CheatSheet #32 zTidBits WAS' WLM Basics

- Proper WLM goals can significantly affect your application throughput. The WebSphere Application Server address spaces should be given a fairly high priority. When setting the WLM goals for your z/OS system, you might want to:
  - Classify location service daemons and controllers as SYSSTC or high velocity.
  - Use STC classification rules to classify velocity goals for application servers.
    > **Java garbage collection** runs under this classification. Java garbage collection is a CPU and storage intensive process. If you set the velocity goal too high garbage collection can consume more of your system resources than desired. If your Java heap is correctly tuned, garbage collection for each servant should run no more than 5% of the time. Also, providing proper priority to garbage collection processing is necessary since other work in the servant is stopped during much of the time that garbage collection is running.
    > **JavaServer Page file** compiles run under this classification. If your system is configured to do these compiles at runtime, setting the velocity goal too low can cause longer delays waiting for JavaServer Page file compiles to complete.
- Set up an application environment for work running under servants where:
  - The subsystem type is set to **CB (Component Broker).**
  - The environment is classified based on server name, server instance name, user ID, and transaction class.
  - Percentage response time goals are set for enclaves.

    NOTE: You should make the response time goals achievable. For example, a goal that 80% of the work will complete in .25 seconds is a typical goal. Velocity goals for application work are not meaningful and should be avoided.
  - A high velocity default service class for CB subsystem transactions is provided.

    NOTE: The default is SYSOTHER. Your goals can be multi-period. This might be useful if you have distinctly short and long running transactions in the same service class. On the other hand, it is usually better to filter this work into a different service class if you can. Being in a different service class will place the work in a different servant which allows WLM much more latitude in managing the goals.
- Define unique WLM report classes for servant regions and for applications running in your application environment.
- Defining unique WLM report classes enables the resource measurement facility (RMF) to report performance information with more granularity.
- Set your Application environment to **No Limit**. Required if you need more than one servant per application server. See WLM Application Environments panel below.

NOTE: Under WLM, you can control how many servants can be started for each server. If you need more than one servant in a server make sure that **No Limit** is selected for the application environment associated with your server. For information about setting up WLM performance goals, see *z/OS MVS Planning: Workload Management*.

- When the WLM configuration is set to no limit, you can use **wlm_maximumSRCount=*x*** and **wlm_minimumSRCount=*y*** variables to control the maximum and minimum number of servants.
- To specify values for these variables, in the administrative console, *click* **Severs > Application servers** and select the appropriate application server.

  NOTE: If you specify a value for the wlm_maximumSRCount variable, the value must be greater than or equal to the number of service classes defined for this application environment. If the value is less than the number of defined service classes, timeouts might be caused because there is an insufficient number of servants being available.

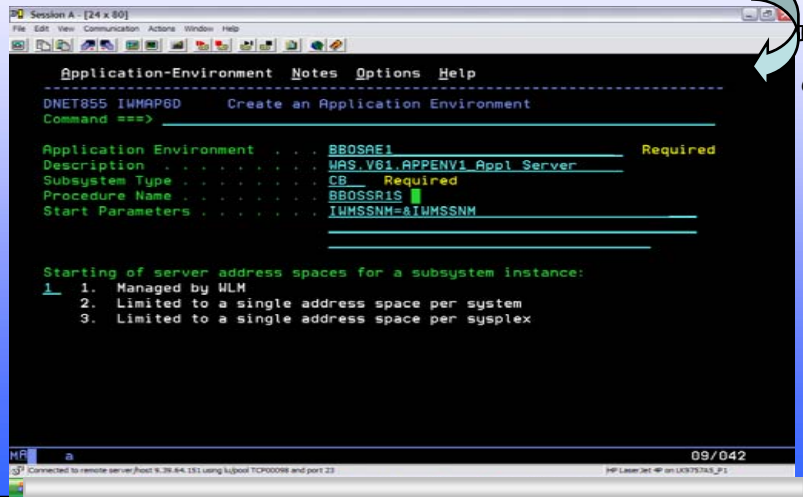
- Results of WAS' WLM is reported in RMF Postprocessor (PP) Workload Activity Report:
  - Transactions per second (not always the same as client tran rate)
  - Average response times (and distribution of response times)
  - CPU time used
  - Percent response time associated with various delays

  NOTE: RMF Monitor III is another great way to locate and solve performance issues on zWAS especially with a complex architecture incorporating MQ, CICS, DB2 in the mix of the transaction flow. This is launched via your TSO Session and also includes details on the Coupling Facility usage.

MORE ON…RMF Monitor III: This product gathers session data with a typical cycle of one second

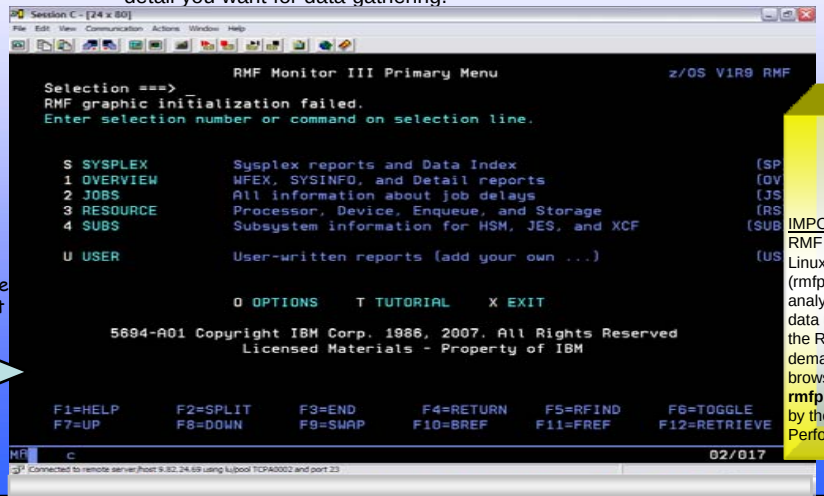
- Consolidated records are written for a range of time (the default is set to 100 seconds).
- You can collect short term data and continuously monitor the system status to solve WAS performance problems.
- You get actual performance data (response times, execution velocity) on a very detailed level to use for future comparison with performance policy goals.
- You can collect data that indicates how fast jobs or groups of jobs are running – this is called *workflow* or *speed*.
- You also get data that shows how resource-intensive jobs are using processor, DASD devices, and storage – the reports describe this utilization under the term ***using*.**
- There is information about ***delays***, which are important indicators of performance problems.

  NOTE: Most data that is available for the Monitor III Reporter is gathered automatically by the gatherer function, but there are also system resources where you can define the level of detail you want for data gathering.



ISPF WLM Panel Option #9

ISPF RMF Performance Management Panel Option #3



IMPORTANT: RMF also has a Linux data gatherer (rmfpms). You can analyze the gathered data using RMF PM or the RMF data on demand in a Web browser. **rmfpms** is also used by the IBM z/VM Performance Toolkit