Throttling Data Load

☆ 11.8

If large amounts of data are poured into Data Hub, performance can strongly be affected. To avoid performance issues, control the stream of data entering Data Hub.

If too much data Loads into the Data Hub all at once, the internal processes can be overwhelmed causing composition and publication to take a very long time to complete. You can actually take several thread dumps of the JVM activity and compare them to determine if Data Hub is still processing. The solution is to construct an application that applies back pressure or throttles the data load process. Other things that can affect Data Hub processing include inadequate JVM resources or inadequate hardware resources.

The way that the Data Hub development team tests Data Hub, and the way that we advise handling loads is to throttle the data load rather than dumping it in all at once. When you dump in a large amount of data all at once, it is similar to taking a very large bite of food. Eventually, you chew it all down, but it slows down the process of ingesting all the food. The best way to achieve the back pressure or throttling is to monitor the pending raw item count in the Data Hub (through the RESTful API). Once the count drops to a certain threshold, you can load more data. The threshold is highly dependent upon the hardware resources available to the JVM and requires experimentation within the environment to determine the best values. For example, in one of our test environments, we monitor the pending raw item count. When it drops below 10k items, we load more data. Throttling is the best way to get data loaded and processed in the fastest way possible. However, it requires some experimentation to determine what works best for the environment and the shape of the data.

A common solution for throttling/back pressure is to use message-oriented middleware (for example, ActiveMQ, RabbitMQ, and so on) in front of the Data Hub's Spring Integration endpoint for importing data. The reason for using message-oriented middleware is that working with loosely coupled systems by passing messages is much easier to design and debug. We have seen a number of customers use message-oriented middleware for this purpose with great success. Furthermore, employing the concepts of throttling data becomes easier when they are mapped to consuming and producing messages. In fact, speaking more generically, the whole idea of providing back pressure/throttling to any system is most typically based on a solution involving messaging. Dealing with message consumers and message producers is a much more stable manner for developing a system because it lends itself to simpler, well-known architectural patterns rather than trying to reinvent the wheel. Also, the vast majority of high scale systems in the world are based on and built using the concepts of message consumers and message producers.