Workload Manager Batch Management

OS/390 & Storage Systems Technical Conference '99

Madrid, Spain

Session 504

Steve
Grabarits
S/390 Performance, Poughkeepsie, New York
STEVEJG@US.IBM.COM

So What's Wrong Anyway?

- Batch management is simple, right?

  Just execute all of my jobs:
  - in the right place
  - at the right time
  - where the required resources are available
  - as efficiently as possible
  - within my batch window (production work)

- Oh, and while you're at it don't forget about
  - load balancing
  - dirty inits (LSQA fragmentation)
  - enqueue contention
  - and the several hundred other things that can go wrong

Many Problems, Many Solutions

- Infrastructure: where and how it runs
  - The JESes
  - Workload Manager (WLM)
  - System Resource Manager (SRM)

- Schedulers: when it is available to run, where it runs
  - Smartbatch ThruPut Manager CA-7
    CONTROL-M OPC/ESA ...etc

- Optimizers: making it run faster
  - data in memory sexy fast hardware Smartbatch
    ...etc

Objectives of WLM Batch Management

- Dynamic, goal-oriented management of the time jobs spend waiting for an initiator
- Multisystem workload balancing
- Reduced operational demands (help? interference?)
- Improved reporting of job response time and pre-execution job delays
- Available with OS/390 JES2 V2R4
- Announced for OS/390 JES3 V2R8

- WLM doesn't do:
  - Deadline scheduling
  - Job history management
  - Consumable or multi-state resources
  - Resolve enqueue contention between initiators
  - Resolve LSQA fragmentation

It is probably as important to understand what WLM batch management is **not** trying to accomplish as it is to understand what it does do.

WLM batch management is not trying to replace your scheduling package. It is trying to get your staff out of the business of messing around with initiator definitions, job class selection lists, etc on an on-going basis. Those things will still need to change based on all sorts of business requirements (or politics). But why should any human need to monitor the job backlog to make sure enough initiators are started to service the load on a given service class?

It uses current system conditions and some recent historical data to decide whether or not your business goals (service policy) are being met, and if not which alterations in current resource allocations will get closer to that ideal. It does not keep a history on every jobname, as OPC/ESA or its like do. It does not attempt to project the impact of a particular job starting; it projects the impact of an additional job in a service class - the "average" job, whatever that is. It does not know when a job "must" be initiated in order to complete on time (since it has no job-based history), nor does it know about inter-job dependencies or networks.

WLM resource affinity support likewise does not implement resources with all of the bells and whistles of other packages: binary states are all for now. What it does do is give you a simple way, integrated into your performance goal specifications, to get jobs running on the right image(s). And it does so in a positive "run it there" sense, not via convoluted setup to ensure that a job "doesn't go anywhere else." Maybe you can delete some job classes if all they do now is enforce affinities

What WLM Manages

- Number of initiators
  - Queue delay samples identify delays due to lack of initiators
  - Goals and importance direct adjustments

- Placement of initiators
  - Awareness of system affinities
  - Work is displaced only if necessary
  - If work must be displaced, goals are used to minimize impact

- Exploitation/migration on a job class basis

- **The previously independent processes of initiation and performance management are now linked**

When a system (and the initiators that resided there) is taken down or "leaves unexpectedly", who/what picks up the work? Does a human need to realize that you just lost 20 production batch initiators and start some replacements elsewhere? How long does that take?

As your workload changes (merged any good datacenters lately?), how do you decide how many inits in each class? Decide what the job selection list for each should be? How fast can you react when some kind user dumps 500 uniquely-named jobs into the queue everyone uses for ad-hoc work? Does it take phone calls from dissatisfied users?

Which systems should be running jobs? At 0200? At 0800? At 1400? Do you actually change the initiator setup multiple times per day as the workload changes, or did you just find one that is (was?) "good enough" a long time ago and give up trying to optimize it?

Does your initiation setup match your business goals for performance?
How do you know?

Do you really enjoy doing this?

WLM can change the setup many times per day, **if** your business goals and system conditions warrant the change.

Prerequisites (aka The Bill)

- Entire MAS must be OS/390 JES2 R4
  ...and staying there, ie stabilized

- ISV/local mod dependencies must be dealt with
  - JES2 has had extensive changes for WLM support and for other JES2 R4 function
  - Any code with its hands in JES's control blocks must be considered suspect
  - Example: ThruPut Manager has a new release to support WLM-managed initiators

- Reformat checkpoint to allow use of R4 functions
  - Be SURE you are stable first...going back requires all member hot start
  - Only JES2 R4 systems can participate in the MAS afterward

A batch work queue is a multisystem resource, so all code interacting with the queue must be at a compatible level.

