Challenges in SMI-S Provider Development

16th September, 2009

Girija Kumar Raghava Kasinadhuni

girija.kasinadhuni@wipro.com

&

Govindan Nampoothiri

govindan.nampoothiriv@wipro.com

Wipro Technologies

Abstract

- This presentation describes the process of SMI-S proxy provider development for a performance sensitive multi-device environment.
- It highlights the decision process in choosing between proxy provider and embedded provider, development tools and choosing a suitable design while developing the provider.
- The presentation goes on to explain approaches to improve provider response time by having an efficient approach for caching device details, multi-threading support etc.
- The presentation concludes by throwing light on resolving issues in a multi-vendor scenario during provider development specifically around triage and defect turnaround times.

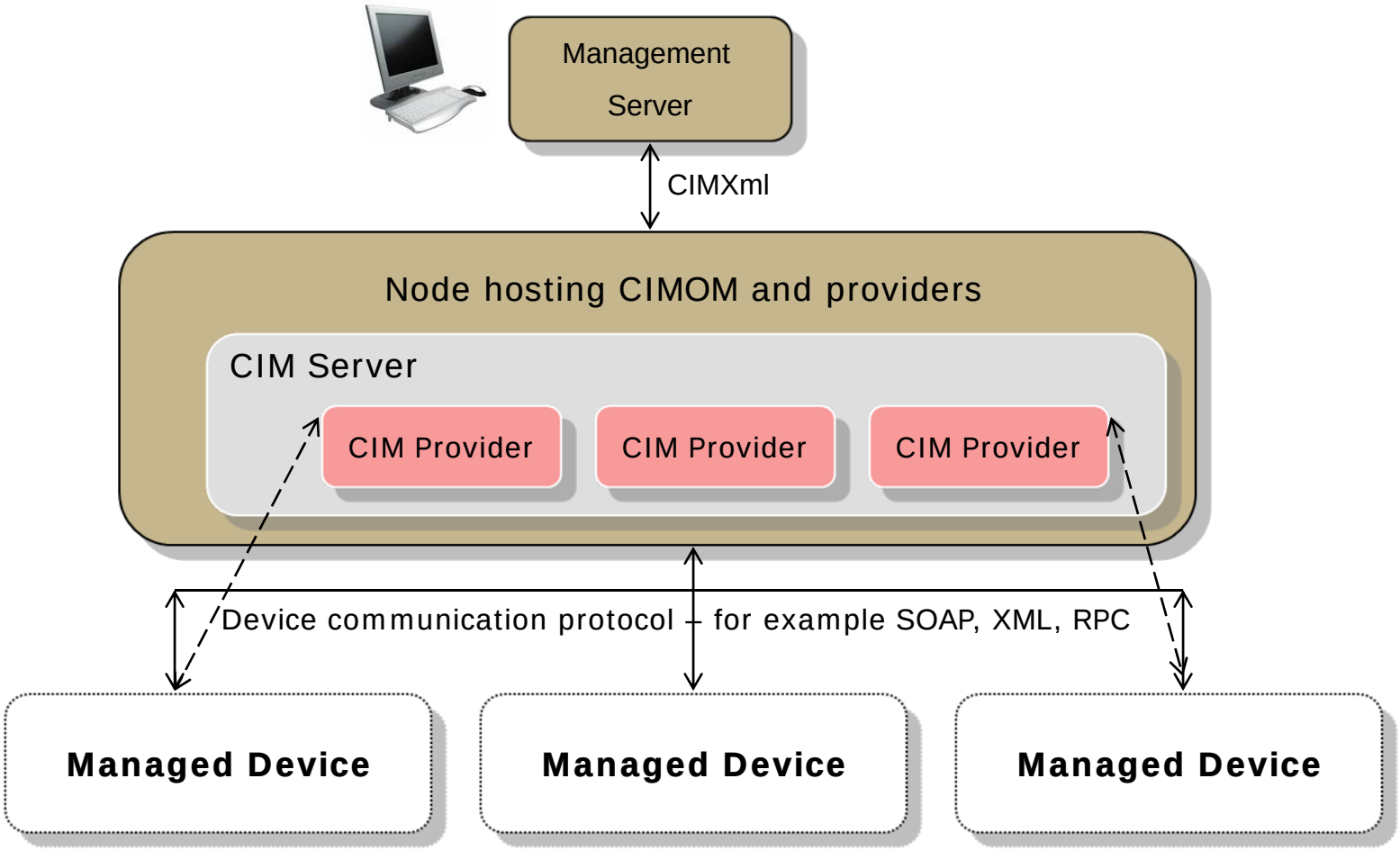
Learning Objectives

- Understanding the process of SMI-S Provider development
- Understanding the design decisions to be made and the challenges associated with it during Provider development
- Understanding how to maximize the provider performance
- Understanding debugging techniques to be used

