15th ANNUAL WORKSHOP 2019
High Speed Networking Monitoring Enhancements
Brett Holman, Jesse Martinez
Los Alamos National Laboratory
March 19th, 2019
LA-UR-19-22147
1. Collect port level traffic bandwidth on cluster fabrics (switch and network card)
2. Collect switch forwarding tables for whole fabric
3. Send and store data for trend analysis
4. Targeting our largest clusters
5. Focus on Intel Omni Path first
6. Integrate with existing monitoring software
7. Higher granularity – 10 seconds or less
HSNMON OVERVIEW
EXISTING MONITORING SOFTWARE: HSNMONT

- Daemon that runs on Fabric Manager (FM) node
- Fails over with FM
- Collection of scripts that check for various error conditions
  - Error counters
  - Performance counters
- Uses FastFabric tools for gathering data (`opaextractperf`)
- Logs results to syslog
- Written by Jesse Martinez based on design of IBmon
  - Susan Coulter
Most scripts located at /usr/local/hsnmon/

- **hsnmon.pl**
  - Main script, calls other scripts
  - Runs as daemon
  - Runs every X minutes, configurable
- **hsn_rosetta.pl**
  - Collects counters using `opaextractperf` and parses it into readable format
  - Looks up local/remote devices
• `hsn_counters.sh`
  – Calls FM to dump counters/clear
• `hsnmon_init.pm`
  – Read in config, define shared functions
• `fabric_status.sh`
  – Pulls FM information to determine if master
• `create_netmap.pl`
  – Generated `hsnnet_map` from fabric links
EXISTING MONITORING SOFTWARE: HSNMON

Start (hsnmon) → Load config (Otherwise Default) → hsnmon.pl → Paused? Not Master?

- Paused? Not Master?
  - No → DST?
    - No → Generate hsnnet_map
    - Yes → Enable DST Mode
  - Yes → Clean log directory
  - Generate host List
    - Generate switch list (Look for missing switches)
      - Pull FM counters/clear
        - Call hsn_rosetta.pl
          - Gather/report errors by link
          - Gather/report performance counters by port
        - Gather link analysis report
        - Sleep
  - Not Master? → Clean log directory
  - Gather/report errors by link
  - Gather/report performance counters by port
  - Sleep


Enable DST Mode → DST? → DST?
EXISTING MONITORING SOFTWARE: HSNMON

Usage: /usr/libexec/hsnmon {start|stop|status|dst-start|dst-stop|pause|clear-counters|help}

Parameters:

- **start** - Start or resume hsnmon (DST mode, if enabled, will resume)
- **stop** - Stop hsnmon (DST mode, if enabled, will remain enabled)
- **status** - Report current status of hsnmon
- **dst-start** - Enables DST Mode
  - Disables error logging ONLY (if enabled)
  - Performance/fabric monitoring will continue (if enabled)
  - DST mode can be enabled with or without hsnmon running
- **dst-stop** - Disables DST Mode
  - Resumes error logging if enabled
  - Clears counters
  - DST mode can be disabled with or without hsnmon running
- **pause** - Disables/halts HSNmon from executing, upon resume (start), will
- **clear-counters** - Clears all counters in FM (configured in OPA config)
- **help, -h** - Displays this usage information

For more information, please read the README: /usr/local/hsnmon/README
For license information, please read the License: /usr/local/hsnmon/License

*Above output for “service” on RHEL6, will be modified for SystemDB*
Challenges:

• HSNmon takes ~1 minute to run on target clusters

• Underlying FastFabric tools are too slow on target clusters
  • (opaextractperf takes >10s to run)

• FastFabric does not provide the ability to gather both port counters and an accurate “sweep time” from PM at speed

• Syslog fails on target clusters with performance counter volume
 Challenges:

- HSNmon takes ~1 minute to run on target clusters
  - spawn separate process for high frequency counter gathering

- Underlying FastFabric tools are too slow on target clusters
  - (opaextractperf takes >10s to run)

- FastFabric does not provide the ability to gather both port counters and an accurate “sweep time” from PM at speed

- Syslog fails on target clusters with performance counter volume
HSNMON MODIFICATIONS

Challenges:

• HSNmon takes ~1 minute to run on target clusters
  - spawn separate process for high frequency counter gathering

• Underlying FastFabric tools are too slow on target clusters
  • (opaextractperf takes >10s to run)
    - write our own tool using OPAMGT API to directly query PA

• FastFabric does not provide the ability to gather both port counters and an accurate “sweep time” from PM at speed

• Syslog fails on target clusters with performance counter volume
Challenges:

- HSNmon takes ~1 minute to run on target clusters
  - spawn separate process for high frequency counter gathering

- Underlying FastFabric tools are too slow on target clusters
  - `opaextractperf` takes >10s to run
  - write our own tool using OPAMGT API to directly query PA

- FastFabric does not provide the ability to gather both port counters and an accurate “sweep time” from PM at speed
  - design the tool to collect the sweep time and sweep duration from PA

- Syslog fails on target clusters with performance counter volume
HSNMON MODIFICATIONS

Challenges:

• HSNmon takes ~1 minute to run on target clusters
  - spawn separate process for high frequency counter gathering

• Underlying FastFabric tools are too slow on target clusters
  • (opaextractperf takes >10s to run)
    - write our own tool using OPAMGT API to directly query PA

• FastFabric does not provide the ability to gather both port counters and an accurate “sweep time” from PM at speed
  - design the tool to collect the sweep time and sweep duration from PA

