Hewlett Packard Enterprise

HPE Server Automation 10.50

Linux Patch Management Performance Characterization

White Paper

Contents

Summary of performance results

HPE tested the Linux Patch Management functionality of Server Automation version 10.50 in the HPE Performance lab. The aim was to validate the overall throughput and resource demand for a well-defined workload.

This study analyzes the new SA Linux Patching code using the Dynamic Patch Policy in comparison with the legacy Software Policy approach. The detailed comparison of the Software Policy vs. Dynamic Patch Policy performance and throughput is presented here, together with the slice scalability performance monitored for both scenarios.

SA Red Hat patching uses the Linux Yum utility to install patch installation on managed servers. Starting with SA 10.50, Red Hat Dynamic Patch Policies are supported. These policies use much of the already existing Ubuntu Dynamic Patch Policy concepts and are based on the results returned by the Yum update that is run against the SA Software Repository.

While the end result does not change, the main enhancement over Software Policies is that the policy is dynamically populated at runtime only with packages applicable to each managed server. This reduces the amount of data processed by the SA deployment. All policy items are then downloaded and installed during the remediation job, and patch compliance is based on the recommended patches for a server.

This new functionality was characterized in the Performance Lab to quantify the improvements provided by the Dynamic Patch Policy capability for Red Hat managed servers. For a given use case, the operational and performance characteristics were measured for both Dynamic Patch Policy remediation and Software Policy remediation code paths using the same software repository and target managed server patch level.

To point out workload distribution across SA slices, the HPE Performance team executed the two patching scenarios on two different SA Core configurations. The first configuration used a single SA Core slice and the other configuration used two SA Core slices. For more information about the SA Core layout, see **Appendix: Test system configuration**.

Test case description

The Performance team selected the following test case based on typical customer usage of the feature **Patch RHEL servers with a monthly erratum set including its dependencies.**

The implemented RHEL Patch management test case:

- concurrently patched up to 1000 RHEL managed servers with the erratum set published in the first month after the General Availability release of the RHEL minor version.
- used unpatched RHEL 6.7 64-bit virtual machine for each managed server.
- used managed server virtual machines evenly distributed across 28 ESXi hosts.
- used a managed server device group with a single policy attached, either a Software Policy or a Dynamic Patch Policy.
  - both policies had an associated *repo.restrict custom* attribute, limiting access to RPM packages used for the purpose of this test.
  - the payload consisted of 29 errata containing 128 RPM patch packages, out of which 32 (157 MB in size) were updated when running the job.

The test case submitted the job via the Twist UAPI through Pytwist.

For more information about the hardware and software configuration of the environment under test, see **Appendix: Test system configuration**.

Performance results

The job performance was measured for a two-slice configuration of a single SA Core, using the two available use cases: Software Policy remediation and Dynamic Patch Policy remediation.

Job throughput is measured in number of managed servers processed per minute. Throughput is computed by dividing the number of targets by the time required to complete the job.

The overall job throughput represents an average of three iterations for each load level: 1, 100, 200, 500 and 1000 managed servers.

Overall job throughput

Under the same test conditions, no performance regressions are observed in the 10.50 release. SA code throughput and stability measurements for Software Policy and Dynamic Patch Policy remediation on Linux servers are comparable, with very close values. The minor differences observed can be attributed to environment variations in between SA jobs.

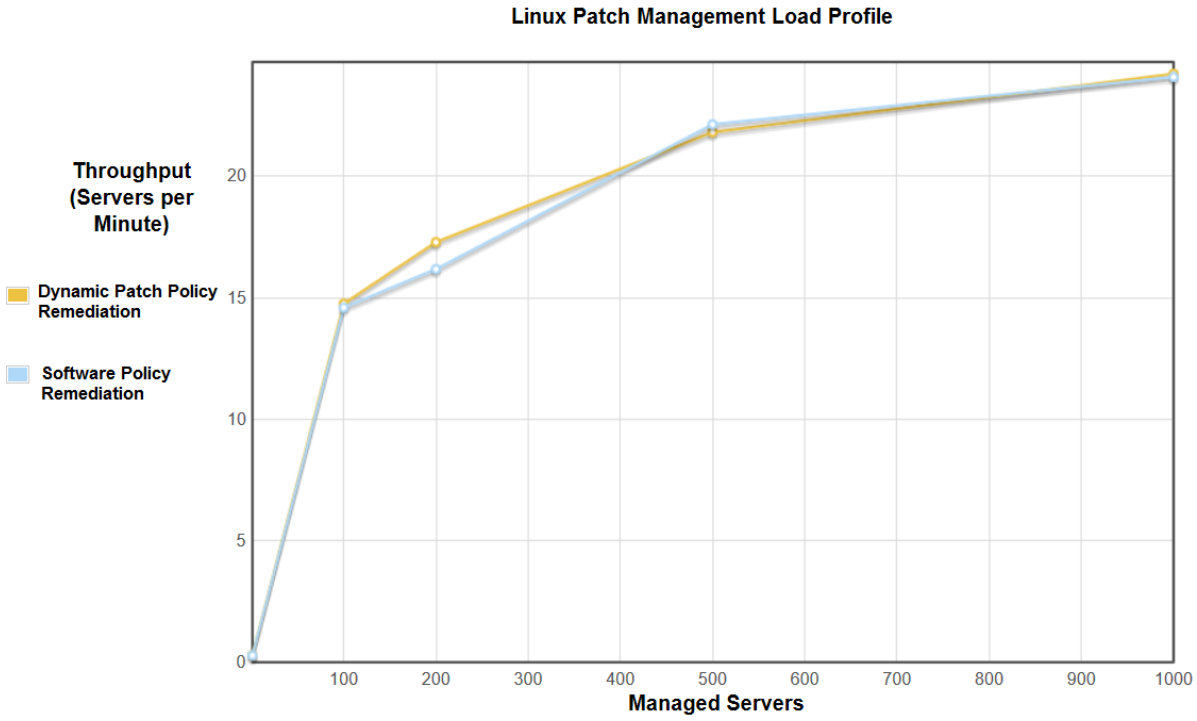

***Figure 1**: Linux Patch Remediation Job Throughput (Managed Servers per minute)*

Workflow characterization for Dynamic Patch Policy vs. Software Policy remediation

The following plots illustrate the similarity between SA workflow stages (SA wayscript thread stages) for Dynamic Patch Policy vs. Software Policy Remediation code paths. Test conditions are kept the same: 32 applicable policy items and 100 managed servers.

