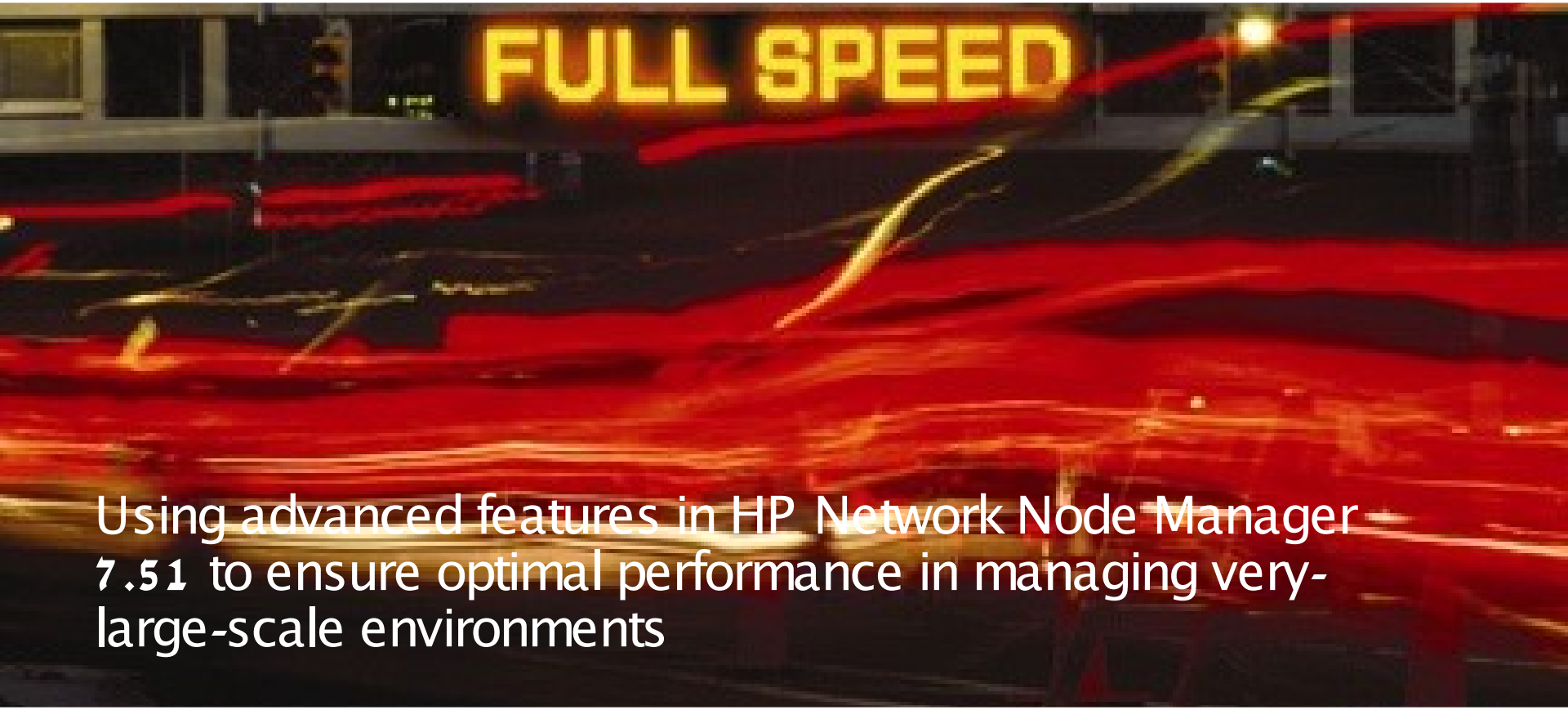
IT Operations Network Management



Using advanced features in HP Network Node Manager 7.51 to ensure optimal performance in managing very-large-scale environments

Using advanced features in HP NNM 7.51 for managing very-large-scale environments

Kevin N. Smith
HP
Lead Software Engineer

Agenda

- Introduction
- Proper machine sizing
- Solid database tuning
- Improving ET discovery
  - Zoning techniques
  - Incremental Node Discovery
  - Interface Filtering
  - Cisco Discovery Configuration
- Reducing Traps and Events
- Tuning APA
- User Interface tuning
- Practical Limitations

Introduction

- Who am I?
- Kevin N. Smith
- kevin.n.smith@hp.com
- Software Developer on NNM
- I do a lot of consultation and help with 7.51 deployments at customer sites

Focus of Presentation

- Primary focus is on Extended Topology and APA
- The basis of this presentation is NNM 7.51 with IP15
- New features have been introduced to help with large scale deployments
- This is not a general purpose "tips and techniques" introduction
- It is complementary to the NNM White Paper documentation
- Much of this presentation comes from a document called
  - "NNM 7.51 Extended Topology and APA Deployment Handbook"
- Please come see me for a copy if you don't have version 1.2 of this document

Defining a large environment

- > **2500** switches and routers
- > **100,000** interfaces
- Many of these techniques can still be applied to smaller environments with good results

Why do large scale?

