

Pragmatic Advice for Handling Market Data Rate Increases

Robert A. Van Valzah

Copyright © 2007 29West, Inc.
March 22, 2007

Table of Contents

1. Introduction.....	1
2. Focus More on Measuring Application Message Latency and Less on Measuring Data Rates	2
3. Assume You Can't Always Keep Up With Spikes.....	2
4. Establish a Latency Boundary for Each Receiver	3
5. Test Application Stability Under Stress	4
6. Test Application Stability with Rising Market Data Rates	4
7. Achieve Stability Through Automatic Policy, Not Human Intervention	5
8. Use Uncertainty to Your Advantage.....	5

Abstract

Our work developing high-performance, low-latency messaging products naturally brings us into contact with firms that are forced to deal with rising market data rates. Some firms seem to have more effective strategies and techniques than others for dealing with rising rates. In this paper, we present pragmatic technical advice and strategies based on what seem to be the best practices in the industry that we have seen.

1. Introduction

It seems like everybody in our industry cites the recent dramatic rise in market data rates as a evidence that some action must be taken in response. The predicted future rise of market data rates is often given as a reason to buy a vendor's product (ours included). At 29West (<http://www.29West.Com/>), we would of course be thrilled if you bought our **LBM** (<http://www.29West.Com/products/lbm/>) or **UME** (<http://www.29West.Com/products/ume/>) messaging products. However we've come to believe that possession of the right software tools must be coupled with the right techniques and strategies in order to deal most effectively with current and future market data rates.

Following sections give pragmatic, technical advice for action that can be taken to cope with current and future market data rates. This represents the "best practices" we've seen in the industry among companies committed to handling increasing market data rates with the lowest-possible latency.

The advice is separated into sections, each of which has a central theme. Benefits will be likely if advice from one section is taken independently of others. But the greatest benefits will come from integrating all of the advice into a cohesive strategy for dealing with market data rate increases.

The sections are presented roughly in order from those which seem to be most often practiced by firms that seem to be effectively coping with rising market data rates down to those which are more weakly correlated with effective coping.

2. Focus More on Measuring Application Message Latency and Less on Measuring Data Rates

We know of no better early warning indicator than application message latency. It's the "canary in the coal mine." Increases in application message latency will often precede more significant problems in a system. Application message latency may be caused by problems in the network, the operating system, the messaging system, or within the application itself.

If you only measure one thing, measure the latency for each message passing through your application. Compare the latency message-by-message. Compare it day-by-day. Use it for forensic analysis when something goes wrong.

Once you're measuring application message latency, you can work to improve it. In contrast, market data arrival rates are generally outside your control. It seems better to measure things that you can change.

Data rate measurements may be useful, but we've found them to be less useful than application message latency measurements. Rate measurements suffer from a "hidden average" effect while message latency measurements do not.

Rate measurements are often expressed in units like messages per second or bits per second with no apparent average. But rate measurements must always be made (and averaged) over an interval that is often unspecified. Even when specified, the measurement interval is often far too long to provide data that's useful in diagnosing latency problems.

For example, network traffic rates are often given in terms of megabits per *second* yet measured only once every *30 seconds*. This averages the data rate over a 30-second interval. Even when a 100 mbps network shows only 20% utilization, it still may have been 100% busy for over 5 seconds if the measurement interval was 30 seconds.

If data rate measurements are to be used in diagnosing latency problems, then we recommend making measurements at intervals that are a fraction of the latency boundary for the application. See Section 4 below for details.

It's very important to make latency measurements in production applications running under real market conditions. Latency should also be measured end-to-end. How much time elapsed between the time a market data message was produced (ideally taken from the exchange time stamp) till the time of final consumption by your application? It's good to make intermediate latency measurements along the way, but don't stop till you hit the end of the chain of events triggered by the arrival of a market data message.

In summary, it may seem counter-intuitive to focus on measuring latency instead of rates when the goal is dealing with increasing rates. However, the actual measurement of consistently low latencies is the only certain evidence we know of that demonstrates effective handling of current market data rates. Coming sections will suggest ways that preparedness for future increases in market data rates can also be demonstrated.

3. Assume You Can't Always Keep Up With Spikes

Why? There will always be spikes. If your application is ready to receive the next message just as it arrives, then that message can be processed with the lowest-possible latency. However, if your application is still processing the

previous message when the next one arrives, then there will be some latency before the arriving message can be processed.

