CHAF - An Object-Oriented Framework for Configuring Applications in a Clustered Environment

Augustus F Diraviam, Ritu Agrawal, Madhur Bansal, Krishna Janakiraman
Sun Microsystems Inc.
{augustus.diraviam,ritu.agrawal,madhur.bansal,krishna.janakiraman}@sun.com

Abstract - **In high availability clustering solutions, an application must be configured properly to run within the framework of the high availability solution. This configuration is often cumbersome, and thus the probability of configuration errors increases. In this paper, we propose a framework, called CHAF, to build configuration tools for high availability clustering solutions. We show how CHAF is scalable to a variety of applications. Finally, we provide implementation snapshots of a feature that has been implemented using CHAF. This feature was successfully released as part of the most recent release of the Solaris Cluster product (Solaris Cluster 3.2).**

I. Introduction

Configuring applications on a cluster requires sound knowledge of the clustering product and the application. Configuring applications is difficult; largely because there are many configuration steps to go through to configure an application. For instance, to configure a typical database application, there will be a data store that will be setup to store the data and there will be a network interface through which the database application can be accessed by its clients. To make this database application highly available (HA), the database application must be configured within a clustering system. In addition the data store and the network interface must be setup to be highly available too. The dependencies between the data store, the network interface and the database application must be configured appropriately. These configuration changes are typically done by humans. The configuration setup is also different for different applications. The difficulty in configuration is more pronounced if there are different types of dependencies between different components of the application. The above fact is severe in the case of complex applications and on clusters with a large number of nodes. It has been often reported that many failures in complex systems are due to people and process issues [1]. In fact a Boeing study [2] indicated that, people or process caused 72% of the fatal commercial aircraft incidents between 1959 and 2001. Majority of the failures on high availability systems are due to wrong configurations on the clustering system made by people [3].

This paper proposes a framework, called CHAF, that automates the configuration of applications in a cluster environment. CHAF is used by front end applications, typically *wizards*. For each application, there is a wizard that will gather application and cluster specific details from the user. At each step, the wizard uses CHAF for discovering and validating data. Once the necessary data is obtained, the wizard uses CHAF to generate the commands necessary to configure the application on the cluster nodes. In case of any failures in the configuration attempt, CHAF provides a way to undo the configuration changes made.

CHAF hides the details about the clustering infrastructure from a user. It enables a user, with only knowledge about the application, to configure the application for HA. CHAF is easily extensible to many applications. The benefit of CHAF is that the complexity of configuring an HA application is hidden from the user.

This paper is organized as follows. Section 2 summarizes other research work related to this topic, and Section 3 describes the architecture of CHAF. Section 4 illustrates the implementation of CHAF as configuration wizards for the Solaris Cluster 3.2 product [9]. Section 5 gives concluding remarks.

II. Related Work

Automating the configuration of highly available cluster applications is a relatively new area of work, but is receiving a great deal of attention from the highly available community due to its impact on system uptime. The Open Framework for Clustering addresses configuration of applications as part of the overall framework [4]. This framework offers a self-configuration API that allows a plug-in to instantiate itself without the need to write new user interfaces. However, this self-configuration API does not provide methods to modify application configuration. In addition, it is not clear how the configuration dependencies with file systems and network resources are resolved with the application plug-ins. There are multiple approaches that try to configure the servers that are part of the cluster dynamically [5], [6]. These approaches focus on the membership of a cluster and the configuration of such a member, but these do not focus on the initial configuration of applications on the cluster. While there are other user interface tools for configuring applications on clusters, a framework for developing such tools are not published for these [7], [8].

III. Architecture of CHAF

In HA systems, an application is made HA, by programmatically moving the application and its components from the compute node that has failed to another compute node. This process of moving applications from one node to another is called, a failover. To perform a failover, the HA system must know how to start, stop and monitor an application and its components. We use the term "resource" to denote a individual component that the HA system recognizes. In the HA system, a resource is way for the system to know the exact set of operations to perform to start, stop and monitor the application. For instance, a network interface should be configured as a "resource" so that the HA system can act on it. Every HA application must be encompassed into a resource. A set of resources grouped together is called a "resource group". We will treat a "resource group" as the unit that will move to another node or get restarted during failures. There will be dependency relationships between resources and between resource groups. These relationships determines the order in which the individual resources provide services, and reach the "online" state. They also determine which compute node is chosen among a groups of compute nodes to host a resource group.

