May 2022

DYNAMIC FLOW ASSIGNMENT IN AI/ML APPLICATIONS BASED ON OBSERVABILITY AND TELEMTRY INFORMATION

Rajendra Kumar Thirumurthi

Deepak Kumar

DYNAMIC FLOW ASSIGNMENT IN AI/ML APPLICATIONS BASED ON OBSERVABILITY AND TELEMTRY INFORMATION

AUTHORS:
Rajendra Kumar Thirumurthi
Deepak Kumar

ABSTRACT

For Artificial Intelligence (AI), Machine Learning (ML), or other Deep Learning applications involving network fabric deployments, the fabric is typically deployed as a Layer 3 (L3) fabric without overlays such that heterogeneous high throughput application traffic is remote direct memory access (RDMA) over Converged Ethernet version 2 (RoCEv2) (e.g., involving User Datagram Protocol (UDP)/Distributed Deep Learning applications) and Transmission Control Protocol (TCP) (e.g., involving storage/HDFC/Network File System (NFS) applications). In such environments, when Graphic Processor Units (GPUs) are increased, deployment completion time starts increasing, so there is typically no benefit from adding extra GPUs due to congestion/microbursts that can be caused by application traffic and quality of service (QoS) configuration options. Presented herein is a solution to improve application performance for RDMA/AI/ML fabric deployments by horizontally scaling computing and storage clusters without causing PAUSE frames to be generated from the network, as PAUSE frames reduce throughput, increase latency, and cause horizontal scaling to stop working. Broadly, the solution presented herein involves monitoring local and remote events at the data path level and moving flows using weighted path selection. Flows may be moved by modifying access control lists (ACLs) and access control entries (ACEs) for load balancing and redirection components. Further, the weights can be auto-adjusted based on various system level events.

DETAILED DESCRIPTION

For network fabric deployments, deep learning/AI/ML deployment completion time should not exponentially increase with a higher number of GPUs being used to deploy a network configuration. Consider various types of traffic, such as UDP and TCP traffic, typically associated with a fabric network deployment. Since RDMA traffic is UDP it is

associated with a No-drop requirement. Conversely, TCP traffic can involve bursts sharing the same port in a network fabric, which can cause congestion. Further, TCP traffic is typically higher bandwidth and requires a higher queue priority, but it is often acceptable to drop TCP traffic.

Based on the various types of traffic that may be present in a given network, this proposal provides a solution through which observability may be used to monitor network traffic and move flows based on various algorithms, as discussed in further detail herein. Figure 1, below, illustrates an architecture through which features of the solution provided by this proposal may be implemented.