In order to support batch management and resource affinity scheduling, the job queue elements (JQEs) in the JES2 checkpoint have been extended. The JQE extension lets JES2 keep track of new data like the scheduling environment requested for a job, the service class for a job, and some times that improve reporting about what happens to a job before execution begins. Details will follow.

In addition, JES2 R4 includes many changes unrelated to WLM batch support. New control blocks, new services, new queues, etc. At the very least all of these internal changes will require rework of code modifications done locally or by other vendor products. And of course most packages dealing with batch problems have their mitts in the JQEs too.

For these reasons, JES2 R4 itself has two "modes". If the JES2 R4 system initiates a cold start or if the new $ACTIVATE command is used, the checkpoint will be formatted with new control blocks. In all other cases the existing checkpoint format will be used. **Once the new checkpoint format is used, only JES2 R4 systems may participate in the MAS.** Returning to the old checkpoint format requires an all member hot start with JES2 APAR OW32920 applied. Without OW32920, a cold start is required.

Don't Underestimate JES2 R4 Migration

- Large-scale command syntax changes
- New checkpoint/JQE structure, CAT changes
  - New services **required** to access/update JQEs and CATs
  - Most batch management packages and local mods will require **changes not just rework**
- Only JES2 R4 systems allowed in MAS once new checkpoint format is $ACTIVATEd
- All member hotstart required to undo $ACTIVATE
- Job selection exits have changed (14, 49)
  - Exit 14 is used by most batch management packages
- JES2 now has to react to WLM policy activation once JES2 R4 functions are activated
  - All jobs are reclassified

WLM-Managed Job Classes

- WLM management is assigned on a job class basis
- Each job class now has a mode assigned via the JES2 init deck or $tjobclass command
  - MODE=JES is the default, same behavior as today
  - MODE=WLM designates WLM-managed job classes that run jobs in WLM-managed initiators
- Initiator definitions do not have to be updated to remove references to WLM-managed job classes

```
JOBCLASS(A)  MODE=JES
JOBCLASS(B)  MODE=WLM
JOBCLASS(C)  MODE=WLM
           :
INIT1 ABC
INIT2 BC
```

JES Initiators vs. WLM Initiators

JES initiators

- Started via INIT proc
- Select jobs from JES-managed job classes only
- Started and stopped by operators
- Active in goal and compatibility modes
- Consume job numbers
- Exits 14 and 49 called

WLM initiators

- Started via INIT proc
- Select jobs from WLM-managed job classes only
- Started and stopped by WLM based on
  - Available capacity
  - Goals
- Active in goal mode only
- Started under MSTR subsystem, so they do not consume job numbers
- Exit 49 called (not 14)

The actual initiator code is the same, regardless of whether the initiator is JES-managed or WLM-managed, and the bulk of the initiator code always runs.

JES-managed initiators is just a new name for the way initiators have behaved in the past. Each initiator is always either running a job or waiting for JES to feed it a job to run. If told to terminate, it will do so. There is no interaction with "MVS" to see whether running the job it is given is a "good idea" or if the job will in fact run at all. If no CPU is available for the type of job it is given, oh well MVS will have to figure that out. Any change to the selection criteria must be done by humans or automation.

WLM-managed initiators add the ability to react to changing conditions. A WLM-managed initiator can be awakened in order to change its selection criteria, in other words to allow it to select work from a different service class. Thus WLM can shift the initiator's insatiable appetite for jobs away from jobs that MVS will not give resources to and toward those that will run once initiated. It can also be told that it is no longer needed at all.

JES exit 14 is called only for JES-managed initiators. JES exit 49, a new exit, is called for both JES-managed and WLM-managed initiators. Exit 14 requires the user to provide an alternate selection if JES's selection is rejected, while exit 49 only allows the user to veto JES's selection. If exit 49 vetoes a selection, JES -- not the exit code -- will select another job to run.

See the JES2 Conversion Notebook Chapter 9 for JES2 R4 migration actions.

