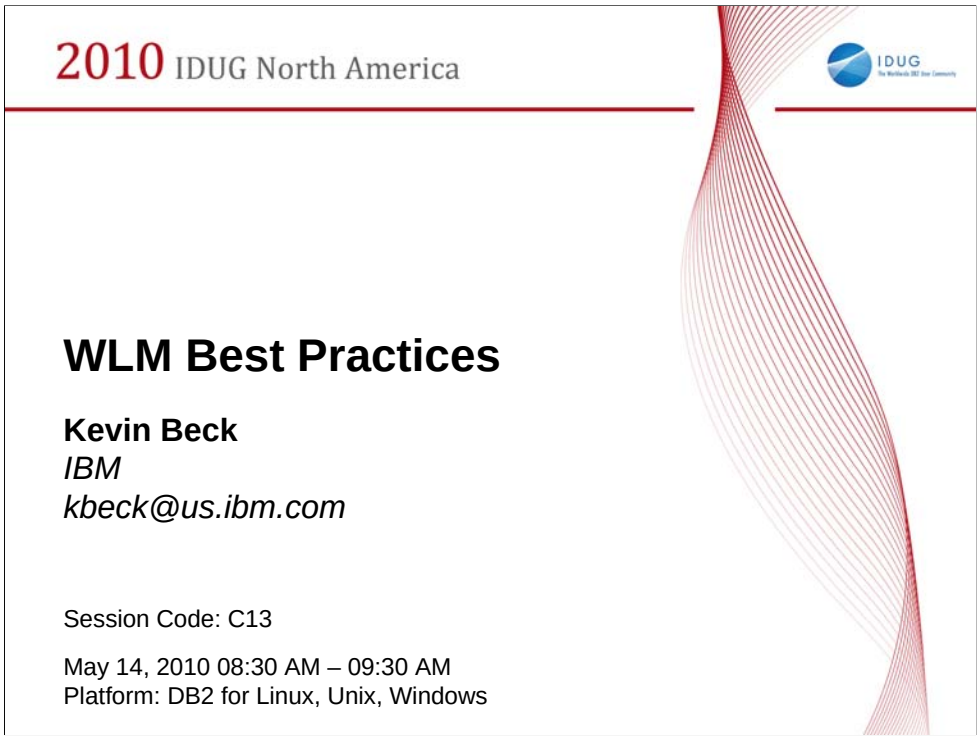


This session provides a step by step approach to creating a complete workload management solution for your enterprise. Learn how you can employ best practices on your Version 9.5 or 9.7 DB2 data server in order to meet business objectives according to their importance and priority. The best practices presented here use the new DB2 Workload Manager (WLM) introduced in Version 9.5. A general methodology will be shown that you can use to achieve real time resource management, work prioritization, system overload protection and enhanced real time monitoring of work running on your data server.

Objectives

- Apply best practices for workload management to build a complete, robust solution
- Leverage new monitoring features to evaluate and understand work running on your data server
- Use the available tools for monitoring and configuring workload management
- Protect your data server from runaway queries
- Understand the basic building blocks of WLM

Disclaimer

Importance of Workload Management

- Meet business objectives according to their priority
  - High priority work addressed in the time required
  - Other work completed without compromising response time of high priority work
- Protect data server from overload
  - Set limits on how long a query can run
  - Restrict CPU resources for a specific line of business, group of users or application
- Monitor your data server
  - Understand what is running
  - Diagnose and correct performance problems

Situation

Data warehouses are growing. Not only is more data being stored, but there are more people using them for more purposes.

You have to deal with more mixed workloads.

Operational activities and analytical activities are competing for resources in the database.

Not only are the user activities competing with each other; remote sites and 24x7 operations mean that

no one has nightly maintenance windows any more.

batch operations like ETL and big reporting jobs need to coexist with operational activities.

Complication

This doesn't prevent customers from demanding more consistent response times, more SLAs or equivalent.

Implication

Without some form of WLM, response times will be inconsistent, servers will experience periods of overload, resources will be wasted on badly written or inappropriate queries.