- With the invention of APA, a typical NNM station can monitor a much larger network than traditional netmon solutions
- Many customers are "collapsing" collection stations running NNM 6.4 to a single 7.51 solution with ET and APA.
- This provides lower hardware costs and easier maintenance of NNM via fewer stations to track
- But adding more managed objects to one station leads to other unique challenges

Additional Documentation

- The new features in NNM **7.51** were not documented via user manual changes.
- All documentation for new features was delivered in White Papers.
- These are located in $OV_DOC/WhitePapers/
- I'll try to point out the White Paper names during the presentation
- Some of these features are off by default and must be enabled (see documentation for details)

Machine Sizing

- 4-CPU box
- CPU speed is very important
  - 3 GHz CISC
  - 1 GHz RISC
- RAM also important
  - 4 gig RAM
  - 8 gig not much better

Database

- $OV_DOC/WhitePapers/SolidPerformanceImprove.pdf
- In general, the embedded database (Solid) works well for large scale after you make the changes recommended in the document
- Changes are made to $OV_DB/analysis/default/solid.ini
  - automatically changed in HPUX
  - Solaris requires one addition change made by user
  - Windows requires all changes to be made by user
- This will greatly improve the Solid performance.
- Oracle can be used but is not required. Some features appear to work better with Solid.

Improving ET Discovery

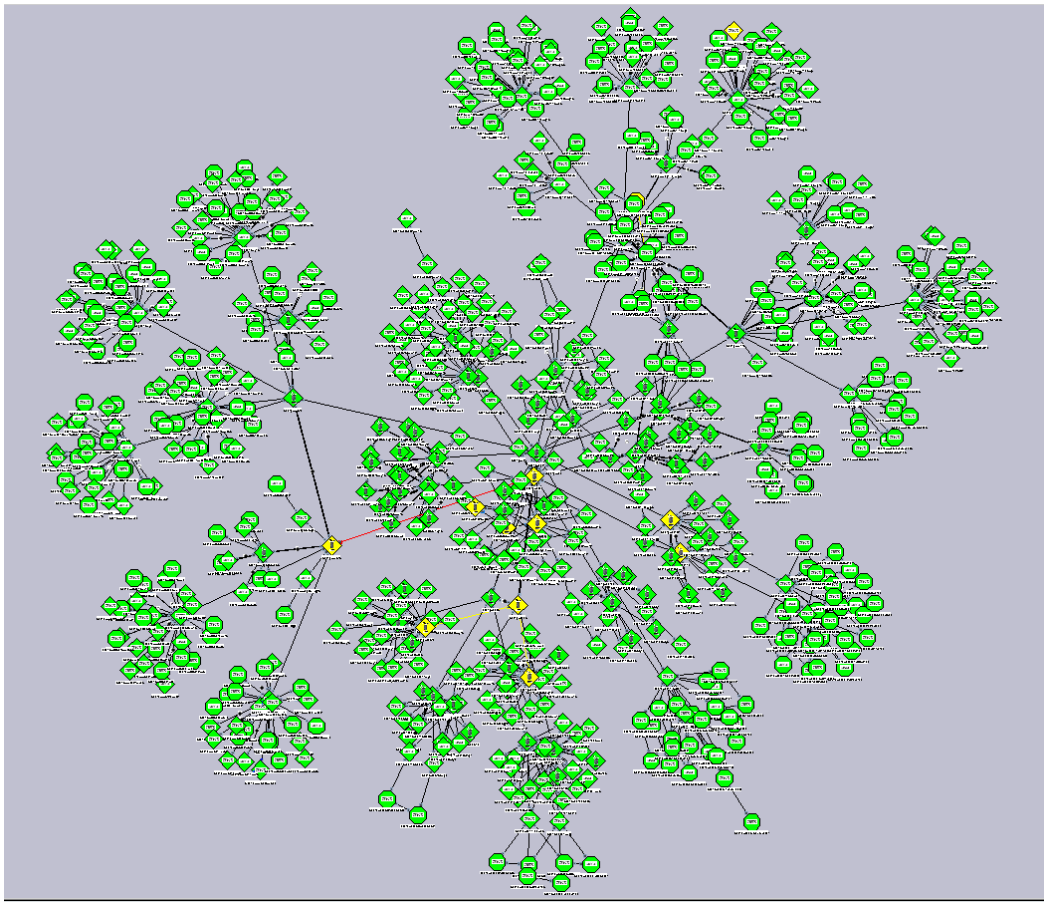
- Zoning techniques
- Interface Filtering
- Cisco Agent Tuning

Zoning Techniques

- Because it's not a new feature, I won't go into zoning here but please see the deployment handbook and user manuals for details
- Good zones make for good topology
- Why does it matter? APA makes decisions primarily based on neighbor analysis and thus needs accurate neighbor relationships
- I always start with autozone but many times end up using manual zones
- At one customer site, switching from autozone to manual zones improved discovery speed by 50%, greatly improved topology accuracy and made APA shine.
- Rich wildcarding is available when creating manual zones (see the deployment handbook for details)

Autozone or manual zones?

- How do I know if I have a good topology?
- It will be well connected in neighbor views
- If poorly connected, consider using manual zones



Incremental Node Discovery

- $OV_DOC/WhitePapers/IncrementalNodeDiscovery.pdf
- Allows newly discovered nodes in classic NNM (either by auto-discovery, loadhosts, or added to map by the user) to be added to ET topology without doing a full discovery.
- Only node and interface information is discovered.
- Node connectivity is not discovered.
- Provides APA with device existence for simple status polling.
- Full or incremental zone discovery is required to discover full device and protocol attributes including connectivity.
- APA behavior for polling new incremental nodes is handled by the isNewNode and isNewInterface polling policies (paConfig.xml) until a new ET discovery is run

Interface Filtering

- $OV_DOC/WhitePapers/InterfaceFiltering.pdf
- Extended Topology Discovery performs better with fewer interfaces
- This feature allows you to selectively remove interfaces from the discovery
- These interfaces are removed from both classic NNM and ET
- This can be useful if trying to place more gear into a single zone via manual zones (like a building with a large amount of L2 interconnectivity)
- It can also reduce the overall discovery time significantly

Interface Filtering (continued)

- This feature was primarily introduced to deal with routers with huge number of virtual interfaces that don't need to be monitored (some routers have over 10,000 interfaces)
- ET can't handle such high interface counts on single devices
- Secondarily, the feature can be used to eliminate other unnecessary interfaces to improve performance
- If you don't think you'll ever care about the interface, consider filtering it out though use some caution.
- This is different than "not monitoring it" via APA filtering. Sometimes you need the interface in topology for neighbor analysis even if you don't think you'll ever monitor it.

