Understanding Fastcapa

Date of Publish: 2018-12-21

https://docs.hortonworks.com/

Contents

Fastcapa

The Fastcapa probe leverages a poll-mode, burst-oriented mechanism to capture packets from a network interface and transmit them efficiently to a Kafka topic. Each packet is wrapped within a single Kafka message and the current timestamp, as epoch microseconds in network byte order, is attached as the message's key.

The probe leverages Receive Side Scaling (RSS), a feature provided by some Ethernet devices that allows processing of received data to occur across multiple processes and logical cores. It does this by running a hash function on each packet, whose value assigns the packet to one receive queues. The total number and size of these receive queues are limited by the Ethernet device in use. More capable Ethernet devices offer a greater number and greater sized receive queues.

- Increasing the number of receive queues allows for greater parallelization of receive side processing.
- Increasing the size of each receive queue can allow the probe to handle larger, temporary spikes of network packets that can often occur.

A set of receive workers, each assigned to a unique logical core, is responsible for fetching packets from the receive queues. There can be only one receive worker for each receive queue. The receive workers continually poll the receive queues and attempt to fetch multiple packets on each request. The maximum number of packets fetched in one request is known as the 'burst size'. If the receive worker actually receives 'burst size' packets, then it is likely that the queue is under pressure and more packets are available. In this case, the worker immediately fetches another 'burst size' set of packets. It repeats this process up to a fixed number of times while the queue is under pressure.

The receive workers then enqueue the received packets into a fixed size ring buffer known as a transmit ring. There is always one transmit ring for each receive queue. A set of transmit workers then dequeue packets from the transmit rings. There can be one or more transmit workers assigned to any single transmit ring. Each transmit worker has its own unique connection to Kafka.

- Increasing the number of transmit workers allows for greater parallelization when writing data to Kafka.
- Increasing the size of the transmit rings allows the probe to better handle temporary interruptions and latency when writing to Kafka.

After receiving the network packets from the transmit worker, the Kafka client library internally maintains its own send queue of messages. Multiple threads are then responsible for managing this queue and creating batches of messages that are sent in bulk to a Kafka broker. No control is exercised over this additional send queue and its worker threads, which can be an impediment to performance.