The typical system management framework, written using Java Management Extensions (JMX) [10], consists of a set of Managed Beans. The user interfaces uses these Managed Beans (MBean) to perform all system management operations on the HA system. Every MBean represents a object of the HA system to be managed. For instance, there is a MBean that manages resources. This MBean exposes interfaces that other components of the system can used to create, delete, modify and perform any allowed operation on the object, the resource. Similarly, there is a MBean for other objects of a HA System, like disks, resource groups and network adapter groups.

CHAF described in this paper is part of a system management infrastructure, as described in the previous paragraph. In CHAF, for each application there are the following components.

- a Service Definition File (SDF). This is a XML file that defines the HA application and its required dependencies. A data structure is built from the SDF for the implementation.
- a wizard to gather the HA application details,
- a MBean, which exposes the discovery, validation and command generation methods. The MBean is specific for a particular HA application.

In the following paragraphs we give brief information on each of the above components, and then we finally give the sequence of steps towards configuring an HA application on the cluster.

A. Components of CHAF

1) Service Definition File: A Service Definition File (SDF) is a XML file that defines the configuration details for a particular application. It can state, for example, the resources and their types, the logical grouping of these resources and the dependencies among the resources and its resource groups. It also provides a path to a library which contains the discovery and validation algorithms for the application. This clear separation of the discovery and validation algorithms of the application, from the actual configuration interfaces, makes CHAF scale to many different applications with ease.

For example, to configure a Scalable Apache application, we need a failover resource group with a shared address resource and a scalable resource group with storage resources and the scalable apache resource. Fig.1 illustrates the Scalable Apache configuration in Solaris Cluster. We will use this apache configuration as a example in this paper.

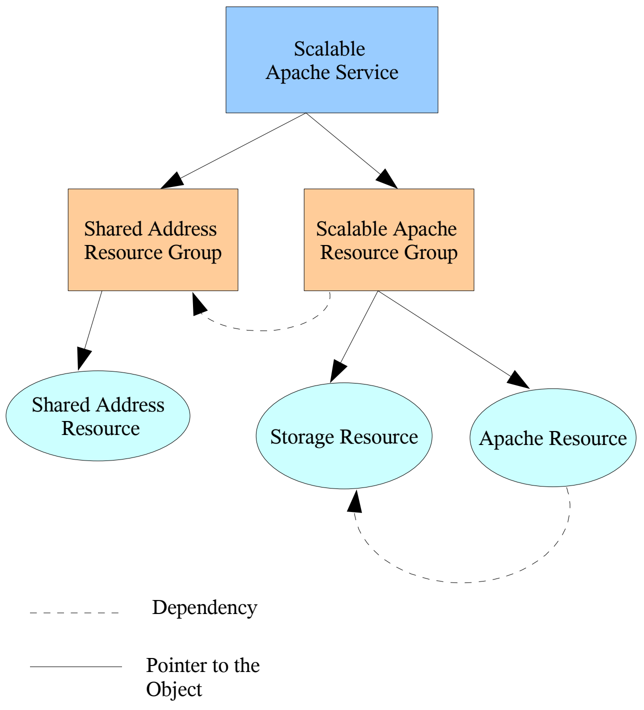


Fig. 1 Scalable Apache Configuration

The application configuration data structure is an in-memory representation of the SDF. The data structure is constructed primarily using the application information, resource group information and resource information classes. These classes encapsulate the information required for configuring a application, a resource group and a resource respectively. Fig. 2 illustrates the general data structure used for configuration. The classes illustrated in Fig. 2 are used to explain the operation of CHAF in the later sections of this paper.

The application information class contains a list of resource group information class instances. The resource group information classes contains a list of resource information class instances. In the example of the scalable apache configuration, the Scalable Apache service will be represented by a instance of application information class. The shared address resource group and the scalable apache resource group boxes in Figure 1 will be represented by instances of the resource group information class. The resources in Figure 1 will be represented by instances of resource information classes.

