IB or Not IB

Cary Whitney

IB tools

- Infiniband Radar (Thank you Martin Gasthuber - DESY)
  - https://github.com/infiniband-radar

- Fabriscale http://fabriscale.com/

  - Paper: https://pdfs.semanticscholar.org/2654/3d0170c11c36fcf6f2eac96624af4c65d485.pdf

Thank You

backup slides

What to do?

- Use existing infiniband-diag tools to collect counters and errors on the fabric.
  - IB base counters are 32 bits.
  - Mellanox extended counters are 64 bit.
  - Counters do not roll.
  - Current tools are an improvement over the older tools in that they are now written in C instead of Perl. They can read both sets of counters.
  - There are some libraries available.
  - There is very little documentation.

- ibnetdiscover -p

```
The -p basically gives a summary of every cable connection in the fabric on a single line.

CA   730  1 0xe41d2d03007ba181 4x FDR10 - SW   772 12 0xe41d2d0300815900
( 'MT25408 ConnectX Mellanox Technologies' - 'MF0;2574-0-SW1-M1:SX6536/L14/U1' )

CA   475  1 0xe41d2d0300789991 4x FDR - SW    39 17 0xe41d2d0300813e80
( 'MT25408 ConnectX Mellanox Technologies' - 'MF0;2574-0-SW1-M1:SX6536/L20/U1' )
```

```
CA - HCA or host card/port
730 - LID number assigned by the SM
1 - Port number on the card
0xe… - GUID for the port
4x - The width of the port
FDR10 - The speed of the port

SW - Switch port
772 - LID assigned to the port
12 - The port number
0xe4… - GUID of the Switch

(stuff) is the descriptions for both sides of the connection.
```

First Pass

- This is more an exercise since I skipped this step mostly. This would be running the tools and parsing the text output with Perl or Python to get the information needed.
- Once parsed. The data could be stored in a method or location desired.
- Pro: Quick, flexible, very little special code
- Con: Easily broken when text changes, can be slow, run once mode only

Second Pass

- Let’s modify an existing infiniband diagnostic tool.
- ibqueryerrors was selected since it gave error and performance counters within the same program.
  - Modified output to JSON format and added in routines to send to RabbitMQ.
- Pro: Builds on existing software
- Con: One really needed to run the command twice, once for performance counters and once for errors, thus required additional modifications.

- ibqueryerror

Normal output:

```
bash-4.1# time ibqueryerrors -f
Errors for "happy HCA-3"
   GUID 0x1175000079da38 port 1: [SymbolErrorCounter == 332] [PortRcvErrors == 6]
Errors for 0x66a00e3003b39 "QLogic 12200 GUID=0x00066a00e3003b39"
   GUID 0x66a00e3003b39 port ALL: [SymbolErrorCounter == 154] [LinkDownedCounter == 2] [VL15Dropped == 23]
   GUID 0x66a00e3003b39 port 35: [SymbolErrorCounter == 154] [LinkDownedCounter == 1] [VL15Dropped == 23]
   GUID 0x66a00e3003b39 port 36: [LinkDownedCounter == 1]
```

First attempt to JSON ibqueryerror by replicating all the print statements with JSON formatted text so it can be sent to Elastic via RabbitMQ.

```
if (json) {
        n += snprintf(buf + n, size - n, " \"%s\": %u, ",
             mad_field_name(i), val);
} else {
        n += snprintf(buf + n, size - n, " [%s == %u]",
             mad_field_name(i), val);
}
```

This just represents one snippet of code. The printing of the output happens throughout the code. Prints a little here and a little there.

Issues with 1 and 2

- Both are single commands. Run -> output
- Cron or script needed to run command periodically.
- Timeouts in the fabric can slow the command down and mess with timing. This can be highly varied.
- Complex use of date to try and address timeout issue, but this can create data skew. (Some data being collected at a different time than timestamp.)
- Timeout can also affect how often the data can be collected.

Third Pass

- Let’s base things on an existing infiniband diagnostic tool but modify it to have it run continuously.
  - This means we can keep state information.
  - Counters can be saved from the previous run.
  - Differences between 64 bit counters can now be calculated.
  - Timestamp per counter can be done. Now we can generate a rate.
  - Timeouts may still affect the program but since each counter has it’s own timestamp, there is little data skew.
  - We can now enrich the data since we control it.

