Distributed Troubleshooting Agents

Charles Earl, Emilio Remolina, Jim Ong

Stottler Henke Associates
{earl,remolina,ong}@shai.com

John Brown
Pentum Group,Inc.
johnbrown@pentum.com

Abstract

Key issues to address in autonomic job recovery for cluster computing are recognizing job failure; understanding the failure sufficiently to know if and how to restart the job; and rapidly integrating this information into the cluster architecture so that the failure is better mitigated in the future. The Agent Based High Availability (ABHA) system provides an API and a collection of services for building autonomic batch job recovery into cluster and grid computing environments. An agent API allows users to define agents for failure diagnosis and recovery. It is currently being evaluated in the U.S. Department of Energy's STAR project.

1. Introduction

In production high-performance cluster computing environments, batch jobs can fail for many reasons: transient and permanent hardware failures; software configuration errors; insufficient computing, storage, or network resources; incorrectly specified application inputs or buggy application code. Simplistic job recovery policies (e.g. blind restart) can lead to low quality of service and inefficient use of cluster resources. To provide high throughput and high reliability, it is necessary to determine the cause of task failure in enough detail to select and execute the appropriate job recovery. While many job failures require human intervention for proper troubleshooting and repair, a significant number can be delegated to autonomic software.

We are developing a platform called the Agent Based High Availability (ABHA) that provides autonomic recovery for batch jobs running on cluster and grid computing environments. ABHA is in use at the U.S. Department of Energy's STAR project [2] at Lawrence Berkeley National Laboratory (LBNL).

2. Architecture

A complete model for autonomic job recovery has to address four problems: 1) recognition of job failure; 2) determination of appropriate failure recovery, which may require diagnosis to select between alternatives; 3) the ability to initiate recovery actions; and 4) using that knowledge to avoid or mitigate the failure in the future.

ABHA uses a collection of distributed agents to address these problems. Agents provide robustness, local monitoring and recovery with global communication, and separation of concerns for creating new error management details.

Figure 1 depicts the core components of the system in a typical configuration. Agents collect information about the system and jobs running on it and share that

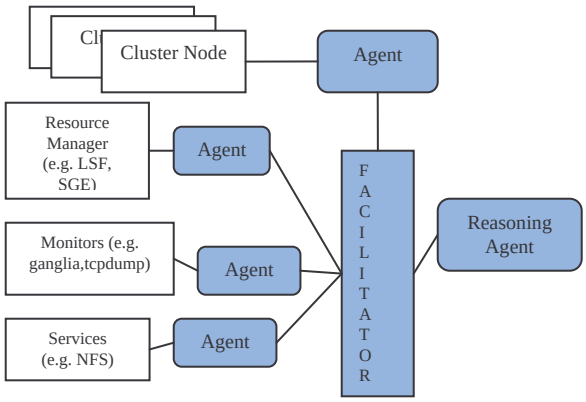

Figure 1: ABHA Architecture

information with other agents by producing events that are distributed by a centralized *Facilitator*. Agents use this shared information to predict and diagnose job failures, make job recovery recommendations, and autonomously perform job recovery. Agents can be deployed on various nodes throughout the cluster as dictated by the configuration of the site. For example, agents can collect information from services deployed through the system (e.g. NFS), gather information from and issue commands to distributed resource managers (e.g. SGE [3] or LSF [4]), filter and interpret information collected from other system monitors (e.g. Ganglia [5], or tcpdump), or perform complex reasoning based on collected information.

ABHA agents can be written in the rule-based language of JESS [6], or C++, Java, and Perl. Agents can be designed and deployed also using our Simbionic[6] agent toolkit.

