OPENFABRICS ALLIANCE

12th ANNUAL WORKSHOP 2016

MONITORING HIGH SPEED NETWORK FABRICS: EXPERIENCES AND NEEDS

Jim Brandt, R&D Staff

Gentile, Allan, Lefantzi, and Aguilar

Sandia National Laboratories

[April 7th, 2016]

Sandia National Laboratories

Exceptional service in the national interest

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2013-6521C

OVERVIEW

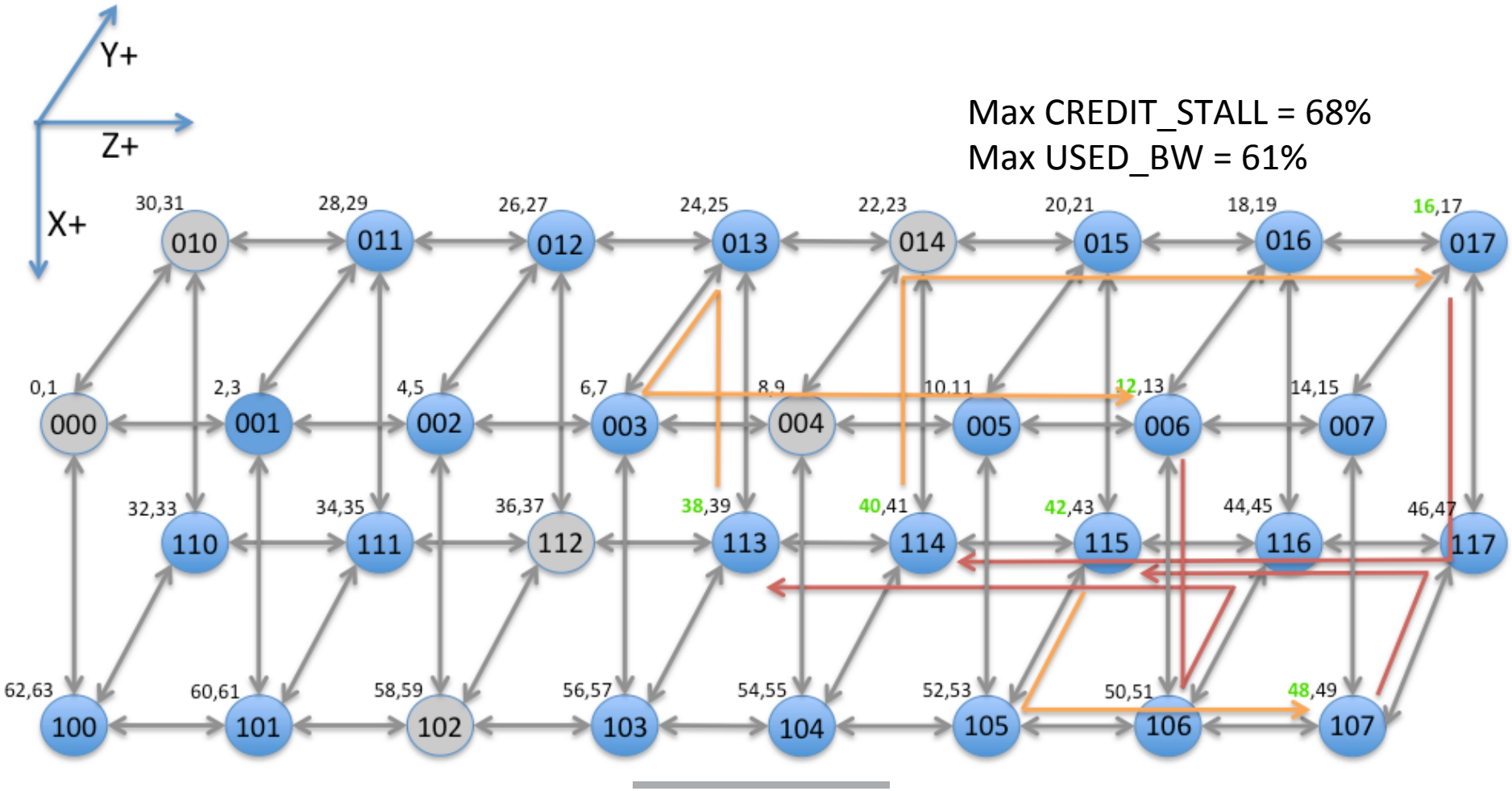
- **Motivation**
  - Understand utilization
  - Understand congestion
  - Build better tools for application <-> system interaction
  - Build more network aware schedulers and resource managers
- **Current Approaches**
  - Data collection
    - In-band vs. out-of-band
    - Synchronous
    - Fidelity
    - Challenges
  - Analysis
    - Scoring at edges
    - Scoring of core or links in core
    - Challenges
  - Visualization
    - Topology relevant layout
    - Challenges
- **Technology specific examples**
    - Infiniband
    - Cray Gemini
    - Cray Aries
    - Intel OmniPath
- **Needs: Information, APIs, tools, and methodologies**

MOTIVATION

MOTIVATION

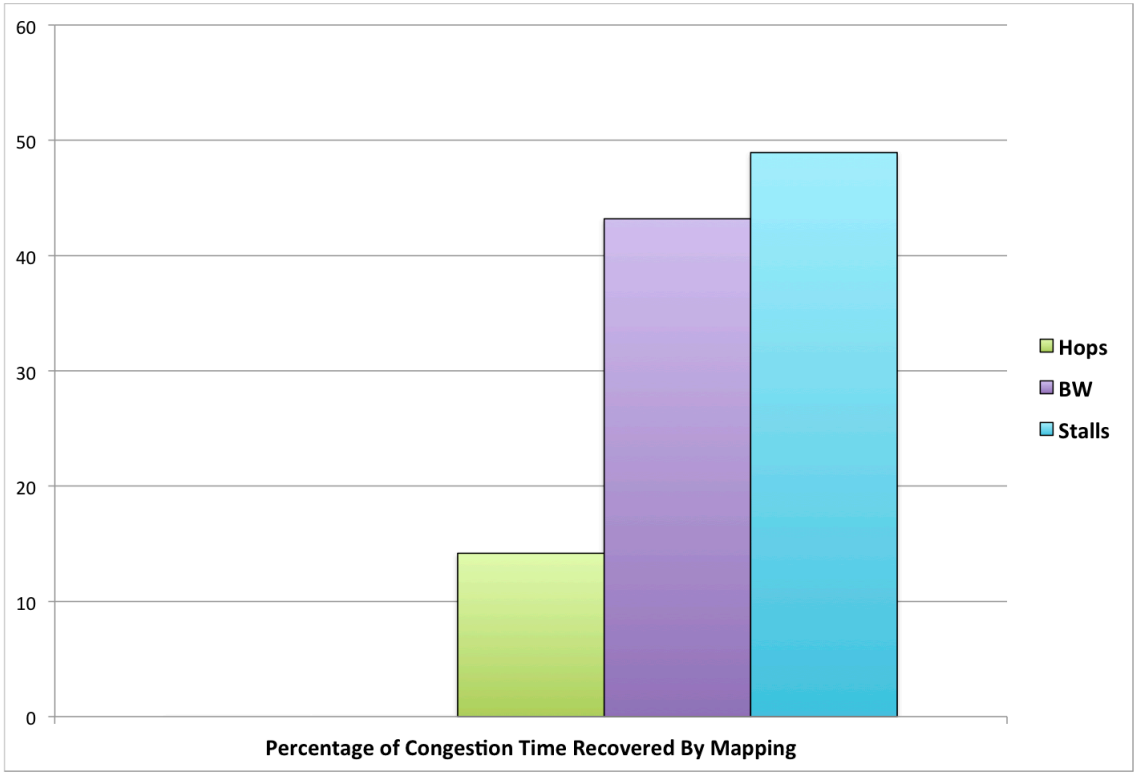
- **Understand utilization**
  - Match fabrics to workloads in acquisitions
  - Match workloads to fabrics in application runs via intelligent scheduling and resource allocation
- **Understand congestion**
  - How well are network hot spots detected and mitigated via dynamic routing algorithms
  - How long lived are hot spots
  - Can we utilize a combination of network and application characteristic aware scheduling and resource allocation, dynamic feedback to applications, and dynamic routing to mitigate the effects of contention for network resources
- **Better scheduling/resource allocation**
  - Require schedulers and resource managers to be network technology/topology and application network behavioral characteristic aware – How?
  - Require allocation-time feedback about network state – perhaps continuous run-time feedback with trending
  - Require either allocation time application information or repository of user-app-input bindings to reference
    - Utility depends on application characteristics and level of behavioral characteristic variability with user and input deck.

COMPETING APPLICATION WITH NETWORK TRAFFIC DEMANDS



DYNAMIC MAPPING RESULTS

- **Percentage Execution Time Recovered by Performing Mapping with Various Metrics (higher is better)**



Remapping based on dynamic network information in a congested environment recovered ~50% of the time lost to congestion.

INFORMATION NEEDED EVEN WHEN HARDWARE IS HEALTHY

- **Injection/Ejection traffic**
- **Backpressure at the host**
  - Send – backpressure from network
  - Receive – host is unable to ingest data fast enough (many to one)
- **When, where, and for how long congestion exists in the network**
  - How well are adaptive routing algorithms working?
  - Should scheduler/resource manager be smarter (application/topology aware) about allocation?
    - Slab allocation on Blue Waters to minimize inter-job contention
  - Can low latency feedback to applications enable better communication strategies?
    - Problem decomposition and partitioning
    - Dynamic response to congestion
  - Are our networks appropriately provisioned for our workloads?
  - Can we correlate application run-time fluctuation with level of network congestion?
  - How does degraded link performance affect overall congestion?
  - Does number and placement of LNET routers adversely impact the network (i.e., cause significant congestion)