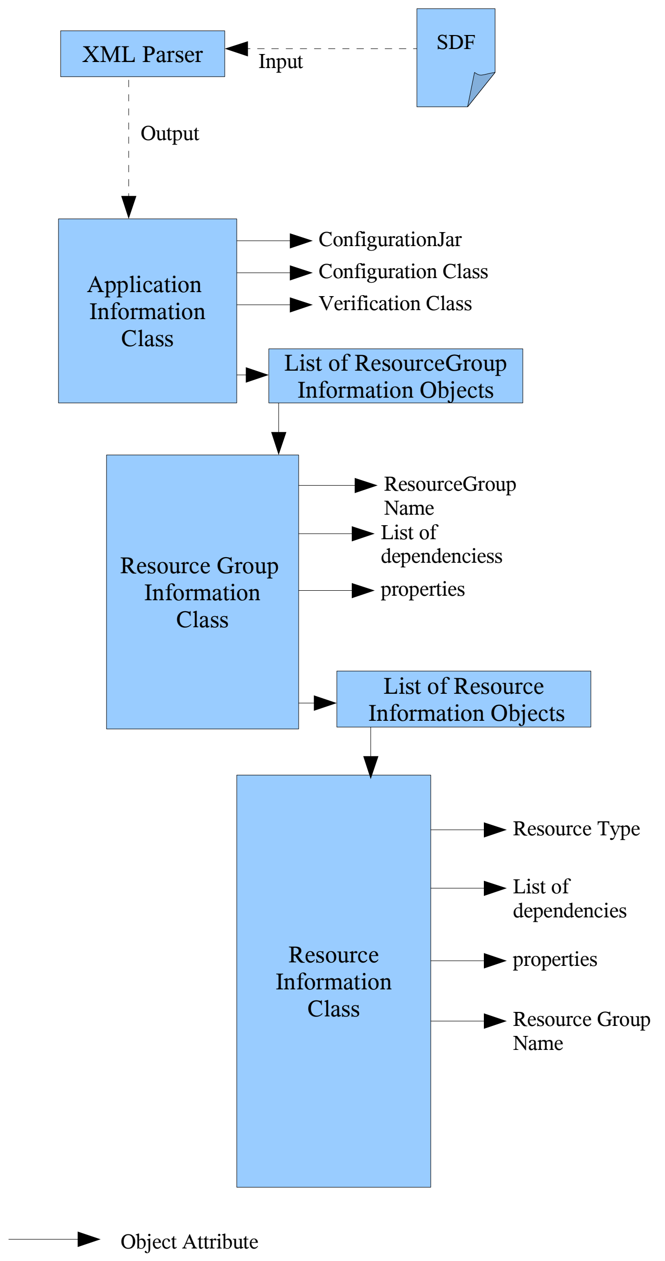


Fig. 2 Application Configuration data structure

2) Wizard: The wizard gathers information from a user on configuring the application on the cluster. For example, it asks the user the exact list nodes of the cluster, on which the application should be configured to run. It asks specific questions related to the cluster objects and the application. At each step the wizard discovers possible user options, using the the discovery algorithms provided by the application specific library. The path to the library is indicated in the SDF.

3) MBean: The Mbean of the application acts as an interface between the wizard and the underlying clustering infrastructure components. The wizard uses the operations exposed by the Mbean to discover, validate and finally configure the application.

The application MBean uses the application information class to get specific application information for a specific configuration. The knowledge of creating different cluster objects are already present in the different MBeans present in the system management infrastructure. Using these, commands are generated by the application MBean to perform the actual configuration. This relationship is illustrated in Figure 3.