- Scan the fabric.

```
main() {
        while(1) {

                if (!(fabric = ibnd_discover_fabric(ibd_ca, ibd_ca_port, NULL, &config))) {
                        fprintf(stderr, "discover failed\n");
                        exit(1);
                }

        /*      ibnd_iter_nodes(fabric, gather, NULL); */
        /*      ibnd_iter_nodes_type(fabric, gather, IB_NODE_CA, NULL); */
                ibnd_iter_nodes_type(fabric, routes, IB_NODE_SWITCH, NULL);
        }
}
```

- Gather

```
void gather(ibnd_node_t * node, void *user_data)
{

/* Loop through all the ports to get stats */
/* pc and ibmad_port are globals */

        if (node->type == IB_NODE_SWITCH)
                startport = 0;

        for (port = startport; port <= node->numports; port++) {
                if (node->ports[port]) {
                        if (node->type == IB_NODE_SWITCH) {
                                ib_portid_set(&portid, node->smalid, 0, 0);
                        } else {
                                ib_portid_set(&portid, node->ports[port]->base_lid, 0, 0);
                        }

                        memset(pc, 0, sizeof(pc));
                        if (!pma_query_via(pc, &portid, port, ibd_timeout, CLASS_PORT_INFO, ibmad_port))
                                fprintf(stderr, "classportinfo query");

                        /* ClassPortInfo should be supported as part of libibmad */
                        memcpy(&cap_mask, pc + 2, sizeof(cap_mask));    /* CapabilityMask */
                        memcpy(&cap_mask2, pc + 4, sizeof(cap_mask2));  /* CapabilityMask2 */
                        cap_mask2 = ntohl(cap_mask2) >> 5;

/*
printf("Port: %d\n", port);
printf("Node GUID: %016lx\n", node->guid);
printf("Port GUID: %016lx\n", node->ports[port]->guid);
*/
```

- Get node state information (partial)

```
static void get_state(struct guids *g, ibnd_node_t * node, int portnum) {

        ibnd_port_t *port = node->ports[portnum];

        if (!port)
                return;

        iwidth = mad_get_field(port->info, 0, IB_PORT_LINK_WIDTH_ACTIVE_F);
        ispeed = mad_get_field(port->info, 0, IB_PORT_LINK_SPEED_ACTIVE_F);
        fdr10 = mad_get_field(port->ext_info, 0,
                              IB_MLNX_EXT_PORT_LINK_SPEED_ACTIVE_F) & FDR10;

        if (port->node->type == IB_NODE_SWITCH)
                info = (uint8_t *)&port->node->ports[0]->info;
        else
                info = (uint8_t *)&port->info;
        cap_mask = mad_get_field(info, 0, IB_PORT_CAPMASK_F);
        if (cap_mask & CL_NTOH32(IB_PORT_CAP_HAS_EXT_SPEEDS))
                espeed = mad_get_field(port->info, 0,
                                       IB_PORT_LINK_SPEED_EXT_ACTIVE_F);
        else
                espeed = 0;
        istate = mad_get_field(port->info, 0, IB_PORT_STATE_F);
        iphystate = mad_get_field(port->info, 0, IB_PORT_PHYS_STATE_F);
```

- Dump counters

```
static void dump_perfcounters(struct guids *g, int extended, int timeout, uint16_t cap_mask,
                              uint32_t cap_mask2, ib_portid_t * portid,
                              int port, int aggregate)
{
        pcSampleTime = time(NULL);
        memset(pc, 0, sizeof(pc));
        if (!pma_query_via(pc, portid, port, timeout, IB_GSI_PORT_COUNTERS, ibmad_port)) {
                fprintf(stderr, "perfquery issue port: %d\n", port);
                g->pmaQueryFailure = -1;
        } else {
                g->pmaQueryFailure = 0;
        }

        /* 1.2 errata: bit 9 is extended counter support
         * bit 10 is extended counter NoIETF
         */
        if (!(cap_mask & IB_PM_EXT_WIDTH_SUPPORTED) &&
            !(cap_mask & IB_PM_EXT_WIDTH_NOIETF_SUP))
                fprintf(stderr, "PerfMgt ClassPortInfo CapMask 0x%02X; No extended counter support
indicated\n", ntohs(cap_mask));

        pc2SampleTime = time(NULL);
        memset(pc2, 0, sizeof(pc2));
        if (!pma_query_via(pc2, portid, port, timeout, IB_GSI_PORT_COUNTERS_EXT, ibmad_port)) {
                fprintf(stderr, "perfextquery2 issue port: %d\n", port);
                g->pmaQueryFailure2 = -1;
        } else {
                g->pmaQueryFailure2 = 0;
        }

       perfcounters_ext(g, cap_mask, cap_mask2, pcSampleTime, pc2SampleTime);
}
```