It only takes two messages to make a spike--the second just has to arrive before the first has been completely processed. If anything, the trend in market data is toward ever larger and more frequent spikes. Hence we recommend accepting spikes as a fact of life and planning to deal with them in the best possible way rather than designing systems as though spikes don't exist.

How will your market data applications behave in response to a spike in market data rates? Broadly speaking, there are three possibilities.

- *No increase in latency*: Your application and the underlying infrastructure have enough reserve capacity to handle the spike without adding latency. This is the desired outcome, but it requires that unused capacity be held in reserve anticipating the spike's arrival. That may be prohibitively expensive. This is the case mentioned above where all previous messages have been completely processed before the next one arrives.
- *Graceful operation with increasing latency*: There is insufficient capacity to handle the spike in real time, so latency must be added to some messages by queuing them until they can be processed.
- *Unstable operation or crashing*: Once arriving messages are queued, there's the question of *how many* should be queued and/or *how long* they should be held. Queues are often fixed size requiring that a message be dropped when a message arrives for a full queue. A dropped message may lead to a request for retransmission of the lost message. Worse, it may trigger a "refresh," requiring retransmission of all state held by the receiver. These are the mechanisms that can lead to unstable operation or crashing as a consequence of spikes.

If you measure application message latency as suggested above in Section 2 and plot that with data rates, you'll be able to see if there's a correlation. Increasing latency with increasing rates is a sure sign of latency as a consequence of queuing. Quantifying the amount of latency due to queuing may help in justifying technology investments toward a goal of reducing latency.

Our **LBM** (<http://www.29West.Com/products/lbm/>) and **UME** (<http://www.29West.Com/products/ume/>) products have extensive features for measuring and monitoring latency.

4. Establish a Latency Boundary for Each Receiver

If spikes are inevitable, then some amount of latency due to queuing is inevitable. Latency-sensitive applications receiving spikes are faced with two alternatives:

- loss-free message delivery with additional queuing latency, or
- message loss with reduced queuing latency.

Some applications would prefer to deal with latency spikes by discarding older messages rather than spending time processing a long but lossless queue. Picking an age at which a message is too old to be worth processing may be difficult, but it may be the best way to keep message latency low and relevance high.

The best latency-sensitive applications that can tolerate loss understand this trade off and gracefully deal with loss as a means to keep latency low.

We've found it helpful if each receiving application is designed and built with a "latency boundary" that is appropriate for its needs. Below this boundary, latency is added as messages are queued, but no messages are lost. Above this boundary, the oldest messages are intentionally dropped to keep latency in check.

29West (<http://www.29West.Com/>) customers using our **LBM** (<http://www.29West.Com/products/lbm/>) product for demanding applications like automated market making often set latency boundaries of 100 ms. Other market data applications may have latency boundaries of 1 or perhaps even 10 seconds.

5. Test Application Stability Under Stress

Even systems that can easily handle today's rates under good conditions might have stability problems when things go wrong. Perhaps a new receiving application is added to the system, making more work for existing sources. Perhaps a network hardware failure causes some packet loss between a source and a receiver. These conditions and similar ones add stress to a system even when it's not dealing with peak market data rates. The more complex a system becomes, the more likely it is that a failure will cause trouble. The more often a system is changed, the more likely it is that a change will have unintended consequences that stress the system.

We recommend that systems be tested by playing back captured market data at its original speed while adding stress to the system and watching for stability problems or latency spikes. It may be impossible to think of every possible source of stress and to test for successfully tolerating it at today's rates. But the process of doing such stress testing ultimately leads to more robust systems that give the expected results, even under unexpected conditions.

The **LBM** (<http://www.29West.Com/products/lbm/>) and **UME** (<http://www.29West.Com/products/ume/>) products from 29West (<http://www.29West.Com/>) have the ability to simulate packet loss in receivers. We use this in our internal product testing and encourage our customers to use it in conducting stability tests of the applications they build using our products.

6. Test Application Stability with Rising Market Data Rates

A serious concern for market data teams is assuring stable application behavior as market data rates rise. Market data applications often exhibit crashes or other unstable behavior in response to short-term surges in market data rates. Hence there's a natural interest in knowing maximum market data rates and assuring stable operation at those maximum rates.

In the old days when market data arrived over serial lines, maximum message rates could be easily calculated. Stability of systems under maximum load could be easily tested by artificially generating market data at line maximum rates or by playing back captured data at the maximum rate. The difference between average and peak rates was relatively small (e.g. 10x).