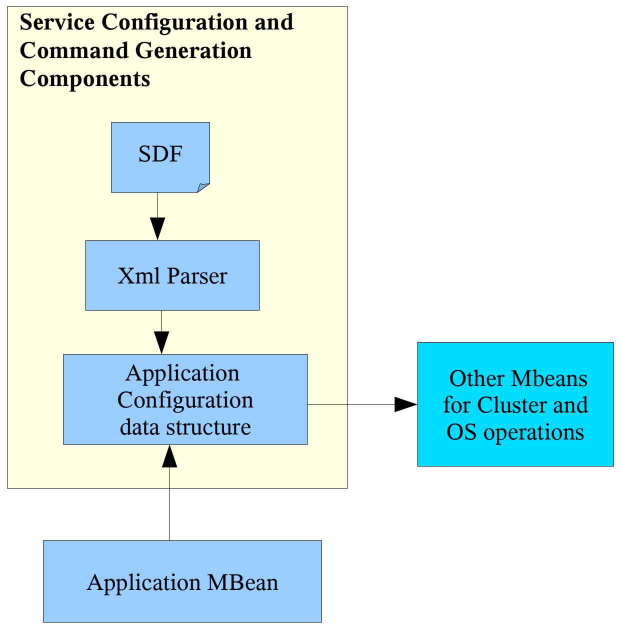

Service Configuration and Command Generation Components
SDF
Xml Parser
Application Configuration data structure
Other Mbeans for Cluster and OS operations
Application MBean

Fig. 3 Command Generation Components

There is a method called "configure", in application information, resource group information, resource information classes. This method generates the necessary cluster and operating system commands for creation, modification and configuration of the individual cluster objects, resource groups and resources. During configuration, the commands are generated using the information present in the instance of each class. The information in the instance of the class is set by the wizard. Once the wizard obtains all the information, the configure method is called on the application information class. This is then cascaded down to invoke the configuration on resource group information class and finally on the resource information class. If configuration fails at any step, the configuration is undone by calling the "rollback" method.

The "configure" method in both resource group information and resource information classes follow the sequence below.

1. If the current object has any other objects that it depends on, call the configure operation on each of

those objects first.
2. Execute the set of commands necessary to create the current object.
3. If object is a resource group, call configure on each of the resources inside the resource group.

Finally, the configure method in the application information class, will perform any configure operation that is specific to the application.

The "rollback" method in the application information, resource group information, resource information classes contains algorithms that will undo any configuration change made to the system. The rollback method is called when there is any failure in the execution of the configure method. Every instance of the resource group class and the resource information class will maintain information about its configuration state during the application configuration. The "rollback" method in both the resource group information and resource information classes follow the sequence below.

1. If there are other objects that depend on the current object, call the rollback operation on each of those objects first.
2. If object is a resource group, call rollback on each of the resources inside the resource group.
3. Execute the set of commands necessary to delete the object.

Finally, the rollback method in the application information class, will perform any undo operation that is specific to the application.

B. Configuration Sequence

The following is a high level sequence of steps for configuring an application. Figure 4 helps is visualizing these steps.

1. The front end application wizard gathers application specific data like : nodes where the application must be configured to run, application specific properties for the resource groups and resources. Most of these values will be presented as a set of possible options, or as valid defaults. The list of possible values and defaults for the application is determined by invoking the application specific discovery algorithms in CHAF. These algorithms are present as part of the application specific library. This information gathered from the user is used to fill the application configuration data structure attributes.
2. A user is allowed to modify any of values presented. If such modifications are made, these values are validated using the application specific validation algorithms. These algorithms are also part of the application specific library.
3. Some applications may need to modify some application specific files on the designated nodes before application configuration starts. For example, in configuring Scalable Apache, the “apachectl” file should be edited on each of the designated nodes depending upon the data gathered from the user. This is done by using the application Mbean on each of the nodes.