Batch Job Flow

READER

CONVERSION

- *Classify*
- *Queue for execution*

JES-managed job classes

WLM-managed job classes

Sampling

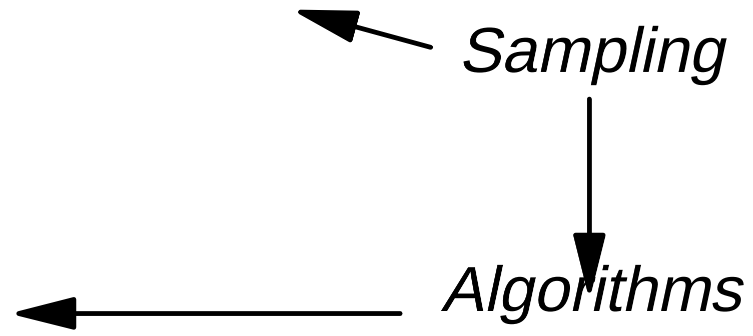
INITIATION

JES initiators

WLM initiators

Algorithms

EXECUTION



The flow of a job into the reader, through the converter, into initiation, execution, and output processing has not changed in any cosmic sense.

Every job is classified by JES2 during conversion (after exit 6) so that it can be queued by service class. JES2 actually maintains two different queues through the JQEs, one for JES-managed mode (oldest within priority) and the other for WLM-managed mode (oldest within service class). This classification call replaces the classification normally done by SRM during job selection; if the level of JES does not supply classification information to SRM, SRM will classify the job as before.

While on the execution queue, a job can be eligible to run (waiting only for an initiator) or it can be ineligible. For reporting purposes JES2 keeps track of 3 times while a job is on the execution queue. Details on the 3 buckets follow shortly. In order to manage batch queue delay, WLM uses the eligible time as queue time.

The other thing that happens while a job in a WLM-managed job class is on the execution queue is that its state is sampled and reported to SRM. A job waiting only for an initiator generates a Queue Delay state sample, a job in any other state generates an Other state sample. When a job is running in a WLM-managed job class with a velocity goal, its queue delay samples are used in the velocity calculation. JES2 samples the job queue while it holds the checkpoint lock and SRM uses that set of samples until JES2 gives it a new one. The shorter the checkpoint cycle, the more accurate batch queue delay samples will be. One cycle is defined here as the time for the checkpoint to be owned by each system in the MAS; anything less than the SRM sampling interval (0.25 seconds wall clock time) will not improve sampling accuracy.

Classification

- JES2 assigns a WLM service class to each batch job at the end of conversion
- Applies to jobs on both JES-managed and WLM-managed queues
- Service class can be changed by JES $tjob command prior to and during execution
- Whenever a service class is changed, SRM and JES2 stay in synch with each other

- Jobs are not ordered by priority on WLM-managed job queues; strictly FIFO queuing
- Classification rules can be used to assign different service classes to jobs based on their priority

```
           $dj19
JOB00019  $HASP890 JOB(JOBA)
$HASP890 JOB(JOBA)    STATUS=(AWAITING EXECUTION),CLASS=C,
$HASP890              PRIORITY=8,SYSAFF=(ANY),HOLD=(NONE),
$HASP890              CMDAUTH=(LOCAL),OFFS=(),SECLABEL=,
$HASP890              USERID=++++++++,SPOOL=(VOLUMES=(SPOOL1),
$HASP890              TGS=1,PERCENT=0.9523),ARM_ELEMENT=NO,
$HASP890              CARDS=2,REBUILD=NO,SRVCLASS=BATCHHI
$HASP890              SCHENV=,SCHENV_AFF=(R4TE),CC=()
```

```
  Subsystem-Type  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                Modify Rules for the Subsystem Type        Row 1 to 2 of 2
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : JES         Fold qualifier names?   Y  (Y or N)
Description  . . . Batch classification rules

Action codes:  A=After    C=Copy          M=Move     I=Insert rule
               B=Before   D=Delete row    R=Repeat   IS=Insert Sub-rule
          -------Qualifier-------------             -------Class--------
Action    Type       Name     Start                 Service     Report
                                          DEFAULTS: BATCHREG    ________
 ____  1  TC         C        ___                   BATCHREG    ________
 ____  2    PRI        >7       ___                 BATCHHI     ________
```

Pre-Execution Job Delays

READER

- *TYPRUN=JCLHOLD delay*

CONVERSION

- *TYPRUN=HOLD delay*
- *Operational delay (job hold, job class hold)*
- *Scheduling delay (class limit, duplicate jobnames)*
- *System or resource scheduling delay*
- *Queue delay (waiting for an initiator)*