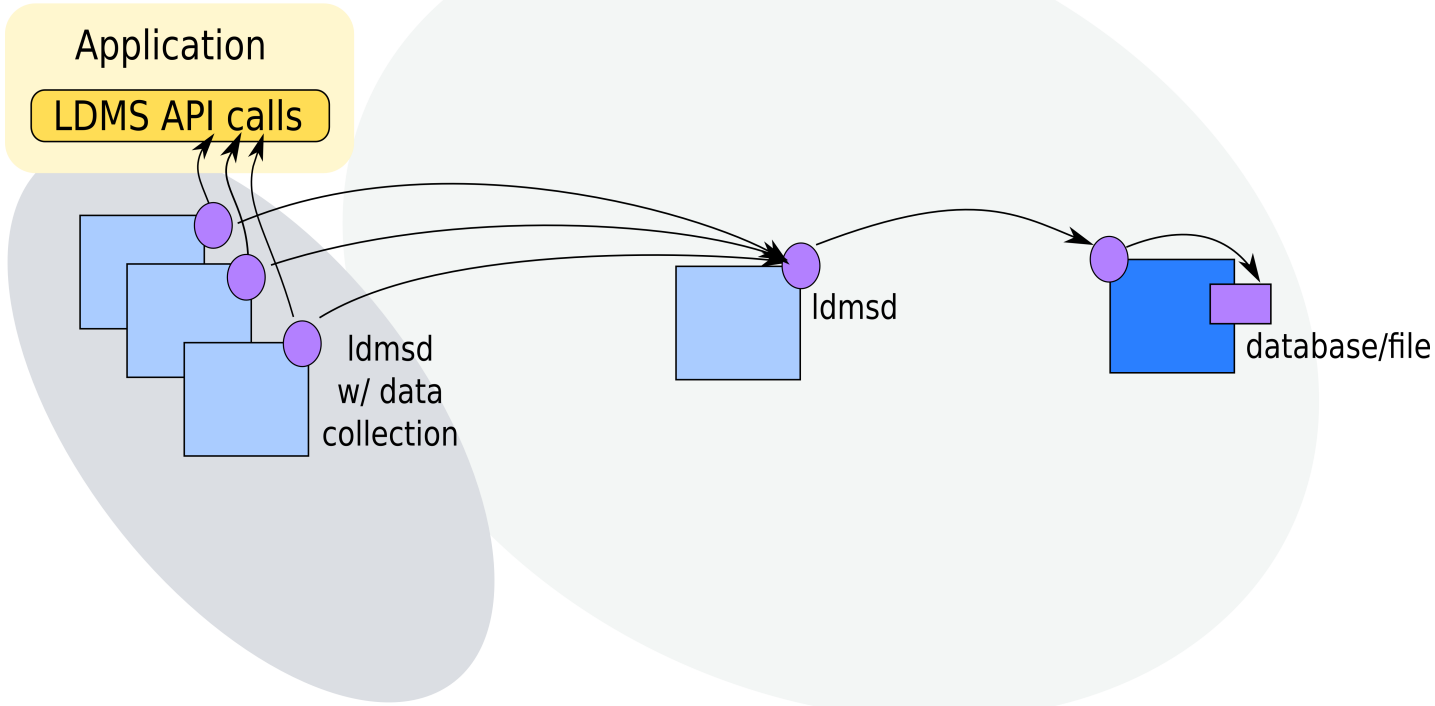
CURRENT APPROACHES

OVERVIEW OF CURRENT APPROACHES

- **Data collection**
  - In-band – active sampling of network switch/router port counters from hosts
    - Better scalability (higher fidelity?)
  - Out-of-band – interaction with subnet/fabric manager to get information
    - Preferred if acceptable fidelity
    - Proof of concept ported of data collection framework to Mellanox IB switch
  - Synchronous – provides a series of “system snapshots”
  - Fidelity – seconds to minutes -- defines lower bound on feedback latency
  - ~1 second collection periods and analysis of bandwidth at edges (hosts)
    - Synchronized
  - ~Minutes collection periods of core fabric port counters via Subnet Manager (SM)
    - Don’t know time skew across collection
  - Challenges
    - High fidelity with no/low impact on application run times due to host and/or network overhead/noise
    - Useful streaming analysis and feedback
    - Storage of large volumes of data for post processing
- **Analysis**
  - Calculate
    - Percentages of network network bandwidth used (job, machine)
    - Network cycles used transmitting data (as opposed to idle/stalled)
  - Challenges
    - Meaning of data and appropriate functional combinations
    - Application attribution
- **Visualization**
  - Color maps overlaid on machine room or network topology layouts
  - Challenges
    - Topology relevant layouts

IN BAND COLLECTION USING LIGHTWEIGHT DISTRIBUTED METRIC SERVICE (LDMS)

LDMS HIGH LEVEL TOPOLOGY VIEW



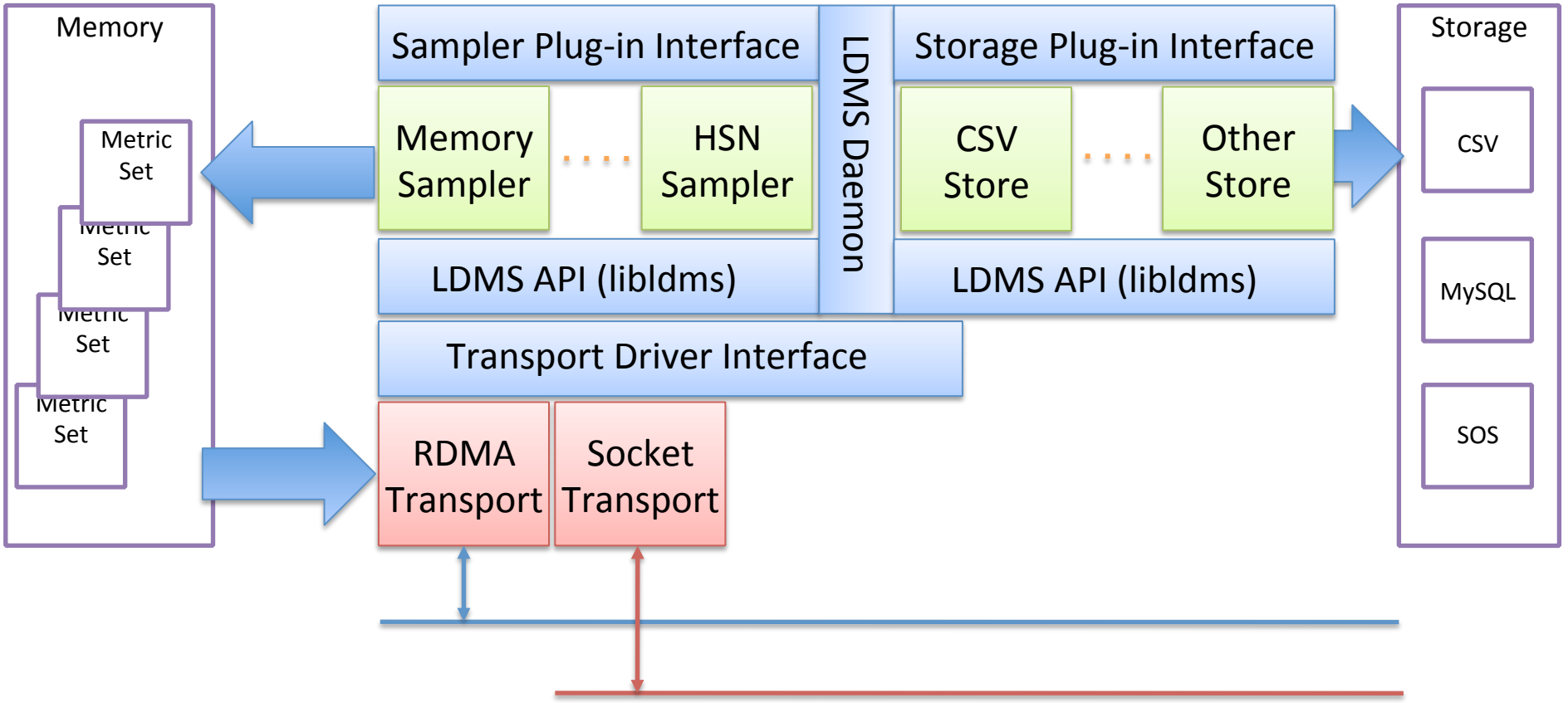
Compute Nodes

Aggregation Node(s)

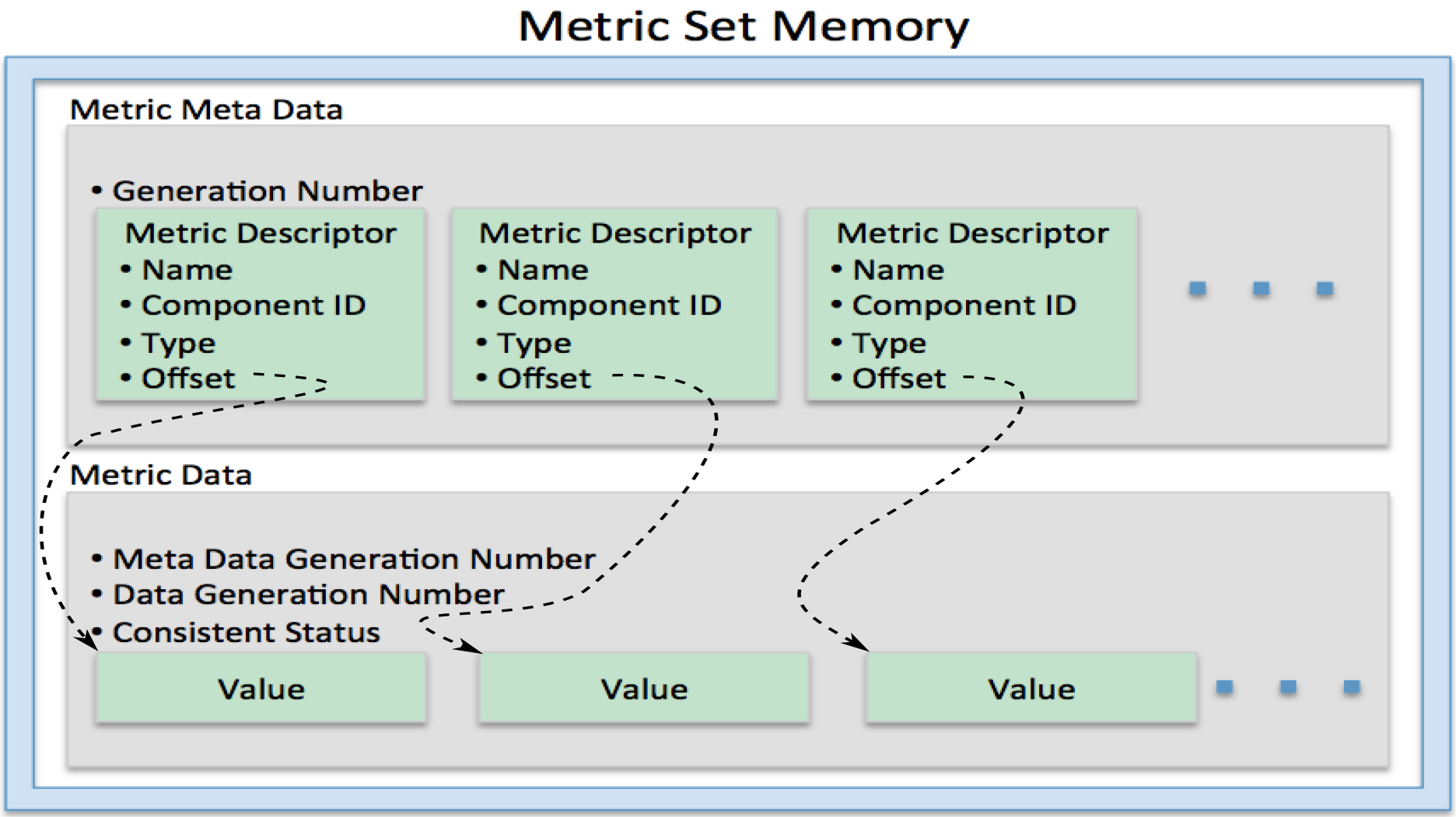
Storage Node(s)

* Only current data is retained in host memory

LDMS PLUGIN ARCHITECTURE



LDMS SET: MEMORY ORGANIZATION



TECHNOLOGY SPECIFIC EXAMPLES

INFINIBAND

- **Currently only gather high fidelity (~1Hz) information at the hosts**
  - Using Lightweight Distributed Metric Service (LDMS) IB sampler – Issues MAD queries to local HCA for extended port counters
  - Host overhead -- ~400usec to collect 22 metrics and compute rates
- **Currently investigating distributed (across compute hosts) switch port query**
  - Divide port queries for entire fabric across compute nodes for whole system high fidelity monitoring
    - Scripts for InfiniBand Monitoring Blake Caldwell April 30, 2015
      - Uses perfquery to get counters of interest along a jobs communication path and distributes across the nodes of the job
  - Sampling Overhead?
    - Host?
    - Switch?
- **Fabric description:**
  - Iblinkinfo, ibnetdiscover

```
vendid=0x2c9
devid=0xbd36
sysimgguid=0x2c9020044816b
switchguid=0x2c90200448168(2c90200448168)
Switch    36 "S-0002c90200448168"              # "Infiniscale-IV Mellanox Technologies" base port 0 lid 5 lmc 0
[1]       "S-0002c902004481e0"[1]              # "Infiniscale-IV Mellanox Technologies" lid 6 4xQDR
[2]       "H-0002c903004cb94c"[1](2c903004cb94d)              # "compton1 HCA-1" lid 39 4xQDR
```