Interface Filtering (continued)

- You can identify nodes based on IP wildcarding or Node Name wildcards
- You can identify interfaces based on the following fields
  - ifType
    - numeric field
  - ifDescr
    - character field
  - ifAdminStatus
    - numeric field
  - ifName
    - character field
  - ifAlias
    - character field

Interface Filtering Example

- Suppose you have many access switches that are connected to computers that you don't manage
- You know that with your configuration, APA only monitors the uplink port (which is the Gig port) and none of the others.
- You decide that you could improve discovery performance by eliminating these interfaces completely.
- Suppose that you have a naming convention that all access switches can be identified with "USR" in the name.
- You also see that the Gig uplink ports have unique IF names starting with Gi (like Gi1/1, Gi1/2)

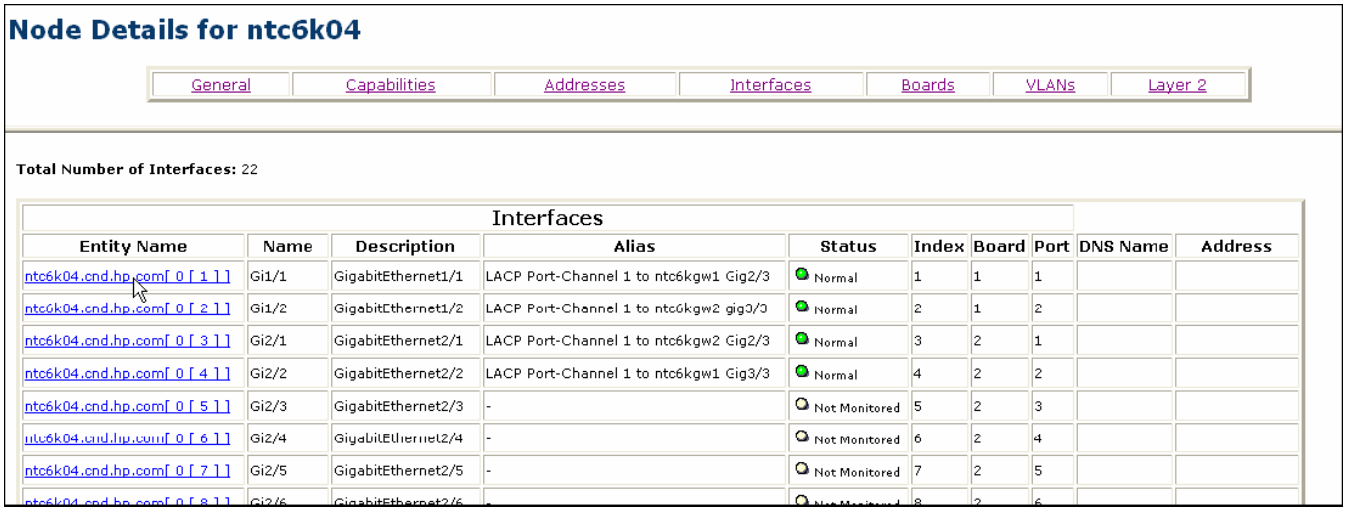
Interface Filtering Example (continued)

- You now edit the file $OV_CONF/netmon.interfaceNoDiscover
- There are some good examples in the file
- There is a Filtering Mode entry that you can read about in the documentation. I usually use "2" meaning "filter any IF of this type"
- You add the following entry stating that you want to "throw out" all interfaces except those beginning with Gi on nodes that have USR in the name.

```
# Only keep Gigabit uplink interfaces on user access switches
*USR*.hp.com  2  ifName!="Gi*"
```

Interface Filtering Example (continued)

- You can get good details about nodes and interfaces from Node Details pages in ET



Node Details for ntc6k04

General | Capabilities | Addresses | Interfaces | Boards | VLANs | Layer 2

Total Number of Interfaces: 22

Interfaces

Entity Name	Name	Description	Alias	Status	Index	Board	Port	DNS Name	Address
ntc6k04.cnd.hp.com[0 [1]]	Gi1/1	GigabitEthernet1/1	LACP Port-Channel 1 to ntc6kgw1 Gig2/3	Normal	1	1	1		
ntc6k04.cnd.hp.com[0 [2]]	Gi1/2	GigabitEthernet1/2	LACP Port-Channel 1 to ntc6kgw2 gig3/3	Normal	2	1	2		
ntc6k04.cnd.hp.com[0 [3]]	Gi2/1	GigabitEthernet2/1	LACP Port-Channel 1 to ntc6kgw2 Gig2/3	Normal	3	2	1		
ntc6k04.cnd.hp.com[0 [4]]	Gi2/2	GigabitEthernet2/2	LACP Port-Channel 1 to ntc6kgw1 Gig3/3	Normal	4	2	2		
ntc6k04.cnd.hp.com[0 [5]]	Gi2/3	GigabitEthernet2/3	-	Not Monitored	5	2	3		
ntc6k04.cnd.hp.com[0 [6]]	Gi2/4	GigabitEthernet2/4	-	Not Monitored	6	2	4		
ntc6k04.cnd.hp.com[0 [7]]	Gi2/5	GigabitEthernet2/5	-	Not Monitored	7	2	5		
ntc6k04.cnd.hp.com[0 [8]]	Gi2/6	GigabitEthernet2/6	-	Not Monitored	8	2	6		

