High Availability through Distributed Control

C. Engelmann, S. L. Scott, G. A. Geist
Oak Ridge National Laboratory

High Availability and Performance Computing Workshop 2004
Overview

- Harness: Concept, research areas and architecture.
- Highly available distributed virtual machine (DVM).
- High availability through distributed control.
- Peer-to-peer distributed control algorithm.
- Group communication: Reliable/Atomic Broadcast.
- Fault-tolerant group membership management.
- What we have learned, what next and why?
What is Harness

- A pluggable, reconfigurable, adaptive framework for heterogeneous distributed computing.
- Allows aggregation of resources into high-capacity distributed virtual machines.
- Provides runtime customization of computing environment to suit applications needs.
- Enables dynamic assembly of scientific applications from (third party) plug-ins.
- Offers highly available distributed virtual machines through distributed control.
- Various experiments and prototypes (C/Java).
Harness Research Areas

- Lightweight, pluggable software frameworks.
- Adaptive, reconfigurable runtime environments.
- Parallel plug-ins and diverse programming paradigms.
- Highly available distributed virtual machines (DVMs).
- Advanced ultra-scale approaches for fault tolerance.
- Fault-tolerant message passing (FT-MPI).
- Mechanisms for configurable security levels.
- Dynamic, heterogeneous, reconfigurable communication frameworks (RMIX).
Harness Architecture

- Light-weight kernels share their resources.
- Plug-ins offer services.
- Support for diverse programming models.
- DVM-enabling plug-ins provide a virtual layer.
- Highly available DVM using distributed control.
- Highly available plug-in services via DVM.
Highly Available DVM

- Parallel virtual machine (PVM).
  - Every node runs a virtual machine (VM).
  - The set of VMs is controlled by a master.
  - The master is a single point of control and failure.

- Distributed virtual machine (DVM).
  - All nodes form a single distributed virtual machine.
  - They equally control the DVM in virtual synchrony.
  - Symmetric state replication assures high availability.
  - No single point of control or failure.
  - The DVM survives while at least one is still alive.
DVM Architecture

DVM Maintains Global State Via Distributed Control

Dynamically Customize And Extend Via Plug-ins

Split and Merge With Other DVMs

User Features

DVM Control

H2O Kernel

- DVM Plug-In
- PVM Plug-In
- FT-MPI Plug-In

HARNESS Daemon

Host A

Host B

Host C

Host D

Another VM

Host Z
Distributed Control

- Steers processes like one virtual process.
- Allows each to have its own independent state.
- Local states form a global virtual process state.
- Mutual exclusive state changes for virtual synchrony.
- Symmetric global state replication for high availability.

Various abstraction models:
- Distributed shared memory.
- Distributed locking of state database.
- Less synchronized peer-to-peer mechanisms.
- Asynchronous distributed agreement (equilibrium).
Peer-to-Peer Distributed Control

- Unidirectional TCP/IP peer-to-peer ring network.
- Message broadcast is completed after one round.
- Acknowledgement round for Reliable Broadcast.
- Message numbering to achieve Atomic Broadcast.
- Collective communication for Distributed Agreement.
- State change type depends on execution target.
Group Membership Management

- Members agree on an initial state:
  - Every new ring node adopts the current state.
- Members maintain a linear history of state changes:
  - Atomic Broadcast of state changes.
  - Distributed Agreement on execution results.
  - State change commit depending on final result.
- Fault tolerance - Members maintain correctness:
  - Faulty ring nodes are immediately removed.
  - Healthy ring node neighbors reconnect the ring.
  - Possibly lost messages are sent again.
What We Have Learned

Many other past and ongoing research in group communication and distributed virtual processes exist.

Main focus is mostly on specific applications and/or communication technologies (e.g. TCP/IP in Harness).

Most solutions are embedded in applications, part of inflexible middleware or pure communication layers.

It is still very hard to implement distributed control.

It is even harder to enable any application or system service to take advantage of high availability.

Implementations very often “reinvent the wheel”.

What Next and Why

A modular, configurable, light-weight framework can provide more flexible support for different:

- Abstraction models.
- Distributed control algorithms.
- Data replication algorithms.
- Communication technologies (like in Open MPI).

Framework goes beyond pluggable protocols.

A common API can reflect all abstraction models.

Existing solutions can be put into pluggable modules.

Enables collaboration of different research groups.
What Next and Why

- High availability services as part of the OS.
- Support for multiple head or service nodes with active/active or active/hot-standby high availability.
- Support for highly available system services, such as:
  - Message passing (e.g. MCA layer of Open MPI).
  - Job schedulers.
  - Parallel file systems.
- Support for highly available scientific applications.
  - For quick response times: earthquake prediction.
  - For long running times: global climate models.
High Availability through Distributed Control

Questions or Comments?

C. Engelmann, S. L. Scott, G. A. Geist
Oak Ridge National Laboratory

High Availability and Performance Computing Workshop 2004