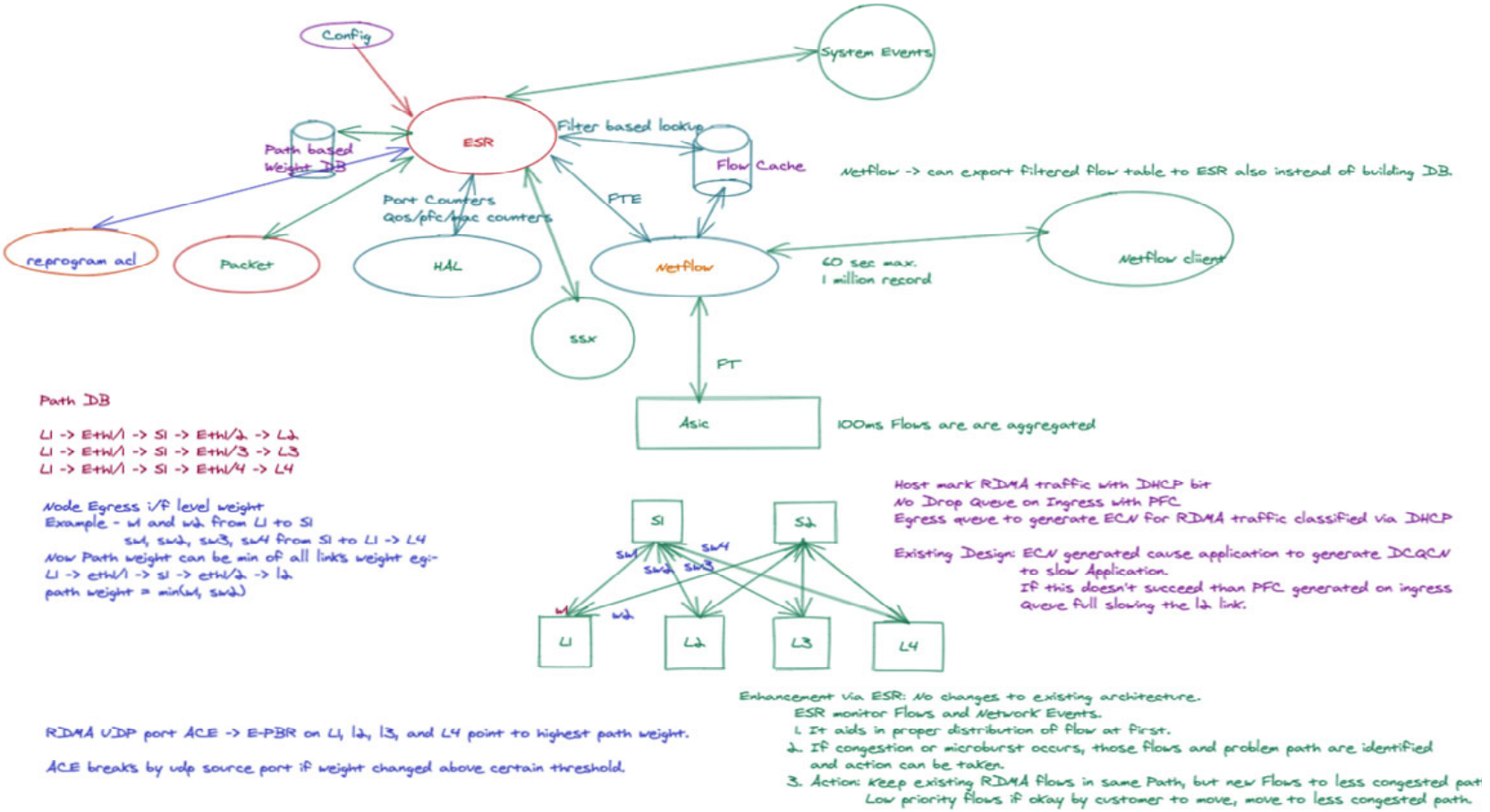

Figure 1: Example Solution Architecture

As illustrated in Figure 1, queue depth, microbursts, and various flow events can be monitored, such that information from the events can be used to take actions to move one or more flow(s) based on various algorithms, as discussed in further detail below. In some instances, flows can be monitored based on user/network administrator configurations provided for one or more network components (e.g., switches, etc.). In one instance, an example component configuration may be provided as follows:

```
Ip access-list acl1
            10 permit ip any range 5,1000 any udp 4791    -> High
Priority
Ip access-list acl2
            10 permit ip any range any any tcp range 9864, 9869  -
> Low Priority
track 1 ip sla 1 reachability
 component device-group <dgroup_name>
      probe icmp
            node 1.1.1.1
            node 2.2.2.2
            node 3.3.3.3
            node 4.4.4.4
 component service <svc_name>
            device-group <dgroup_name>
            access-list acl1
            access-list acl2
            ingress interface <svi/l3-intf>
            failaction bucket-distribute with-load [move acl2]
            no shut
```

In accordance with this proposal, weighted load-balancing can be provided for monitored flows such that TCP flows can be moved, if allowed (per user/network administrator configuration), and RoCEv2 flows can be maintained for an established path. Future RoCEv2 flows may be assigned to a higher weighted path by updating the weights of a path, which is important, as RoCEv2 flows are averse to being dropped. Thus, the solution may involve identifying flows that can be moved.

Various considerations may be provided for identifying flows that may be moved. All flows are not moved under the solution of this proposal, as this can cause trust issues

and may not decrease congestion. For example, flows that may potentially cause application issues might not be moved. Further, it may be desirable to move flows that may be more resilient to packet drops, such as TCP flows.

In one example, large flows that may be moved can be identified if they exceed any combination of a minimum monitoring time threshold, a byte threshold, and/or a minimum bandwidth threshold. Further, microburst flows can be identified through various Flow Table Events (FTEs) in which consistent microburst flows identification can be provided by identifying a given flow associated with microbursts and determining whether the flow matches a given event for a threshold number of times, which can help to avoid misidentifying short-lived flow issues.

Further congestion/delay for all possible paths can be monitored. Leaf paths can receive local monitoring information and also weighting information from other network elements (switches). For example, Internet Protocol (IP) Service Level Agreement (SLA) extensions and/or Internet Control Message Protocol (ICMP) type-length-value (TLV) objects can be provided to carry such information. Load-balancing weight logic for each leaf path can apply weighting based on this received information for a single direction of a flow.

When an existing flow is finished, it is reverted to the latest weight as determined by load-balancing weight logic such that new flows can be distributed based on congestion/delay. In some instances, low priority flows can be moved to alternate paths to help avoid congestion issues.

Consider an example algorithm that can be used to determine path weights and move flows in accordance with the solution provided by this proposal. In instance, the algorithm, may involve changing the weight of a given path based on the number of paths upon determining whether any error has occurred on an egress interface involving a leaf path. Consider an example involving four egress interfaces for a switch such that the interface paths may have an equal weight, such as: 8,8,8,8 that may be double the number of paths, for flexibility (e.g., if the port range is 400, it can be broken into four parts of 100 each).

One or more local and/or remote events can be determined, which may trigger moving one or more flows. For example, one potential event may be a reduction in

bandwidth (e.g., a 5% range, after only 25% of bandwidth is remaining), which may trigger load-balancing weight logic for the switch to reduce the egress interface weight by 1 from an existing value. For any existing flow information that may be present in the flow cache of the switch, the ACL policy of the port can be broken in such a way that existing flows are not altered and new flows are weight distributed according to the new weight. For example, if existing flows are present on ports 1, 2, 3, and 4, the existing flows would remain weighted at 8,8,8,8. Any new flows will be assigned to a port other than 1, 2, 3, and 4, so the algorithm can include breaking 5-400 in 4 parts, having weights 7,8,8,8.

In another example, a potential event may involve a link being down or removal of a router, which may trigger load-balancing weight logic to change the weight of a path to zero and distribute all flows to other remaining nodes. In another example, a potential event may involve a new link coming up or the addition of a router, which may trigger load-balancing weight logic to change the weight for one or more paths after a threshold period of time based on bandwidth. Other events may include Policy Flow Control (PFC) events, Explicit Congestion Notification (ECN) events, packet drop events, buffer drop events, or other such events, which may trigger load-balancing weight logic to also change the weight for one or paths after a threshold period of time based on bandwidth.

In summary, the solution provided herein provides for the ability to monitor local and remote events at the data path level (e.g., interface events, Media Access Control (MAC) events, queue depth events, buffer events, ECN events, etc.) and move flows using weighted path selection. Flows may be moved by modifying ACLs/ACEs for load balancing and redirection components. Additionally, the weights can be auto-adjusted based on various system level events, as described herein.