ABHA: A Framework for Autonomic Job Recovery

Charles Earl,
Emilio Remolina, Jim Ong

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

John Brown
Pentum Group,Inc.

johnbrown@pentum.com

Chris Kuszmaul
chris_kuszmaul@
hotmail.com

Brad Stone
bstone@
aspirinsoftware.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 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 [1] 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 being tested in the context of the U.S. Department of Energy's STAR project [2] at Lawrence Berkeley National Laboratory (LBNL). We are now evaluating it on production facilities there.

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.

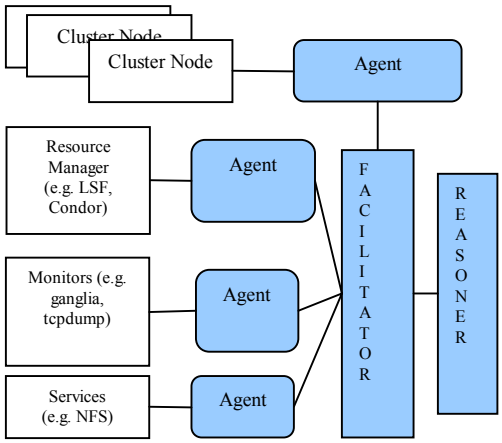

Figure 1: ABHA Architecture

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 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 gather information from and issue commands to distributed resource managers (e.g. Condor [3] or LSF [4]), filter and interpret information collected from other system monitors (e.g. Ganglia [5]), provide detailed information from specific jobs, or collect information from services deployed through the system (e.g. NFS).

ABHA deploys a centralized Reasoner (based on the Java Expert System Shell [6]) that interprets rules that are run against the events sent to the Facilitator. The behavior of remote agents can also be specified using rules. ABHA provides C++, Java, and Perl APIs for developing agents. The Facilitator is implemented using the Java Message Service (JMS) API and can be configured to provide fail-over and persistent event storage. A graphical user interface allows inspection of events and control of agents.

3. An Example

One example provides an illustration of the functionality of ABHA and the kinds of recovery issues that it can address. 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 typical STAR 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. An administrator can suspend jobs accessing the overloaded vault, adjust their resource requirements, and shepherd each job them the queue until the load on the vault reaches acceptable levels.

We developed and tested a solution to this problem on our local cluster. Rules loaded by the Reasoner agent direct diagnosis and recovery. The main rule is paraphrased below.

```
IF(high_diskvault_load ON ?dv AT ?T1)
 AND(max_dvio_consumer ?dv ?node ?T1)
 AND (lsf_job ?node ?job ?T1)
 AND (job_mounts ?job ?dv)
THEN
 (lsf_suspend (jobs_using_vault ?dv))
(restart ?job)
 (UNTIL(normal_diskvault_load ON ?dv)
      (lsf_restart (pick ?jobs)))
```

A ganglia agent filters information from the ganglia monitor, sending `high_diskvault_load` when the load on one of the disk vault machines exceeds a threshold.

The Reasoner agent then requests the tcpdump agent to determine which machine consumes the most I/O bandwidth with respect to the vault. The tcpdump agent posts this information as a `max_dvio_consumer` event.

The Reasoner then requests the lsf agent to determine the jobs running on the offending host, and returns these in an `lsf_job` event. The rule then requests mount information from local_node_monitor agent on the node on which the job is running. The local_node_monitor agent returns this information in a `job_mounts` event. The Reasoner then follows the THEN part of the rule: it suspends jobs running against the disk vault, adjusts the priority of the offending job, and once the offending job has finished, restarts remaining jobs, until the load on the disk vault returns to normal.

4. Remaining Work

We are evaluating on the PDSF production cluster. A Grid service implementation of ABHA is also being developed for the STAR Grid project [7].

References

1. Chess, D., Kephart, J.: The Vision of Autonomic Computing. IEEE Computer Magazine 1 (2003) 41-50.
2. STAR experiment website http://www.star.bnl.gov/.
3. Condor project website http://www.cs.wisc.edu/condor/.
4. Platform Computing LSF http://www.platform.com
5. Ganglia project website http://ganglia.sourceforge.net/.
6. JESS website at http://herzberg.ca.sandia.gov/jess
7. Parallel Distributed Systems Facility website http://www.nersc.gov/nusers/resources/PDSF/