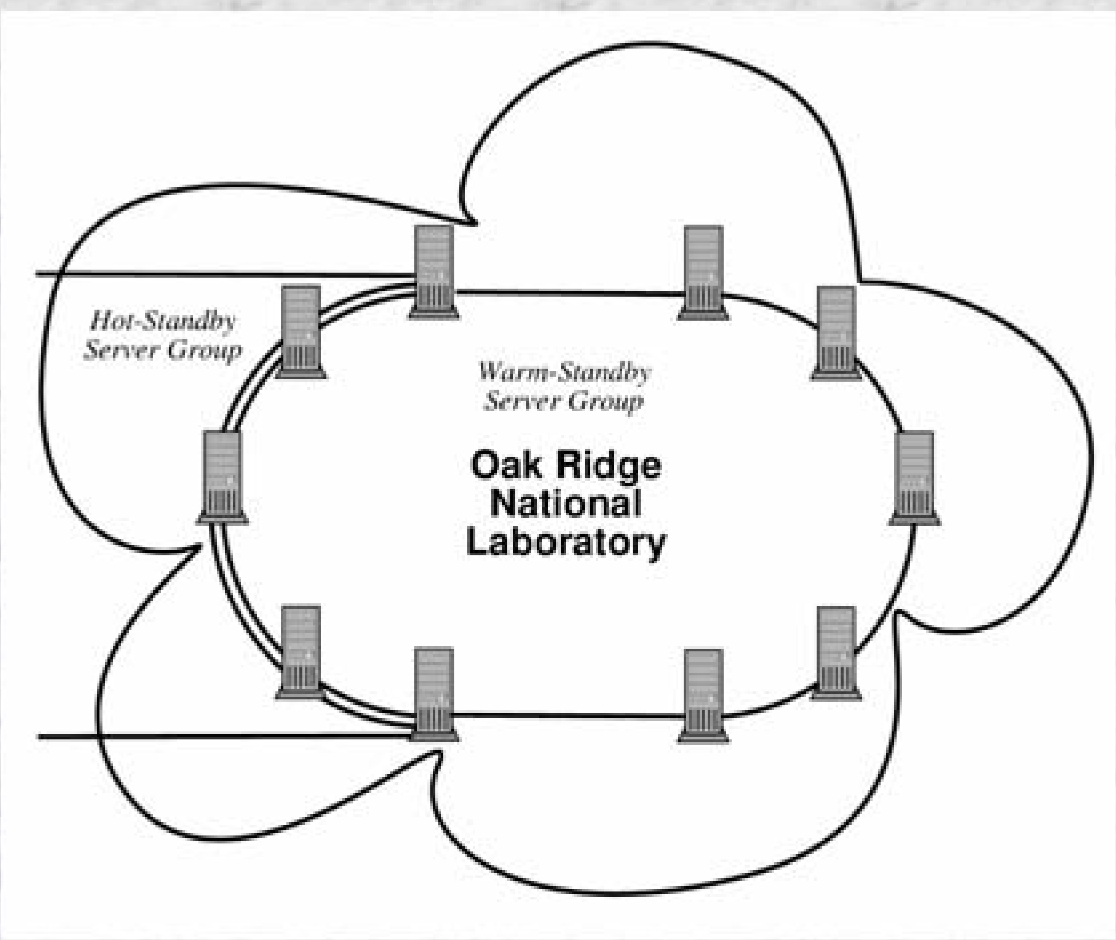
- **Logical Architecture**
  - **Hot-Standby Server Group with consistent State**
  - **Warm-Standby Server Group with backup State**
  - **Adjustable Degree of Fault-Tolerance**
    (1 < Hot-Standby Group Size < Number of Members)

- **Physical Architecture**
  - **Different geographical Locations of collaborating HPC Facilities**
  - **Linear-Scalable and Heterogeneous**
    Peer-to-Peer Network Architecture (TCP/IP-Ring)

Distributed System Architecture



Local System Architecture



Group Communication

- **Reliable Broadcast**
  - State changes must be broadcasted reliably.
- **Atomic Broadcast**
  - Broadcasted state changes must have an unique order.
- **Distributed Agreement**
  - All members must agree on a state change.
- **Transaction Control**
  - All members must have a linear history of state changes.
- **Membership**
  - All members must agree on an initial state.

Group Communication in a Ring

- **Group Communication in an unidirectional Ring**

Algorithm	Fully Connected	Unidirectional Ring
Reliable Broadcast	$O = 2, \quad C = n^2$	$O = 2n, C = 2n$
Atomic Broadcast	$O = 2, \quad C = n^2$	$O = 2n, C = 2n$
Distributed Agreement	$O = n^2, \quad C = n^3$	$O = 3n, C = 3n$
Transaction Control	$O = 2n^2, C = n^3 + n^2$	$O = 4n, C = 4n$

- **Group Communication in a bidirectional Ring**

Algorithm	Fully Connected	Bidirectional Ring
Reliable Broadcast	$O = 2, \quad C = n^2$	$O = n, \quad C = 2n$
Atomic Broadcast	$O = 2, \quad C = n^2$	$O = n, \quad C = 2n$
Distributed Agreement	$O = n^2, \quad C = n^3$	$O = n + n/2, C = 3n$
Transaction Control	$O = 2n^2, C = n^3 + n^2$	$O = 2n, \quad C = 4n$

Fault Detection and Recovery

- **Fault Detection**
  - The neighbor members detect faulty members.
  - Any occurring TCP/IP error starts the fault recovery.

- **Fault Recovery**
  - The neighbor members recover the ring architecture.
  - All not reliably broadcasted messages are sent again.
  - Already received messages are filtered.
  - A state change removes faulty members from the list of members.

Implementation

- **Prototype**
  - **Bidirectional Server Ring**
  - **Transaction Control and Membership**
  - **Fault Detection and Recovery**
  - **In C++ on Linux**
- **Test**
  - *Almost Correct and Almost Fault-Tolerant*
  - **Some Problems with the Ring Synchronization and Fault Recovery**
  - **Heterogeneous, Efficient and Linear Sclalable**
  - **No Starvation of Members due to a fair share**

Conclusion

- **Solved Problems**
  - Heterogeneous & Linear Scalable Distributed System Architecture
  - Efficient & Linear Scalable Group Communication Algorithms
  - Starvation of Members, Control Token, Time Stamps
- **Unsolved Problems**
  - Correctness of Ring Synchronization
  - Correctness of Fault Recovery
- **Future Work**
  - Complete Review and System Analysis based on the Results
  - Implementation in C (Optimization)

Distributed Peer-to-Peer Control for Harness

by

Christian Engelmann,

Oak Ridge National Laboratory