Interface Filtering Example (continued)

- Restart netmon to pick up the change and it will begin removing these interfaces
- Running an ET discovery will remove these interfaces from ET (you can do this step before the netmon step)
- I recommend against having netmon remove these interfaces while running an ET discovery if removing a lot of interfaces (over 10,000)
- There is usually a high overhead when removing huge numbers of interfaces from classic NNM. Expect to see some processes running high CPU like ovwdb and netmon for the next 24 hours after making this change.
- I usually stop (or completely disable) netmon while I experiment with ET. After I'm convinced it's going to work well, I let netmon run for 24 hours without running an ET discovery during this removal phase.

Cisco Discovery Configuration

- $OV_DOC/WhitePapers/CiscoDiscoveryConfiguration.pdf
- Allows you to control how much detail the ET discovery process collects for certain Cisco network devices
- This can improve ET discovery performance making discovery complete faster

Cisco Discovery Configuration

- This feature can be highly tuned but usually the greatest benefit comes from Forwarding Database Tables (FDB)
- For Cisco devices, ET primarily uses CDP and FDB tables to derive connectivity
- CDP is much lighter weight than FDB
- If you have certain Cisco devices that are always connected strictly to other Cisco devices running CDP, you could use this feature to exclude FDB gathering from these devices
- You need to be cautious with this feature. FDB is the principle method of discovering connectivity between certain devices and you must be careful to not degrade the integrity of the NNM Topology
- There aren't great methods to audit the results. I usually watch the connectivity via the Topology Summary page or various neighbor views involving the nodes.

Reducing Traps and Events

- Many customers have large trap volumes received by the NNM station
- This can cause poor performance and stability
- NNM 7.51 introduced enriched capabilities for filtering traps

What traps are used by NNM?

- APA responds to a set of traps when monitoring devices
- Currently, these traps include:
  - ColdStart 1.3.6.1.6.3.1.1.5.1
  - WarmStart 1.3.6.1.6.3.1.1.5.2
  - LinkDown 1.3.6.1.6.3.1.1.5.3
  - LinkUp 1.3.6.1.6.3.1.1.5.4
  - HSRPState 1.3.6.1.4.1.9.9.106.2.0.1
  - chassisChangeNotifOID 1.3.6.1.4.1.9.5.11.2.0.2
  - StackMIBModuleDown 1.3.6.1.4.1.9.5.0.4
  - StackMIBModuleUp 1.3.6.1.4.1.9.5.0.3
- All others are simply passed through the event pipeline and written into the Binary EventStore (BES) and not used by NNM
- If you don't need a record of these traps, you can filter them out
- If you need to forward specific traps to other applications, you can still trim down a number of extraneous traps to improve performance.

Trap logging

- Setting a trap to "log only" still allows it to pass through the pipeline which can bog down the system
- A high volume of traps also causes the Binary Event Store (BES) to "roll" frequently and will cause you to lose your history of alarms quickly.
- You can increase the size of the BES but this is discouraged.
- It is best to filter it when the trap first arrives

ovtrapd.conf

- A feature was introduced in an earlier release that helped NNM survive trap storms called ovtrapd.conf.
- In addition to protected against storms, this file can be tuned to selectively discard traps
- Previously ovtrapd.conf could only filter traps from a specific node.
- Now you can filter specific traps base on OIDs as well as specific nodes. These can both use wildcard expressions.
- Some documentation can be found in $OV_DOC/WhitePapers/EventReduction.pdf
- As well as a man / reference page on ovtrapd and ovtrapd.conf

An example of using ovtrapd.conf

- (Note that ovtrap must be configured to use ovtrapd.conf values. Read the man pages for ovtrapd and ovtrapd.conf)
- You notice that PMD runs at high CPU
- You consider that you might have a high volume of traps affecting performance
- To see the most frequent traps, run "ovdumpevents" to get an readable format of the current BES

```
ovdumpevents > /tmp/dumpevents.txt
```

- Next run these results through "processEvents"

```
/opt/OV/support/processEvents dumpevents.txt summaryfile
```

- This program may give you some errors but the results are still fairly good.

ovtrapd.conf example continued

- The summary file will contain information about the number of trap occurrences in the BES. Lines will look similar to:

```
Total Number for trapId .1.3.6.1.4.1.9.9.43.2.0.1 = 5274
```

- If you wanted to find out what node this trap is coming from, simply grep for the OID in the dumpevents.txt file
- You decide this trap isn't of interest and you don't want to receive it from any node
- Place the following entry in ovtrapd.conf and restart ovtrapd (assuming you configured ovtrapd to use ovtrapd.conf)

```
<*.*.*.*,.1.3.6.1.4.1.9.9.43.2.0.1>
```

- A common misunderstanding is that you can't have multiple "lines" for the same IP wildcard. So if you want to filter two traps for all nodes, the syntax would look like

```
<*.*.*.*,.1.3.6.1.4.1.9.9.43.2.0.1,.1.3.6.1.4.1.9.5.0.*>
```

- I don't know how long this line can go so try to use wildcarding and ranges as much as possible

Tuning APA

- The deployment document gives some detailed examples
- Instead I'll cover general concepts plus a new tool