- Calculate Stat

```
calc_stat(uint8_t *p, struct stats *item, int stat, time_t sampleTime) {
        item->ptimestamp = item->ctimestamp;
        item->ctimestamp = sampleTime;
        item->statId = stat;
        item->pvalue = item->nvalue;
        mad_decode_field(p, stat, &item->nvalue);
        if((item->nvalue - item->pvalue) < 0) {
                item->cvalue = 0;
                item->rate = 0;
        } else {
                item->cvalue = item->nvalue - item->pvalue;
                if((item->statId >= 197 && item->statId <= 204) || (item->statId >= 137 && item-
>statId <= 141)) {
                        item->rate = (item->cvalue / (item->ctimestamp - item->ptimestamp)) * 4;
                } else {
                        item->rate = 0;
                }
        }
}
```

- Generate JSON in one location and send to RabbitMQ

```
void print_route_structure(struct ibroutes *g) {

        while(g != NULL) {
                status = asprintf(&message, "{ \"switchGuid\": \"%lx\", \"lid\": %d, \"port\": %d,
\"routeLid\": %d", g->switchGuid, g->lid, g->port, g->routeLid);

                rmessage = message;
                status = asprintf(&message, "%s, \"node_name\": \"%s\", \"system_name\": \"%s\",
\"location\": \"%s\", \"remote_name\": \"%s\", \"remote_system\": \"%s\" }",
                                rmessage, lookup_node(g->switchGuid, 1), lookup_node(g->switchGuid,
2), lookup_node(g->switchGuid,3), lookup_node(g->remotePortGuid, 1), lookup_node(g->remotePortGuid,
2));

                free(rmessage);

                if(DEBUG) {
                        printf("FULL: %s\n", message);
                } else {
                        rabbitmq_send(message);
                }
                free(message);
                g = g->next;
        }
}
```

- ibroute switchGUID

This is only available on switches and is a simple table listing every LID in the network and what port on the switch should that LID be routed out.

```
0x0000 255
0x0001 020
0x0002 021
0x0003 033
0x0004 019
0x0005 028
0x0006 028
0x0007 031
```

Since the LID is the final destination, I also get the port’s next hop. Both the remoteGUID and remotePort are also gathered.

Plus the LID information is also enriched by listing it’s name and GUID at the time. (LID’s can change at major sweep time by the SM.)

- Route

```
void routes(ibnd_node_t *node, void *user_data) {

        if (node->type == IB_NODE_SWITCH)
                startlid = 0;

        ib_portid_set(&portid, node->smalid, 0, 0);

        if (!smp_query_via(&sw, &portid, IB_ATTR_SWITCH_INFO, 0, 0, ibmad_port)) {
                printf("query fail\n");
                return;
        }

        mad_decode_field(sw, IB_SW_LINEAR_FDB_TOP_F, &endlid);

        top = head_route;
        startblock = startlid / IB_SMP_DATA_SIZE;
        endblock = ALIGN(endlid, IB_SMP_DATA_SIZE) / IB_SMP_DATA_SIZE;
        for (block = startblock; block < endblock; block++) {
                int status;

/*              printf("BLOCK: %d\n", block); */
                if (!smp_query_status_via(lft, &portid, IB_ATTR_LINEARFORWTBL, block, 0, &status, ibmad_port)) {
                        fprintf(stderr, "SubnGet() failed" "; MAD status 0x%x AM 0x%x\n", status, block);
                        return;
                }
                i = block * IB_SMP_DATA_SIZE;
                e = i + IB_SMP_DATA_SIZE;
                if (i < startlid)
                        i = startlid;
                if (e > endlid + 1)
                        e = endlid + 1;

                for (; i < e; i++) {
                        unsigned outport = lft[i % IB_SMP_DATA_SIZE];
                        unsigned valid = (outport <= node->numports);

                        portguid = 0;
                        lidport.lid = i;
```