- **Topology: Fat tree, 3D Torus, Hypercube**
- **Adaptive routing: Yes**

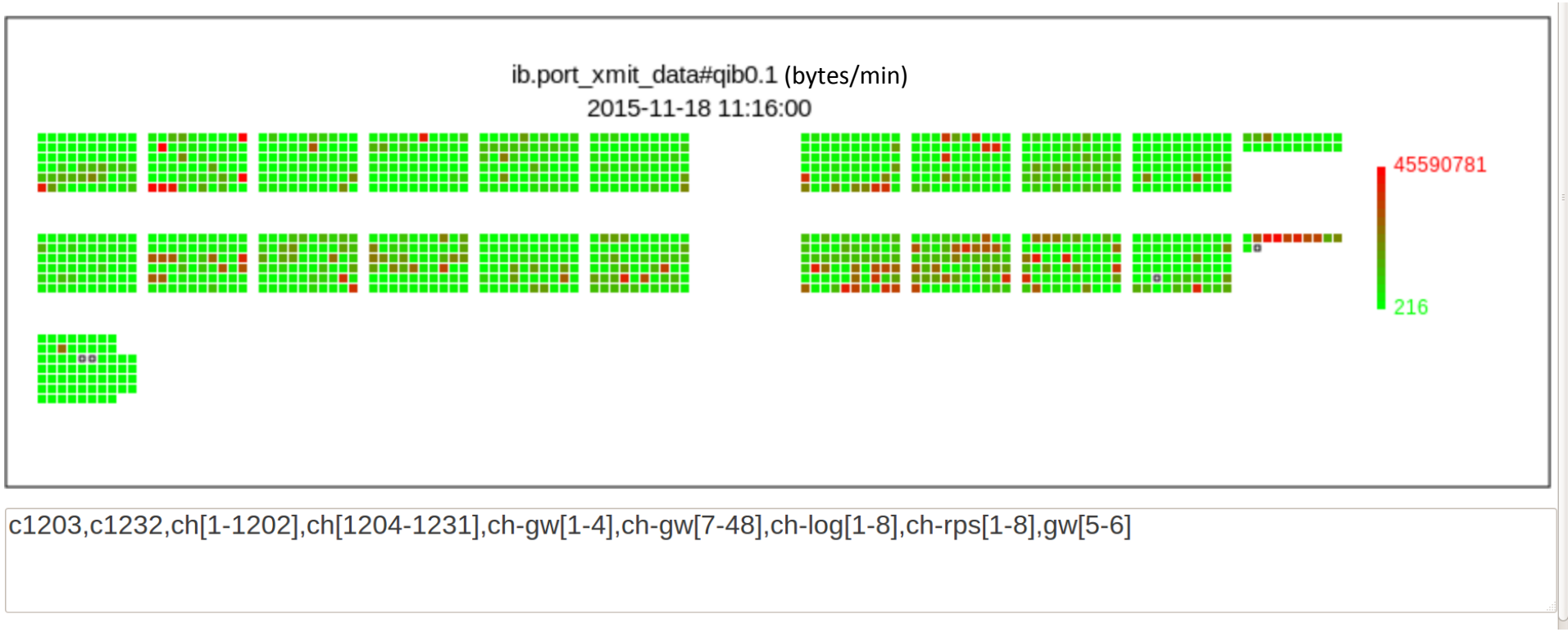
HOST BASED IB METRIC SET

symbol_error
link_error_recovery
link_downed
port_rcv_errors
port_rcv_remote_physical_errors
port_rcv_switch_relay_errors
port_xmit_discards
port_xmit_constraint_errors
port_rcv_constraint_errors
local_link_integrity_errors
excessive_buffer_overrun_errors
VL15_dropped
port_xmit_data
port_rcv_data
port_xmit_packets
port_rcv_packets
port_xmit_wait
port_unicast_xmit_packets
port_unicast_rcv_packets
port_multicast_xmit_packets
port_multicast_rcv_packets

SWITCH BASED IB METRIC SET

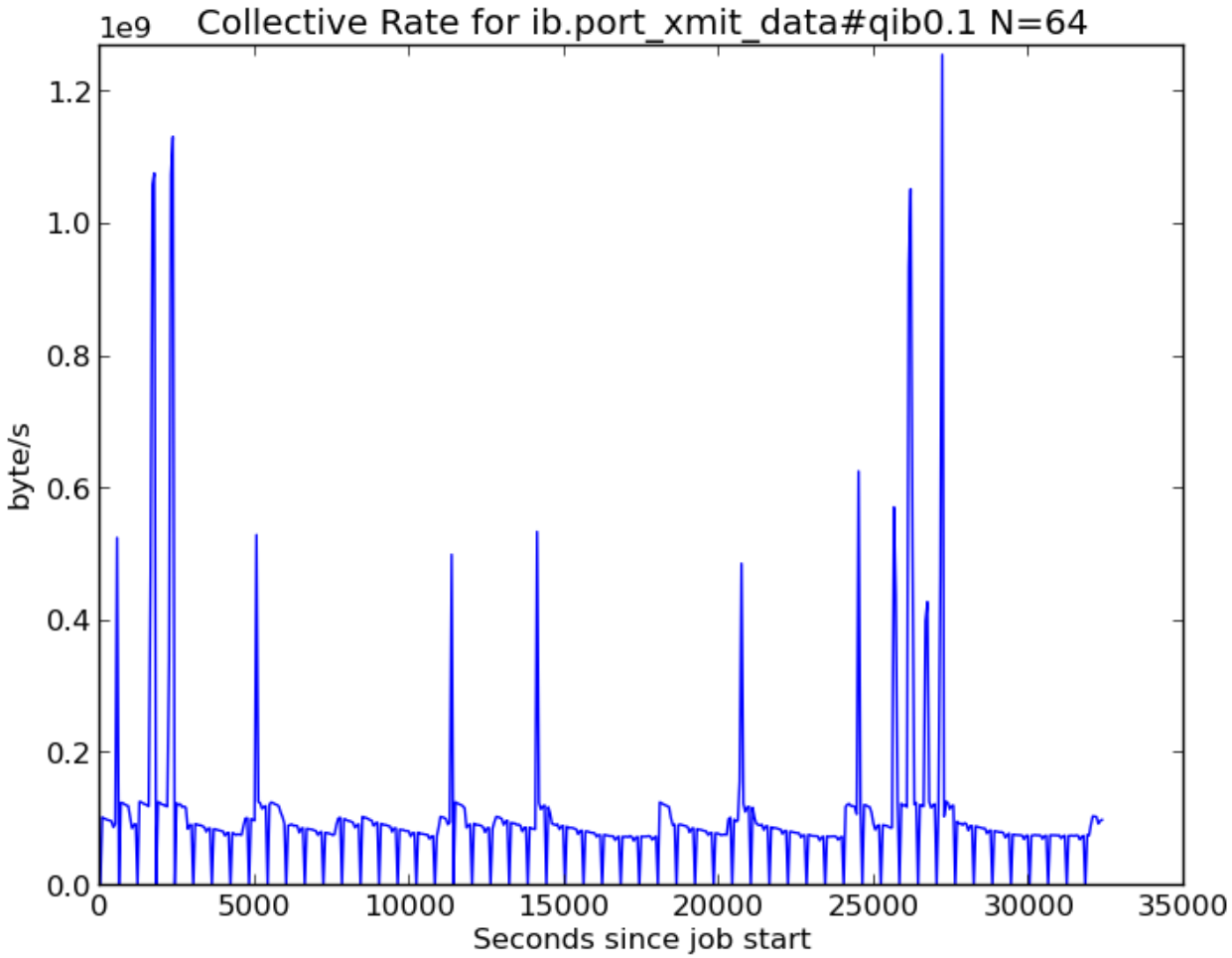
- SymbolErrorCounter
- LinkErrorRecoveryCounter
- LinkDownedCounter
- PortRcvErrors
- PortRcvRemotePhysicalErrors
- PortRcvSwitchRelayErrors
- PortXmitDiscards
- PortXmitConstraintErrors
- PortRcvConstraintErrors
- LocalLinkIntegrityErrors
- ExcessiveBufferOverrunErrors
- VL15Dropped
- PortXmitData
- PortRcvData
- PortXmitPkts
- PortRcvPkts
- PortXmitWait
- PortXmitDataExt
- PortRcvDataExt
- PortXmitPktsExt
- PortRcvPktsExt
- PortUnicastXmitPktsExt
- PortUnicastRcvPktsExt
- PortMulticastXmitPktsExt
- PortMulticastRcvPktsExt
- LinkWidthEnabled
- LinkWidthSupported
- LinkWidthActive
- LinkSpeedSupported
- LinkState
- PhysLinkState
- LinkDownDefState
- ProtectBits
- LinkSpeedActive
- LinkSpeedEnabled
- NeighborMTU
- VLCap
- VLHighLimit
- VLStallCount
- PartEnforceInb
- PartEnforceOutb
- FilterRawInb
- FilterRawOutb
- MkeyViolations
- PkeyViolations
- QkeyViolations
- ClientReregister
- LocalPhysErr
- OverrunErr

HOST BASED VIEW OF IB TRAFFIC



- **Host based view of Infiniband transmit activity (bytes/min)**
  - No core based views due to lack of core switch port counter sampling

JOB IB PORT TRANSMIT PLOT

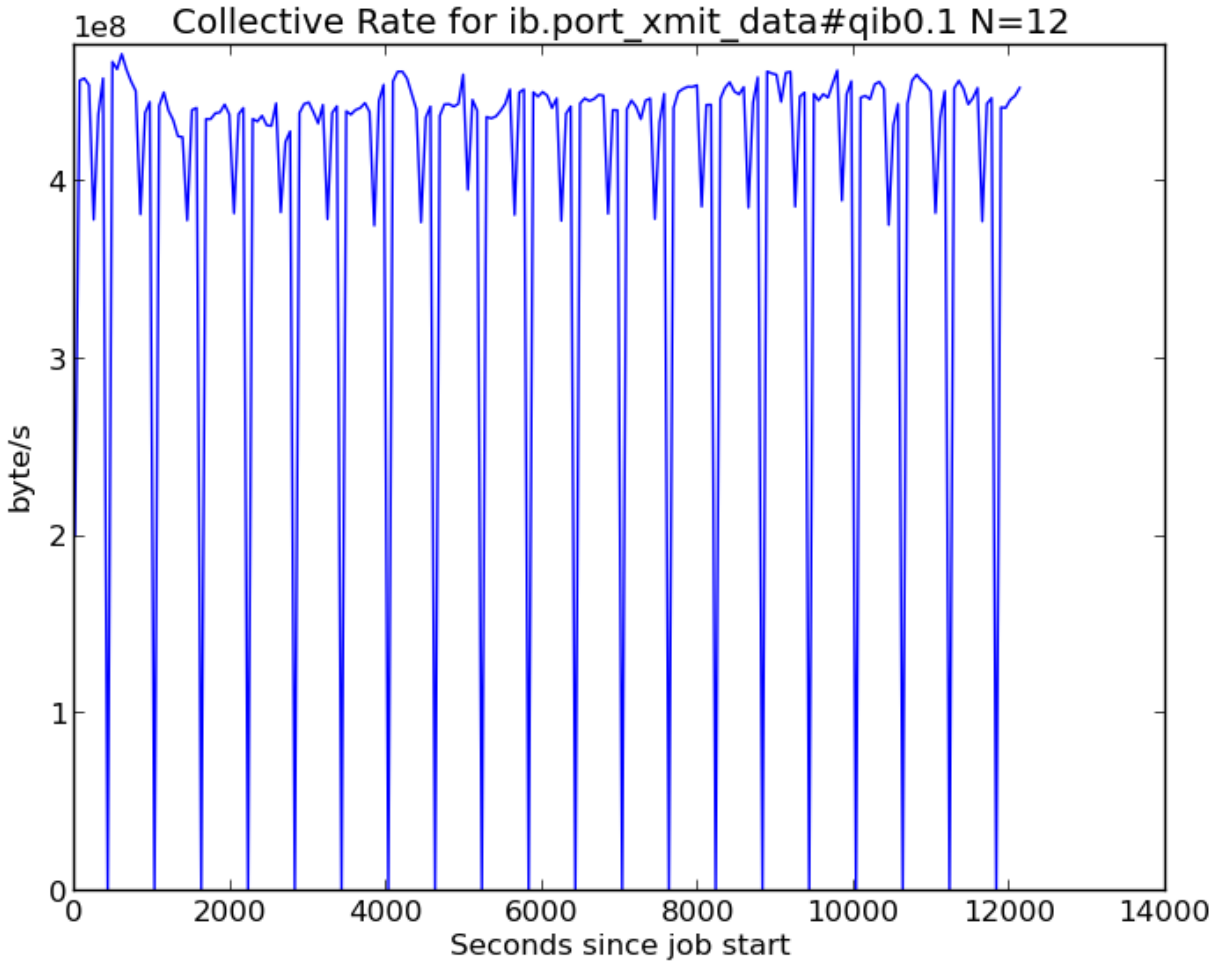


- Data collected at 60 second intervals.
- Spikes to 0 due to counter resets every 600 seconds.