Contents

- Introduction to SMI-S Architecture
- Design Considerations for
  - Embedded Providers & Proxy Providers
- Development Challenges
- Performance Improvement Techniques
  - Caching Mechanism, Multithreading, Association Traversal
- Triage and Debugging Challenges.
- Testing Challenges.

Introduction to SMI-S Architecture



Storage arrays, RAID Controllers etc. (Managed Devices)

Requirements for Provider development

- Device to be managed
- Operating Environment
- Footprint of the provider and the cimom being used
- Profiles, packages and sub-profiles to be implemented
- Vendor specific methods, attributes, properties and values that need to be supported

Types of Providers

- Embedded
- Proxy

Embedded Provider Environment

CIMXml

Management Server

Third Party CIM Client

Embedded Software Stack

CIM Server

CIM Provider

API Interface For Managed Device

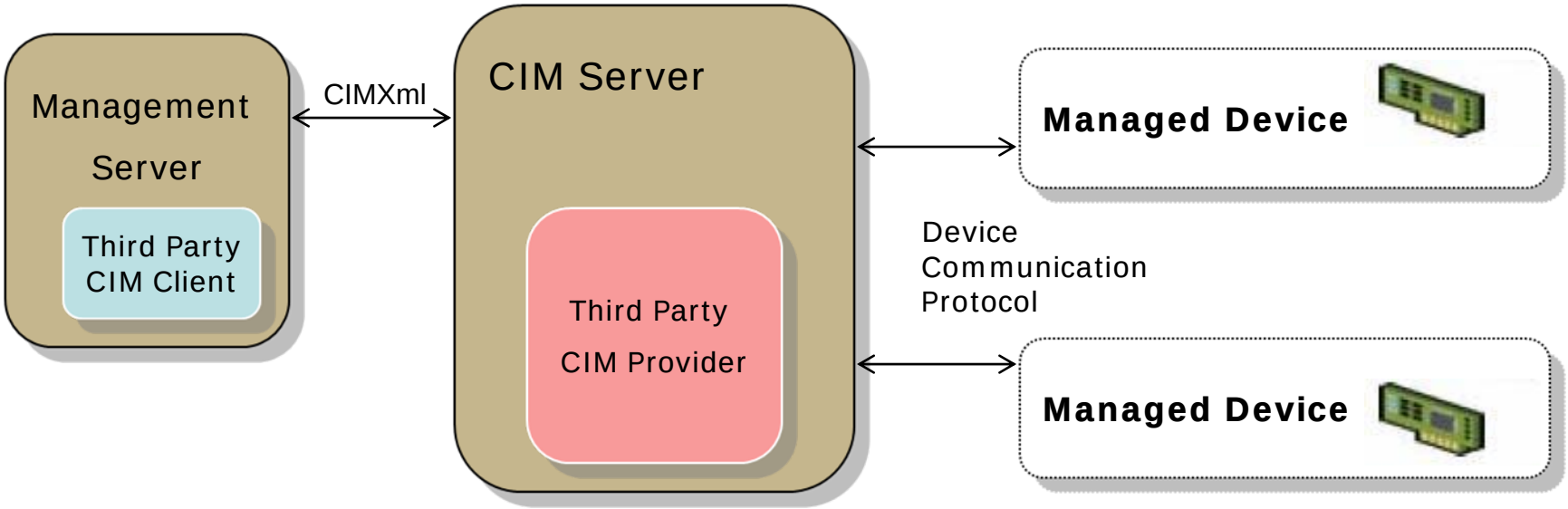
Device Driver For Managed Device

Managed Device

Design Considerations: Embedded Providers

- Embedded Providers manage the device they are present on - PCI RAID cards, HBA, flash devices etc .
- Smaller footprint.
- Scenarios where entire software including the provider has to be packaged together and installed as a whole. For e.g.. ESXi servers of VMWare, Flash devices which need to have all the software embedded on them.
- With embedded providers, the issue of scalability vanishes
- Architecture / system resource limitations. Memory space is a limitation