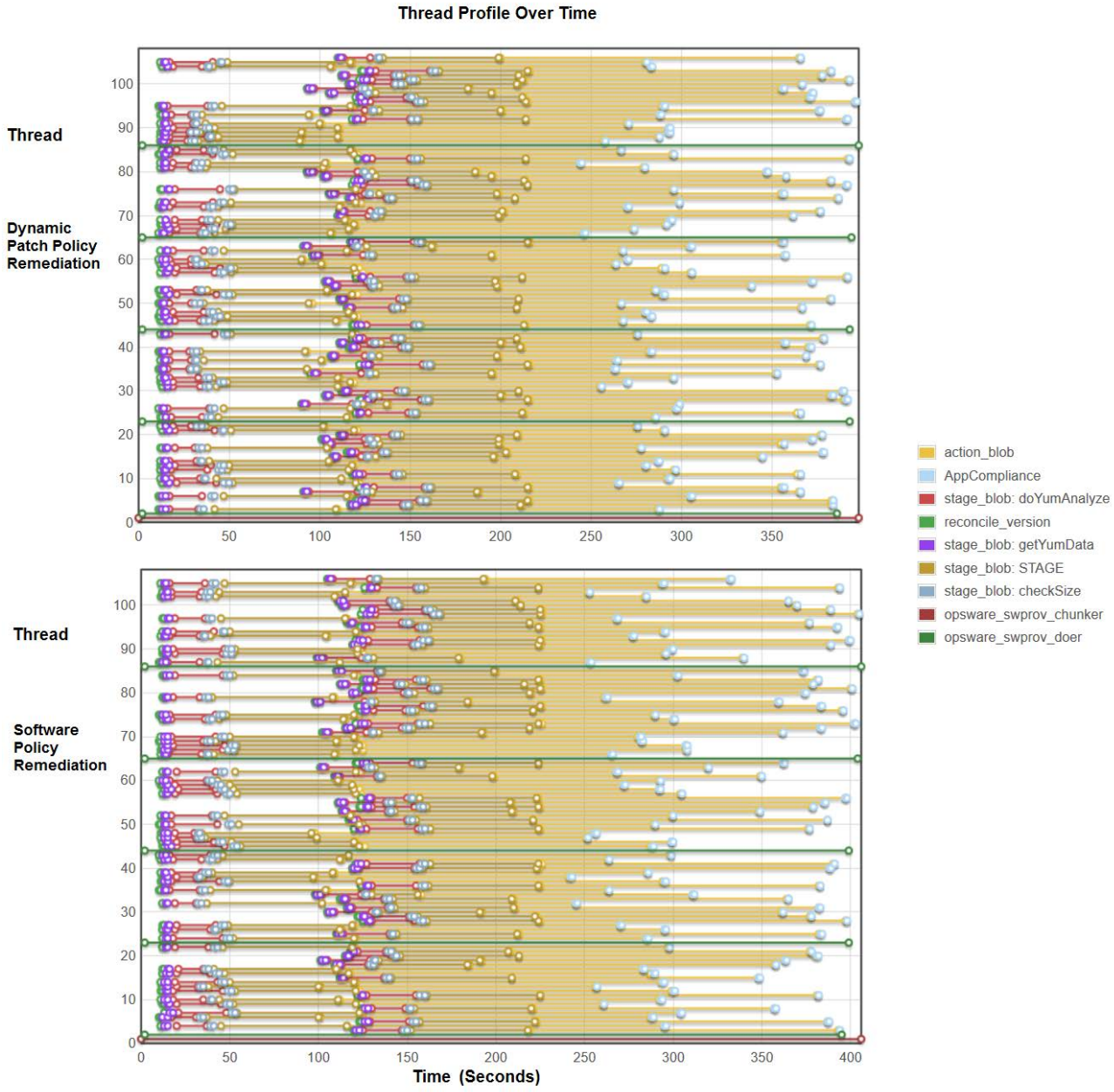

***Figure 2**: Concurrent patching operations against 100 managed servers*

A Red Hat patch management job consists of a single “chunker” session, which spawns one or more “doer” sessions. Each doer operates on one or more managed servers. Several tunable parameters modify how the chunker session distributes and limits the job across these doer sessions and the SA Core as a whole.
In Figure 2 above, five doers operate on 20 managed servers each.
Job sessions with more managed servers would have more doer sessions to handle them. If an SA Core has more than one slice, then the doers are evenly distributed across slices and each slice operates on a similar number of managed servers. Each managed server in Figure 2 is managed by the first doer session below it and all doers are managed by the chunker session at the bottom.

Impact on core servers

Core server performance is characterized by lower resource demand on the database and slice servers in the case of Linux Dynamic Patch Policy remediation when compared to Software Policy remediation. A detailed analysis is presented below for both database and slice servers.

Resource usage on SA truth database

The following graphics show by comparison, the resource demand on the database CPU and the network usage.
The remediation job runs on 100 managed servers.