JOB IB PORT TRANSMIT SUMMARY

```
Summary for ib.port_xmit_data#qib0.1 of job.011287899/sysclassib.1450249200 N=64
-------------------------------------------------------------------------------------------
activity_score  10      Fraction of intervals where derivative of the metric is nonzero (scaled to 1-10)
peak_score      1       Fraction of local upper bound of largest single-node value (scaled to 1-10)
usage_score     1       Fraction of available rate (scaled to 1-10)
balance_score   10      Node-to-node similarity of time-averaged usage (nonlinear scale; 10=similar, 1=wildly different)
-------------------------------------------------------------------------------------------
activity_pct    90.0    100*nonzero count / interval count
peak_pct        0.59261123726   100 * local max / local upper bound
usage_pct       9.90999541122   100 * resource-seconds used / resource-seconds available
sigma_pct       0.000346035391978       100*stddev(per-node-resource-seconds)/(total resource-seconds used)
-------------------------------------------------------------------------------------------
gdifference     Total usage     3.4539960677e+12        byte
grate_min       Min. single node rate   0.0     byte/s
grate_max       Max. single node rate   25452458.8327   byte/s
grate_avg       Avg. rate per node-second       1850778.0684    byte/s
grate_avgnz     Avg. nonzero delta per node-interval    99942015.8478   byte
grate_minnz     Min. nonzero rate per node      941697.504249   byte/s
grate_minsum    Min. sum across nodes in any interval   0.0     byte/s
grate_maxsum    Max. sum across nodes in any interval   1257071419.7    byte/s
grate_time      Sum of time intervals accounted.        2073599.9705    seconds
grate_time_pct_missing, %-age node-wall_time unaccounted        0.0
ublocal Per-node upper bound on rate    4000000000.0byte/s
collmax Cluster collective upper bound on rate  256000000000.0byte/s
-------------------------------------------------------------------------------------------
```

JOB IB PORT TRANSMIT PLOT

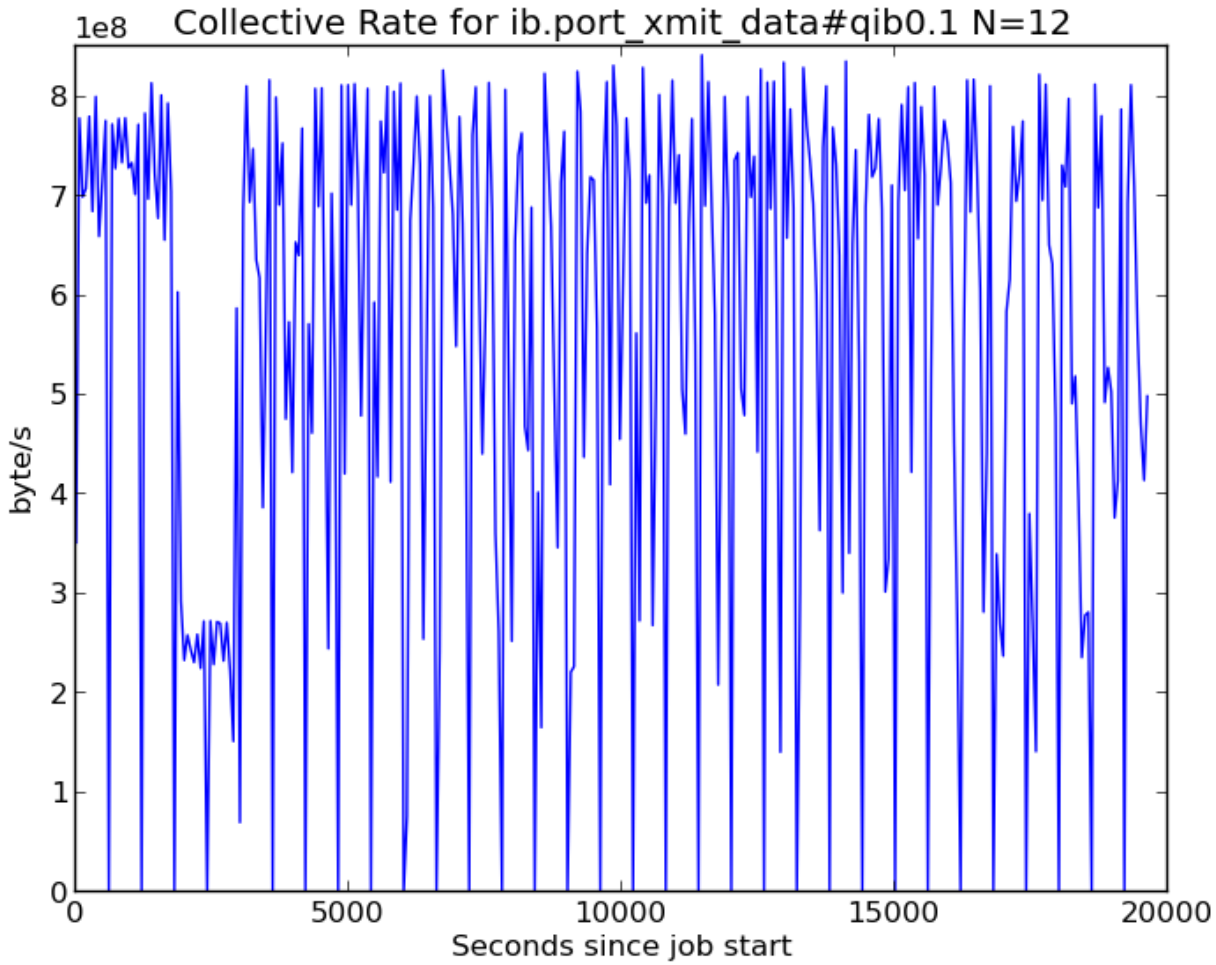


- Data collected at 60 second intervals.
- Spikes to 0 due to counter resets every 600 seconds.

JOB IB PORT TRANSMIT SUMMARY

```
Summary for ib.port_xmit_data#qib0.1 of job.011317645/sysclassib.1450249200 N=12
-------------------------------------------------------------------------------------
activity_score  10      Fraction of intervals where derivative of the metric is nonzero (scaled to 1-10)
peak_score      1       Fraction of local upper bound of largest single-node value (scaled to 1-10)
usage_score     1       Fraction of available rate (scaled to 1-10)
balance_score   10      Node-to-node similarity of time-averaged usage (nonlinear scale; 10=similar, 1=wildly different)
-------------------------------------------------------------------------------------
activity_pct    90.1193906958   100*nonzero count / interval count
peak_pct        1.5978768081    100 * local max / local upper bound
usage_pct       36.6136607266   100 * resource-seconds used / resource-seconds available
sigma_pct       0.017951031908  100*stddev(per-node-resource-seconds)/(total resource-seconds used)
-------------------------------------------------------------------------------------
gdifference     Total usage     4.81198694748e+12       byte
grate_min       Min. single node rate   0.0     byte/s
grate_max       Max. single node rate   68628286.3384   byte/s
grate_avg       Avg. rate per node-second       36637637.2753   byte/s
grate_avgnz     Avg. nonzero delta per node-interval    1981073555.76   byte
grate_minnz     Min. nonzero rate per node      7537627.74321   byte/s
grate_minsum    Min. sum across nodes in any interval   0.0     byte/s
grate_maxsum    Max. sum across nodes in any interval   472109238.8     byte/s
grate_time      Sum of time intervals accounted.        145739.984478   seconds
grate_time_pct_missing, %-age node-wall_time unaccounted        0.285976522304
ublocal Per-node upper bound on rate    4000000000.0byte/s
collmax Cluster collective upper bound on rate  48000000000.0byte/s
-------------------------------------------------------------------------------------
```

JOB IB PORT TRANSMIT PLOT



- Data collected at 60 second intervals.
- Spikes to 0 due to counter resets every 600 seconds.

JOB IB PORT TRANSMIT SUMMARY

```
Summary for ib.port_xmit_data#qib0.1 of job.011317652/sysclassib.1450249200 N=12
-------------------------------------------------------------------------------------
activity_score  10      Fraction of intervals where derivative of the metric is nonzero (scaled to 1-10)
peak_score      1       Fraction of local upper bound of largest single-node value (scaled to 1-10)
usage_score     2       Fraction of available rate (scaled to 1-10)
balance_score   10      Node-to-node similarity of time-averaged usage (nonlinear scale; 10=similar, 1=wildly different)
-------------------------------------------------------------------------------------
activity_pct    90.4447268107   100*nonzero count / interval count
peak_pct        1.77198176537   100 * local max / local upper bound
usage_pct       51.252822484    100 * resource-seconds used / resource-seconds available
sigma_pct       0.00187951785676        100*stddev(per-node-resource-seconds)/(total resource-seconds used)
-------------------------------------------------------------------------------------
gdifference     Total usage     1.08633758422e+13       byte
grate_min       Min. single node rate   0.0     byte/s
grate_max       Max. single node rate   76106037.3138   byte/s
grate_avg       Avg. rate per node-second       50872787.4735   byte/s
grate_avgnz     Avg. nonzero delta per node-interval    2760425830.04   byte
grate_minnz     Min. nonzero rate per node      824.780940842   byte/s
grate_minsum    Min. sum across nodes in any interval   0.0     byte/s
grate_maxsum    Max. sum across nodes in any interval   842545045.767   byte/s
grate_time      Sum of time intervals accounted.        236099.976741   seconds
grate_time_pct_missing, %-age node-wall_time unaccounted        0.0253331949082
ublocal Per-node upper bound on rate    4000000000.0byte/s
collmax Cluster collective upper bound on rate  48000000000.0byte/s
-------------------------------------------------------------------------------------
```