Why tune APA

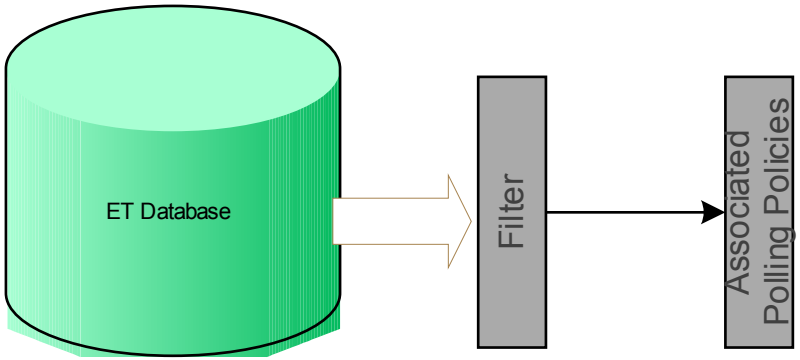
- Throwing the "big switch" and moving from netmon to APA usually has limited success
- Netmon settings are not migrated to APA settings
- You must tune APA to get the desired effects
- The process for tuning APA is a paradigm shift from netmon

Netmon tuning vs. APA tuning

- In netmon, the primary method of tuning was done by unmanaging interfaces on individual nodes and using various files like netmon.noDiscover
- You could use ovautoifmgr to unmanage interfaces based on filters
- APA is primarily tuned based on filters and polling policies
- Individual nodes and interfaces can be "unmanaged" via ovet_toposet but it is command line only and inefficient

APA Tuning Paradigm

- APA works with collections of objects
- Objects can be nodes, interfaces, addresses, boards, HSRP groups, and others
- Collections of objects are defined via filters (TopoFilters.xml)
- Collections of objects then have polling policies associated with them (paConfig.xml)
- The same filters can are used in other parts of NNM like Dynamic Views, topology dumps, etc.
- The hard part is that the filters and polling policies are written with XML. This makes the filtering extremely powerful yet harder to learn.



What not to do

- Don't enabled polling on every node and interface and then use ovet_toposet to unmanage select objects.
- This is very inefficient and hard to maintain
- I also don't recommend going with the out-of-the-box default APA settings

Issues with default settings

- The number one issue with default APA settings is that switch/routers are treated as routers.
- APA polls all interfaces on routers by default.
- This causes many unwanted alarms if the node is acting as a switch connected to computers.
- A better set of defaults is to poll only "connected" interfaces on all devices along with interfaces with IP addresses on them.
- Interfaces are considered "connected" if they are found to connect to another object in the NNM ET topology. This usually excludes end computers.

Simple Approach

- If you don't have time to learn APA, I recommend consulting the Deployment Handbook and making these changes:
  - implementing the Switch/Router change (swouter)
  - Setting switches to be "pinged" by default
- After that, consider disabling polling of interfaces based on IF types
- Last, consider adding polling to all interfaces that have IP Addresses on them if appropriate
- Contact me if you are stuck with any of these steps.

Additional Steps

- After you get more comfortable, you'll be able to develop sophisticated filters that can make for powerful tuning
- When doing development of filters and policies, I recommend using the program RCS or a similar tool to track your changes.
  - Don't have lots of copies of paConfig.xml and TopoFilters.xml.
  - You'll get confused easily.
  - RCS is free on all platforms and easy to use.
  - Tutorials are available on the web.
- Consider using an XML editor. There are some very nice free ones available on the web.
- Maybe buy an XML book or find a tutorial on the web
- A hint that most people miss: If you develop node filters for APA, you usually need to develop interface filters to define interfaces that reside on these nodes.
- Please see the Deployment Handbook for further details.

Debugging polling policies

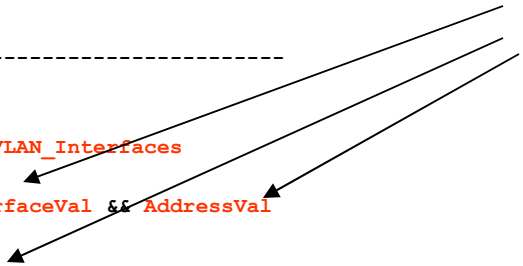
- One of the harder things in APA tuning is diagnosing problems with your filters and polling policies.
- A very handy tool was developed to help

```
ovet_demandpoll.ovpl -d
```

- Previous to this tool, it was difficult to know what polling policy (via a filter) was being applied to a node, interface or address.
- Now you can know exactly what policy is being applied.
- Only drawback is that APA must be running. If you make changes, you must restart APA to pick up the new changes.

Example of ovet_topodump.ovpl -d

```
ovet_demandpoll.ovpl -d testnode.corp.com
==========================================================================
ADDRESS  10.28.16.33

OBJ_TYPE   SHORT_OBJECT_NAME    snmpEnable ET_FILTER
-----------------------------------------------------------------------------
all-types       *                  true    DEFAULT
NODE       hp3sw3                  true    isSwitch
INTERFACE  hp3sw3[0[53]]           false   My_Unmonitored_VLAN_Interfaces
ADDRESS    10.28.16.33             true    DEFAULT
Composite  10.28.16.33             false   Always false for Addresses


OBJ_TYPE   SHORT_OBJECT_NAME    pingEnable ET_FILTER
-----------------------------------------------------------------------------
all-types       *                  true    DEFAULT
NODE       hp3sw3                  true    isSwitch
INTERFACE  hp3sw3[0[53]]           false   My_Unmonitored_VLAN_Interfaces
ADDRESS    10.28.16.33             true    DEFAULT
Composite  10.28.16.33             false   NodeVal && InterfaceVal && AddressVal


OBJ_TYPE   SHORT_OBJECT_NAME    interval   ET_FILTER
-----------------------------------------------------------------------------
all-types       *                  900     DEFAULT
NODE       hp3sw3                  900     isSwitch
INTERFACE  hp3sw3[0[53]]           300     UnconnectedAdminUpOrTestSwitchIF
ADDRESS    10.28.16.33             900     DEFAULT
Composite  10.28.16.33             300     Interface Value
```



You can tell exactly which policy is applied to each object and the composite result

Controlling the management address

- The "management address" is the address that NNM uses to speak SNMP to the node.
- 7.51 gives you more control over the management address
- Most customers with Cisco gear prefer to always communicate with nodes via the loopback address
- Usually netmon will chose to communicate with the loopback address but under certain conditions, netmon and APA will change the management address of a node.