The two SA remediation job types show similar low CPU and demand across all stages.

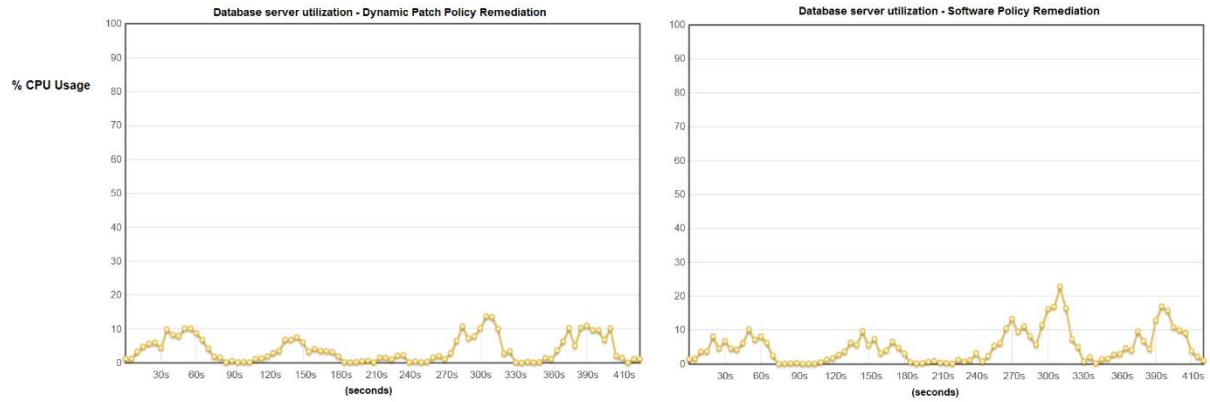

***Figure 3**: CPU utilization on Truth Database/Infrastructure Server*

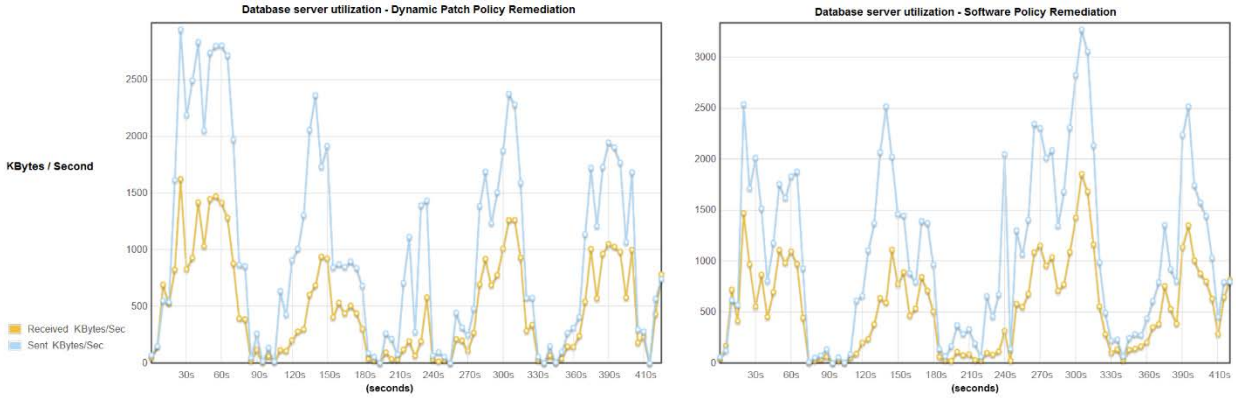

***Figure 4**: Network utilization on Truth Database/Infrastructure server*

Resource utilization on SA slice server

The following graphics show, by comparison, the CPU demand and network utilization on the Infrastructure/Slice server.
The remediation job runs on 100 managed servers.

The the two SA remediation job types show similar high CPU usage in the analysis stage and lower demand while processing the workload on managed servers.