Proxy Provider Environment



Design Considerations: Proxy Providers

- Proxy providers give us the advantage of managing multiple devices within a network.
- Disk space/System Resources not a constraint
- Can be installed on any machine in the network and manage the devices in that network. Eg. Enterprise servers using disk drives which use external RAID controllers
- Scalability is a big challenge as proxy providers typically manage more than one device. Provider design must be robust to seamlessly manage all the devices
- Design should handle delays due to network latency

Development Challenges

- To make providers, CIMOM agnostic we use CMPI interfaces
- Developing providers using native CMPI requires expertise in CMPI
- CIMOM might not support CMPI – WMI
- For C++ providers, we need to consider class hierarchies and design classed based on MOF files
- Using native CMPI interfaces is time consuming

Development Tools

- Using open source development tools helps us overcome earlier mentioned issues - Cimple, Konkrete etc.
- Usage of open source tools make the abstraction for CMPI very simple.
- All the developer needs to concentrate on is the design and working of the internals of the provider (threading, caching and persistency of data structures), and the interface to the proprietary solution.
- Consideration of cases like no support for CMPI in WMI (Windows CIMOM) would be taken care of by a development tool which would have an adapter to WMI.
- Development time is reduced considerably

Development Tools continued…

- Provides stubs to implement several of the provider WBEM operations like enumeration of instances, getting instances, association traversal etc.
- Generates real classes in the target language from MOF files
- Generates the provider skeleton and CIM interface automatically
- More expertise required for native CMPI

Performance Improvement Techniques

- Improving command response time - Caching Mechanism.
- To Handle Multi Device Scenario - Threading.
- Association Traversal – device specific.

Performance Improvement Techniques continued..

Caching Mechanism

- Fetching information every time from the device will be time consuming. Hence cache the information.
- Improves the response time significantly.
- Need to have mechanism to update the cache when required.
- Need to limit the cache size

Performance Improvement Techniques continued..

Synchronous Caching Mechanism …

- Cache updation can be time based or event based
- In time based cache updation provider polls the managed device on predefined time intervals and updates the cache.
- Time based caching has the disadvantage that there will be delay between the configuration changes and the actual cache updation.

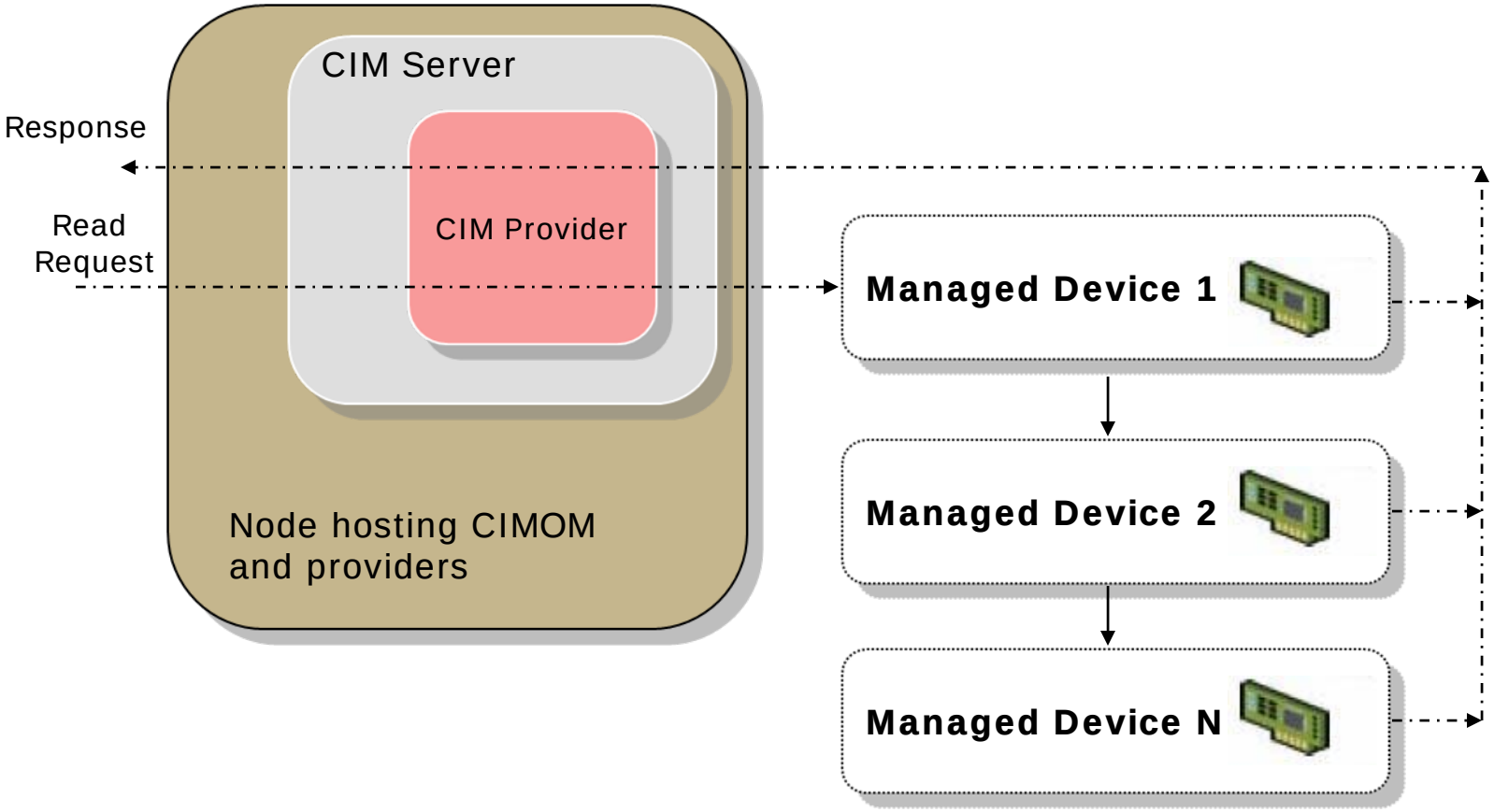
Performance Improvement Techniques continued..