Management address continued

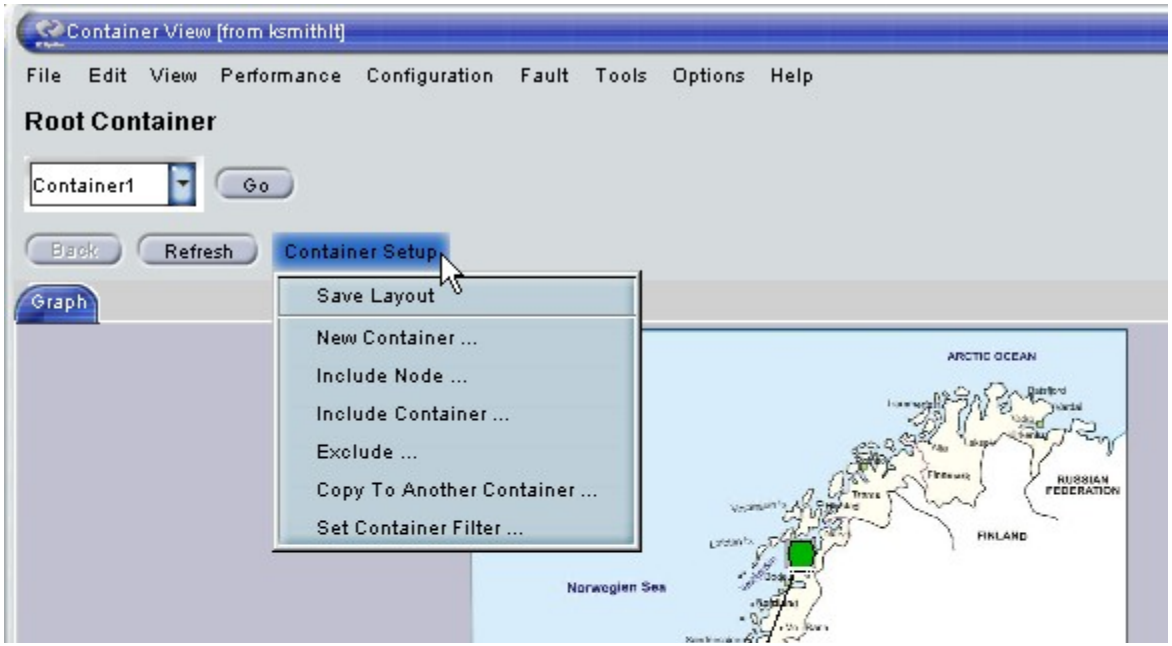
- If you don't want netmon to change management address based on reachability, you can disable this in netmon.lrf
  - -k adjustNodeSnmpAddr=false
- APA will do a similar behavior. If you want to prevent it from changing in APA. Edit paConfig.xml and change the default parameter for MgmtAddrInhibited from false to true.
- What if the address has already been changed and you want to get it back to the loopback?
  - Delete the node (ovtopofix -r) and rediscover the node
  - Come see me for a handy tool that can help

Using Container Maps

- Customers can place nodes into container views and create/modify a hierarchy of containers from the GUI
- Nodes can reside in more than one container
- Container placement can be based on logic (ET filters) or placed by hand (though I recommend only using ET filters if possible)
- Container maps are "static" (nodes retain position) and can have customized background graphics
- Containers can be hierarchical and show cumulative status based on sub-containers and nodes
- Optional Container-to-Container and Container-to-Node connectivity

Hard parts

- Container menu is hard to spot
- There is no "delete container" menu item
- There is no "rename container" menu item



Other general rules

- Try to have less than 50 nodes in a container
- Have fewer than 500 containers
- Consider turning off connectivity as the results can be misleading (we only display nodes that pass the filter). Use neighbor view for connectivity instead.

Advanced Container View Tasks

- Some procedures require editing the file $OV_DB/nnmet/containers.xml
  - Delete Container
  - Rename Container
- I usually stop ovas while messing with this file.
- Scale Container Icons versus Background Graphics

Delete Containers

- Remove any containerReference tags
- Remove the entire container tag
  - Note: blank lines in the file might mislead you. Watch out for that. You can remove them.

```
<container name="Fort Collins">
    <containerReference name="MPLS nodes" x="356.8442951222268" y="182.8442951222268"/>
    <containerReference name="NTC" x="531.3368696289949" y="180.3368696289949"/>
</container>

<container name="MPLS nodes">    <filter name="MplsName"/>
    <topoNode name="mplspe06.cnd.hp.com" ... />
    <topoNode name="ntc6k01.cnd.hp.com" ... />
</container>
```

Rename Container

- Simply replace name="old name" to name="new name" in all locations
- http://<hostname>:7510/topology/home?reloadContainers=true or restart ovas

