ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 3, March 2014

Diagnostic Bundle Consolidation

Patel Rushi J[1], Smitha G. R[2]

Student, Department Of Information Science, R.V. Collage of Engineering, Bangalore, India[1]

Assistant professor, Department Of Information Science, R.V. Collage of Engineering, Bangalore, India[2]

Abstract: Failure diagnosis is finding the correct cause of failure, and gives the enough information to correct it. And it is one of the major challenges that home users and software administrators face today in the distributed software. The problem is more because there are so many different components which collaborate to realize a particular service and these components belong to different functional domains as well as physical locations. With increasing number of such services, it is important to design systems which enable easy diagnosis of problems encountered and allow determining the root cause of the failures. Diagnostic bundle consolidation mechanism provides a single point of management for creation and collection of the diagnostic bundle within the Snap Creator Framework. The log files, command outputs, etc. are scattered around the Snap Creator agent. It can consolidate diagnostic files based on various scope parameters as needed to meet effective troubleshooting requirements. This diagnostic bundle consolidation work is based on java REST architectural style, which provides light weighted communication between server and agent.

Keyword: Snap Creator Framework (SCF), Snap Creator (SC), REST, Restful.

I. INTRODUCTION

Snap Creator framework is a product used for data protection (backup, restore, and clone) of application and virtualization environment consuming remote storage. It is a distributed ecosystem involving server and multiple agent hosts running various components related to their respective application/virtualization domains. It is OS-independent. Snap Creator Framework integrates NetApp data protection with a broad range of third-party applications. Snap Creator plug-ins integrates different features with third-party applications, operating systems, and databases. Snap Creator also accommodates custom plug-ins and has an active developer community. The Snap Creator Framework provides:

- **Application-consistent data protection.** Users get a centralized solution for backing up critical information, integrating with existing application architectures to assure data consistency and reduce operating costs.
- **Extensibility.** Achieve fast integration using Storage modular architecture and policy-based automation.
- **Cloud readiness.** OS-independent Snap Creator functionality supports physical and virtual platforms and interoperates with IT-as-a-service and cloud environments.

The current troubleshooting model used in Snap Creator has limitations in its capability, where collecting diagnostic information is not exhaustive and flexible. The troubleshooting capability needs to be extended to enable the following:

- Exhaustive agent side diagnostic information collection
- Diagnostic information collection based on various scopes like Snap Creator profile, Snap Creator configuration, Snap Creator plugin, Snap Creator workflow etc.
- Driving the diagnostic bundle creation and collection from the centralized Snap Creator server, and pulling the diagnostic bundle to the server.

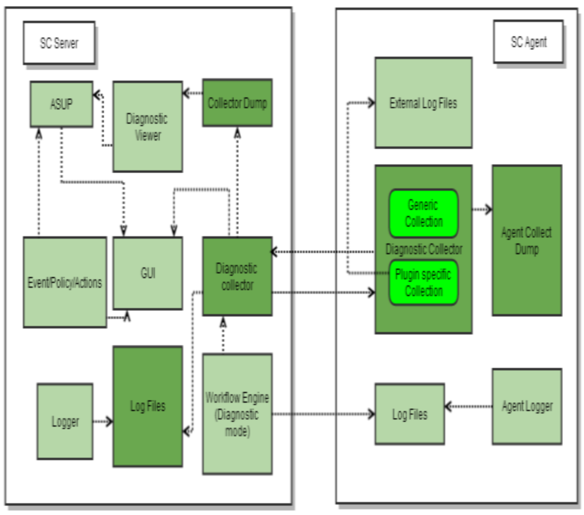

Figure 1: Working architecture diagram of diagnostic bundle consolidation.

This paper have included an enhanced way of collecting diagnostic information spread across multiple components on multiple hosts, for more effective troubleshooting in the Snap Creator environment. The agent implements 2 REST APIs one to create the diagnostic bundle, second to pull the bundle from the agent to the server. Log files and other diagnostic data can be filtered based on various scope parameters specified as part of the REST API.

The agent implementation of diagnostic bundle creation consists of 2 parts one a generic module which consolidated generic diagnostic data, second a plugin module where a Snap Creator plugin can implement diagnostic data collection specific to the agent calls the generic as well as plugin specific collection logic based on the scope parameters passed to it, and consolidates it to form a single archive. The entire process is driven from the SC server which invokes the agent REST APIs to consolidate and download the bundle from the agent to the server.

