Closing the Batch Window with WebSphere MQ

Background

Traditional batch streams would typically comprise of a job or jobs that first perform a backup, and then update master files and then finally produce reports. This type of batch stream has been used by legacy applications for many years and at many mainframe shops this is pretty much the way that many batch streams still run.

For data integrity reasons file updates to the same file must be performed sequentially, a consequence of this that badly tuned JCL streams can place enormous stress on ever decreasing batch windows. This is compounded as demands for the availability of business critical online systems increase. As the demand increases so the available overnight batch window decreases. However, as the business grows and the amount of data increases accordingly, so the size of the files used increases and this leads to the batch process time extending. Point break will inevitably occur. The overnight batch will eventually take longer and longer, and the timeline will become more and more critical until eventually the time needed to run the batch job streams exceeds the time available.

Job Schedulers go some way to providing a mechanism to improve batch streams by providing monitoring job streams and utilizing automation to remove much of the manual intervention for submitting jobs in a timely and correct manner. Job schedulers alone cannot provide all the answers. The breaking down of huge monolithic jobs and the tuning of those job streams is a major step to improving the batch throughput times. This work, however, can be a major undertaking. It also relies on cooperation between application programmers and operation analysts, as it requires a detailed knowledge of the application flow. For maximum efficiency these job streams should be designed to use as much parallelism as possible.

Using WebSphere MQ

WebSphere MQ can be used as, what at first flush looks like, an unlikely ally in the battle of the shrinking batch window. Of course, the benefits that WebSphere MQ can bring to the batch window will differ depending on the business applications and way that those applications have been written and how much inter-application file sharing is taking place.

Let's look at a scenario where WebSphere MQ can be used as a mechanism to allow multiple batch jobs to run updates against the same master file. See the following figure:

Copyright © Abbydale Systems LLC. 2017-2020

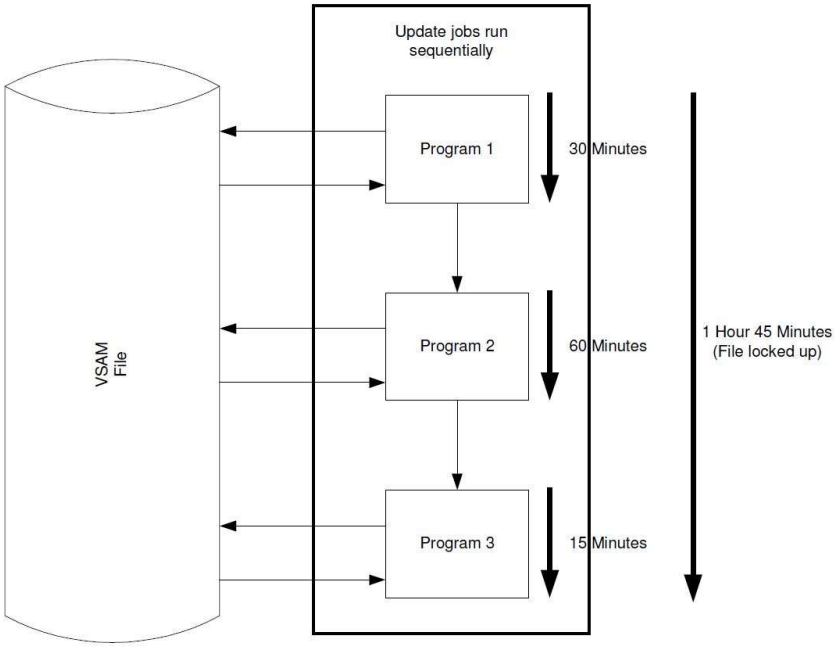

Figure 1

Let us assume that we have 3 batch jobs that are single step file update jobs. In a traditional sense these jobs will have to run sequentially. Let us further assume that Program 1 takes 30 wall clock minutes to end, then program 2 takes 60 wall clock minutes to run and finally program 3 takes 15 wall clock minutes. This means that the entire batch process will take 1 hour 45 wall clock minutes to complete before any reports can be run against that particular file, or indeed any other file that are exclusively enqueued by the batch process. The whole situation can be compounded if the batch stream gets interrupted or delayed.