CRAY GEMINI

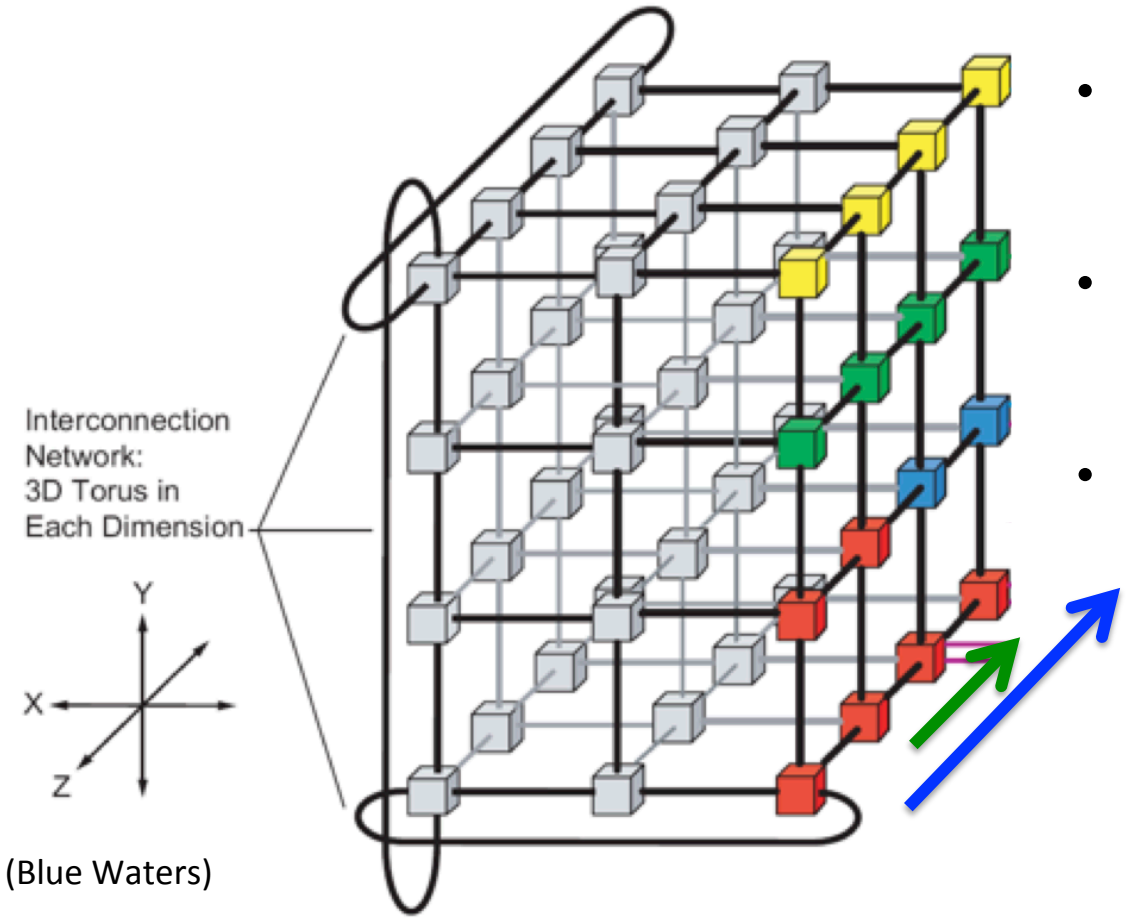
- **Performance counter API**
  - In-band only – no network fabric manager to query
- **Fidelity**
  - ~1 second to 1 minute sampling periods
- **Sampling overhead**
  - Host – ~200usec to collect 160 metrics distilled into 24 aggregates
    - Each of 2 hosts associated with a Gemini router chip collects all data
      - 2x redundancy in case a host goes down
  - Router – none?
- **Topology – 3D Torus**
- **Interconnect details**
  - On SMW: rtr –interconnect

    C0-0c0s0g0l00[(0,0,0)] Z+ -> c0-0c0s1g0l32[(0,0,1)] LinkType: backplane
    C0-0c0s0g0l01[(0,0,0)] Z+ -> c0-0c0s1g0l21[(0,0,1)] LinkType: backplane
    C0-0c0s0g0l02[(0,0,0)] X+ -> c0-0c1s0g0l02[(1,0,0)] LinkType: cable11x
    C0-0c0s0g0l03[(0,0,0)] X+ -> c0-0c1s0g0l03[(1,0,0)] LinkType: cable11x
    C0-0c0s0g0l05[(0,0,0)] X+ -> c0-0c1s0g0l05[(1,0,0)] LinkType: cable11x
    C0-0c0s0g0l06[(0,0,0)] Z- -> c0-0c0s7g0l26[(0,0,7)] LinkType: cable11z
    C0-0c0s0g0l07[(0,0,0)] Z- -> c0-0c0s7g0l35[(0,0,7)] LinkType: cable11z
- **Analysis and/or visualization tools available**
  - System software looks for threshold crossings and errors that are used to trigger network throttling or network quiesce events
  - None available to the user
  - Link aggregation and well defined static routing rules make this particularly easy to understand
- **Adaptive routing: No**

CURRENTLY COLLECTED PERFORMANCE COUNTER METRICS

X+- traffic (Bytes)
Y+- traffic (Bytes)
Z+- traffic (Bytes)
X+- packets (Num)
Y+- packets (Num)
Z+- packets (Num)
X+- credit_stall (ns)
Y+- credit_stall (ns)
Z+- credit_stall (ns)
X+- inq_stall (ns)
Y+- inq_stall (ns)
Z+- inq_stall (ns)
X+- sendlinkstatus (Num good sublinks)
Y+- sendlinkstatus (Num good sublinks)
Z+- sendlinkstatus (Num good sublinks)

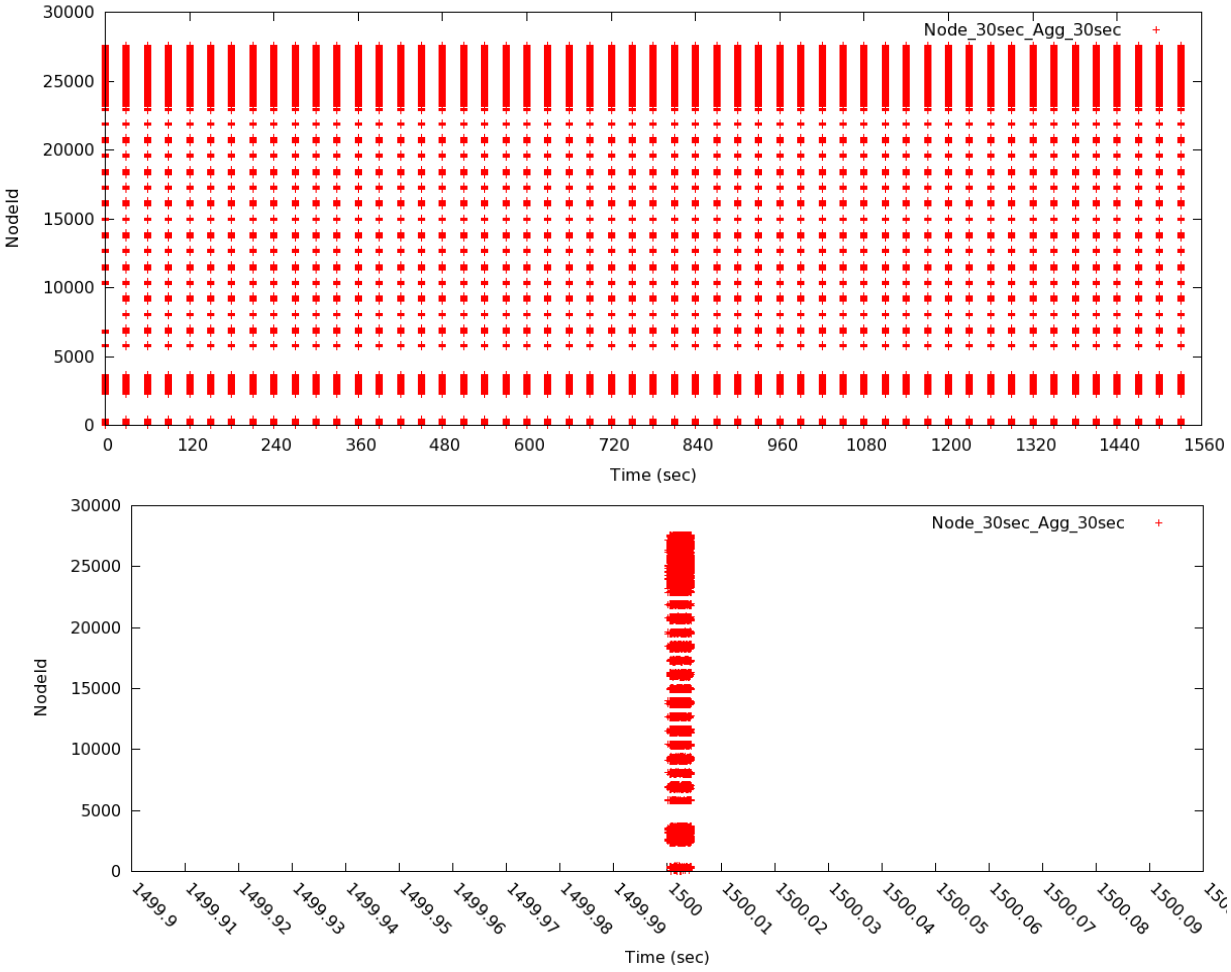
UNDERSTANDING NETWORK CONTENTION *REQUIRES* WHOLE SYSTEM MONITORING



(Blue Waters)

- An application may be impacted by the traffic of other applications.
- An application cannot get a measure of contention from its view alone.
- Note: 2 nodes share a Gemini router

SYNCHRONOUS COLLECTION



Synchronized collection across all nodes:

- Enables a coherent system snapshot

Asynchronous option spreads network load

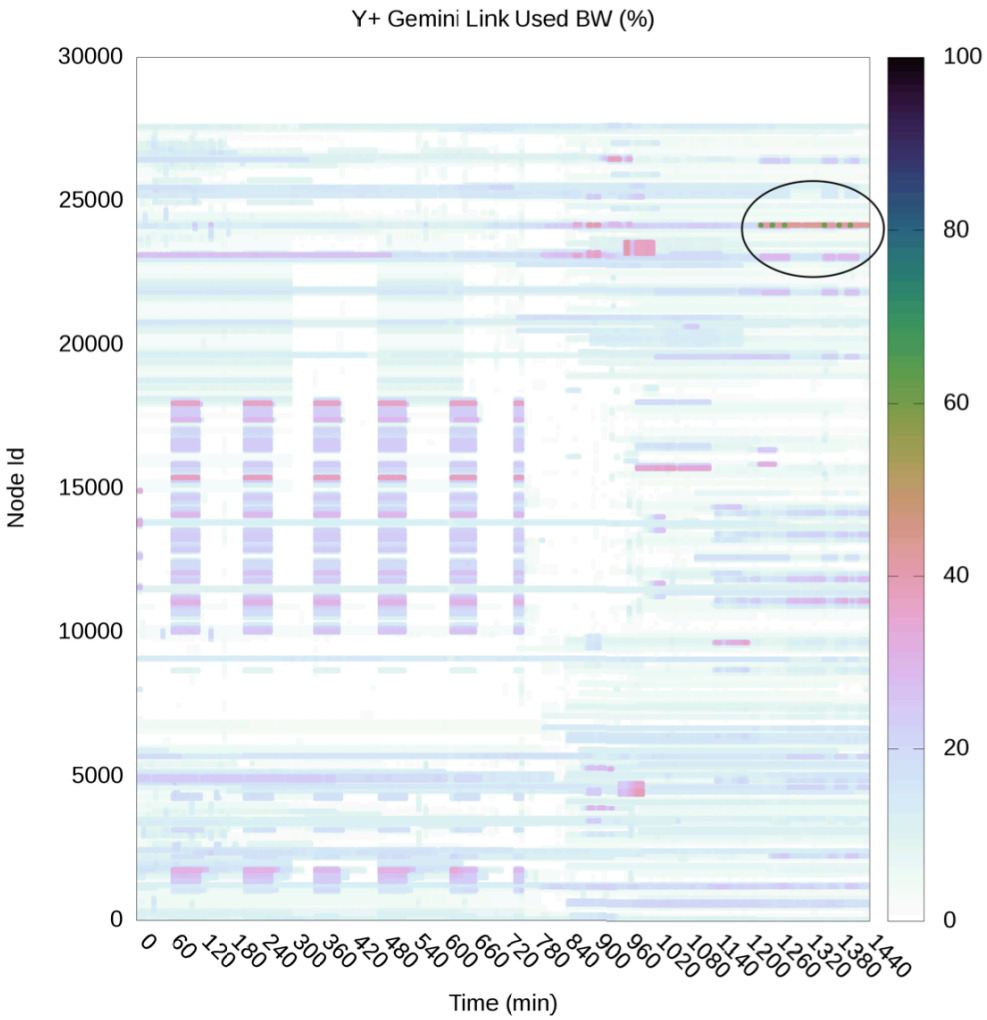
Synchronous:

