A New Batch Service Level

BMC Compuware ThruPut Manager
Uncomplicates Batch Initiation Performance
Compuware joins BMC to empower the next generation of developers to mainstream the mainframe. With unparalleled mainframe operations management and agile application development and delivery, BMC Compuware provides a mainframe-inclusive DevOps toolchain that increases volume and velocity.

Bring apps that run on the mainframe to market faster with increased quality throughout the process of analyzing, building, testing, deploying, monitoring, and tuning apps and services. Our combined offering accelerates each client’s evolution to an Autonomous Digital Enterprise.
Introduction

Console operators have historically been the focal point for all things mainframe. On their desks, next to their consoles can be found the bane of their existence—a telephone, for which everyone seems to know the number. Anybody and everybody hesitate NOT in the least to call them and ask them to solve their problems. They are left to juggle a mixture of chainsaws, anvils, and rabid squirrels, seemingly responsible for everything, and at the same time relatively helpless amidst the myriad of pressures endemic to their craft.

Also, historically, the service level for batch we have established and can sustain with some exactitude has been based upon job completion time-of-day. That being a job way down in a schedule of jobs—like down at the end—that job that must finish before the online regions can be started, for example. In some shops, this practice can be like hitting a barn door; the target is made so big, they generally won’t miss it, unless something falls from the sky and pins them down—good for them. Or, the other extreme, where batch has become a daily wrestling match with the clock. Regardless, this form of Service Level Agreement (SLA) is critical and essential to keep the business humming.

About the Author

Kelly Vogt is a BMC Compuware solution consultant with expertise in BMC Compuware ThruPut Manager for automated batch processing as well as BMC Compuware Strobe for application performance management and analysis. He recently retired from Humana Inc. where he managed an evolving batch strategy, improving batch services for his customers, while reducing operating costs and risks by fully automating batch processing. He has 39 years of z/OS systems programming, performance tuning, capacity planning, operations, and management experience.
Time-to-Initiate

But the service level we’ve also needed for decades, yet found impossible to achieve, stems from the operator’s most beloved question: When will my job start? That’s a great question, but not one we can answer with any precision; “soon” or “ASAP” being the most common answers … resulting in more phone calls (rinse/repeat). This is essentially a lack of control that has a serious impact on batch performance. When harried, or when priorities are not strictly defined, the operator’s tendency is simply to force the job into execution leading to over-initiation, and subsequent general degradation of batch throughput, missed SLAs, etc.

So, why is it so difficult to answer this simple question? The operation of mainframe batch at modern volumes is an intricate exercise in Queuing Theory, where the more complex the initiator/job class structure, the more difficult to intuit an estimate in real time. Job arrival rates, inter-queue job priorities, and multiple job class initiator assignments further complicate determining an answer. And finally, the supply of CPU, which is subject to change suddenly and dramatically for numerous reasons, can chuck all your good intentions out the window to pave a road somewhere.

What is needed is simplification AND automation. The fundamental problem is that mainframe batch is still manually operated. Manual operation was sufficient for the early decades, but as volumes have increased, it has become exponentially more difficult to manage using unscalable, antiquated methods. And of course, to err is human … not something you want to base meeting your SLAs upon.
Why Is This So Difficult?

So, why has batch initiation performance become so complicated? Mainly, because the job class/initiator structure has become unwieldy, begging the question: Why do shops have so many job classes? Job classes have historically been used to organize and control the flow of jobs. In many shops, this has been taken to extremes, i.e. departments, even individuals, demanding and receiving their own job classes. The politicization of batch processing! Subsequently, all those job class queues must have some, generally more than one, initiators out there somewhere to process the jobs.

What administrators fail to understand when they cave in is, when you cut your pumpkin pie into umpteen pieces, nobody gets much dessert. Decisions, yours and IBM’s, come with consequences. When job classes A-Z and 0-9 were exhausted—36 being bad enough!—IBM gave us 8-character job classes, enabling you to “support” up to 512 execution job classes in a MAS. Admittedly, there is a serious cool factor in being able to assign witty names to your classes, but the ensuing complexity only makes the already difficult rush headlong towards impossible.

Excessive- or over-initiation is like over-booking an airplane flight; someone gets home late tonight, or worse, tomorrow! There is only so much CPU capacity for batch at any point in time for all work to run at machine speed. And when you’re getting your money’s worth out of your investment, a simple job step change from, for instance, an I/O bound program to a CPU soaker program, can severely upset how your Swiss watch runs. Workload performance starts to drag. Address spaces start waiting for CPU dispatch inflating your batch clock time and endangering your SLAs.

So, if you’ve already bloated your initiator/class structure to near bursting, or see yourself headed that way, what do you do? You first must admit it could work better using another methodology—a simpler method, where less is more.
There’s Got to Be a Better Way!

Enter **BMC Compuware ThruPut Manager**, a rules-based, policy-driven control system delivering automatic execution management for JES2 batch workloads. ThruPut Manager assumes responsibility for managing your initiators for peak batch performance. No over-initiation, where job execution drags, and batch throughput declines. Nor under-initiation, where CPU goes unused, and the opportunity to get more work completed is wasted.

ThruPut Manager assesses system instrumentation and batch service class PIs\(^2\) every 10 seconds and adjusts to ensure your system is always optimally initiated. Why start another initiator and select a job for it, if doing so will only over-commit your CPU and degrade batch performance? And for JES2 MAS implementations, for jobs that are LPAR agnostic, ThruPut Manager will balance workload across LPARs; no more lopsided systems.

Using ThruPut Manager policy, workloads can be categorized into five business importance\(^3\) levels and assigned individual initiation SLAs. You specify target, acceptable, and optionally critical time-to-initiate goals. ThruPut Manager will prefer selection of more important jobs, and jobs aging through their time-to-initiate goals, ahead of less important jobs, and jobs more likely to meet their SLAs. All jobs eventually bubble to the top of the queue for selection and jobs with higher business importance and urgency to initiate rise faster.

Users can see their job’s performance against time-to-initiate goals with an optional in-job report showing all delays before and after initiation as well as execution speed. Operators can monitor overall SLA performance using a modern interface, whereby from across the room, they can see how well their SLM policy is performing to goals.

ThruPut Manager will afford you that new service level we’ve always needed, plus your highest possible batch throughput, and improved application scalability. Oh, and your telephone will once again be your friend!

---

\(^1\) The optimal state of program performance with infinitesimal waits for CPU dispatch and I/O screams

\(^2\) WLM Performance Indexes

\(^3\) Importance of each job to your business