Issues with Pass 3

- Libraries included printf statements and did not just pass back information.
- Difference between OFED and Mellanox caused issues since some parts of the libraries are being used but symbols are not included.
- Code was dragged into my program just to make it work. Code is not used but is needed to satisfy symbol errors.
- enum type skew. Different versions of infiniband-diag may have added enum types in the counters list.
    - Types have the same name between the 32 bit and 64 bit counters. Thus try to use the enum type place as id.
    - New types are sometimes added into the middle of the list, creating off by one issues.

- Inside the libraries?

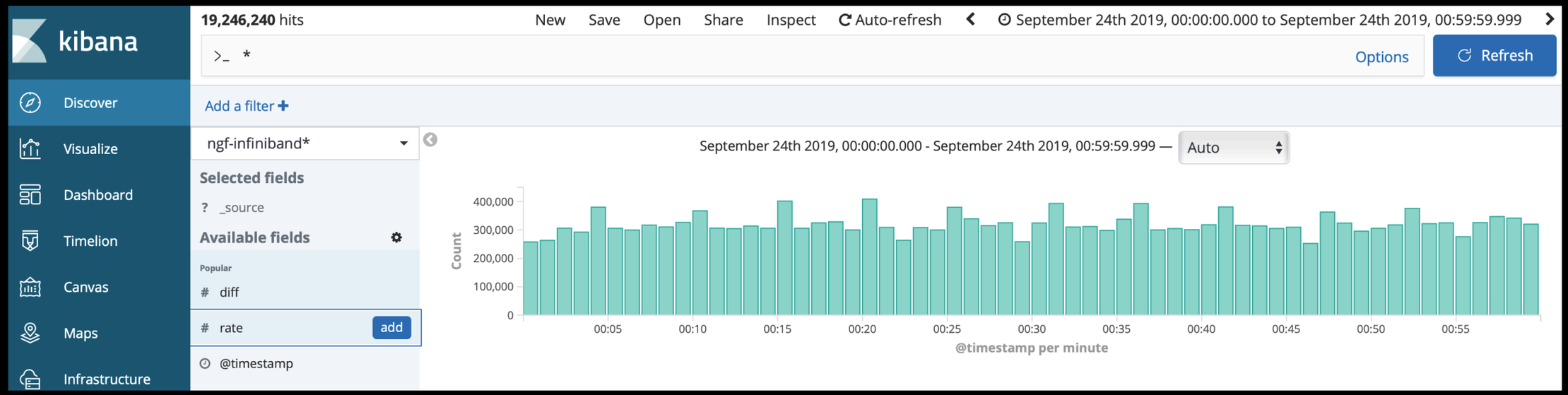
```
void mad_dump_node_type(char *buf, int bufsz, void *val, int valsz)
{
        int nodetype = *(int *)val;

        switch (nodetype) {
        case 1:
                snprintf(buf, bufsz, "Channel Adapter");
                break;
        case 2:
                snprintf(buf, bufsz, "Switch");
                break;
        case 3:
                snprintf(buf, bufsz, "Router");
                break;
        default:
                snprintf(buf, bufsz, "?(%d)?", nodetype);
                break;
        }
}

static void dump_endnode(ib_portid_t * path, char *prompt,
                         ibnd_node_t * node, ibnd_port_t * port)
{
        char type[64];
        mad_dump_node_type(type, sizeof(type), &node->type, sizeof(int));
        printf("%s -> %s %s {%016" PRIx64 "} portnum %d lid %d-%d \"%s\"\n",
               portid2str(path), prompt, type, node->guid,
               node->type == IB_NODE_SWITCH ? 0 : port->portnum,
               port->base_lid, port->base_lid + (1 << port->lmc) - 1,
               node->nodedesc);
}
```