INITIATION

EXECUTION

OUTPUT

Response time definition in OS390 R4 (JES2 only)

One of the problems lingering with batch was the definition of response time. Prior to OS/390 R4, user-initiated hold times (TYPRUN=HOLD) were included in the response time. In goal mode this often led installations to use velocity goals for batch, since velocity goals do not use the response time. Velocity goals have their own problems (see March 1997 SHARE, 1996 MVS Expo, or CMG 96 proceedings), so this was not really desirable for the long term. As of OS/390 R4 JES2, TYPRUN=HOLD time is not included or reported. If you are currently using velocity goals for batch work because of the response time definition, re-evaluate your position. If you decide to use response time goals for batch, you can make this change as soon as JES2 R4 functions have been $ACTIVATEd and it will save you from having to recalibrate the velocity goals when you enable WLM initiator management.

JES2 R4 with R4 functions activated maintains three time buckets:

Queued: job is waiting only for an initiator

Resource/System Affinity delay: job is waiting for availability of scheduling environment or system

Ineligible delay: job is waiting for some other reason. This corresponds to the Operational delay and Scheduling delay categories on the previous
. page.

If JES2 R4 functions are not activated, the times will be reported as before. Queued time will contain all 3 times, and the other two categories will be 0.

SMF 30 and SMF 72 Data

- Time to convert job(s)
- Time following conversion during which job(s) could not run due to eligible systems being down or not having needed resources (scheduling environment)
- Time following conversion during which job(s) could not run due to job hold, job class hold or limit, or duplicate jobname
- Time between end of conversion and initiation during which job was eligible for initiation

- Available at both the job (smf 30) and performance group / service class (smf 72) granularities
- Available in both goal mode and compatibility mode once JES2 R4 functions are activated.

Note: some of this data is in support of functions other than batch management. They are listed here for completeness.

Additional SMF 30 (job level) data:

- Job was reset into a new service class before initiation
- Job was reset after initiation
- Job was started using the $SJ command
- Job has been restarted
- Job scheduling environment

SMF 26 (job purge) data:

- Job scheduling environment
- Service class
- Executed by a WLM-managed initiator
- Job was started using the $SJ command

SMF 90

- Subtype 30 contains some new flags for existing functions
- New subtype 32 describes scheduling environment definition after a change

SMF 99

- Subtype 2 contains sysplex view of initiator work queues

RMF

- Job delay data is available on the WLMGL and WKLD reports in RMF monitor 1
- Job delay data is also available on the GROUP, SYSRTD, and SYSSUM reports in RMF monitor 3

```
                   W O R K L O A D   A C T I V I T Y


----------------------------------------------------------------
REPORT BY: POLICY=POLICY1 WORKLOAD=BATCH  SERVICE CLASS=HOTBATCH

TRANSACTIONS   TRANS.-TIME HHH.MM.SS.TTT
AVG     23.03  ACTUAL                3.883        . . .
MPL     23.03  EXECUTION             3.312
ENDED       6  QUEUED                  571
END/SEC  0.00  R/S AFFINITY              0
#SWAPS     13  CONVERSION              310
EXECUTD     0  INELIGIBLE                0
               STD DEV               6.617
```

```
                    RMF 2.4.0  Sysplex Summary - MCLXCF01        Line 1 of 4

 WLM Samples: 399     Systems: 2  Date: 08/04/97 Time: 17.36.40 Range: 100   Sec

                     >>>>>>>>XXXXXXXXXXXXXXXXXX<<<<<<<<

 Service Definition: PRIMSHFT             Installed at: 08/04/97, 17.25.29
      Active Policy: PRIMSHFT             Activated at: 08/04/97, 17.29.06

               ------- Goals versus Actuals --------  Trans --Avg. Resp. Time-
               Exec Vel  --- Response Time ---  Perf  Ended  WAIT EXECUT ACTUAL
 Name      T  I  Goal Act  ---Goal--- --Actual--  Indx  Rate   Time   Time   Time

 NRPRIME   S          86                              0.140 48.06  52.03  1.67M
           1  3       96  5.00M 70%        100%  0.50  0.100 0.188  1.945  2.134
           2  4    20  87                         0.23  0.000 0.000  0.000  0.000
           3  D       82                              0.040 2.80M  2.95M  5.75M

              RMF Response Time Components Data

The following details are available for NRPRIME/1
Press Enter to return to the report panel.

 Response Time Components:

  Actual         : 2.134
  Execution      : 1.945
  Wait           : 0.188
  - Queued       : 0.188
  - R/S Affinity : 0.000
  - Conversion   : 0.000
  - Ineligible   : 0.000
```

Place cursor on period 1 row
and press ENTER to get details

JES2 R4 functions not yet activated
so data is zero.