These days, market data arrives over packet-based IP networks making it much more difficult to calculate a maximum possible rate or to generate traffic at that rate. In many cases, the exchanges generating market data can't meaningfully specify maximum possible peak message rates today, let alone accurately predict maximum possible peak rates in the future. (This is yet another good reason to favor latency measurements over data rate measurements. See Section 2 above.) The difference between average and peak rates is now much larger--often 50x or more.

We believe it's best to capture arriving market data with high-resolution time stamps so that it can later be played back at multiples of the actual arrival rate for load testing. As suggested above, measure application latency while replaying. Compare message latency as you increase replay rates to find the rates where queuing and loss-recovery latency becomes significant.

Imagine an experiment where you successfully replayed market data captured on your busiest day to date at 2x the actual arrival rate. Wouldn't it be comforting to know that your applications wouldn't crash if market data arrived tomorrow at 2x today's rate? Wouldn't it be helpful if you could quantify the latency that would be added to your applications if market data rates jumped to 2x tomorrow while using today's hardware?

The **LBM** (<http://www.29West.Com/products/lbm/>) and **UME** (<http://www.29West.Com/products/ume/>) products from 29West (<http://www.29West.Com/>) have features that allow you to build applications that are stable at any market data rate. There are also features that allow fine-grained measurement of latency.

7. Achieve Stability Through Automatic Policy, Not Human Intervention

Messaging systems can be monitored to detect unstable operation. When trouble is detected, a human can be alerted to find the cause of the problem and remove it.

A classic example of this is a "crybaby receiver" causing a "NAK storm." A crybaby receiver (<http://www.29West.com/docs/THPM/group-rate-control.html#CRYBABY-RECEIVER>) is simply a receiver enduring so many lost packets that it generates many NAKs (Negative Acknowledgments) which request retransmission. The retransmissions at least add latency for lossless receivers. In some cases, retransmissions completely prevent even lossless receivers from receiving new messages.

Monitoring can detect a NAK storm and alert a human who can find and fix the crybaby receiver to allow data to once again flow to receivers without loss. We see this as an example of detecting unstable operation and relying on human response to restore stable operation.

We believe it's better to simply establish policies that prevent the system from reaching the point of unstable operation. Automatic policies should be in place that prevent sources from retransmitting so fast that receivers stop receiving new messages. With policies like this in place, human intervention is not required to maintain stable operation.

Such automatic policies do not remove the need for monitoring, but they do reduce the need for prompt human attention and intervention. They effectively shift the focus of monitoring toward capacity planning and forensics. See Section 2 above.

The **LBM** (<http://www.29West.Com/products/lbm/>) and **UME** (<http://www.29West.Com/products/ume/>) products from 29West (<http://www.29West.Com/>) allow policies to be set that limit the rate of retransmission so that the impact of crybaby receivers can be limited. Such policies allow stable operation of your applications even when some receivers are suffering massive losses.

8. Use Uncertainty to Your Advantage

If spikes are a fact of life and your applications won't always be able to keep up with them, then it may be important for your applications to take advantage of uncertainty. If processing a spike without loss would push your application over its established latency boundary, then it might be best to configure the messaging system to deliver unrecoverable loss events instead of adding additional latency attempting to recover lost messages. See Section 4 above for details.

This could leave your application in a position where it knows the current price, but also knows that it has missed some of the trades from the market that lead to the current price. Applications built requiring full knowledge of the market before trading may be at a disadvantage compared with applications that could still trade (though perhaps less aggressively) with current prices and an imperfect knowledge of market history.

Messaging systems can detect packet loss and request retransmission to repair the loss. This provides lossless message delivery to receiving applications, but adds latency at the receiver while the loss is repaired. This provides another opportunity for applications to deal with uncertainty.

The most latency-sensitive applications may want to know as soon as packet loss is detected by the messaging system. Such applications would continue to receive messages after the loss, hence avoiding the repair latency on them. If and when repair is completed, the application would then receive the older messages that had been lost. This requires the receiving application deal with uncertainty of loss repair and to make sense of messages that arrive in a different order than they were sent. In return for this additional complexity in the application, messages that did not have to be retransmitted will have no latency due to loss repair in the messaging system.

Avoiding repair latency requires sequence number coordination between the messaging system and the application. The **LBM** (<http://www.29West.Com/products/lbm/>) and **UME** (<http://www.29West.Com/products/ume/>) products from 29West (<http://www.29West.Com/>) have the necessary features to support applications that want to eliminate loss repair latency.