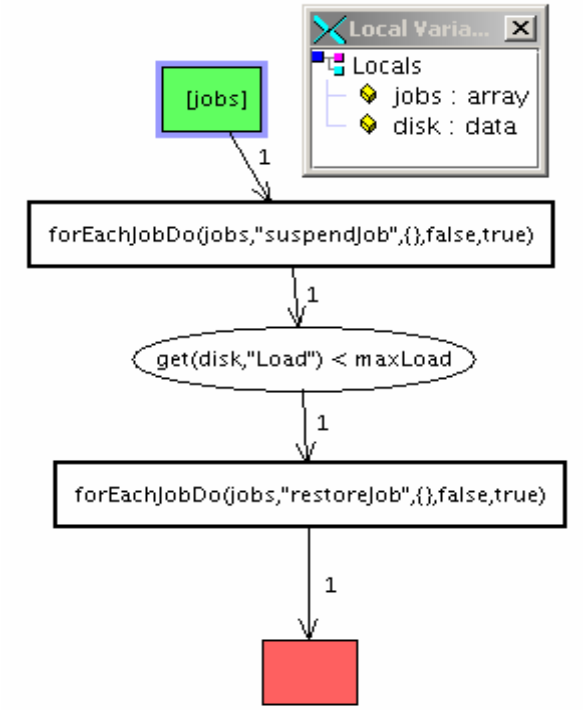

Figure 2. Defining Agent behaviors in Simbionic. Agent behaviors are defined using a flowchart-like graphical language but with the constructs of state-of-the-art programming languages. In the figure, the agent suspends some jobs accessing a disk vault (described in Section 3), waits until the disk vault is acceptable, and then resumes the jobs.

Simbionic provides developers an IDE to specify agent behaviors by drawing Behavior Transition Networks (BTN) (Figure 2). A BTN is a kind of finite state machine. Unlike state machines, BTNs provide constructs common to programming languages: they (i) are hierarchical, (ii) have local variables, (iii) have access to blackboards, (iv) are polymorphic, and (v) can sent messages to other BTNs. Patterns useful to the job management domain can easily be represented as BTNs: for instance, monitors, triggers, timeouts, and parallel execution of repair procedures. Finally, a BTN can execute arbitrary Java/C++ code and can be fully integrated with rule-based systems (i.e., JESS) giving developers a full range of programming power while maintaining a graphical behavior representation that can be understood and modified by non-developers.

3. Example

The STAR production cluster at LBNL [7] maintains a clustered file system for storage of experimental data. Each node is referred to as a disk vault. The batch job will be assigned to run on one of 344 compute nodes and will access data that is remotely mounted on one of the 65 disk vaults. If too many jobs try to read data at the same time, the disk vault goes into a thrashing mode and only reboot can bring it back. A reboot can be avoided by intervening when disk vault I/O reaches a critical value, by suspending jobs accessing the overloaded vault, adjust their resource requirements, and shepherd each job when the load on the vault reaches acceptable levels.

We developed a set of ABHA agents for addressing this case using the Simbionic. These agents interact as follows:

- A ganglia agent filters information from the Ganglia monitor, sending an event when the load of any disk vault exceeds a threshold.
- The Reasoner agent (a Simbionic-JESS mixture) then requests the tcpdump agent to determine which nodes access the vault and what I/O bandwidth they use.
- The Reasoner then requests the lsf agent to determine the jobs running on the offending nodes.
- The Reasoner then determines users causing the problem by correlating users, jobs, nodes where the jobs run, and node I/O access to the disk vault.
- Emails to alert the users and the administrators are then run, as defined by some administrator policy.
- Under the administrator command, the repair policy in Figure 2 is executed.

The ABHA system addressing the diskvault problem as described above has been in use on the STAR production cluster at LBNL since January 2005, providing support for analysis and troubleshooting of diskvault failures.

4. Remaining Work

A Grid service implementation of ABHA is also being developed for the STAR Grid project [7].

References

1. STAR experiment website http://www.star.bnl.gov/.
2. SGE website http://www.cs.wisc.edu/condor/.
3. Platform Computing LSF http://www.platform.com
4. Ganglia project website http://ganglia.sourceforge.net/.
5. Simbionic website at http://www.simbionic.com/
6. JESS website at http://herzberg.ca.sandia.gov/jess
7. Parallel Distributed Systems Facility website http://www.nersc.gov/nusers/resources/PDSF/