Algorithms: Adding Inits to a Service Class

- Policy adjustment - every 10 seconds wall clock time
  - A service class period is chosen as a receiver (needs help) based on the goals vs. actuals in the active policy
  - Queue delay samples are significant source of delay
  - Adding more initiators is projected to help measurably
  - Existing initiators for the service class are busy
  - CPU and storage to support another average job in the service class are available
  - **More than one initiator can be added at a time**

Selection of a receiver period and of queue delay as the bottleneck are unchanged from previous releases. Batch queue delay is only addressed based on the sysplex performance index (PI) since the queue is a sysplex-scoped resource (technically the queue is MAS-scoped, but the point is that more than one system is involved so the local PI cannot be used).

The additional initiators must improve the receiver's performance relative to the service policy in effect or the delay will not be addressed. If starting more initiators will replace queue delay with paging delay, WLM will learn that, project no improvement, and try to help using other resources. If existing initiators are not busy most of the time (90%ish), the presumption is that most jobs have affinities to other systems.

And of course, enough capacity must exist to run another **average** job...average for period 1 of the service class, based on recent history.

Resources may be obtained to add initiators by stealing from other work that is over-achieving its goals or is of lower importance.

Other cases where initiators may be added:

Resource adjustment, whose interval is adjusted by CPU speed, if the system is underutilized.

Jobs with affinity to only systems having no initiators for its service class may cause an initiator to be started. This attempts to guarantee that the job will not starve for initiation when the service class is meeting its goals.

Algorithms: Draining Inits from a Service Class

- Housekeeping - every 10 seconds wall clock time
  - Existing initiators swapped out
    - Indicates storage is needed for more important work
  - Existing initiators idle
    - Workload may have completed
    - Job affinities may have changed leaving this system eligible to run few/no jobs in the service class

- SRM algorithms tell WLM to remove the initiator from the pool for the service class
- Once the current job completes, WLM can
  - Re-assign the initiator to another service class immediately
  - Keep it around for a while and re-assign it later
  - Terminate it

Housekeeping and a change into compatibility mode are the only way WLM-managed initiators are removed from a service class. Policy adjustment can take MPL to reclaim storage, which then triggers housekeeping to reassign initiators.

There are a couple of other housekeeping cases, such as significant over-commitment of the CPU, not described. They all have the same idea though: use it or lose it. If the service class to which initiators are assigned cannot make good use of them, SRM returns them to WLM, and WLM will find a service class that can use them if possible. If things are so rosy that no service class is needy, they will sit idle for a while and if the condition persists they will be stopped. For the detail-oriented, "a while" is measured in minutes and is variable.

Syslog - what you see when a WLM-managed initiator starts

```
$HASP373 BAT1001  STARTED - WLM INIT  - SRVCLASS BATCH1   - SYS SY#C
IEF403I BAT1001 - STARTED - TIME=13.11.11
IWM034I PROCEDURE INIT STARTED FOR SUBSYSTEM JES2 730
APPLICATION ENVIRONMENT SYSBATCH
PARAMETERS SUB=MSTR
IEF196I        1 //INIT     JOB MSGLEVEL=1
IEF196I        2 //STARTING EXEC INIT
...
IEF196I          XX*       The INIT procedure is used to start the MVS
IEF196I Initiator      */ 00900000
```

Algorithms: Initiator Placement

- Same peer-peer distributed decision-making SRM goal mode has always done
- Each system has a view of remote goal mode systems' CPU capacity, shortages, and initiator queue data
- When policy adjustment wants to add initiators, resources can be found to support them, and adding them helps measurably try to do what a human might do:
  - Use idle capacity if available
  - If nothing idle, then proceed to
    - Have each system find which work would have to donate
    - Choose amongst the donors based on the policy goals

Algorithms: Initiator Placement

- Start with all systems in the MAS that owns the job queue
- Subtract out compatibility mode systems, those in shortages
- Look for systems with the most excess CPU
- If a remote system has the most CPU available, wait(*) for it to help
- If work must be displaced, perform damage control:
  - If the local donor would still meet goals, displace him
  - Else get local assess data from all systems and pick the system which causes the least harm based on goals

- *** All waits are until a remote system helps or a small number of policy adjustment intervals have passed, not forever**
- Rebalancing is strictly demand-based

Assumptions

- There is a direct measurable relationship between queue delay and the number of initiators serving the queue
  - Does this hold true for your batch submission pattern(s)?
- Any job in a WLM-managed job class may incur some queue delay if resources to run more jobs are not available
  - If your scheduler holds jobs until the moment they should run, your scheduler is controlling initiation and you don't want any queue delay by definition.