Now suppose that WebSphere MQ were to be brought into the picture. Let us assume that a single program is written to perform the file updates based upon the contents of messages placed on a single queue. If those same three programs in the previous example were rewritten to drop a 'update the file with this' message onto this single common queue then these programs can run in parallel. Theoretically this could trim as much as 45 minutes off the 'file unavailability' time. See the following figure.

Copyright © Abbydale Systems LLC. 2017-2020

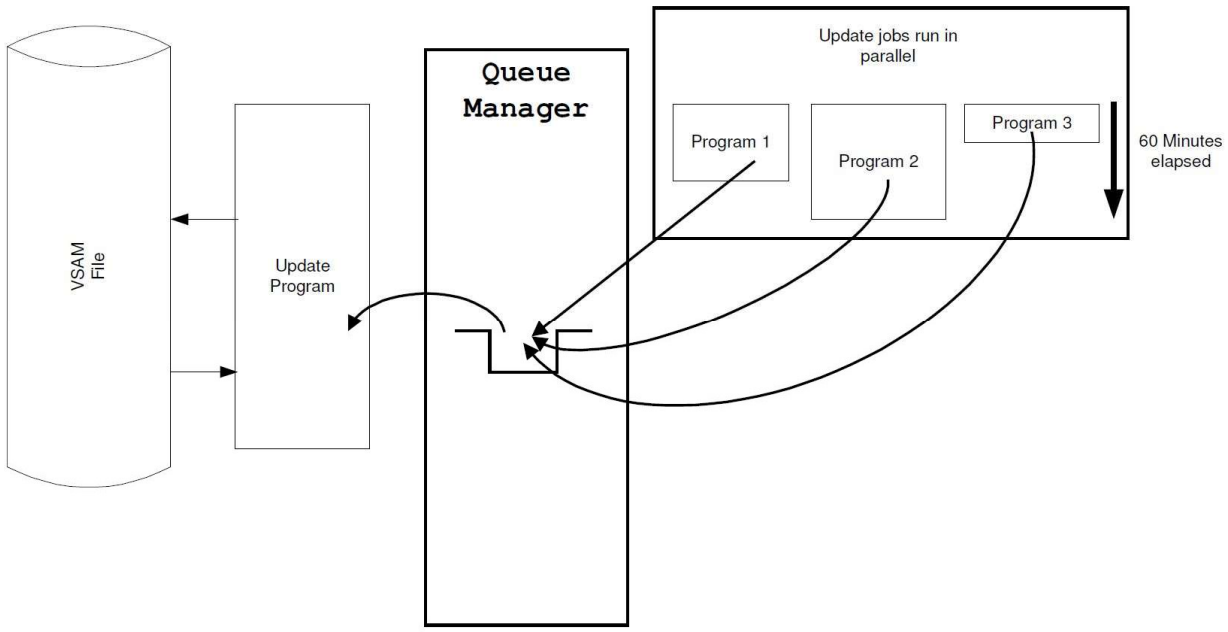

Figure 2

You should also bear in mind that using this method you could also have programs from other systems and platforms update the file at same time. If this 'file update' WebSphere MQ program were to be a CICS transaction, then the whole process could take place without the need to drop the file from CICS.

It may also be that the programs that previously updated the file will run quicker than before as the I/O overhead is removed. The writing of a message to a queue is a very fast operation, especially if this message does not require a response.

Additional benefits are that the use of persistent messages guarantees that updates to the file will take place, regardless of if the update program is currently available or not.

This concept raises some very interesting possibilities for, not just the batch window, but for the CICS processing and, potentially, porting applications to other platforms.

A working copy of an example of how to update a VSAM file using a WebSphere MQ enabled program is available for our website at:

https://abbydalesystems.com/Q2VSAM.php

Copyright © Abbydale Systems LLC. 2017-2020