• Syslog fails on target clusters with performance counter volume
  - Use AMQP to send data to destination for analysis
• get_port_counters.c
  – Uses OPAMGT API to poll for counters and accurate counter time

• pipe_to_rmq.py
  – Sends stream or file via AMQP using Pika

• hsn_send_if_diff.sh
  – Executes argument, saves output to file and sends file if different from last run (used to send LFT)
NEW HSNMON CODE

Usage:

get_port_counters [poll frequency in seconds]

Sample Output

Node="edge1" Port=38 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=30197 RcvData=33089
Node="edge1" Port=39 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=363279 RcvData=364887
Node="edge1" Port=40 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=351113 RcvData=350227
Node="edge1" Port=41 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=370357 RcvData=369490
Node="edge1" Port=42 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=337909 RcvData=337060
Node="edge1" Port=43 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=351113 RcvData=350227
Node="edge1" Port=44 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=370357 RcvData=369490
Node="edge1" Port=45 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=337909 RcvData=337060
Node="edge1" Port=46 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=364560 RcvData=363670
Node="edge1" Port=47 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=30197 RcvData=33089
Node="edge1" Port=48 Image_Start="Mon Mar 11 10:24:54 2019" Image_Duration=0.108s Image_ID=0x800003b40d110000 XmitData=348041 RcvData=351410
Duplicate image [0x800003b40d110000] skipping...
Duplicate image [0x800003b40d110000] skipping...
Duplicate image [0x800003b40d110000] skipping...
Duplicate image [0x800003b40d110000] skipping...
Usage:

get_port_counters [poll frequency in seconds]

Sample Output

Node="kit-master hfi1_0" Port=1 Image_Start="Mon Mar 11 10:47:19 2019" Image_Duration=0.003s Image_ID=0x800003b40d110000 XmitData=666717582633873344 RcvData=0
Node="kit-fe hfi1_0" Port=1 Image_Start="Mon Mar 11 10:47:19 2019" Image_Duration=0.003s Image_ID=0x800003b40d110000 XmitData=666717582633873344 RcvData=0
Node="kit-iol hfi1_0" Port=1 Image_Start="Mon Mar 11 10:47:19 2019" Image_Duration=0.003s Image_ID=0x800003b40d110000 XmitData=666717582633873344 RcvData=0
...
Node="core1" Port=40 Image_Start="Mon Mar 11 10:46:19 2019" Image_Duration=0.002s Image_ID=0x800003b407110004 XmitData=666717556864083584 RcvData=0
Node="core1" Port=41 Image_Start="Mon Mar 11 10:46:19 2019" Image_Duration=0.002s Image_ID=0x800003b407110004 XmitData=666717556864083584 RcvData=0
Node="core1" Port=42 Image_Start="Mon Mar 11 10:46:19 2019" Image_Duration=0.002s Image_ID=0x800003b407110004 XmitData=666717556864083584 RcvData=0
...

Duplicate image [0x800003b40d110000] skipping...
Duplicate image [0x800003b40d110000] skipping...
Duplicate image [0x800003b40d110000] skipping...
Duplicate image [0x800003b40d110000] skipping...

Non-counter data is written to stderr for easy parsing
HSNMON MODIFICATIONS

Flow Chart

- **Start (hsnmon)**
- **Load config (Otherwise Default)**
- **Pause? Not Master?**
  - **No**
    - **DST?**
      - **No**
        - **Generate hsnnet_map**
        - **Clean log directory**
        - **Generate host List**
      - **Yes**
        - **Enable DST Mode**
        - **Save Linear Forwarding Table**
  - **Yes**
    - **Generate switch list (Look for missing switches)**
    - **Pull FM counters/clear**
    - **Perfscope?**
      - **Yes**
        - **Fork() -> “perfscope” process**
      - **No**
        - **Call hsn_rosetta.pl**
        - **Gather/report errors by link**
        - **Gather/report performance counters by port**

- **Sleep**
HSNMON MODIFICATIONS

Flow Chart

Start (hsnmon) → Load config (Otherwise Default) → hsnmon.pl → Paused? Not Master?

- Yes: Generate hsnnet_map → Clean log directory → Generate host List → Save Linear Forwarding Table
- No: DST?
  - Yes: Enable DST Mode
  - No: Gather/link analysis report → Gather/report errors by link

Sleep

DST?

- Yes: Fork() -> execute “perfscope” process
- No: Generate switch list (Look for missing switches) → Pull FM counters/clear → Call hsn_rosetta.pl → Gather/report performance counters by port

Perfscope?

- Yes: New
- No: Gather/report performance counters by port

Has Linear Forwarding Table?

- Yes: Fork() -> execute “perfscope” process
- No: Sleep

Note: DST? checks for DST mode, and if not, it continues with other processes. The flowchart includes decision points for various conditions, such as whether the process is paused or not, if it is not the master, and if there is a DST (Destiny) mode.
NEW HSNMON DATA

*legend not shown
FUTURE WORK
FUTURE WORK

HSNMON Work:
- Continue deploying on clusters
- Find way to track when PM sweep times out
  - data integrity
- Make open source

Applications To InfiniBand:
- Apply lessons learned to IBmon
  - Share new HSNmon scripts (if they are helpful)
- Look into making an OpenSM Perfmanager Event Plugin
THANK YOU

Brett Holman, Jesse Martinez

Los Alamos National Laboratory