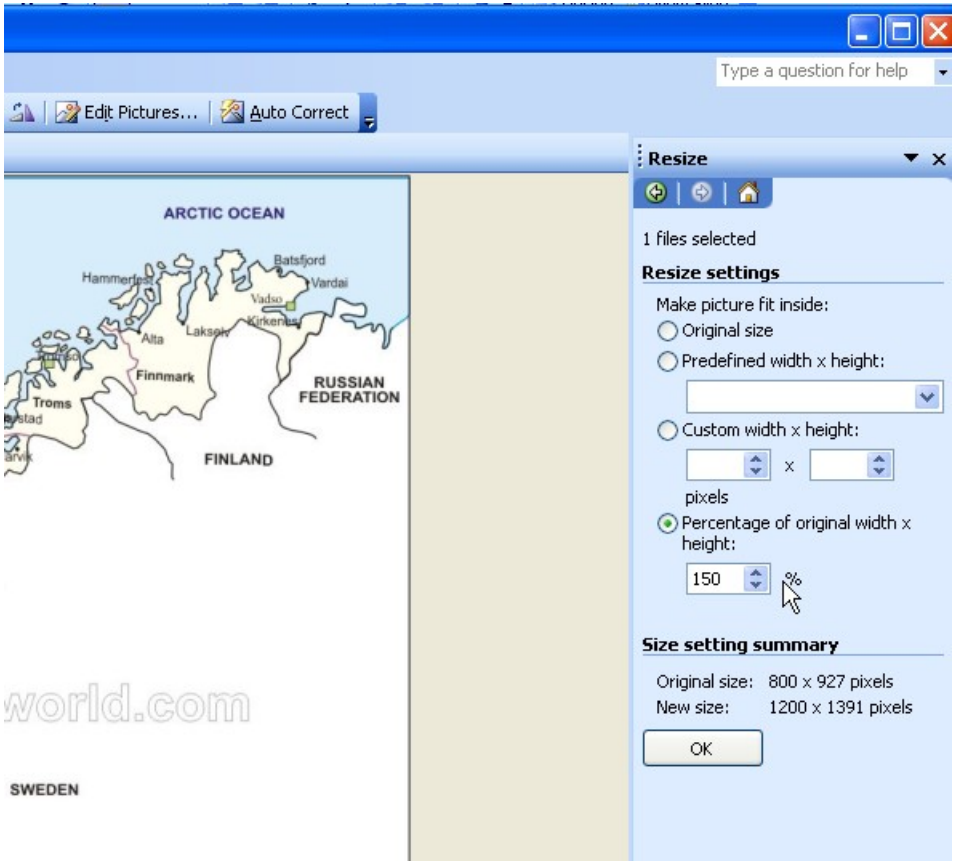
```
<container name="Fort Collins">
    <containerReference name="MPLS nodes" x="356.8442951222268" y="182.8442951222268"/>
    <containerReference name="NTC" x="531.3368696289949" y="180.3368696289949"/>
</container>

<container name="MPLS nodes">    <filter name="MplsName"/>
    <topoNode name="mplspe06.cnd.hp.com" ... />
    <topoNode name="ntc6k01.cnd.hp.com" ... />
</container>
```

Scale Container Sizes

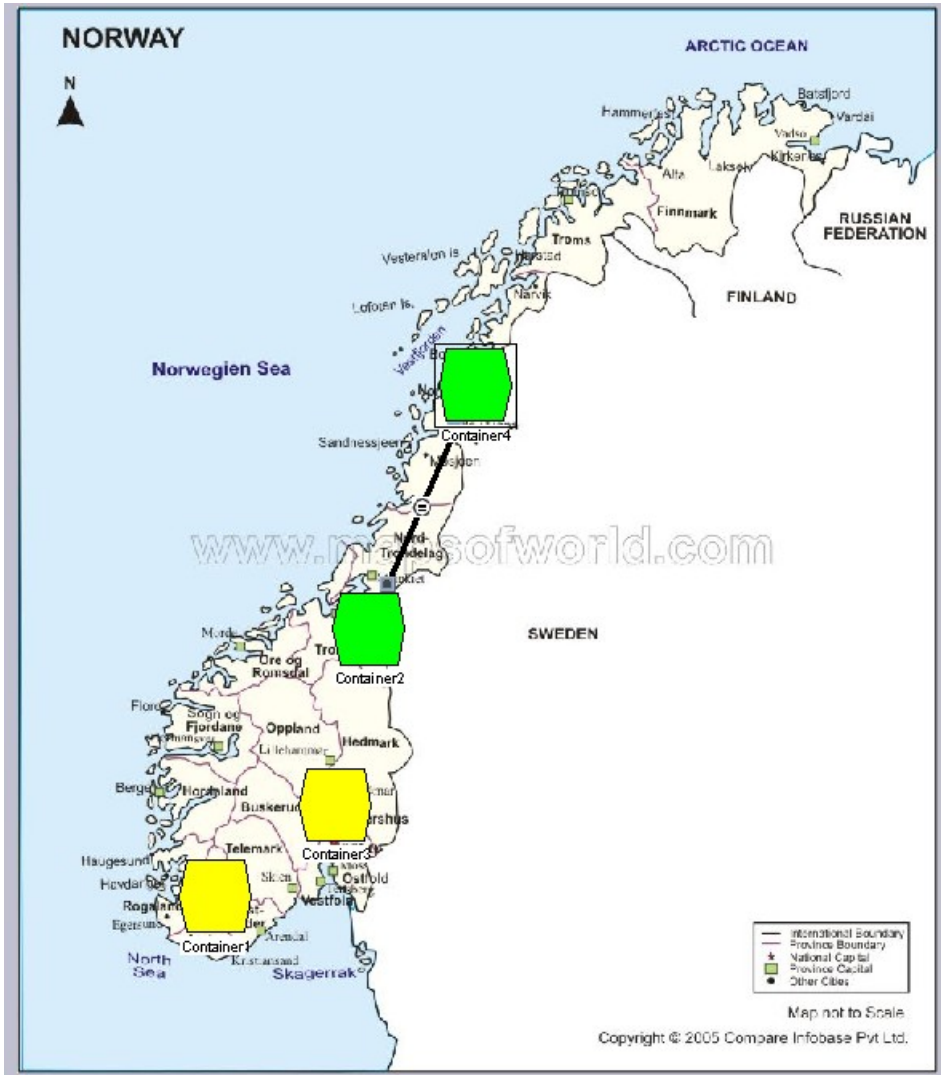
- If you want to change the size of the container relative to the background graphic, you must increase the size of the background graphic.
- I use Microsoft Office Picture Editor