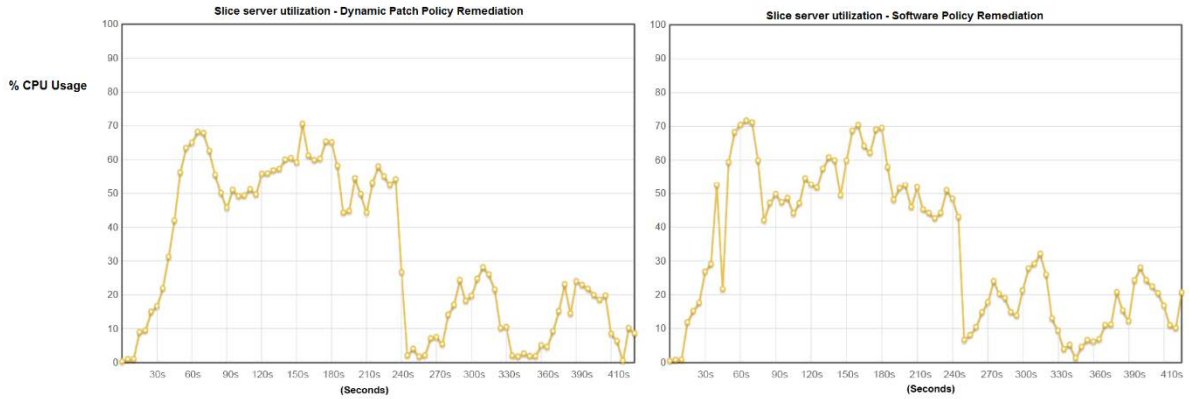

***Figure 5**: CPU utilization on Slice server*

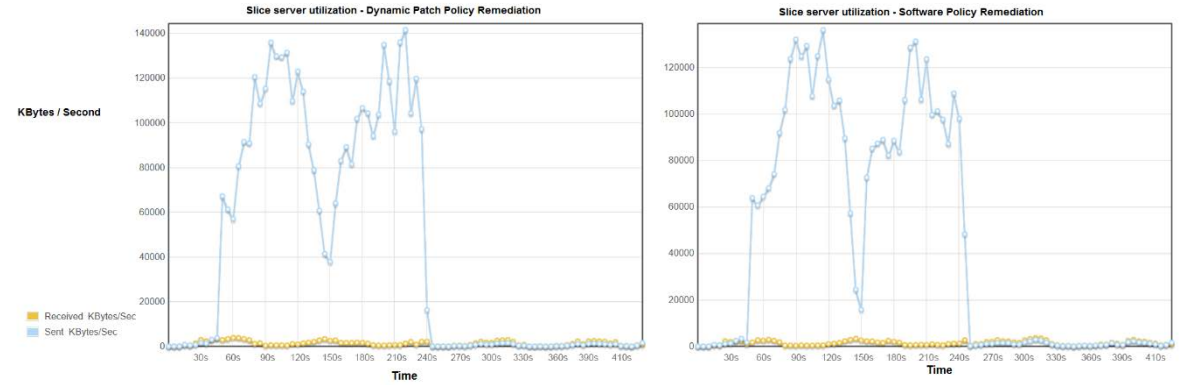

***Figure 6**: Network utilization on Slice server*

Scalability within SA Core

The SA Core automatically distributes remediate workload across all SA slice servers within the Core. In this way, for large patch management job submissions, throughput can benefit from the horizontal scalability of the slice server. Overall throughput increases as the number of slice servers is increased.

Both dynamic patch management and software policy patch management test cases show a significant increase in throughput.

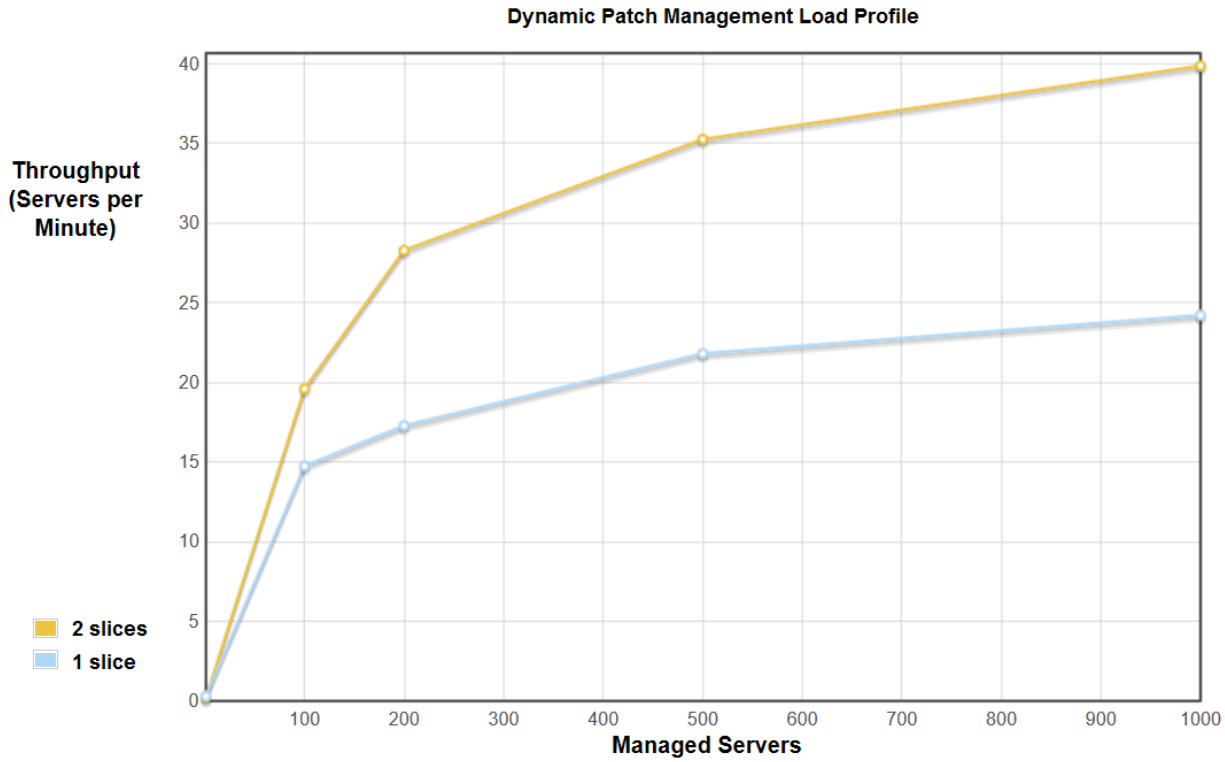

***Figure 7**: Dynamic Patch Management throughput, one slice vs two slice*

The following table gives the horizontal scalability factor at the workload level of 1000 managed servers for the Dynamic Patch Policy remediation job, as the number of slice servers is increased.