The single most fundamental assumption for using WLM initiator management is that there is a predictable relationship between the size of the job queue and the number of initiators serving it. For WLM's purposes, the queue includes only jobs in a single service class.

Some installations essentially throw all their production jobs in at once, over-initiate, and let MVS manage whatever happens. Since only a small percentage of the jobs can actually be initiated at once, the incremental effect of additional initiators on the queue delay becomes negligible and SRM's algorithms would conclude that there is no value in adding more. This can happen with both response time and velocity goals.

The reverse can also happen with velocity and response time goals: since most of the delay stems from queuing, adjustments to any other resource (CPU, storage) show little value toward achieving the goal and are abandoned. If you submit many jobs at once and you intend to use WLM initiator management, we recommend that you use discretionary goals for the batch work submitted en masse. Resource group minimums can optionally be used to give preferential treatment to different subsets of the batch workload.

In other cases a scheduling package is used to minimize queue time today. If sufficient system resources are available to run the jobs today or if some queue time is tolerated today after the scheduler releases jobs, no change is required. If the scheduler is being used to effectively eliminate queue time or system resources are constrained, proceed cautiously.

Assumptions

- The goal for period 1 of the service class is the most stringent goal in the service class
- Batch work running in the last period of a multiperiod service class
  - still makes some progress over time
  - is not a large percentage of the work in the service class
- Each service class uses either WLM or JES to control initiation, not both

Users of multiperiod service classes to manage batch work need to be aware of a situation in which things can theoretically go awry. WLM-managed initiators are a service class resource; any given initiator can serve any period of a service class. Because all jobs begin in period 1, period 1 gets all queue delay samples and its goal is used to evaluate the value/impact of adding initiators. If the pattern of transaction completions is such that period 1 has a relatively small number of completions and the goals on later periods are much less stringent (e.g. discretionary) and resources are constrained, the following can occur. Each initiator selects a job, the majority end up in last period and get swapped out. The existence of swapped out busy initiators is a criteria used to cease adding new initiators for the class; since they are being swapped out, they are doing no good to the service class and there is no point in adding more. So jobs in period 1 starve for initiators due to the relatively poor treatment of the work once it moves into later periods.

Using both WLM-managed and JES-managed initiators to service work in the same service class is simply an attack on the data. Doing so weakens the relationship between the number of WLM-managed initiators and the queue delay observed. There are no known cases where this would cause dire consequences, but managing based on suspect data is never a good idea. The recommendation is not to mix the different types of initiation control within a single service class.

Migration Considerations

- Satisfy the prerequisites mentioned earlier
- Multiple JESplexes within the sysplex
- Secondary JESes
- Compatibility mode
- Priority Aging

Job class definitions must be consistent within a JESplex (enforced by JES2 R4), but could vary across multiple JESplexes within a single sysplex. WLM controls on the other hand are sysplex-wide, period. Make sure the two are not at odds with one another.

"Boss" JES has nothing to do with objects, attitude, or surfer-dude coolness. It does have to do with which JES is allowed to create WLM-managed initiator queues when more than one JES2 instance in the same MAS is running on a single MVS image. JES2 has some rules for selecting the Boss JES2, which their publications document. The short version is: if the primary JES is running, primary is the Boss; if the primary is not running, the first secondary encountered in IEFSSNxx that is running is the Boss.

WLM compatibility mode is not where you want to be using WLM-managed initiators. They will drain and terminate if the system is changed from goal to compatibility mode while they are running, and will not be started at all if running in compatibility mode. Since JES2 effectively partitions JES-managed jobs/initiators from WLM-managed jobs/initiators, jobs submitted to WLM-managed job classes while running in compatibility mode will sit on the queue looking forlorn. If the system is switched into goal mode afterward or the job class is changed to JES-managed they will get their chance to be initiated, but not before.