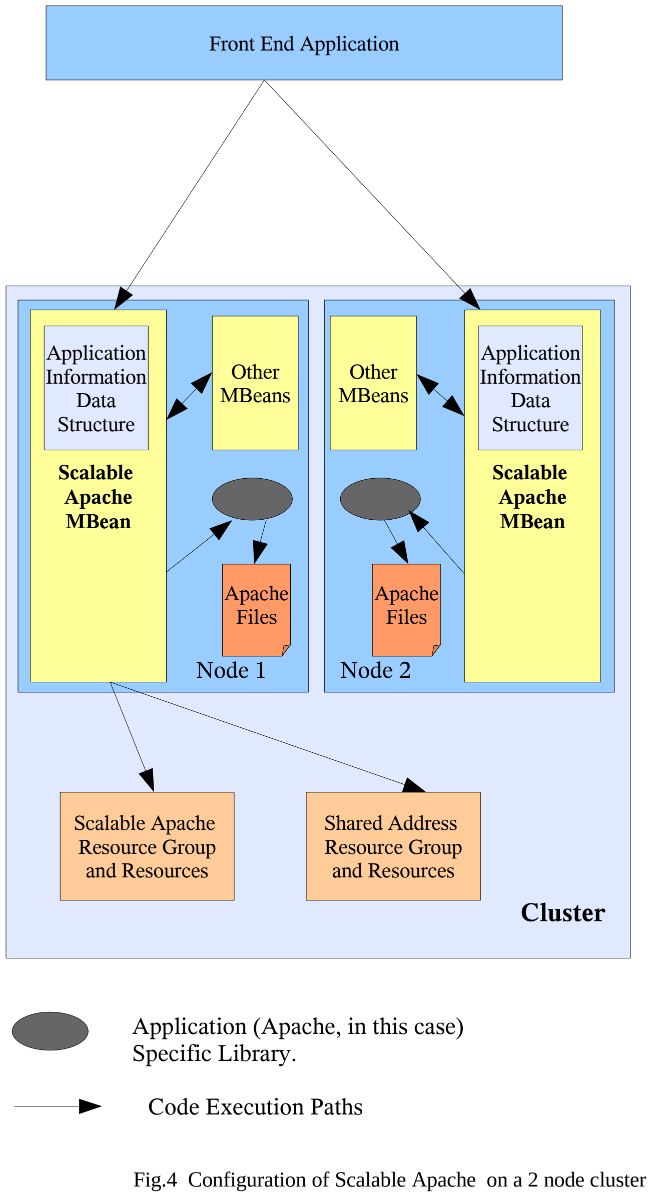

Front End Application
Application Information Data Structure
Other MBeans
Scalable Apache MBean
Apache Files
Node 1
Other MBeans
Application Information Data Structure
Scalable Apache MBean
Apache Files
Node 2
Scalable Apache Resource Group and Resources
Shared Address Resource Group and Resources
Cluster
Application (Apache, in this case) Specific Library.
Code Execution Paths

Fig.4 Configuration of Scalable Apache on a 2 node cluster

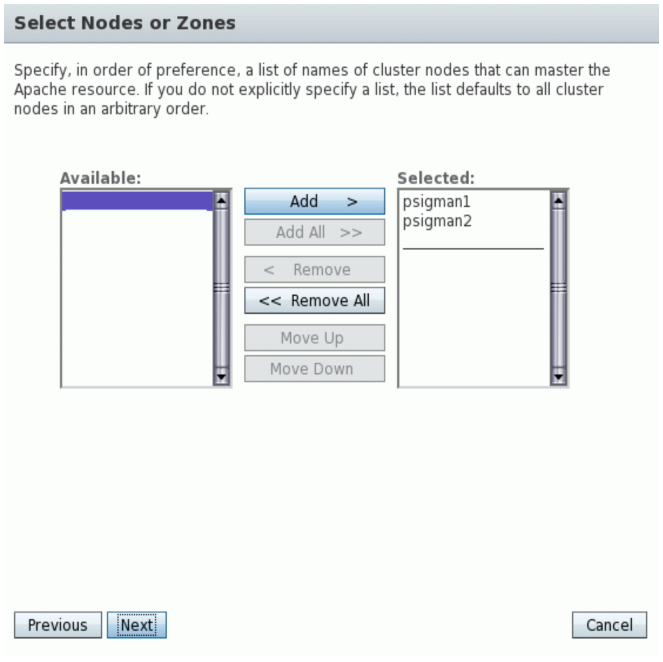
4. The front end application calls the configure operation on the application information class. CHAF configures the application on the cluster by executing appropriate configuration commands.
5. If the configure operations fails for any reason, the CHAF will call the rollback operation on the application information class. This operation will execute commands that will undo the configuration changes made so far.
6. All the configuration commands executed by CHAF are logged and presented to the user by the wizard. This helps users understand the changes performed

on the system.

7. For some applications, the underlying storage that it uses must be online and providing service, before the application itself can be configured in the cluster. For some applications, some application specific files should be modified while configuration is going on. There are resource attributes like, 'bringOnline' and 'customConfigureMap' in the resource information class that will help to solve these cases for any particular resource.