- Variance in collection timestamps ~ 4ms

Note: Clock skew not accounted for

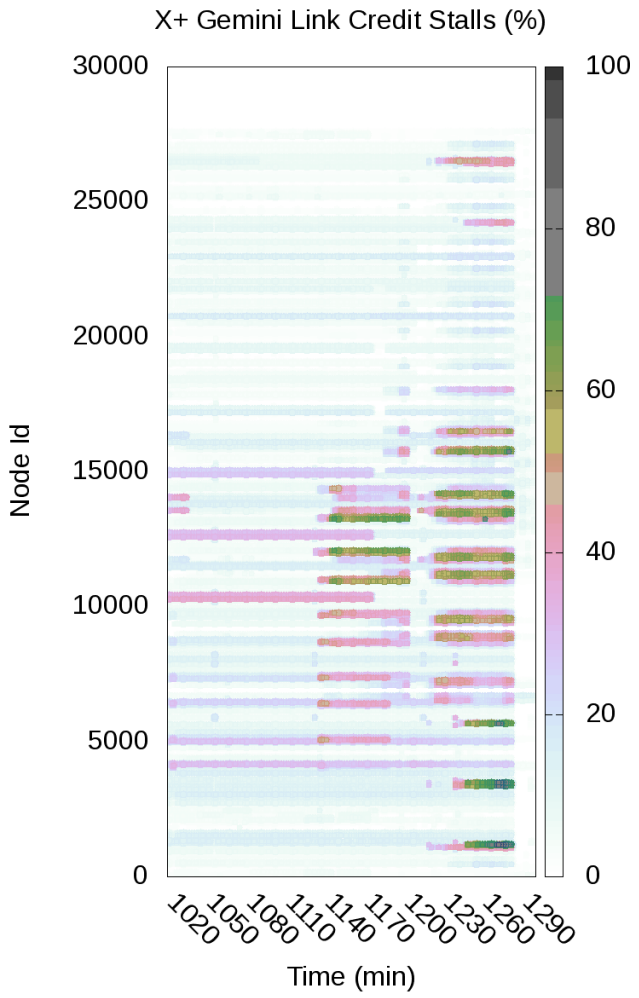
Collection occurrences over 10000 nodes on Blue Waters

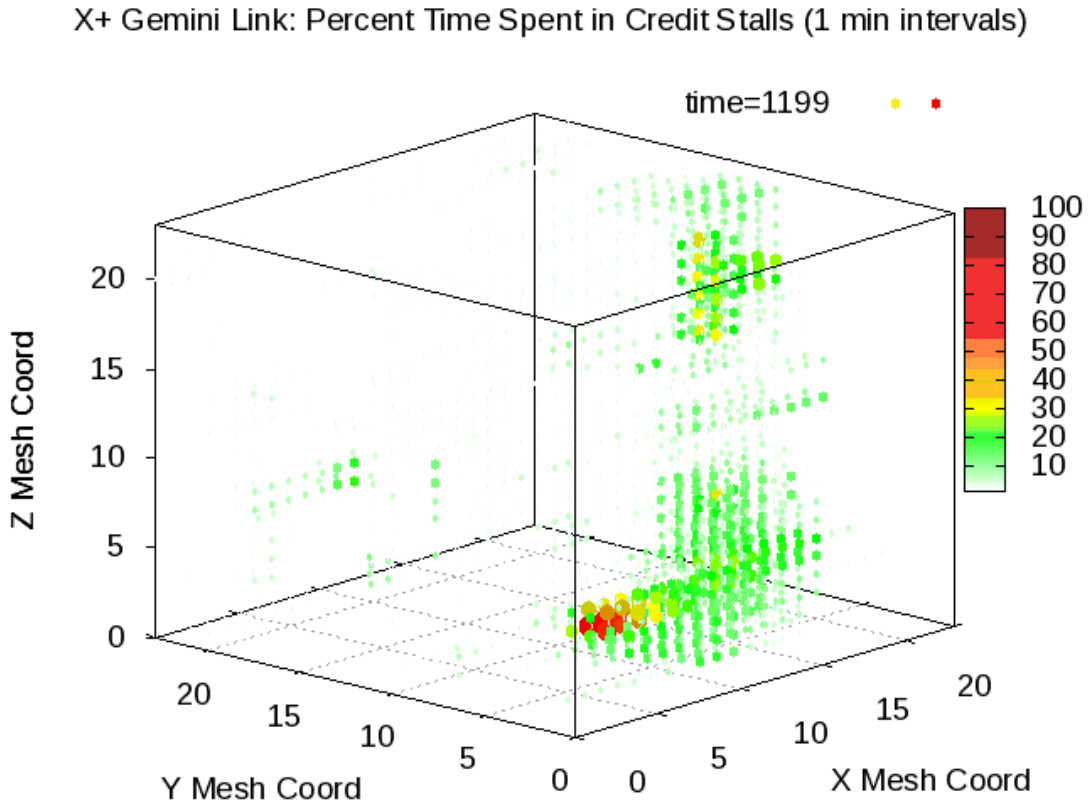
PERCENT USED BANDWIDTH – NODE VIEW (BLUE WATERS)



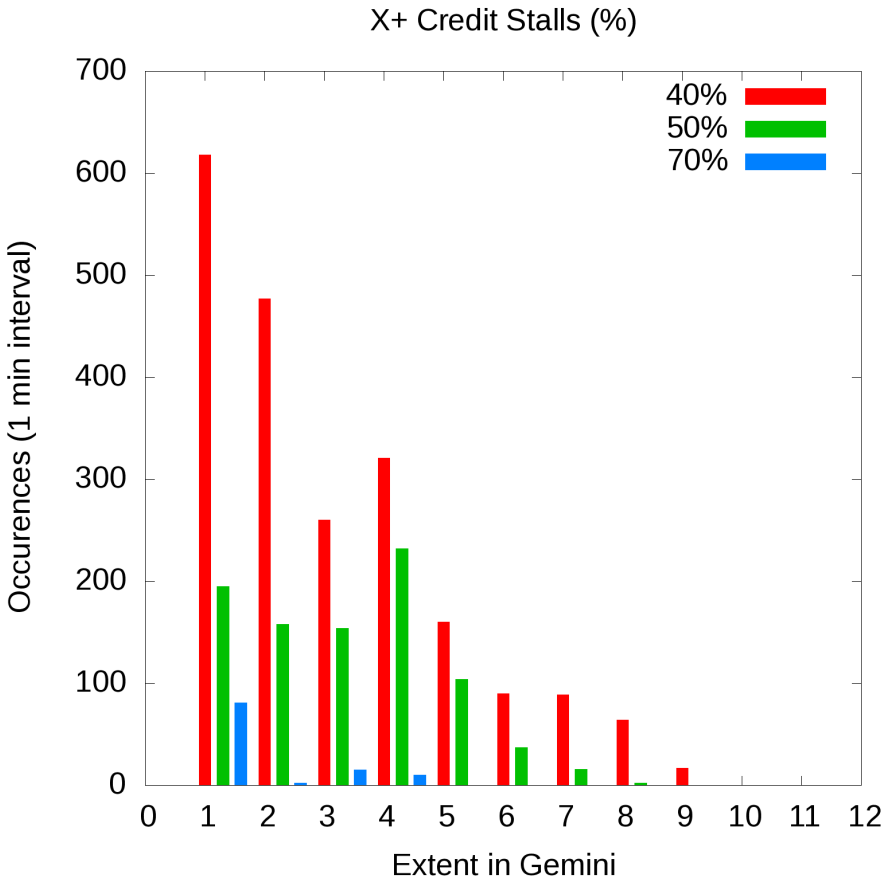
Max value = 63%

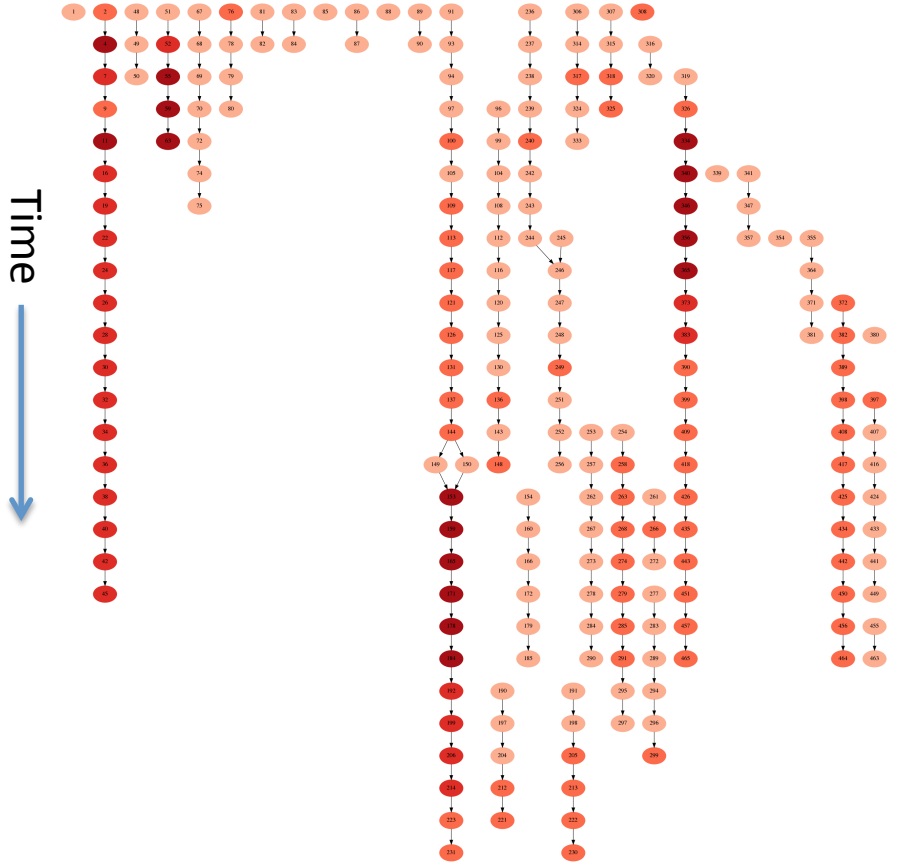
MESH TOPOLOGY REPRESENTATION
X+ CREDIT STALLS ANIMATION: 4 HRS @ 1059
(BLUE WATERS)





CONGESTION REGION CHARACTERISTICS: SPATIAL, SEVERITY, EVOLUTION





- **Number of Congestion Regions vs Size**
- Evolution of ROI through time colored by max value

CRAY ARIES

- **Performance counter API**
  - In-band only
    - NIC (13 metrics being collected)

      AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_PKTS
      AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_FLITS
      AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_STALLED
    - RTR (800 metrics being collected)

