Table of Contents

More realistic Felix data-flow generation pattern

We developed a custom distribution to represent the traffic generation of the felix servers. This distribution and tests are described in this page.

See meeting Caracterizando la generacion de trafico de los servidores felix (Jorn, Matias, Andy)

Conclusions:

1. The felix behavoir in HIGH_BANDWITH expect to add ~2600 - 2900us latency for the data flow (for 10 GBT lins and 24 GBT links respectively). This corresponds to 2500us of latency due to the felix buffering and 130-400us due to the burst queueing)
2. Bursts are absorved in the felix NIC, which get to a maximum queue size of 3.5MB for 10 connections (GBT links). Although there wont be any single buffer with this size (it will be distributed in main memory, OS and NIC memory), the delay/latency produced by this buffer will exisit in the real world.
3. Traffic pattern is more realistic: 1) bursts of packet from the felix application using per connection 1MB buffers 2) bonded link hash per connection (GBT link) 3) we simulate the arrival of each GBT message (@100kHz per GBT)

- Implementation

The distribution "simulates" the arrival of messages throught the GBT links at a given rate. In Low_LATENCY mode, the messages are forward without delay. In High_throughput mode, the messages are queued and forwarded when the buffer is full.
Distributions only have a nextValue method which return a random number distributed according to the distribution. In this case, it returns the time period between one outmessage from the felix server to the next outmessage. Because of felix behavior outmessages are sent in bursts: nothing is sent while the buffer is being filled, and then several messages are sent to the networking stack all together (the buffer is partitioned in several messages according to TCP MTU).

The idea is to have 1 flow in the simulation per felix connection (because there will be 1 buffer per connection). Also this will help routing each connection propertly in the bonded links: the bonded link will always use the same link for the same connection.

Parameters for the distribution

1. **Period:** (in seconds) This is a distribution parameter. The period (1/rate) of the incomming messages in the GBT links.
2. **Mode**: low_latency or high_throughput
3. **incomming size (only in high_throughput mode) :** (in bytes) This is a distribution parameter. Size of the incomming GBT messages (which will fill up the buffer)
4. **Buffer size:** (in bytes) this is the size of the buffer.
5. **timeout:** maximum time without sending a message (if the buffer does not get filled)
6. **outSize:** size of each outmessage (the buffer is partitioned in several messages according to TCP MTU)

```
flowFlow0_1.period = DISTRIBUTION_FELIX;
flowFlow0_1.period_period = DISTRIBUTION_EXPONENTIAL;
flowFlow0_1.period_period_mu = 1/(10*M); // 1000 MB/s (80Gbps)
flowFlow0_1.period_mode = FELIX_MODE_HIGH_THROUGHOUT;
flowFlow0_1.period_size_bytes = DISTRIBUTION_NORMAL;
```

```
flowFlow0_1.period_size_bytes_mu = 1*k ; // 1000 MB/s (80Gbps)
flowFlow0_1.period_size_bytes_var = 1*k;
flowFlow0_1.period_buffer_bytes = 1 * M;
flowFlow0_1.period_timeout = 1; // (seconds)
flowFlow0_1.period_out_size_bytes = TCP_MTU_bytes;
flowFlow0_1.packetSize = DISTRIBUTION_CONSTANT; // (in bits)
flowFlow0_1.packetSize_value = TCP_MTU_bytes * 8; // value for the constant distribution
```

The implementation of the FelixDistribution includes the generation of 2 random numbers (size and period) per each simulated arrival of a GBT message. This can severely impact the performance if there are many FelixDistributions, with high GBT rate and small GBT message size.