# OF SLICES	THROUGHPUT	SCALABILITY FACTOR
1	24.22	1
2	39.88	1.65

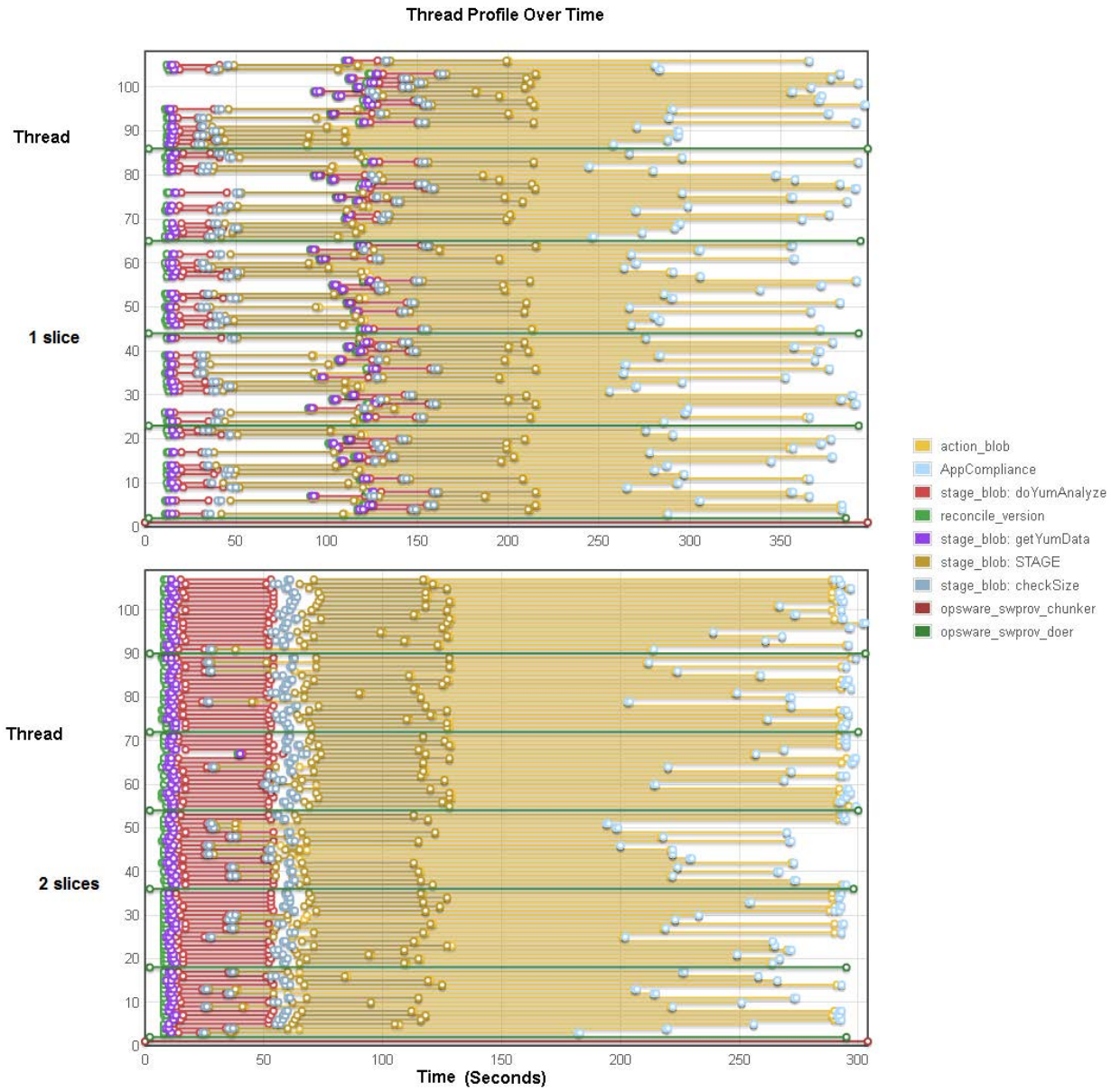

***Figure 8**: Concurrent Dynamic Patch Policy patching operations against 100 managed servers, 1 slice vs 2 slices*

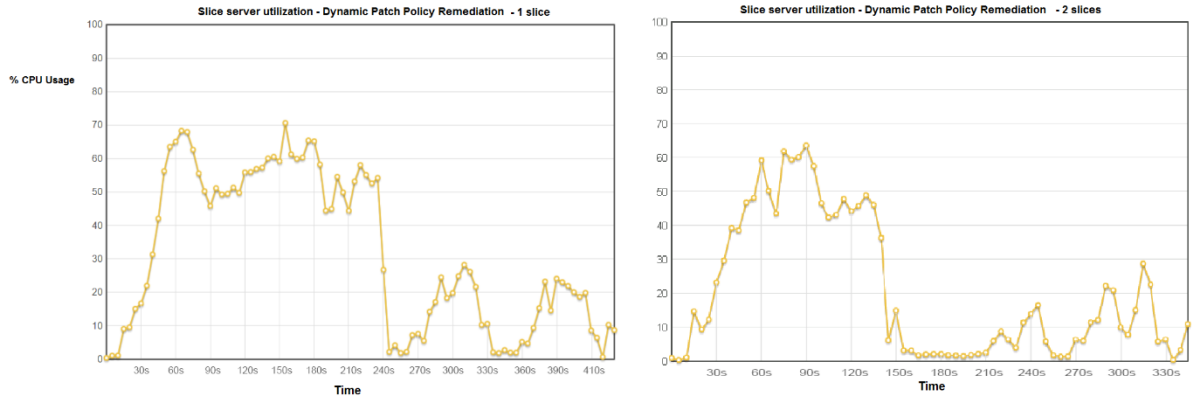

***Figure 9**: CPU usage on a single slice server for Patch Management using Dynamic Policy, configuration with one slice vs two slices, 100 managed servers*