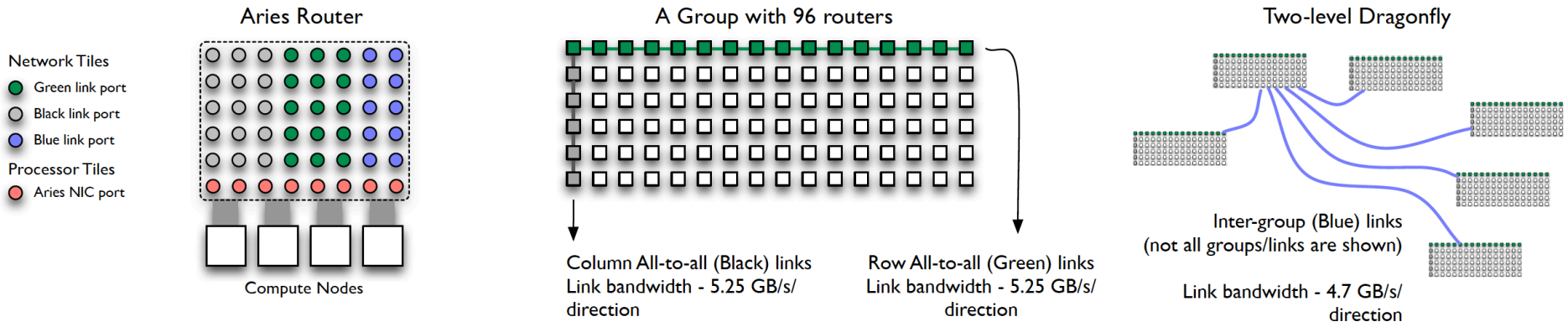
      AR_NL_PRF_REQ_PTILES_TO_NIC_0_FLITS
      AR_NL_PRF_REQ_PTILES_TO_NIC_0_PKTS
      AR_NL_PRF_REQ_PTILES_TO_NIC_0_STALLED
      AR_RTR_0_1_INQ_PRF_INCOMING_FLIT_VC0
      AR_RTR_0_0_INQ_PRF_INCOMING_PKT_VC0_FILTER_FLIT0_CNT
- **Fidelity**
  - ~1 second sampling period
- **Sampling overhead**
  - Host -- ~135usec to collect 213 raw metrics
    - Each of 4 hosts associated with an Aries router chip collects a quarter of data
      - No redundancy – loss of data if host goes down
  - Router – none?
- **Topology – Dragonfly**
- **Interconnect details**
  - On SMW: rtr --interconnect

    c0-0c0s0a0l00(0:0:0) green -> c0-0c0s6a0l10(0:0:6)
    c0-0c0s0a0l01(0:0:0) blue -> c2-0c0s0a0l01(1:0:0)
    c0-0c0s0a0l02(0:0:0) blue -> c2-0c0s0a0l02(1:0:0)
    c0-0c0s0a0l03(0:0:0) blue -> c2-4c0s0a0l02(25:0:0)
    c0-0c0s0a0l04(0:0:0) blue -> unused
    c0-0c0s0a0l05(0:0:0) blue -> unused
    c0-0c0s0a0l06(0:0:0) blue -> unused
    c0-0c0s0a0l07(0:0:0) black -> c1-0c2s0a0l16(0:5:0)
    c0-0c0s0a0l10(0:0:0) green -> c0-0c0s2a0l21(0:0:2)
    c0-0c0s0a0l11(0:0:0) green -> c0-0c0s7a0l00(0:0:7)
- **Adaptive routing: Yes**

CRAY ARIES CONTINUED

- **Analysis and/or visualization tools available**
  - System software looks for threshold crossings and errors that are used to trigger network throttling or network quiesce events
  - **None available to the user**
  - To build analysis tools for understanding the user has to read documents such as Cray Docs S-0045-20 and find statements in text descriptions of counters such as:
    - “for a contended resource. A high ratio of stalls to flits is an indication of possible congestion. Blocked counters are included at some arbiter inputs within Aries™. Typically, these blocked counters increment each flit-time a flit is available at the input, but arbitration has, instead, selected a different input.”
  - and/or solicit information directly from the vendor engineers
    - “for any interface for which both PKT and FLIT counts are provided, the ratio of increase in flit count to increase in packet count (both measured over the same time interval) provides an indication of average packet packet size over that time interval. The flit count increase, alone, is the best measure of traffic volume at the interface.”
    - RE: AR_RTR_r_c_INQ_PRF_ROWBUS_STALL_CNT – “compute the ratio of the increase in this stall count to the sum of the increases in the flit counts across all of the tile’s VCs, where the stall and flit increases are measured across the same time interval.”
  - Typical counter descriptions:
    - AR_NL_PRF_REQ_PTILES_TO_NIC_n_FLITS -- NIC request flits to NIC n
    - AR_NL_PRF_REQ_PTILES_TO_NIC_n_PKTS -- Request packets to NIC n
    - AR_NL_PRF_REQ_PTILES_TO_NIC_n_STALLED -- Clocks all request PTILEs have stalled to NIC n
    - AR_RTR_r_c_INQ_PRF_INCOMING_FLIT_VCv – counts network flits received.
    - AR_RTR_r_c_INQ_PRF_INCOMING_PKT_VCv_FILTER_FLITv_CNT -- counts the number of flit-times for which VC 'v' is stalled while waiting to forward a flit.
- **Dynamic routing: Yes**

CRAY XC ARIES ROUTER CONFIGURATION FOR DRAGONFLY NETWORK TOPOLOGY



Various components and logical units in the Cray Cascade (dragonfly) design: An Aries router has 40 network and 8 processor tiles (left). 96 such routers form a group (center) and several groups connect together to form the two-level network (right).

- **Green** links connect a router to all routers in a chassis
  - One **Green** link per connection
- **Black** links connect a router to all routers in same blade position across 6 chassis
  - Three **Black** links per connection
- **Blue** links connect routers in a 2 cabinet "group" to all other groups
  - Minimum of 4 **Blue** links between any two groups

From: *Analyzing Network Health and Congestion in Dragonfly-based Systems*, Bhatele et. al., IPDPS 2016

CRAY XC ARIES DRAGONFLY CONFIGURATION

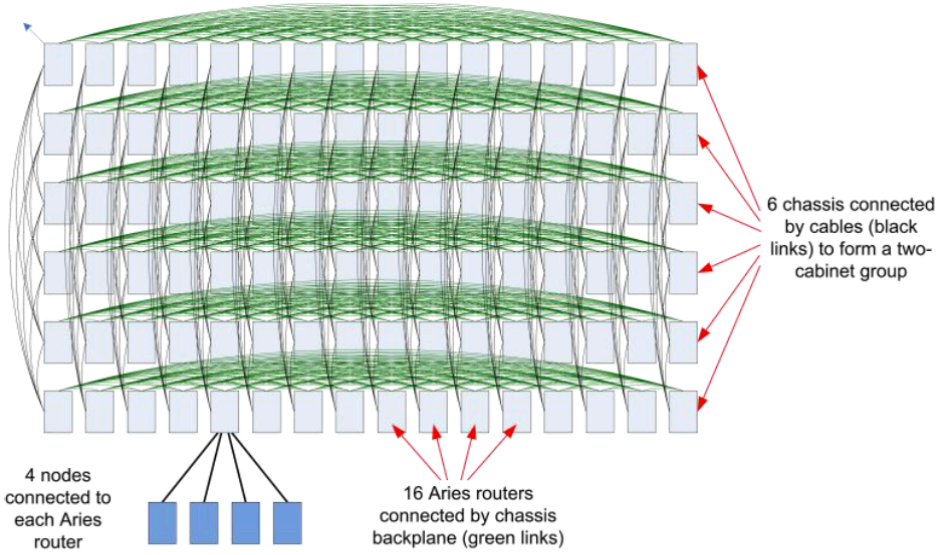


Figure 4. Structure of a Cascade electrical group. Each row represents the 16 Aries in a chassis, with 4 nodes attached to each, and connected by the chassis backplane (green links). Each column represents an Aries in one of the six chassis of a two-cabinet group, connected by electrical cables (black links).

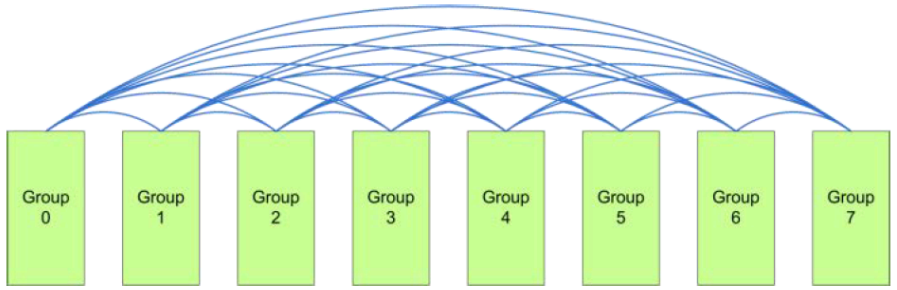
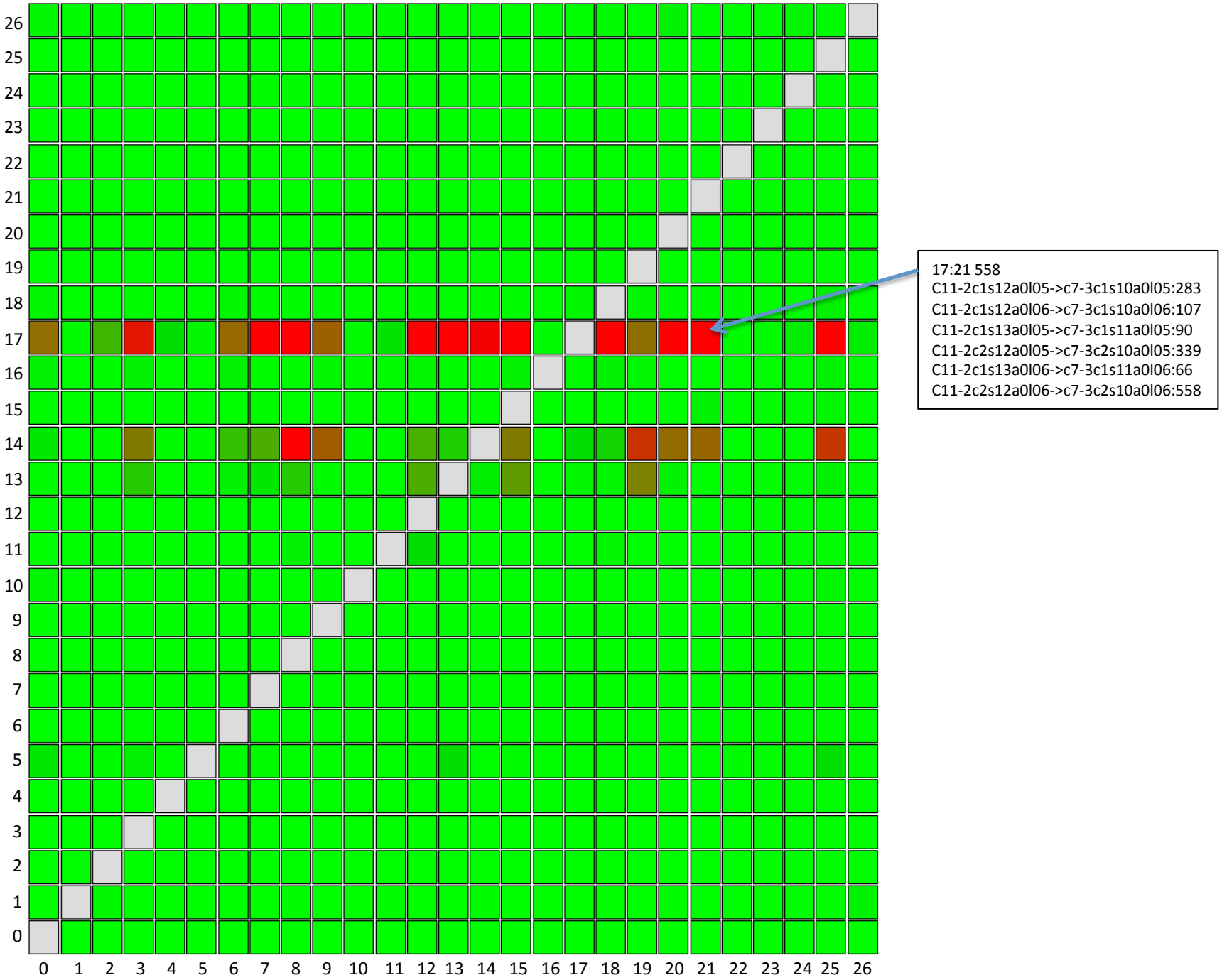


Figure 5. The global (blue) links connect Dragonfly groups together. In a large system these links are active optical cables.