Tests - comparison

With exponensial distribution

```
flowFlow1_1.period = DISTRIBUTION_EXPONENTIAL;
flowFlow1_1.period_mu = 1/(1*M); // mean for the exponential distribution.
flowFlow1_1.packetSize = DISTRIBUTION_CONSTANT;
flowFlow1_1.packetSize_value = 1000.0; // value for the constant distribution
```

With Felix distribution

```
flowFlow0_1.period = DISTRIBUTION_FELIX;
flowFlow0_1.period_period = DISTRIBUTION_EXPONENTIAL;
flowFlow0_1.period_period_mu = 1/(10*M); // 1000 MB/s (80Gbps)
flowFlow0_1.period_mode = FELIX_MODE_HIGH_THROUGHOUT;
flowFlow0_1.period_size_bytes = DISTRIBUTION_NORMAL;
flowFlow0_1.period_size_bytes_mu = 1*k ; // 1000 MB/s (80Gbps)
flowFlow0_1.period_size_bytes_var = 1*k;
flowFlow0_1.period_buffer_bytes = 1 * M;
flowFlow0_1.period_timeout = 1; // (seconds)
flowFlow0_1.period_out_size_bytes = TCP_MTU_bytes;
flowFlow0_1.packetSize = DISTRIBUTION_CONSTANT; // (in bits)
flowFlow0_1.packetSize_value = TCP_MTU_bytes * 8; // value for the constant distribution
```

Tests with 10 GBT links per felix (32 Gbps) - 2 felix servers

Configuration

https://docs.google.com/presentation/d/1hbVbfWeP610hO88F7t5n_U10j3norKcGcVpiJhzkeT4/edit#slide=id.p

FELIX_GBT_PERIOD_sec = ExponentialDistribution.new 1.0 / (100*K) # distribution period in seconds
FELIX_GBT_SIZE_bytes = NormalDistribution.new 4.0*K, 1.0*K # (in bytes)
FELIX_GBT_BUFFER_bytes = 1*M # (in bytes)
FELIX_GBT_TIME_OUT_sec = 2 # (in seconds)
FELIX_GBT_OUT_SIZE_bytes = TCP_MTU_bytes # (in bytes)

FELIX_GENERATION_PERIOD = FelixDistribution.new FELIX_GBT_PERIOD_sec,
FelixDistribution::FELIX_MODE_HIGH_THROUGHOUT,
FELIX_GBT_SIZE_bytes,
FELIX_GBT_BUFFER_bytes,
FELIX_GBT_TIME_OUT_sec,
FELIX_GBT_OUT_SIZE_bytes
FELIX_GENERATION_SIZE = ConstantDistribution.new TCP_MTU_bytes*8 #distribution size in bits

Results

Throughput at the SWROD

With 10GBT links (each generating at 400MB/s=100KHz * 4KB) each felix server generates 4GB/s. It is expected for each SW_ROD to receive this data as no congestion is expected.

It can be seen that the very first 0.01s there was less data received by the SW_RODs. This is because data is first buffered in the felix servers.

Latency

A big increase in latecy is observed cause by the felix server bursts (last packet in the burst will have higher latency becuase of the queueing effect).

IMPORTANT: It is important to note that the latency here the network latency: starting to count when the packet leaves the felix server. It does not include de latency added by the felix server buffering (counting

when the message arrives from the GBT).

Just as as estimation, **the added latency for the queueing effect is 2500us** (the latency missing in the plot) is approximatly 1/400 seconds = 2.5ms=2500us (the 1MB buffer is filled at 400MB/s, so it is flushed every 1/400 seconds).

Link Usage

Felix servers

As expected both links are equally used in average (~2GB/s).

It is important to note the difference with the previous basic scenario where both links where exactly equally used (now only equally in average). This is because before the bonded link was doing a RR with each packet. Now the bonded link chooses always the same path for the same connection (and flows are configured so that 5 flows go through one link and the other 5 through the other link). This is much more realistic than the previous scenario as the bonded link hash will be probably based on connection properties.

switches

Same is observed at the switches.

Queue sizes

Note on the queue plots: the figures plot the **MAXIMUM** queue size in a given sampling period (in this case samplingPeriod=0.01s). This is because we are interested in the queue required to achieve no discards. In the legends of the figures, the **TIME _ AVERAGE** is shown: this is the queue size average taking time into consideration => queueSize{i} / totalTimeWithSize{i}. See SamplerLogger and TimeAvg definition for more details.