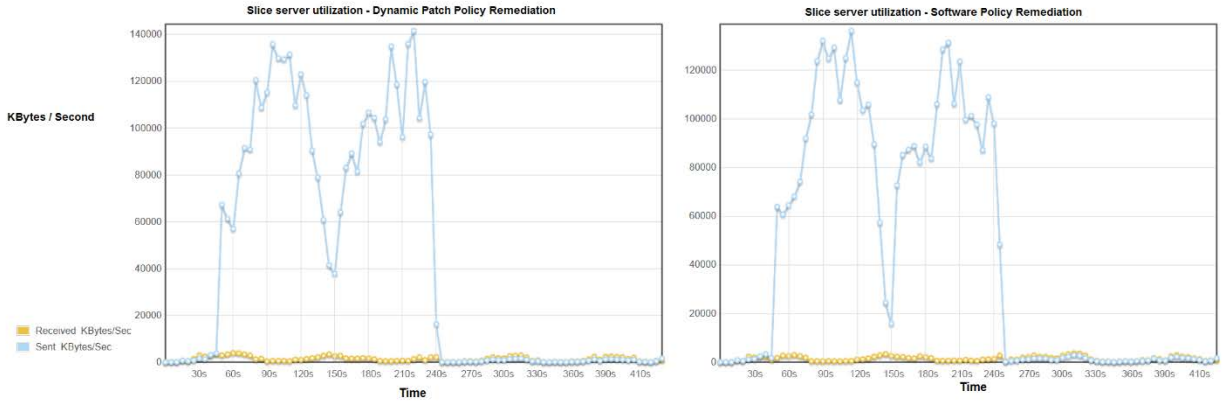

***Figure 10**: Network usage on a single slice server for Patch Management using Dynamic Policy, configuration with one slice vs two slices, 100 managed servers*

The Dynamic Patch Policy Remediation shows lower resource usage (**Figure 9** and **Figure 10**) in the download phase of the patching jobs.
The overall job duration is also shorter on a two-slice SA Core, which gives a higher throughput.

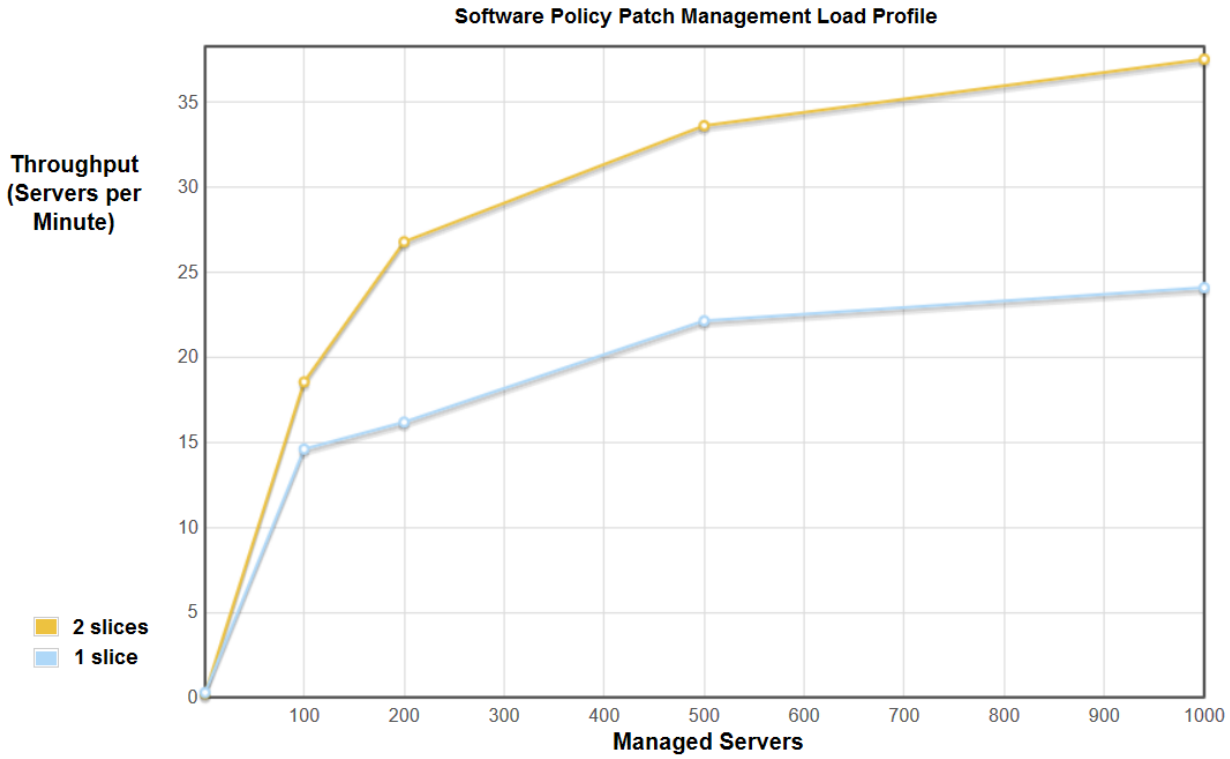

***Figure 11**: Software Policy Patch Management throughput, one slice vs two slice*

The following table gives the horizontal scalability factor at the workload level of 1000 managed servers for the Software Policy Patch management job, as the number of slice servers is increased. The observed scalability factor increase is lower in comparison with the one obtained when leveraging the Dynamic Patch Policy mechanism.