IV. Implementation Screen Snapshots

CHAF described in this paper was implemented and released as part of Solaris Cluster 3.2 [9]. The CHAF wizards were implemented in a browser-based and a command-line based user interface. The figures 5, 6, 7, and 8 are screen snapshots of configuration wizards that configure the Apache application using a browser-based user interface.



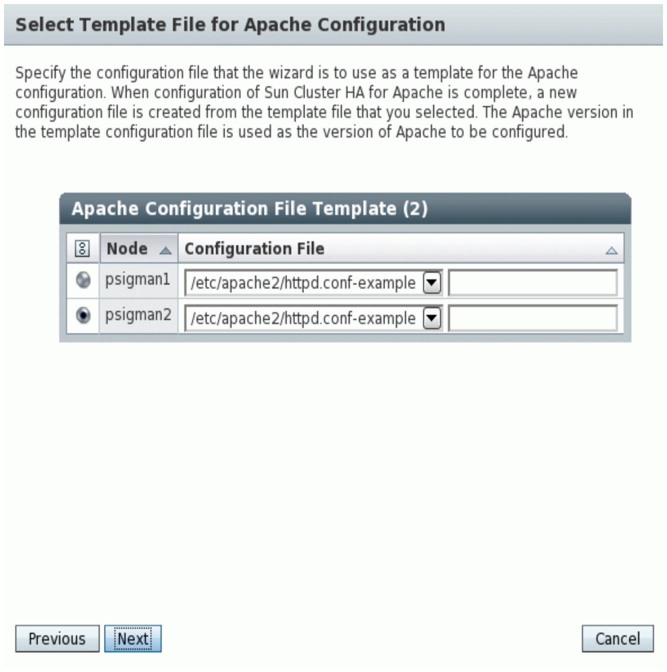
Select Nodes or Zones

Specify, in order of preference, a list of names of cluster nodes that can master the Apache resource. If you do not explicitly specify a list, the list defaults to all cluster nodes in an arbitrary order.

Available:

Selected:
psigman1
psigman2

Fig.5 Node selection screen



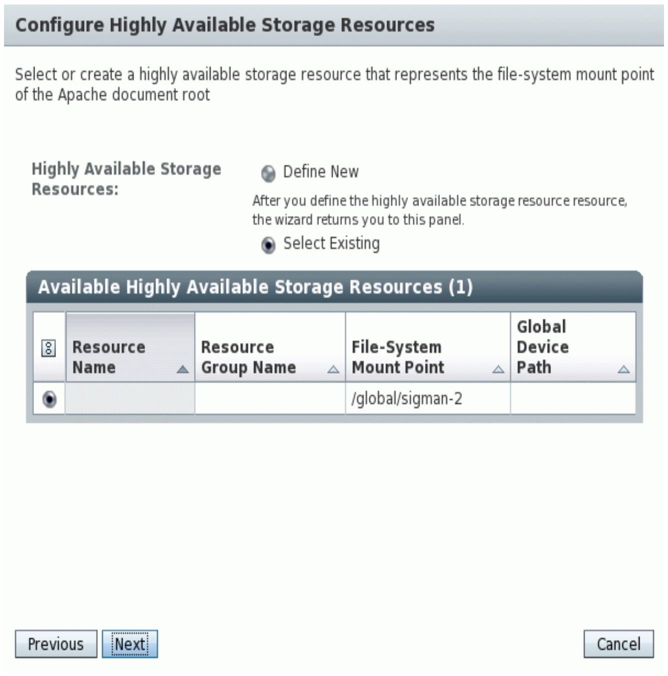
Select Template File for Apache Configuration

Specify the configuration file that the wizard is to use as a template for the Apache configuration. When configuration of Sun Cluster HA for Apache is complete, a new configuration file is created from the template file that you selected. The Apache version in the template configuration file is used as the version of Apache to be configured.