- Counter/Error Record

```
@timestamp  September 24th 2019, 00:59:59.997
_index  infiniband-2019.09.24
counterName  PortXmitPkts
counterTime  1,569,311,983
desc  MF0;ib1:SX6536/S18/U1
diff  748,388
iSpeed  53.125
iWidth  4
id  198
lid  82
location  (null)
ndc.system  fs
ndc.type  infiniband
nodeGuid  2c90300713ed0
nodeName  ib1
nodeTime  1,568,924,304
physicalState  LinkUp
pmaQueryFailure  0
pmaQueryFailure2  0
port  31
portGuid  2c90300713ed0
rate  272,140
recordType  0
remoteLid  26
remoteName  ib1
remoteNodeGuid  e41d2d03017fc680
remotePort  36
remotePortGuid  e41d2d03017fc680
remoteSystem  leaf_31
speed  undefined (4)
state  Active
systemName  spine_18
tags  infiniband, metric, ndc
value  4,606,326,506
width  4X
```

- Routing Record

```
@timestamp  September 24th 2019, 00:57:48.469
_index  infiniband-2019.09.24
lid  25
location  (null)
ndc.system  fs
ndc.type  infiniband
nodeName  ib1
port  20
recordType  1
remoteLid  1
remoteName  ib1
remotePort  3
remotePortGuid  e41d2d0300102a40
remoteSystem  spine_02
routeGuid  98039b03016db30c
routeLid  611
routeName  node04
routeSystem  home_filesystem
routeTime  1,569,311,859
switchGuid  e41d2d03017fc600
systemName  leaf_03
tags  infiniband, metric, ndc
```

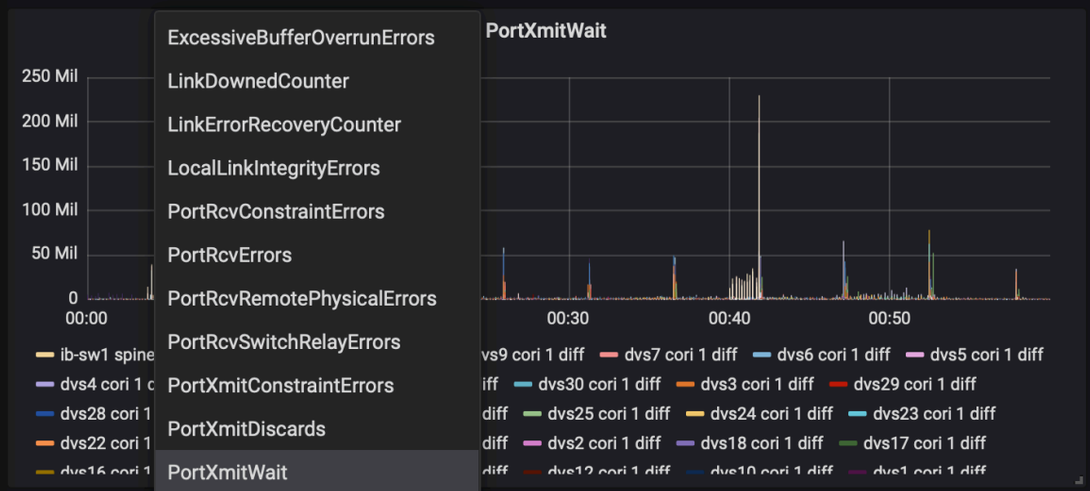

kibana
19,246,240 hits
New
Save
Open
Share
Inspect
Auto-refresh
September 24th 2019, 00:00:00.000 to September 24th 2019, 00:59:59.999
>_ *
Options
Refresh
Add a filter +
Discover
Visualize
Dashboard
Timelion
Canvas
Maps
Infrastructure
ngf-infiniband*
Selected fields
? _source
Available fields
Popular
diff
rate
add
@timestamp
September 24th 2019, 00:00:00.000 - September 24th 2019, 00:59:59.999 —
Auto
400,000
300,000
200,000
100,000
0
Count
00:05
00:10
00:15
00:20
00:25
00:30
00:35
00:40
00:45
00:50
00:55
@timestamp per minute