# OF SLICES	THROUGHPUT	SCALABILITY FACTOR
1	24.22	1
2	39.88	1.65

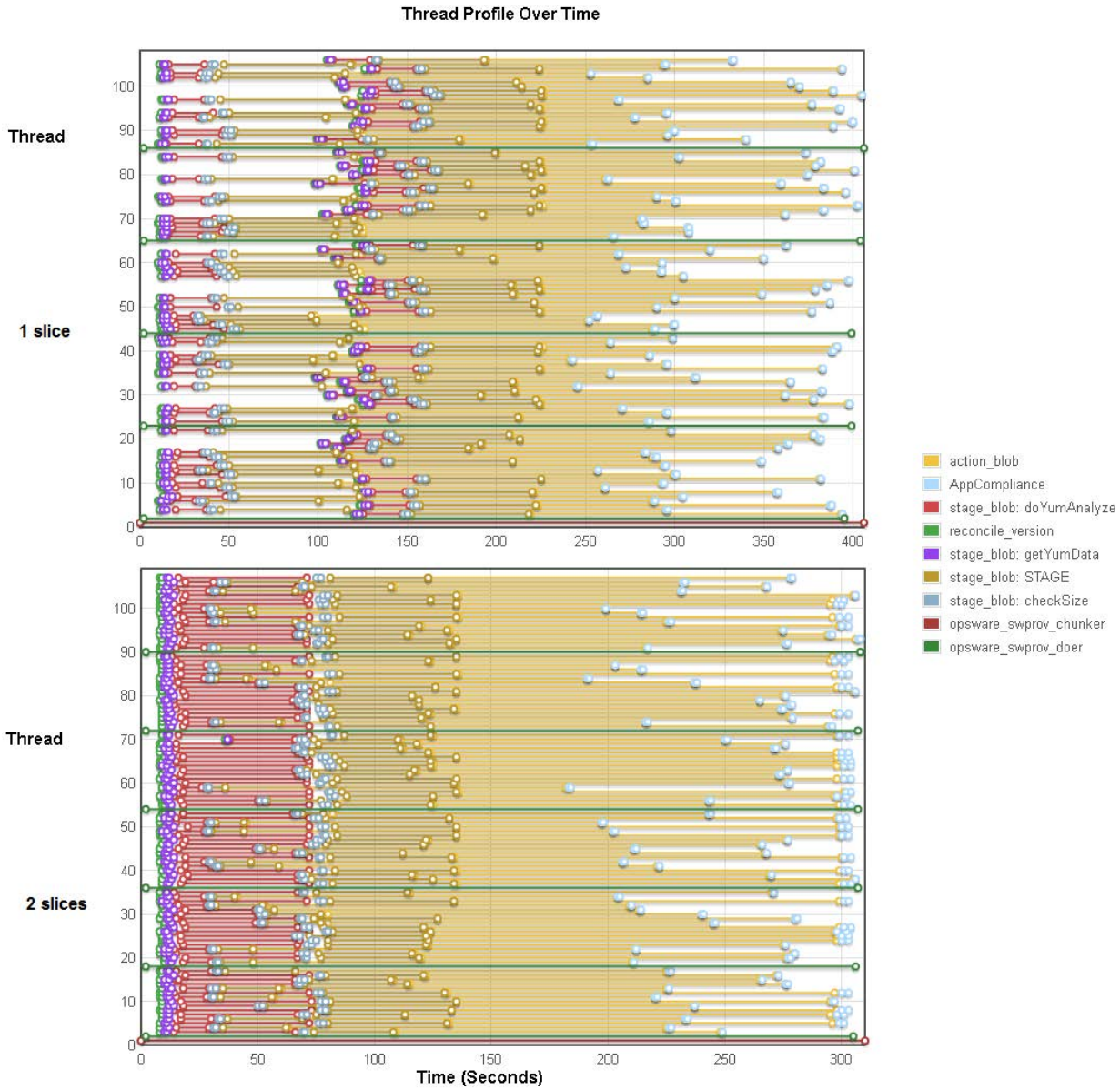

***Figure 2**: Concurrent Software Policy patching operations against 100 managed servers, 1 slice vs 2 slices*

***Figure 3**: CPU usage on a single slice server for Patch Management using Software Policy, configuration with one slice vs two slices, 100 managed servers*

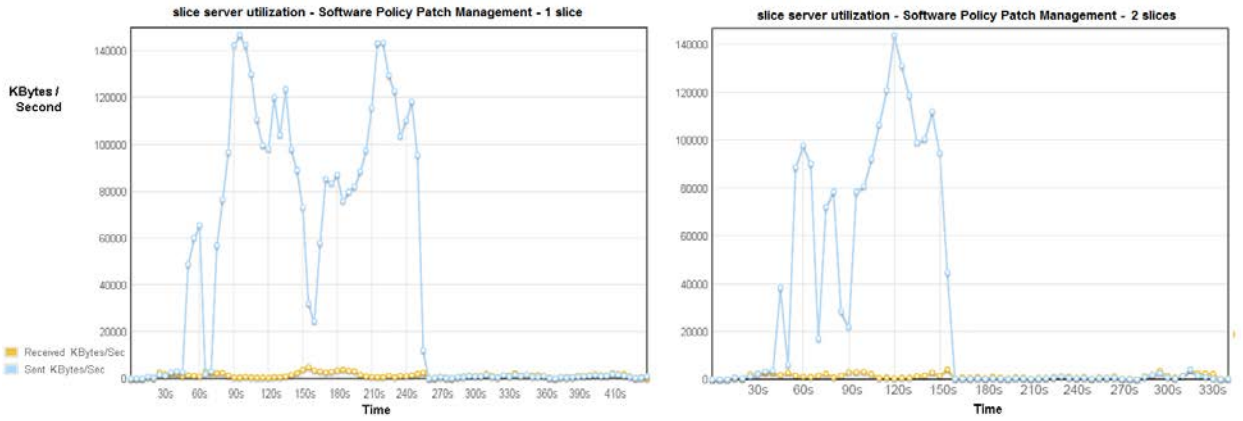

***Figure 14**: Network usage on a single-slice server for Patch Management using Software Policy configuration with one slice vs. two slices and 100 managed servers.*

Altough the Software Policy Patch Management mechanism shows a shorter job duration on a two-slice Core (310s vs 406s) for 100 managed serves, the CPU and Network usage on a two-slice SA Core environment is higher that the Dynamic Patch Policy mechanism on a two-slice SA Core.