JES2 priority aging will reclassify all jobs whose priority is changed to see if the service class changed. Only JES-managed job classes are priority-aged since job priority does not affect queueing order for WLM-managed job classes.

Operational Changes

- Starting and stopping initiators
- Running a job immediately
- Limiting concurrent jobs

Controlling Job Initiation

- $SI and $PI commands continue to control JES initiators only

- $S XEQ command
  - Starts job initiation on the system where it's entered
  - Use for system startup

- $P XEQ command
  - Stops job initiation on the system where it is entered
  - Use for system shutdown
  - Active jobs are allowed to finish
  - WLM initiators drain and terminate
  - JES initiators drain

Initiating a Job Immediately

- Need the ability to "force" a job into execution on user request
- $SJ command - "start job"
- Job must have a WLM-managed job class
- WLM selects the system on which to execute the job based on
  - System affinity
  - This system's view of available capacity
- A WLM initiator is started solely to run the specific job, runs it, and terminates
- No relationship between initiation and performance management

$SJ exists to fill an operational niche, "run this now" or demand initiation, that WLM-managed initiators remove. It is intended for occasional use.

It is not in any way, shape, or form intended or designed for frequent use.

Initiator placement is based on a less sophisticated version of the algorithm, so that if there is idle CPU capacity in the set of eligible systems no damage control is attempted. The only use of goals is when displacing work, to choose the system with the most CPU available at the importance of the demand-initiated work. The impact to any displaced work is not considered, nor is optimal displacement (aka damage control) attempted, since this is a short-term decision.

Job Class Limit

- Need ability to set the maximum number of jobs in a job class that can run concurrently due to service level agreements or physical resource constraints (tape drives)
- Previously able to do this by limiting how many initiators can select jobs from the class
- A new control is added to the JOBCLASS definition and $tjobclass

```
JOBCLASS(E)    XEQCOUNT=(MAXIMUM=5)
```

Can be a WLM-managed
or JES-managed job class

Multi-access spool (MAS) scope

SDSF Changes

- Job information line command

```
                                 Job Information

Job name          BAT1155    Job class limit exceeded? NO
Job ID            JOB01140   Duplicate job name wait?  NO
Job schedulable? NO          Time in queue             N/A
Job class mode   WLM         Average time in queue     N/A
Job class held?  NO          Position in queue         N/A     of N/A
                             Active jobs in queue      N/A

Scheduling environment:                  available on these systems
```

- Management mode (JES or WLM) on STatus display

```
JOBNAME  SRVCLASS WPOS  SCHEDULING-ENV   DLY MODE
BAT1280  BATCH1      1                   NO  WLM
BAT1281  BATCH1      2                   NO  WLM
BAT2283  BATCH2      0 OVERNIGHT         YES WLM
BAT1282  BATCH1      3                   NO  WLM
```

WLM Batch Scheduling Enhancements

Resource Affinity Scheduling

Classic Batch Workload Scheduling

Mmmm...need a job class for a job which uses:
IMS system IMSA, and only when database XYZ is unavailable to on-line transactions...
.... how can it be that they're all taken?

- Uses a combination of:
  - JES job class
  - Priority or job class hold
  - System affinity
  - Data set availability
- Attempt to get the job scheduled on the correct MVS image at the proper time.

Resource Affinity Scheduling

- New capability provided by WLM in OS/390 R4.
- Services for available for use by any scheduler that must accommodate asymmetric environments.
  - JES2 is first exploiter.
    - JES and WLM managed initiators.
  - Does <u>not</u> require "Goal Mode".

Resource Scheduling Environment

- Defined in customer terms:
  - Server address space (e.g., IMSA, CICSB) available
  - Data base 'XYZ' access available to batch or on-line processing
  - Vector Facility
  - Prime Shift/Off Shift
  - Production/Test
  - etc.

- Available or unavailable on each image in the sysplex
  - Multiple environments available on each MVS image
  - Available on none, some or all MVS images

Defined and controlled by installation!

Definitions

- Resource Elements
  - 16 character names
    - Maximum 999
    - SYS_ prefix is reserved
  - 32 character description
  - Exist in one of three states:
    - "ON"
    - "OFF"
    - "RESET"
  - Known throughout the sysplex
  - Scheduling states are:
    - "ON" or "OFF"
  - State on each system independent of state on other systems
  - Building blocks for scheduling environments

- Prime_Shift
- Vector_Facility
- DB2_Prod
- C_Compiler

Definitions...