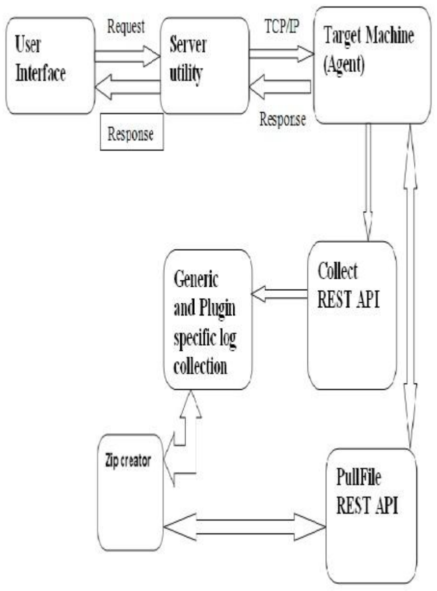

Figure 2: Block diagram for diagnostic consolidation bundle.

II. STATEMENT OF THE PROJECT PROBLEM

Despite spite the fact that manufacturers of the distributed architecture software put great strengths only in its perfection and increase of work reliability periodically, nevertheless, there arise refusals of the software component at remote location. The diagnostic process of this distributed component is complex system. Qualified experts have to go to the remote locations where, the software components are installed and correct it. However, elimination of the found error frequently is limited only to simple actions, (for example, replacement of the failed component) not demanding high qualification. As a rule, for detection and elimination of malfunctions the consumer should address the manufacturer of the software, It frequently causes significant transport expenses. If the software is supplied with system allowing the expert to have remote diagnostics it remote, giving the consumer the recommendation on elimination of the found errors, above it is possible to avoid transport expenses in many cases. All above shows, that development of such system is necessary and justified.

III. PRACTICAL REALIZATION

To make the troubleshooting mechanism easy, diagnostic should be done on the server side based on diagnostic collection method which includes the vital information regarding the failure in the snap creator operations. The diagnostic information's are collected from the agent side by invoking the corresponding APIs.

Initially when the error occur in the snap creator framework, users invoke the diagnostic consolidation method from the server side, In that it will collect local diagnostic information on the server side and invoke createDump API on the agent side, this API include scope parameter for limiting the diagnostic information. This parameter can be profile name, configuration name, workflow ID, and plugin name. The agent side the collection is done in two parts. One the agent does the generic diagnostic collection. After that it will invoke external plugin specific diagnostic collection with the scope parameter, the plugin do diagnostic collection based on the scope defined and send the generated diagnostic information to the agent. The agent will combine both generic diagnostic information and plugin specific diagnostic information in to the archive file and store it to temporary location and send that location information to the server. Then server invokes pullfile API to download archive file on the agent side to the server side, and it will combine both the diagnostic information and give it to troubleshooter.

IV. THE RESTFUL WEB SERVICES

REST is not a protocol; it's a style of the Web architecture; abstraction and description to the design principle of Web architecture. In other words, Web is the instance of the REST system. REST described how to design and develop distributed system.

The Goals and Design Principle of REST

The goals [1] of REST are the followings:

1. Scalability of component interactions
2. Generality of interfaces
3. Independent deployment of components
4. Intermediary components to reduce
5. Intermediary components of reducing interaction latency, enforcing security, and encapsulating legacy systems

The components in the REST system must comply with the following constraint [1]:

1. Identification of resources
2. Manipulation of resources through representations
3. Self-descriptive messages
4. Hypermedia as the engine of application state

The Idea of REST

In the REST system, all the resources had the URI. Using the GET, POST, DELETE and PUT as the general interface of the resources, user visited the resources via them. Developers must consider [4] each method's expected semantics to decide which methods are suitable for each resource. GET, PUT, and DELETE, for example, must be idempotent, and GET must be safe for clients to call repeatedly because all it does is return a representation of a resource. The PUT method lets a client replace a resource state with a new state, whereas clients use DELETE to remove resources. Both obviously have side effects, but both are idempotent because calling them repeatedly has the same effect as calling them once. POST can be made to perform virtually any action, but in RESTful systems, it's normally used to create or extend resources, and so it isn't expected to be idempotent or free of side effects.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

Not everyone agrees [4] that REST is easy, of course. One frequently mentioned issue is a lack of tools specifically; those that fit within the interactive development environments (IDEs) that many enterprise developers use to help them write and maintain their code. Given that IDEs are helpful only because they automate activities and approaches that developers have already manually proven to be worth automating, this "lack of tooling" argument is somewhat off the mark. With the right language-specific patterns and idioms to follow, existing IDEs work just fine for RESTful service development.