Tunable SA configuration parameters

The following system configuration parameters have been adjusted to facilitate stable operations:

Increase way.remediate.package_alarm_timeout

The remediate operation includes the action phase, which implies a workload on the managed server proportional with the payload being remediated.

When the number of managed servers in a job is large enough, this operation may reach the default timeout value of 3600 seconds. In this study, this value has been increased to 10200 seconds. You can turn off the value of this parameter from the SA Client[1].

Increase way.remediate.get_dicts_timeout

The *get_dicts_timeout* tuning parameter in SA is similar to *package_alarm_timeout.* It limits the number of seconds allowed for getting a list of installed software in the remediation action phase.

By default, this parameter is set to 1800 seconds but has been increased to 7200 seconds to work around timeout issues encountered at heavy workload levels. You can change the value of this parameter from the SA Client[2].

This test uses the default parameters for all the other configuration parameters.

Conclusions

For the tested configuration, Software Policy remediation and Dynamic Patch Policy remediation show similar performance characteristics:

- Decreased resource demand on both database and infrastructure/slice servers. The two job types show similar overall job duration for the same payload and job characteristics.
  These representative results depend on the number and size of the installed patches. Different patch sets would exhibit correspondingly different characteristics.
- Using the new Dynamic Patch Policy functionality for Linux improves resource demand. When compared to the alternate Software Policy remediation flow, this functionality has positive influence on the throughput for large payloads like Red Hat Channel policy remediation.
- Dynamic Patch Policies significantly decreases the maintenance time as you only need to manage the content repository. Policy management is performed by SA. Managed servers are scanned automatically or on-demand for the RPMs already installed. SA then marks for upgrade only the RPMs with a newer version than the one installed. Therefore, a remediation of the Dynamic Patch Policy always brings the server up-to-date and in compliance.
- Managed servers are hosted virtual machines running on a number of physical servers. The resources of CPU cycles and bandwidth to SAN (hosting the root disk images) available to the managed servers are limited. At higher levels of concurrent managed server operations, test system limitations may skew the observed results.
- For concurrent operations on a workload level of 1000 managed servers, the Dynamic Patch Policy patching operation achieves a throughput of 24.22 servers per minute and the Software Policy patching operation a throughput of 24.09 servers per minute.
  Adding an extra slice greatly increases the throughput to 39.88 server per minute for the Patch Management using Dynamic Patch Policy and to 37.52 servers per minute for the Patch Management using Software Policy.

[1] Administration View -> System Configuration / Configuration Parameters -> way.remediate.package_alarm_timeout
[2] Administration View -> System Configuration / Configuration Parameters -> way.remediate.get_dicts_timeout

Appendix: Test system configuration

SA Core servers

SA Core infrastructure	• Infrastructure & Slice services • Model Repository Multimaster Component (vault) • Data Access Engine (Spin - primary) • Gateways (mgw) • Media Repository (Word storage on NFS, SMB) • Model Repository Database (Truth)
ESXi host Specifications	• ESXi 5.1 • HW: Model: HP ProLiant BL460c Gen9 • CPU: 16 CPUs x 2.6 GHz Intel Xeon E5-2640 • Memory: 256 GB
VM specifications	• Disk: 150 GB Linux ext4 • CPU: 8x vCPU @ 2.60 GHz , Memory: 32 GB
Network configuration	Network: 10 GBPS LAN, dedicated VLAN
Software specifications	• OS: RHEL6.7 64-bit • SA 10.50 (Build 65.0.70496.0)
SA Core slice #1 and #2	• “Slice” scalable services • Command Engine (Way) • Secondary Spin • Web service API (Twist) • Opsware Global File System (Hub) • Word • Tsunami • Gateways (cgw, agw)
ESXi host specifications	• ESXi 5.1 • HW: Model: HP ProLiant BL460c Gen9 • CPU: 16 CPUs x 2.6 GHz Intel Xeon E5-2640 • Memory: 256 GB
VM specifications	• Local Disk: 150 GB Linux ext4 • CPU: 8x vCPU @ 2.60 GHz , Memory: 32 GB
Network configuration	Network: 10 GBPS LAN, dedicated VLAN
Software specifications	• OS: RHEL 6.7 64-bit • SA 10.50 (Build 65.0.70496.0)

SA database	Oracle database
ESXi host specifications	• ESXi 5.1 • HW: Model: HP ProLiant BL460c Gen9 • CPU: 16 CPUs x 2.6 GHz Intel Xeon E5-2640 • Memory: 256 GB
VM specifications	• Local Disk: 150 GB Linux ext4 • CPU: 8x vCPU @ 2.60 GHz , Memory: 32 GB
Network configuration	• Network: 10 GBPS LAN, dedicated VLAN
Software specifications	• OS: RHEL6.7 64-bit • Oracle Database 12c Standard Edition Release 12.1.0.2.0 – 64bit Production • SA 10.50 (Build 65.0.70496.0)

Managed servers

Managed servers	6.7 VMware VMs
ESXi host specifications	• ESXi 5.1 • HW: Model: HP ProLiant BL460c Gen8 • CPU: 16 CPUs x 2.6 GHz Intel Xeon E5-2670 • Memory: 192 GB
VM specifications	• Local Disk: 20 GB Linux ext4 • CPU: 1 vCPU @ 2.60 GHz , Memory: 2 GB
Network configuration	Network: 10 GBPS LAN, dedicated VLAN
Software specifications	OS: RHEL 6.7 64-bit
Additional notes	VMs are evenly distributed across 28 VMware ESXi hosts

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Linux Patching Performance White Paper (Server Automation 10.50)

Just add your feedback to the email and click send.
If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to hpe_sa_docs@hpe.com.

We appreciate your feedback!

February 2017

Hewlett Packard Enterprise