Select Resize in MS Picture Editor

Increase the size of the image

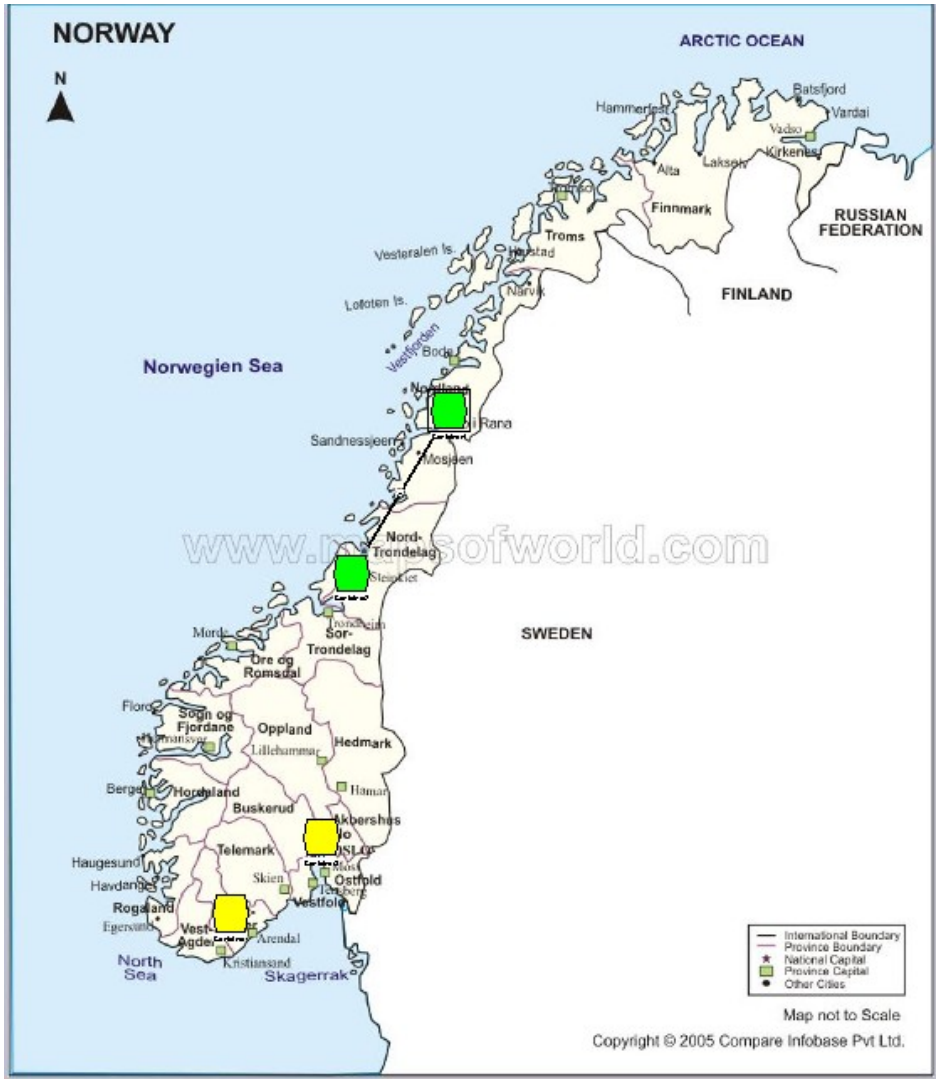


Containers before scaling JPEG file



invent

Containers after scaling JPEG file



Other handy GUI changes

- In the file $OV_AS/webapps/topology/WEB-INF/web.xml, you can set the following:
  - Set default Neighbor View Hop to 1 (defaultNumHops)
  - Change Node View to use ET Filters (enableETFilters)
  - Change Interface View to not show disabled interfaces (includeDisabled)
  - Limit the NeighborView to use ET Database only (restrictToET)

Practical Limitations

- Even new technology has its limits
- In practice, NNM ET seems to be able to scale to about 10,000 nodes and 200,000 interfaces. If you need to manage a larger environment than this, consider using two NNM stations and splitting the management responsibilities.
- APA can handle these large environments well but is sensitive to ICMP timeouts so watch for these.
- It's okay to increase the threads in APA
- Discovery is more challenging so work hard to make discovery work well since everything else depends on a good topology.
- You really don't need more than 4 CPUs in almost all cases

Conclusion

- NNM can scale very large with tuning using these new features
- Please consult the Deployment Handbook plus the White Papers for full details
- Support specializes in defect fixes not deployment
- Please contact me if you need help ☺
- You can also contact nnm.docs@hp.com for deployment questions.

HP Software Universe

June 18-22, 2007 | The Venetian | Las Vegas, Nevada