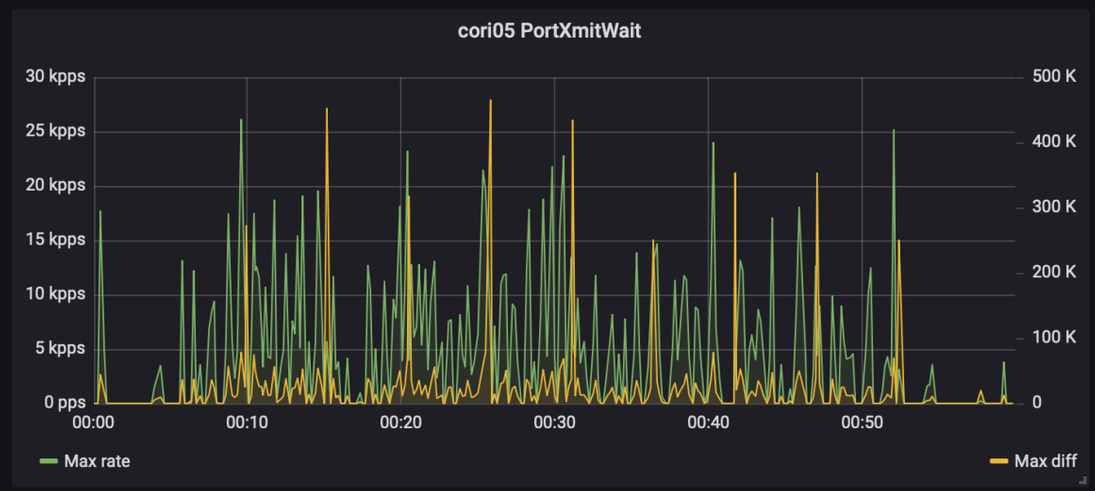
IB Errors

Trouble Node: cori05 | switchPorts: All | Switch: ib-sw1 | Blade: leaf_03 | Port: 4 | Host(s) Route through Switch Port: ngfgw15 | Counter Type: Data | pswitch: All | rlid: All

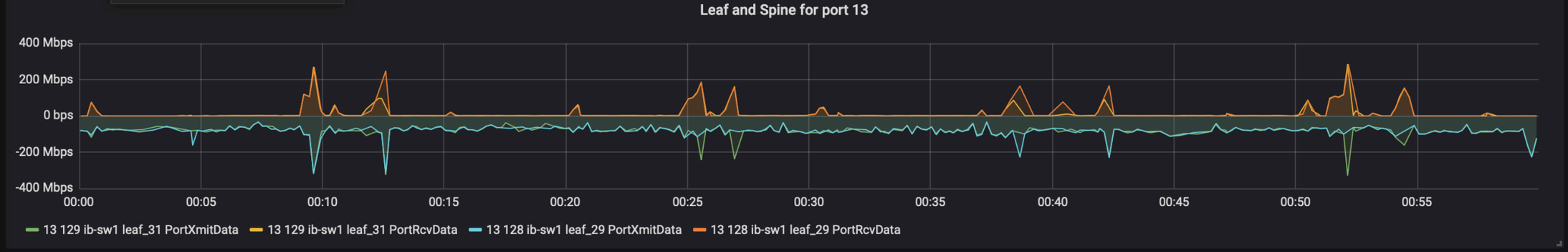
Error Counters

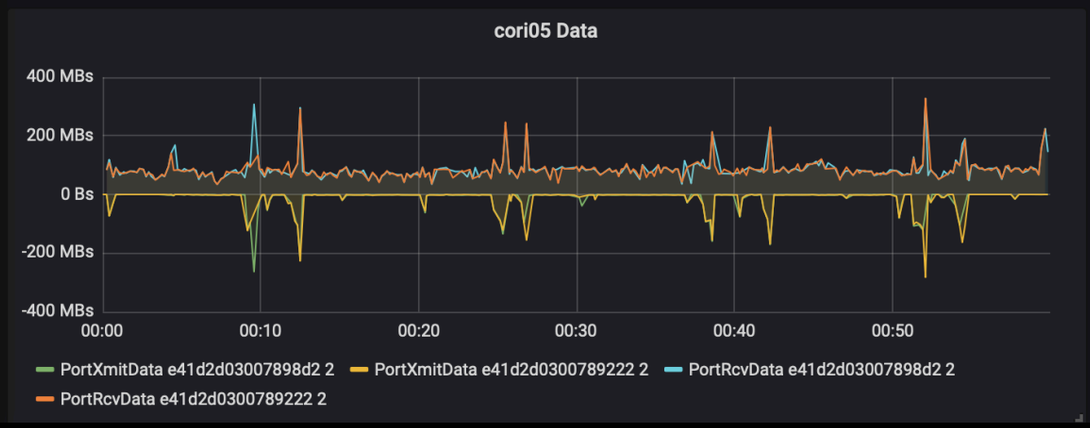
- ExcessiveBufferOverrunErrors
- LinkDowned Counter
- LinkErrorRecoveryCounter
- LocalLinkIntegrityErrors
- PortRcvConstraintErrors
- PortRcvErrors
- PortRcvRemotePhysicalErrors
- PortRcvSwitchRelayErrors
- PortXmitConstraintErrors
- PortXmitDiscards
- PortXmitWait
- SymbolErrorCounter
- VL15Dropped