- Scheduling Environments
  - "External" used to request an installation defined environment.
  - 16 character names
    - Maximum 999
  - 32 character description
  - Composed of 0 - 999 resources
    - Each resource has a specified state: "ON" or "OFF"
  - All resources must be in specified state for an environment to be available.
  - A scheduling environment may be available on none, one or more systems in the sysplex.
  - None, one or more scheduling environments may be available on each system of the sysplex.

Cheap_Cycles
- Prime_Shift(OFF)

Cheap_Fortran
- Prime_Shift(OFF)
- Vector_Facility(ON)

Visibility

- Scheduling environments and resources are defined through WLM ISPF dialog
- Scheduling environments visible through SDSF, JES and MVS operator commands
- Resources visible through SDSF and MVS commands
- User needs to know about scheduling environments not resources underneath
- Only one scheduling environment can be associated with a "job".

External

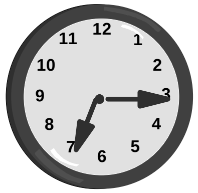
```
//MYJOBF   JOB (6565,C003),
//             SCHENV=CHEAP_FORTRAN,
//             CLASS=A,
//             MSGCLASS=T,...

//MYJOBC   JOB (6565,C003),
//             SCHENV=CHEAP_CYCLES,
//             CLASS=A,
//             MSGCLASS=T,...
```

Or with JES input service exits, converter exits, or scheduling product

Example

ROUT SY1,F WLM,RESOURCE=PRIME_SHIFT,OFF

ROUT SY2,F WLM,RESOURCE=PRIME_SHIFT,OFF



System Eligibility

MYJOBF
- SY1

MYJOBC
- SY1
- SY2

		SY1	SY2
Resource Elements	Vector_Facility	ON	OFF
	Prime_Shift	OFF	OFF
Scheduling Environments	Cheap_Cycles	Available	Available
	Cheap_Fortran	Available	Unavailable

Key Control Points

- WLM Operational Control
  - Inquire on Scheduling Environments
    - Which environments are available on which MVS images
    - What resource elements/states comprise an environment
  - Inquire on Resource Elements
    - What is the state of resource element
  - Modify Resource Elements
    - Set the scheduling state to "ON" or "OFF"
    - Set the resource to unscheduleable state "RESET"
  - Cannot directly modify Scheduling Environments

- JES Operational Control
  - Display the Scheduling Environment for a specified job
  - List jobs waiting for a Scheduling Environment to become available.
  - Schedules job into execution on MVS image only if Scheduling Environment is available!

Summary

- Dynamic, goal-oriented management of the time jobs spend waiting for an initiator
- Multisystem workload balancing
- Reduced operational demands
- Improved reporting of job response time and pre-execution job delays
- Support for all users/exploiters of resource affinity scheduling

Recent Service

- See info APAR II10760 for required JES2 service
- For OS/390 V2R6 and up, WLM APAR OW37405

Sources of Information

- **http://www.ibm.com/s390/wlm** - 2 papers on batch
- redbook: "Batch Processing in a Parallel Sysplex", SG24-5329
- JES2 Migration Notebook, chapter 9
- JES2 Init & Tuning Guide, chapter 2 Job Selection and Execution topic
- MVS Planning: Workload Management
  - classification by job priority
  - velocity migration
  - scheduling environments

WLM Education References:

IBM Education courses

H3995	Parallel Sysplex Implementation (45 minutes on WLM)
H4011	Parallel Sysplex Overview
H4012	Parallel Sysplex Overview
H4013	Workload Manager Workshop
H3986	Migration
H3988	Measurement and Tuning - compatibility to goal mode migration
H3989	Migration
ES540	OS/390 Workload-Based Performance Management (new in 1999!)

(check **http://www.training.ibm.com** for availability)

MVS Planning: Workload Management

Every exploiting customer should be familiar with it.

MVS Programming: Workload Management Services

Probably interesting only to vendors or customers that write their own extensions.

WLM Technical References:

WLM home page

http://www.ibm.com/s390/wlm

IBMLink/TalkLink

MVSWLM forum - customers, consultants, and development participate

SHARE, CMG, MVS Expo proceedings

User experiences from Boeing, Fairfax County of Virginia, MCI, Trigon Blue Cross/ Blue Shield, others