Asynchronous Caching Mechanism …

- Incase of an event based cache updation, the periodic polling of the managed device is not required.
- Provider will wait for events that are triggered when there is any change in the managed device's state.
- Event – based mechanism helps the cache to remain updated asynchronously

Performance Improvement Techniques continued..

Single Thread Scenario



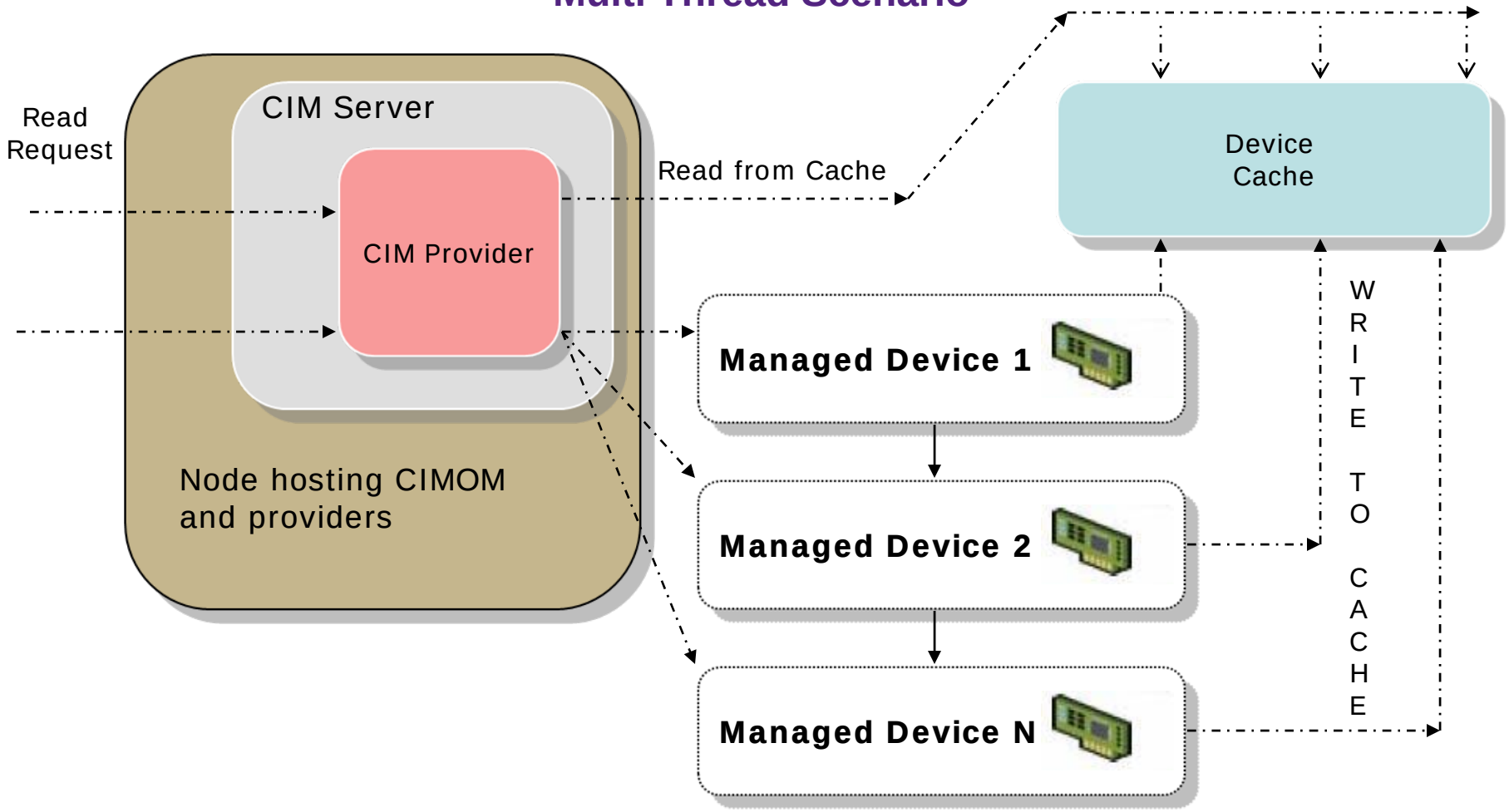
Performance Improvement Techniques continued..