From: *Cray Cascade: a Scalable HPC System based on a Dragonfly Network*, Faanes et. al.,SC 2012

Blue Links


26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
17:21 558
C11-2c1s12a0l05->c7-3c1s10a0l05:283
C11-2c1s12a0l06->c7-3c1s10a0l06:107
C11-2c1s13a0l05->c7-3c1s11a0l05:90
C11-2c2s12a0l05->c7-3c2s10a0l05:339
C11-2c1s13a0l06->c7-3c1s11a0l06:66
C11-2c2s12a0l06->c7-3c2s10a0l06:558

INTEL OMNIPATH

- **Sampling methodology**
  - Pull model
  - Fabric and Performance manager with both graphical and text user interfaces
    - Extract, display, and analyze counters and provide performance summaries
  - Inside each device there is a Performance Manager Agent (PMA) with 64 bit counters.
    - Fabric Manager (FM) can query counters whenever it likes (pull model)
    - Error counters are consolidated into a summary count
      - Don't need to query all the error counters (reduce overhead)
- **Fidelity**
  - ~10 second fabric sweeps by default (can be changed)
    - How does this scale?
- **Overhead**
  - Switch – is overhead only network traffic load or can high frequency queries disrupt traffic routing?
  - Host – any value in distributing fabric sampling over hosts?
- **Topology**
  - Fat tree, other
- **Fabric description**
  - Commands: Many tools for doing fabric/link analysis and report generation. Many of these appear to use **opareport** which provides fabric analysis and reporting capabilities.
- **Adaptive routing: yes**

Information from: *Intel® Omni-Path Fabric Suite FastFabric Command Line Interface Reference Guide*

PERFORMANCE RELATED METRICS

- **Link Integrity**

  These counters reflect errors in the Physical (PHY) and Link Layers, as well as errors in firmware. The typical cause is a hardware problem such as a poor connection, marginal cable, incorrect length model cable for signal rate, or damaged/broken hardware, such as bad connectors.

  - **Link Quality Indicator**
    - This is a status indicator, similar to the signal strength bar display on a mobile phone, that enumerates link quality as a range of 0-5, with 5 being very good. Values in the lower part of the range may indicate hardware problems such as port, cable, and others that surface as signal integrity issues, leading to performance and other problems.
  - **LocalLinkIntegrityErrors Counter**
    - This counter indicates the number of retries initiated by the link transfer layer.
  - **PortRcvErrors Counter**
    - This counter indicates the total number of packets containing an error that were received by the port, including Link Layer protocol violations and malformed packets. It indicates possible misconfiguration of a port, either by the SM or, more likely, by user intervention. It can also indicate hardware issues or extremely poor signal integrity for a link.
  - **ExcessiveBufferOverrunErrors Counter**
    - This counter, associated with credit management, indicates an input buffer overrun. It indicates possible misconfiguration of a port, either by the SM or, more likely, by user intervention. It can also indicate hardware issues or extremely poor signal integrity for a link.
  - **LinkErrorRecovery Counter**
    - This counter indicates the number of times the link has successfully completed the link error recovery process.
  - **LinkDowned Counter**
    - This counter indicates the total number of times the port has failed the link error recovery process and downed the link. These events can cause disruptions to fabric traffic.

Information from: *Intel® Omni-Path Fabric Suite FastFabric Command Line Interface Reference Guide*

PERFORMANCE RELATED METRICS

- **Congestion**

  These counters reflect possible errors that indicate traffic congestion in the fabric.

  - **PortXmitWait Counter**
    - This counter indicates the amount of time (in flit times) any virtual lane had data but was unable to transmit due to no credits available.
  - **SwPortCongestion Counter**
    - This switch-only counter indicates the number of packets that were discarded as unable to transmit due to timeouts.
  - **PortRcvFECN Counter**
    - When a device receives a packet with the FECN (Forward Explicit Congestion Notification) bit set to one, this counter is incremented.
  - **PortRcvBECN Counter**
    - When a device receives a packet with the BECN (Backward Explicit Congestion Notification) bit set to one, this counter is incremented.
  - **PortXmitTimeCong Counter**
    - This counter indicates the total number of flit times that the port was in a congested state for any data VL.
  - **PortMarkFECN Counter**
    - This counter indicates the total number of packets that were marked FECN (Forward Explicit Congestion Notification) by the transmitter due to congestion.

Information from: *Intel® Omni-Path Fabric Suite FastFabric Command Line Interface Reference Guide*

PERFORMANCE RELATED METRICS

- **SMA Congestion**

  These counters reflect congestion in the fabric specific to communication between the Subnet Manager and Subnet Manager Agents using the management VL (VL 15).

  The category is calculated exactly as the Congestion category using the same weights and the correct VL15 utilization counters.

  - **PortVLXmitWait[15] Counter**
    - This counter behaves the same as PortXmitWait, but it is restricted to VL 15 which carries only SM traffic.
  - **SwPortVLCongestion[15] Counter**
    - This counter behaves the same as SwPortCongestion, but it is restricted to VL 15 which carries only SM traffic.
  - **PortVLRcvFECN[15] Counter**
    - This counter behaves the same as PortRcvFECN, but it is restricted to VL 15 which carries only SM traffic.
  - **PortVLRcvBECN[15] Counter**
    - This counter behaves the same as PortRcvBECN, but it is restricted to VL 15 which carries only SM traffic.
  - **PortVLXmitTimeCong[15] Counter**
    - This counter behaves the same as PortXmitTimeCong, but it is restricted to VL 15 which carries only SM traffic.
  - **PortVLMarkFECN[15] Counter**
    - This counter behaves the same as PortMarkFECN, but it is restricted to VL 15 which carries only SM traffic.

Information from: *Intel® Omni-Path Fabric Suite FastFabric Command Line Interface Reference Guide*

PERFORMANCE RELATED METRICS

- **Bubble**

  These counters occur when an unexpected idle flit is transmitted or received. The transmit port sends idle flits until it can continue sending the rest of the packet. The category is calculated as follows:

  - 1. The maximum value between the sum of the XmitWastedBW and XmitWaitData or the neighbor's PortRcvBubble.
  - 2. Then divide the previous value by the port's utilization to provide context.

- **PortXmitWastedBW Counter**
  - This counter indicates the number of flit times where one or more packets have been started but the transmitters are forced to send idles due to bubbles in the ingress stream. Also, the VLs that have data to be sent are not permitted to preempt the currently transmitting VL.
- **PortXmitWaitData Counter**
  - This counter indicates the number of flit times where one or more packets have been started but interrupted due to bubbles in the ingress stream.
- **PortRcvBubble Counter**
  - This counter indicates the total number of flit times where one or more packets have started to be received, but the receiver received idle flits from the wire.

Information from: *Intel® Omni-Path Fabric Suite FastFabric Command Line Interface Reference Guide*

PERFORMANCE RELATED METRICS

- **Security**

  These counters reflect possible security problems in the fabric.

  Security problems can occur if a PKey or SLID violation occurs at the port during the ingress or egress of a packet.

  The category is calculated as the sum of the neighbor's PortRcvConstraintErrors and the local port's PortXmitConstraintErrors

  - **PortRcvConstraintErrors**
    - This counter is incremented when partition key or source LID violations are detected in a received packet, indicating a possible security issue or misconfiguration of device security settings.
  - **PortXmitConstraintErrors**
    - This counter is incremented when partition key violations are detected in a packet attempting to be transmitted, indicating a possible security issue or misconfiguration of device security settings.

- **Routing**

  These counters reflect possible routing issues. When a routing issue occurs, the offending packet is dropped.

  A typical cause of this error is the routing to a wrong egress port or an improper Service Channel (SC) mapping. These errors can be a side effect of a port or device going down while traffic was still in flight to or through the given port or device.

  - **PortRcvSwitchRelayErrors**
    - This counter indicates the number of packets that were dropped due to internal routing errors. It indicates possible misconfiguration of a switch by the SM.

- **Other**

  These counters do not fit into any of the previous categories.

  - **PortRcvRemotePhysicalErrors**
    - This counter indicates the number of downstream effects of signal integrity (SI) problems. It indicates an SI issue in the upstream path.
  - **UncorrectableErrors Counter**
    - This counter indicates the number of unrecoverable internal device errors. It indicates a severe hardware defect or data corruption inside the device.
  - **FMConfigErrors Counter**
    - This counter indicates inconsistencies of low level SMA configuration on both sides of the link. It indicates possible misconfiguration of a port, either by the SM, or, more likely, by user intervention.

NETWORK VENDOR AND TECHNOLOGY INDEPENDENT NEEDS

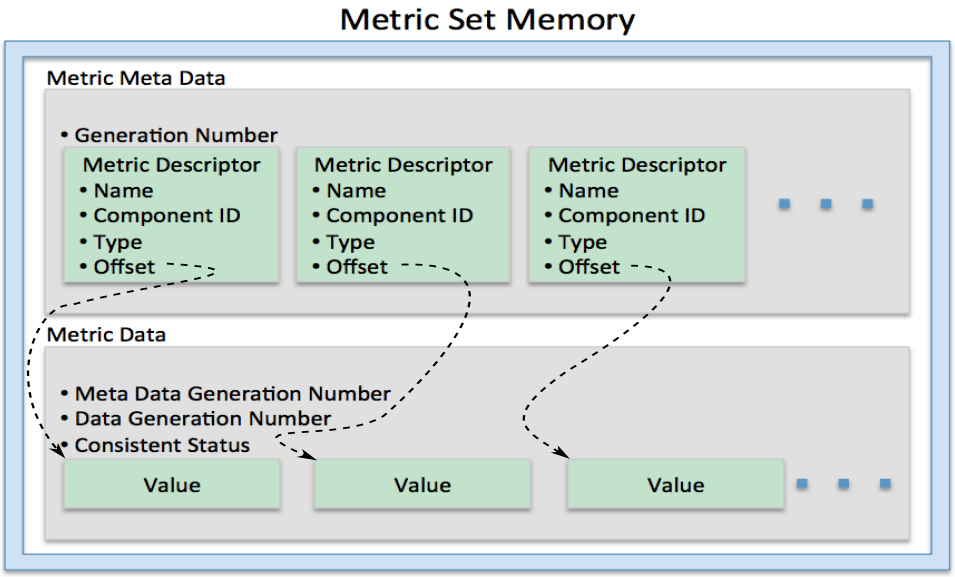
NEEDS

- **APIs and approaches for lightweight network performance counter acquisition**
  - Querying a single subnet manager is a serial process and thus has scaling and synchronization issues issues
    - Could this be distributed?
  - What overhead is incurred in querying a switch directly?
  - Possibility of DMA to contiguous switch register memory to grab all counters for a port at one go?
  - Timestamps relative to counter updates?
- **Reasonably high fidelity querying without performance degradation**
  - What is high enough and why?
- **System synchronized (to within some time delta) fabric sampling**
  - How synchronized and what mechanism?
- **Host to Traffic attribution would be nice including internal fabric**
- **Descriptions of how to best use counters to gain insight**
  - How do I know level of congestion?
  - Normal vs. abnormal operation and why
- **Tools for analysis, visualization, and feedback to admin staff, users, system software and applications**
  - Post processing and run-time
    - Published analytics allowing the user to understand health of fabric (not just a number)
    - Visualization to aid in understanding
  - Enable development through published APIs and good documentation
    - Open source examples could help

PROVIDE SIMPLE UTILITIES FOR EFFICIENT ACQUISITION OF COUNTERS OF INTEREST

Vendors

- Would be nice to enable the user to define their set of counters of interest, update rate, and time synchronization and enable RDMA read of whole set at a time.
  - Switch – define sets that are groups of ports metrics to enable distributed queries
  - Host – kernel sampler with same properties but map set memory into user space



Vendors + Community

- A common way of doing this across all network technologies would enable re-use of tools while still enabling vendors to add value via a rich set of counters
- Platform/network technology independent tools for analysis and visualization

OPENFABRICS
ALLIANCE

12th ANNUAL WORKSHOP 2016

THANK YOU

Jim Brandt, R&D Staff
Gentile, Allan, Lefantzi, and Aguilar
Sandia National Laboratories

[LOGO HERE]