Apache Configuration File Template (2)

	Node	Configuration File
○	psigman1	/etc/apache2/httpd.conf-example
●	psigman2	/etc/apache2/httpd.conf-example

Fig.6 Apache template selection screen



Configure Highly Available Storage Resources

Select or create a highly available storage resource that represents the file-system mount point of the Apache document root

Highly Available Storage Resources: ○ Define New

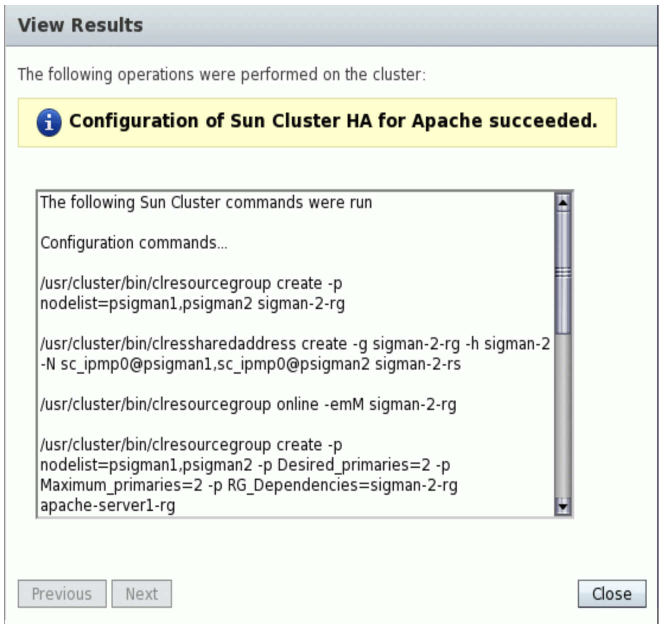
After you define the highly available storage resource resource, the wizard returns you to this panel.

● Select Existing

Available Highly Available Storage Resources (1)

	Resource Name	Resource Group Name	File-System Mount Point	Global Device Path
●			/global/sigman-2	

Fig.7 Storage selection screen



The following operations were performed on the cluster:

Configuration of Sun Cluster HA for Apache succeeded.

```
The following Sun Cluster commands were run

Configuration commands...

/usr/cluster/bin/clresourcegroup create -p
nodelist=psigman1,psigman2 sigman-2-rg

/usr/cluster/bin/clressharedaddress create -g sigman-2-rg -h sigman-2
-N sc_ipmp0@psigman1,sc_ipmp0@psigman2 sigman-2-rs

/usr/cluster/bin/clresourcegroup online -emM sigman-2-rg

/usr/cluster/bin/clresourcegroup create -p
nodelist=psigman1,psigman2 -p Desired_primaries=2 -p
Maximum_primaries=2 -p RG_Dependencies=sigman-2-rg
apache-server1-rg
```

Fig.8 Final results screen

V. Conclusion

In this paper we propose a JMX based framework, called CHAF, for configuring applications in a highly available clustering environment. We showed how CHAF is extensible to many different applications. It offers the ability to do application specific configuration on the cluster nodes, and the ability to undo configuration changes on failure. CHAF enables a user with knowledge about the application, configure the application for HA easily. CHAF is implemented in the Solaris Cluster 3.2 product.

References

[1] D. Patterson et al, "Recovery Oriented Computing (ROC): Motivation, Definition, Techniques and Case Studies", UC Berkeley Computer Science Technical Report, UCB/CSD-02-1175, Mach 15, 2002.
[2] "Statistical Summary of Commercial Airplane Accidents: Worldwide Operations, 1959-2001", published by Airplane Safety, Boeing Commercial Airplane, June 2002.
[3] Ira Pramanick, “Modeling Sun Cluster Availability,” *Proc. SUPerG-2002, 2002.*
[4] Alan Robertson, "An Open Framework for Clustering", *opencf.org/documents/HAFramework.html*
[5] Sung, Hocheol; Choi, Byounguk; Kim, Heemin; Song, Jungwook; Han, Sunyoung; Ang, Chee-Wei; Cheng, Wang-Cho; Wong, Kim-Sing, "Dynamic Clustering Model for High Service Availability" in *Autonomous Decentralized Systems, 2007, ISADS '07. Eighth International Symposium*
[6] Jiun-Kai Wang; Jeen-Shing Wang, "A self-regulating clustering algorithm for identification of minimal cluster configuration", *Neural Networks, 2004, Proceedings. 2004 IEEE International Joint Conference*
[7] Veritas Cluster Server, www.symantec.com
[8] Cluster Systems Management, www.ibm.com
[9] Solaris Cluster 3.2, www.sun.com
[10] Java Management Extensions Specification, openjdk.java.net/groups/jmx/