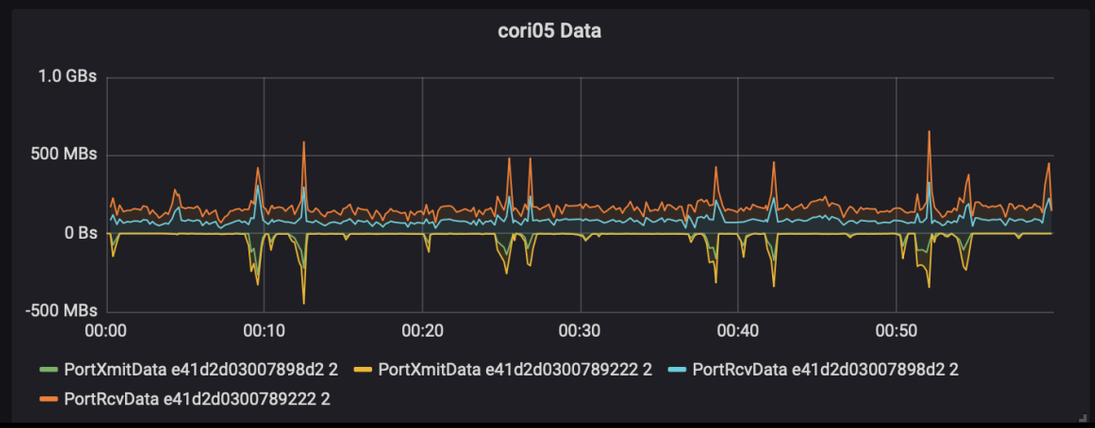




Counters







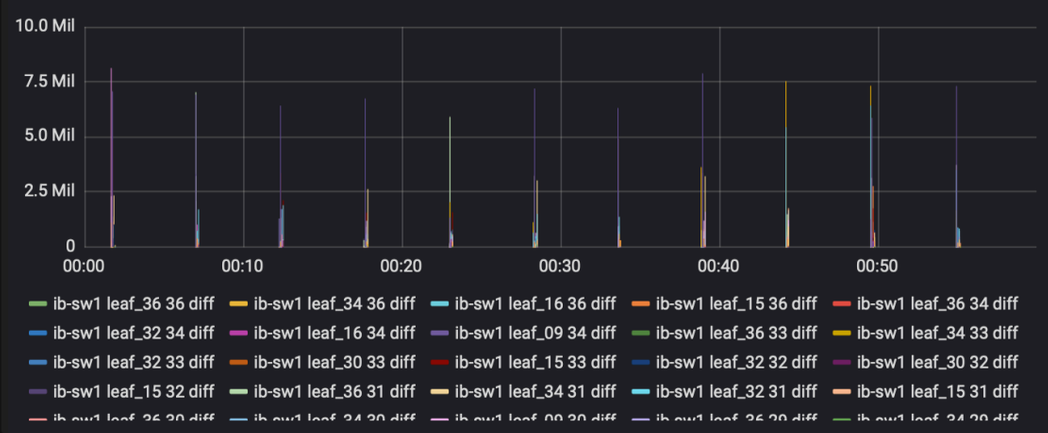
IB Errors

2019-10-08 00:00:00 to 2019-10-08 00:59:59

Trouble Node: cgpu07 | switchPorts: All | Switch: ib-sw1 | Blade: leaf_03 | Port: 23 | Host(s) Route through Switch Port: ib-sw1 | Counter Type: Data | pswitch: All | rlid: All