Multithreading

- When there are multiple devices to be handled and with only one thread, retrieving device information becomes slower as it will happen sequentially. Hence it will be more time consuming.
- Having multithreading framework can resolve this bottleneck.
- Separate threads can handle each device instead of a single thread keeping the request to each device in the queue. This will speed up the provider response time.

Performance Improvement Techniques continued..

Multi Thread Scenario



Performance Improvement Techniques continued..

Association Traversal

- CIM model works based on the various profiles and the relationships between them. Association traversal is used to traverse the class relationships and to obtain the requested information.
- Associations are the most common and important WBEM operations.
- Cimserver response time largely depend on the way the association traversal has been implemented.

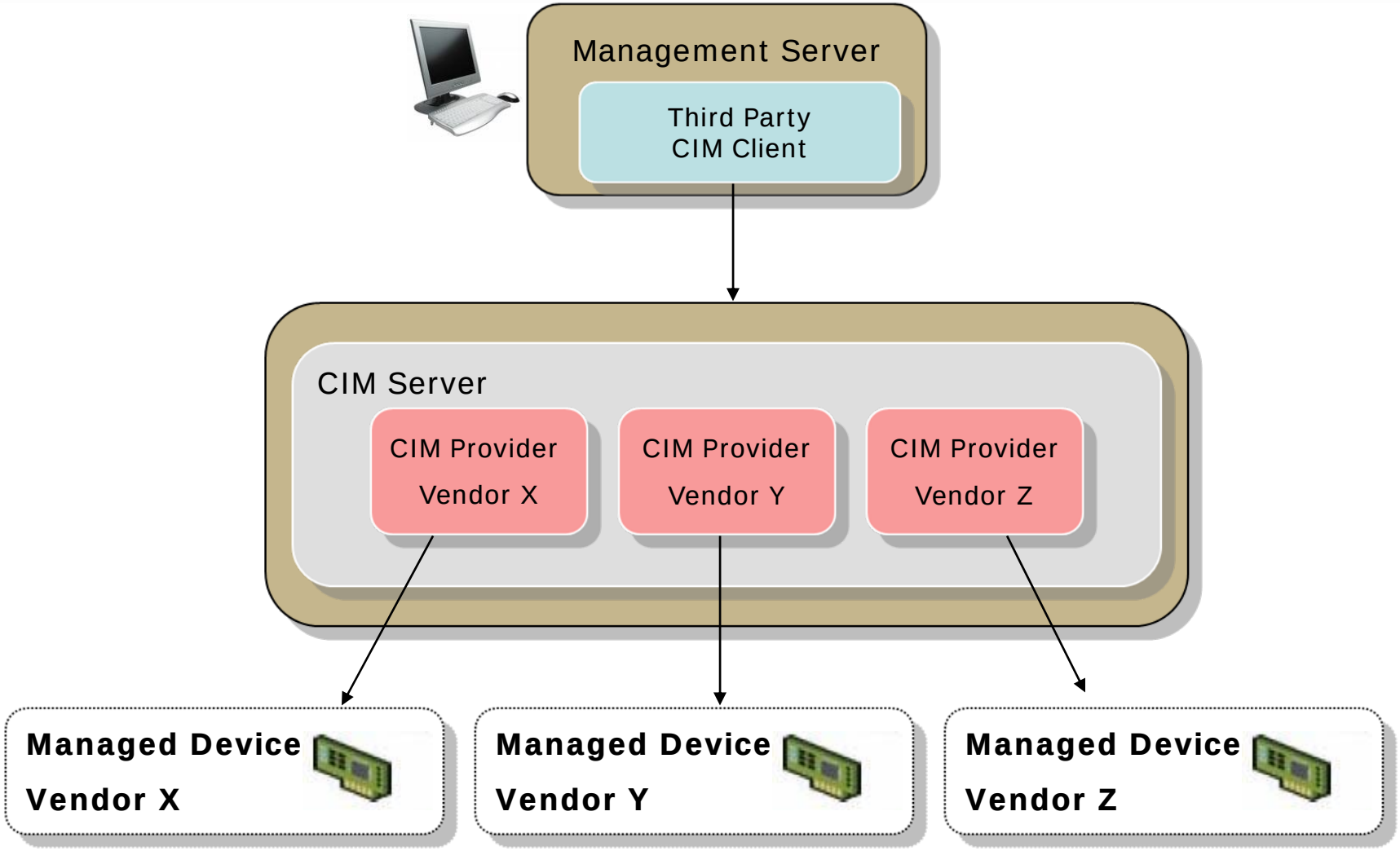
Performance Improvement Techniques continued..