Felix servers

The output-queues on the felix server NICs are considerably bigger (which affects latency as seen before). This is because the GBT messages are buffered and causing a burst of packets when flushed. This burst of packets from the felix application is buffered at the felix NIC.
The maximum usage of the queue reached 3.5MB, which means that 3-4 connection buffers are flush simultaneously in average. The timeAverage usege of the buffer is much less ~300kB.
The big difference between the maxium usage and the avgUsage denotes that the queues increase with bursts (comming from the flushing of the buffers) and quickly emptied.

Switches

As expected and also observed in the previous basic scenario the queues at the switches are always completly empty. This is because all bursts are absorved by the felix NICs and then flows don't share ports (incoming traffic from a 40Gbps link going to a 40Gbps link).

Performance

Number of generated packets: 2.666 M ==> 2 servers generating 2.666 Mpackets/s (4GB/s with MTU=1500B) during 0.5 seconds =>
Total metric points (not including T) logged to Scilab: 2023766

Simulation execution:

- Initialization TOTAL time: 40398 (ms) [in basic scenario 24384 (ms)]
- Simulation time (not including init): 93930 (ms) [in basic scenario 73334 (ms)]
- **TOTAL execution time: 134649 (ms) [in basic scenario 98540 (ms)]**

=> ~0.05ms of execution per simulated packet (not including init time) **[in basic scenario 0.035 (ms)]**

Compared to the basic scenario:

- initialization time duplicated: this is becuase lot more parameters are read from scilab (x7 for each flow, now with 10 flows per felix)
- Simulation time per packet increases <x2 : although the implementation of the MultiFlow and the FelixDistribution is not very performant and can be improved, it affects the overall simulation performance slightly.

Tests with 24 GBT links per felix (76.8 Gbps) per felix - 13 felix servers

Configuration

git commit: b33d6c8..cf1bc38

backup:
/afs/cern.ch/work/m/mbonaven/public/SimuResults/PhaseI/LArSlice_1_to_1/LAr_slice_13felix_20161206

Topology same as in previous scenario, but with full LAr slice of 13 felix servers (instead of only 2) and each server generating 80Gbps (24 GBT links)

NUMBER_OF_FELIX_SERVERS = 13 # this generates 1:1 connections with sw_rod, so NUMBER_OF_FELIX_SERVERS=numberOfSWRODServers
LINK_BW_BITS_S = 40 * G # 40 Gbps
FELIX_FLOW_PRIORITY = 0
FELIX_GBT_ELINKS = 24 # #GBT e-links in each felix server. There will be one flow created per e-link (because there will be 1 thread, one connection per e-link)

# felix distributions
FELIX_GBT_PERIOD_sec = ExponentialDistribution.new 1.0 / (100*K) # distribution period in seconds
FELIX_GBT_SIZE_bytes = NormalDistribution.new 4.0*K, 1.0*K # (in bytes)
FELIX_GBT_BUFFER_bytes = 1*M # (in bytes)
FELIX_GBT_TIME_OUT_sec = 2 # (in seconds)
FELIX_GBT_OUT_SIZE_bytes = TCP_MTU_bytes # (in bytes)

#FELIX_GENERATION_PERIOD = ExponentialDistribution.new (1.0 * FELIX_GBT_ELINKS) / (70*M) # distribution period in seconds
#FELIX_GENERATION_PERIOD = ExponentialDistribution.new (1.0 * FELIX_GBT_ELINKS) / (1*M) # distribution period in seconds

```
#FELIX_GENERATION_SIZE = ConstantDistribution.new 1.0*K #distribution size in bits
FELIX_GENERATION_PERIOD = FelixDistribution.new FELIX_GBT_PERIOD_sec,
FelixDistribution::FELIX_MODE_HIGH_THROUGHOUT,
FELIX_GBT_SIZE_bytes,
FELIX_GBT_BUFFER_bytes,
FELIX_GBT_TIME_OUT_sec,
FELIX_GBT_OUT_SIZE_bytes
FELIX_GENERATION_SIZE = ConstantDistribution.new TCP_MTU_bytes*8 #distribution size in bits
```

Results

Throughput at the SWROD

With 24GBT links (each generating at 400MB/s=100KHz * 4KB) each felix server generates 9.6GB/s = 76.8Gbps. It is expected for each SW_ROD to receive this data as no congestion is expected.