Error Counters: PortXmitWait

PortXmitWait

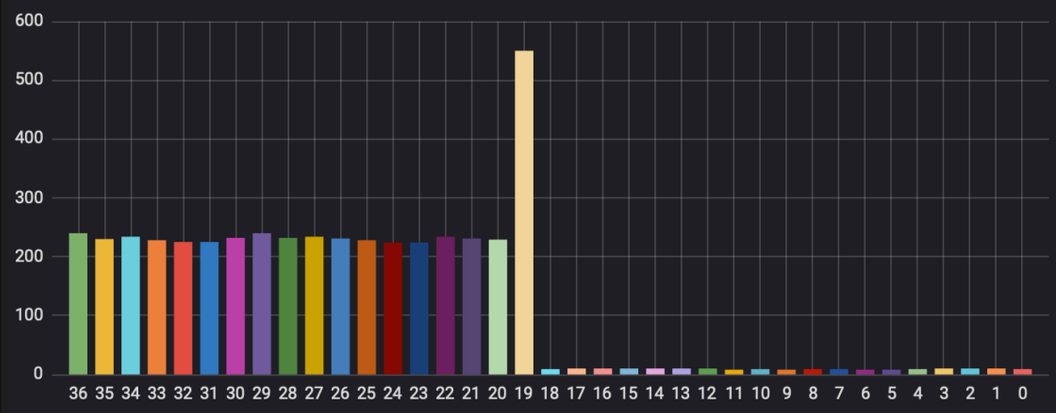


cgpu07 PortXmitWait

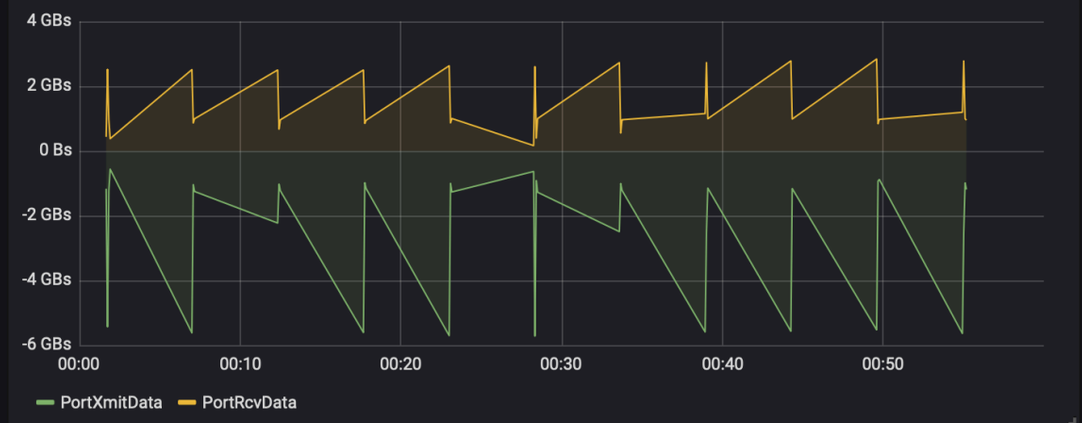
Counters (3 panels)

Switch and Node Stat

ib-sw1 leaf_03



ib-sw1 Data



Modifications to Pass 3

- Instead of adding code snippets into my code, should just rewrite the libraries to send information instead of printf.
- This would address Mellanox changes that are not needed.
- There are some version information that I miss when pulling to the code but would be retained with a library rewrite.
- Library rewrite would be easier to incorporate back into the community.
- Will make it easier for others to write IB utilities.

ToDo

- Create custom visualizations. Grafana and Kibana really only show basic graph relationships.
    - What links are really needed?
    - Correlate IB links to Lustre servers. Include Lustre Jobstats. Now we can see if there are network hotspots, especially PortXmitWaits.
- Can this be abstracted to Slingshot?
- Communicate route changes effectively. (This includes adaptive routing.) I currently collect every 5 minutes since we do no do adaptive but could collect faster.
- Monitoring. Trying to present more information to the POC so they can respond to issues faster. (Instead filesystem slow, we can say, this node and this network is showing errors and these jobs are affected.)
    - Using elasticsearch_dsl, any python script can do this and work within our Nagios environment.
    - Also using fn - FaaS to correlate multiple indexes into the alert.

Thank you, again.