Association Traversal ….

- Intelligently routing the association traversal only for the required device instead of traversing through all the classes will save time and improve the performance .

- Add a qualifier to the association class that we want to traverse, using the possible shortest path. This algorithm of shortest path traversal could be applied to all those classes whose qualifier is set.

Triage and Debugging of Providers in a Multi-Vendor Scenario



Multi Provider Environment

Triage and Debugging of Providers in a Multi-Vendor Scenario ...

- In a real-world provider development scenario, the provider is normally tested along with the SMI Agent (cimom) that manages Agent Extensions (providers) from different vendors simultaneously.

- Triaging defects becomes a challenge in a multi-vendor, multiple provider scenario. There is a lot of scope for misunderstanding that could result in faulty triaging.

Sample Scenarios

- Cimserver crash.
- Memory leaks.

Sample Scenarios – Cimserver crash.

- Provider from a vendor ‘X’ quietly causes the cimserver to crash while testing a provider from vendor ‘Y’.

- This can happen as the threads that support indications can keep running in infinite loops which causes all the providers managed by the cimom to be running though no specific queries to them are made.

Sample Scenarios – Memory Leaks.

- Another case is triaging memory leaks.
- A memory leak caused by the SMI Agent could be misinterpreted as that being caused by an Agent Extension.
- This could happen because the memory leak causing portion of code is triggered only when a particular Agent Extension is loaded by the Agent.

Sample Scenarios …

Best Practices

- Carry out individual testing of providers with the cimom, and then, carry out the integration testing with all the providers for all the managed devices
- Good to test the cimom standalone before testing with all the providers that are to be loaded

Testing Challenges

Adoption of a testing framework

- Manual testing of provider can be very tedious and error prone.
- Hence it is good to automate the test framework to validate the provider. This will be helpful during regression testing as well.
- Reduces the burden of testing every class and ease of validating the attributes of every class.

Testing Challenges …

Adoption of a testing framework

- The test framework can query for
  - Enumerate instance
  - Get Instances
  - Invoke methods
  - Association traversals

- The values mentioned in the mofs for the respective classes be matched with those obtained from the provider.

  Ex. The expected Operational Status from a provider may be in a range which is not compatible with the mof.

References

- http://www.dmtf.org/standards/wbem/
- http://www.snia.org/forums/smi/tech_programs/smis_home

Acknowledgements

We gratefully acknowledge and are thankful to the inputs of following team members:

Vivekananda Kar, Suyog Tiwari,
Naveen Narayanamurthy, Michael Franco,
Biswamohan Mohapatra & Vipul Srivastava.

We are thankful for the reviewers:
Padmavathy Madusudanan & Govindan Thiruvengada

Q & A

Thank You !!!