V. RESULT

The diagnostic capability in Snap Creator shall have the capability to be driven from multiple contexts. The diagnostic component in Snap Creator shall function based on the "scope" in which it is operating on, which shall be determined from the context. Following are the scopes in which diagnostic collection can be done:

1. Global scope - Typically run by the SC admin user from a diagnostic page in SC GUI, multiple profiles/configs are specified for collecting diagnostic information, and it will collect all the information available on SC agent and SC server.
2. Profile scope - SC user specifies SC profile(s) for collecting diagnostic information based on the profile name specified.
3. Config scope - SC user specifies SC config(s) for collecting diagnostic information based on the config name specified.
4. Workflow/Operation scope - SC user specifies SC workflow/Operation scope collecting diagnostic information based on the workflow/Operation scope.

Table 1: Diagnostic Bundle Consolidation Result.

S. No.	Scope For Filtering	Number Of Files Present	Number Of Files Collected	Time Taken For This operation(in ms)
1	Global	100	100	7450
2	Config_Name	100	78	6859
3	Plugin	100	54	6319
4	Operations	100	10	3829
5	Config_Name+Plugin+Operations	100	2	3530

Here, if user doesn't specify any scope then it is global scope, and this mechanism will collect all the data from the entire component. And as the data will be in huge amount the time takes to do this operation is more. If config_name is provided as one of the scope parameter then also the data file collected will be in huge numbers and time taken is slightly lesser then the global scope. As if user specify all the scope parameter then the data collected will be only limited to that scope only and the time taken to collect it will be very less, and troubleshooter can easily troubleshoot the error.

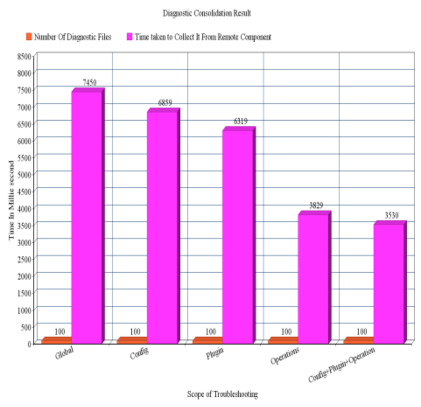

Figure 4: Diagnostic Result

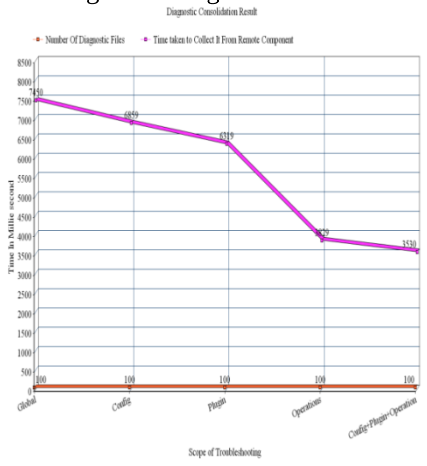

Figure 5: Diagnostic Result

VI. CONCLUSION

In general distributed software diagnostic can be done by subject being co-located with the person or system doing the diagnostics, instead of doing so with remote diagnostics the subjects can be separated by physical distance and important information is exchanged either through wire or wireless. This approach can be used to improve reliability of vital or capital-intensive installations and reduce the maintenance costs by avoiding unplanned maintenance, by monitoring the condition of the system remotely. This can be even making simpler by using scope field to filter out unwanted diagnostic data.

REFERENCES

[1] Mohsen Rouached and Hassen Sallay, RESTful Web Services for High Speed Intrusion Detection Systems, 2013 IEEE 20th International Conference on Web Services
[2] Hongjun Li, RESTful Web Service Frameworks in Java, 2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)
[3] P. Cesare and W. Erik. Restful web services: principles, patterns, emerging technologies. In Proceedings of the 19th international conference on World wide web, WWW '10, pages 1359–1360, New York, NY, USA, 2010. ACM.
[4] Koizumi, N, Construction Methodology for a Remote Ultrasound Diagnostic System, 2009 Robotics, IEEE Transactions on (Volume:25 , Issue: 3)
[5] J. Sandoval, Restful Java Web Services . Packt Publishing, 2009.
[6] L. Richardson and S. Ruby, RESTful Web Services .O'Reilly Media, 2007.
[7] S.Vinoski, RESTful Web Services Development Checklist.
[8] D.Brown,C.M.Davis and S.Stanlick, Struts 2 in Action. Manning,2008