It can be seen that the very first 0.01s there was less data received by the SW_RODs. This is because data is first buffered in the felix servers.

 Latency

A big increase in latecy is observed cause by the felix server bursts (last packet in the burst will have higher latency becuase of the queueing effect).

IMPORTANT: It is important to note that the latency here the network latency: starting to count when the packet leaves the felix server. It does not include de latency added by the felix server buffering (counting when the message arrives from the GBT).

Just as as estimation, **the added latency for the queueing effect is 2500us** (the latency missing in the plot) is approximatly 1/400 seconds = 2.5ms=2500us (the 1MB buffer is filled at 400MB/s, so it is flushed every 1/400 seconds).

Link Usage

Felix servers

As expected both links are equally used in average (~4.8GB/s).

switches

Same is observed at the switches, all 26 links are almost with the same usage

Queue sizes

Note on the queue plots: the figures plot the **MAXIMUM** queue size in a given sampling period (in this case samplingPeriod=0.01s). This is because we are interested in the queue required to achieve no discards. In the legends of the figures, the **TIME _ AVERAGE** is shown: this is the queue size average taking time into consideration => queueSize{i} / totalTimeWithSize{i}. See SamplerLogger and TimeAvg definition for more details.

Felix servers

The output-queues on the felix server NICs are considerably bigger (which affects latency as seen before). This is because the GBT messages are buffered and causing a burst of packets when flushed. This burst of packets from the felix application is buffered at the felix NIC.
The maximum usage of the queue reached 9.5MB at the very begging (where more buffer flushes coincide) and 6.5MB afterwards. This means that 6-10 connection buffers are flush simultaneously in average. The timeAverage use of the buffer is ~2MB.

It is important to note that this simulated buffers dont exist in the read world. In the real world the 1MB buffer for each connection sits in main memory, which is flushed to the NIC buffer by the OS. The NIC buffer wont have enough capacity to buffer all the information, so the OS will transfer data when possible. So, although there wont be any single buffer with the size shown in the plots (it will be distributed in main memory, OS and NIC memory), the delay/latency produced by this buffer (see plot above) will exisit in the real world.

Switches

As expected and also observed in the previous basic scenario the queues at the switches are always completly empty. This is because all bursts are absorved by the felix NICs and then flows don't share ports (incoming traffic from a 40Gbps link going to a 40Gbps link).

Performance

Number of generated packets: 41.6M packets simulated => each felix generates 76.8Gbps (9.6GB/s). The TCP MTU is 1500B, so each felix generates 6.4M packets/s. There are 13 felix servers => total of 83.2M packets/s. We simulated 0.5s => 41.6M packets simulated

Simulation execution:

- Initialization TOTAL time: 321458 (ms)
- Simulation time (not including init): 9465597 (ms)
- **TOTAL execution time: 9788834 (ms) --> 2h 42min**

=> ~0.23ms of execution per simulated packet (not including init time) **[in basic scenario 0.035 (ms)]**

Compared to the previous scenario:

- initialization time increased x100 times: this is becuase lot more parameters are read from scilab (x7 for each flow, now with 26 flows per felix with 13 servers)
- execution time increases x10 times: this is because we are now using 338 FelixServerDistributions. Each of these distributions simulate the arrival of each indiviual message from the GBT links (@100kHz), so it needs to generate several random numbers which affects performance.

-- MatiasAlejandroBonaventura - 2016-12-05

This topic: Main > PhaseISimulation_FelixDistribution
Topic revision: r5 - 2016-12-08 - unknown