Issue Date: AIX EXTRA eNewsletter
July 2004, Posted On: 7/1/2004

AIX EXTRA: A Look at Workload Manager

Jaqui Lynch

Increasing AIX system implementation costs make it expensive to run individual applications on separate servers. Consolidating those workloads onto a single server using logical partitioning (LPAR) technology still keeps the applications on separate OS instances, but consolidates them to one partitioned server. Workload Manager (WLM) implemented on individual instances on partitioned systems or on non-partitioned systems manages heterogeneous workloads on the same OS instance.

WLM works on percentages of resources, managing CPU time, memory and I/O bandwidth using a combination of hierarchical classes and tiers. It assigns threads to a class, either automatically (using the rules file) or manually (by superuser), based on class rules and assigns minimum and maximum amounts for CPU, memory and I/O throughput to each class. To correctly assign a process to a class, WLM goes through process identification, analyzing the attributes of the process to see how it matches up with the WLM class definitions.

The owner or group ID, the full application path and name, the process type or a series of application tags identify processes. WLM assigns each class a set of resource shares and limits, then can further assign a class to a tier. Tiers add a layer of granularity to prioritize groups of classes.

Classes
A class, which is a set of processes with a single set of resource limits, is either a superclass or a subclass. WLM assigns resource shares and limits to a superclass based on the total resources in the system, and then defines subclasses to further divide the superclass' assigned resources amongst its assigned jobs.

Five predefined superclasses exist by default, and one can add up to 27 user-defined superclasses. Each superclass can have 12 subclasses; two predefined and 10 user-defined. Each class is assigned a name with a maximum of 16 characters. Superclass names must be unique, and subclass names must be unique within their assigned superclass.

Predefined superclasses

- Default—The default superclass is always defined. All non-root processes not automatically assigned to another superclass are assigned to the default class.
- System—All privileged (root) processes not assigned to another class by rules are assigned to the system superclass. All memory pages belonging to the system memory segments, kernel processes and threads are assigned to this class.
- Shared—All memory pages shared by processes in more than one superclass are assigned to this class.
- Unclassified—When WLM starts, it classifies processes. If it's unable to relate pages to a specific process, it assigns them to the unclassified class—e.g., pages of a closed file in memory are unclassified. A few kernel processes such as LRUD and wait also appear in the unclassified superclass.
- Unmanaged—This class tracks memory usage for all pinned pages in the system that are not managed by WLM. No processes are assigned to this class.

Predefined subclasses

- Default—All processes not assigned to a specific subclass of the superclass are assigned to the default subclass.
- Shared—As with superclasses, all shared memory pages are assigned to this class.

Tiers
Tiers define class importance relative to other classes. You can define up to 10 tiers (0 through 9) to prioritize classes, with 0 being most important and 9 least important. WLM assigns resources to the highest tier process that is ready to run.

Classification

The files necessary to classify and describe classes and tiers are defined in the /etc/wlm directory, beneath which is a directory created for the schema to be used. For example, if the /etc/wlm/prodsys95 directory is the production systems definition for 9-5, an additional directory could be created for outside those hours. A simple cronjob command could switch between the two definition sets.

Classification requires several files, including classes (see Table 1), which lists each class, its description, the tier to which it belongs and other class attributes; rules (see Table 2), where control is exerted over the class resources; limits (see Table 3) and shares (see Table 4), which define resource limits and shares, respectively; and a description file, which includes a description of the classes.

WLM reads rules from top to bottom and assigns a process to the first matching rule, which makes it important to list rules from specific (top) to general (bottom). Processes are generally assigned to a class based on the userid, group or fully qualified path and application name. Type and tag fields can also be used. Type can be 32-bit, 64-bit, plock (the process called plock() to pin memory) or fixed (the process is a fixed priority process).

Activation and Monitoring
Once the rules and other files are coded, it's time to start WLM in one of two modes: passive or active. Start with passive mode to monitor its potential effect without actually making it active. Do this using the command /usr/sbin/wlmcntrl -d prodsys -p (where prodsys is the directory name in which the configuration files live). Once the configuration is proofed and monitored the configuration, WLM can be started in active mode.

WLM Commands
Various commands exist to control changes to and reports from WLM, including:

- wlmassign—Manually assign a process to a class.
- wlmcheck—Check assignment rules to see how to classify a resource.
- wlmcntrl—Stop and start WLM, switch it from passive to active mode and query what mode it's running in.
- wlmstat—Show WLM per class resource utilization.
- wlmmon and wlmperf—Provide graphical views of resource activities by class.
- acctcom—Updated with a -w flag to list WLM information and a -c flag to list only specific classes, many other commands, such as ps, were also updated.

Steps to Implementation
The first, most critical step is to design the classification criteria. It's important to know what classes will be defined and how they will be classified—i.e. what user ID or application would be defined.

The second step is actually defining the class, limits, shares and rules files and starting WLM in passive mode. Then use WLM to refine the definitions before activating them.

Summary
WLM has added the capability to drive up utilization on AIX systems while still ensuring that applications get the percentage of resource required to provide the performance necessary for success. As systems become more complex and expensive, it's apparent that workloads will need to be combined onto the same OS instances to take better advantage of system resources. Going through the exercise of classifying the workloads running on your systems now, even if you only run WLM in passive mode, can give you an additional tool to report on what's happening on your systems, as well as a potential management tool, should the need arise.