Resolution

Workload management, new in DB2 LUW 9.5, provides tools to align prioritization of activities by DB2 with your business objectives.

This presentation outlines a methodology you can use to apply WLM in your enterprise.

You can create criteria that define the work you consider critical to your business, and through WLM, DB2 will give it higher priority.

You can also restrict the resources allocated for lower priority work, to reduce its impact.

WLM also provides ways for you to set policies that protect your data server from becoming overloaded.

For example, you can set limits on how long particular classes of queries can run.

Or you can restrict the CPU resources allocated to a particular line of business or application.

Best Practices Roadmap

- Divide incoming work into categories (service classes)
  - Categorizing work is safe (no change in behavior)
- Monitor to validate work is properly categorized
- Apply controls to service classes
  - Prioritize work to match business needs
  - Impose limits to reduce wasted resources and enforce policies
  - Monitor validate and troubleshoot performance

This presentation assumes you are working with DB2 v9.5.

However, it also mentions a few useful features new in DB2 v9.7. These are marked as such.

A service class groups activities for which you want the same priorities and controls applied.

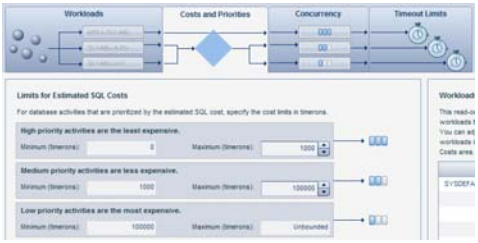
Service classes are structured in a two level hierarchy of subclasses grouped within superclasses.

The first half of this presentation assumes a single service superclass with a few service subclasses in it.

Depending on your requirements, you might never need to create additional service superclasses.

Simplified Configuration via Web Based Tooling

- Optim Performance Manager 4.1
  - Integrated WLM monitoring and configuration
  - Choice of solution templates
  - Works for DB2 LUW v9.5 or v9.7
- Admin Console in InfoSphere Warehouse v9.7
  - Template configuration
  - Based on 3 query classes
  - Adjustable parameters



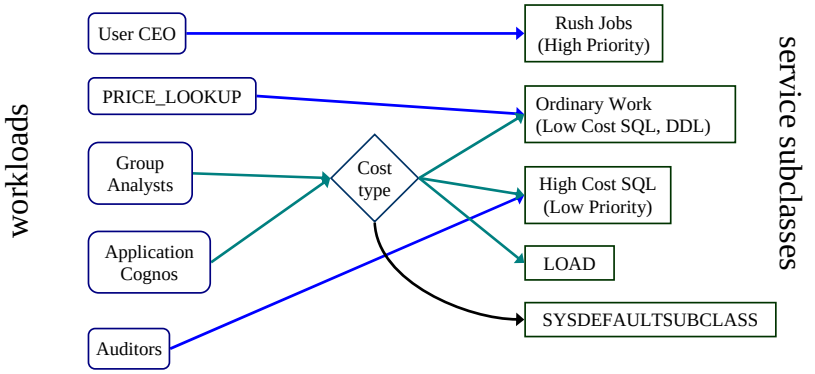
Categorizing Work – Available Criteria

- Connection attributes
  - User or group
  - Application
  - Tags inserted by middleware
  - Etc.
- Type of activity
  - DML
    - READ (SQL Queries)
    - WRITE (insert, update and delete)
  - DDL
  - LOAD
  - Etc.
- Estimated cost (SQL queries only)

Categorizing Work - Mechanisms

WLM uses a two step evaluation to categorize activities

1. Workloads categorize connections
2. Work action sets further categorize by activity type and/or cost



Workloads categorize connections based on connection attributes

- Key connection attributes include user ID or group, application name, various tags set by the client
- Typically, workload assignment happens once at connection time

Work action sets categorize based on activity type or estimated cost

- Type SQL (DML, DDL, Read, Write), LOAD, CALL
- Estimated cost or cardinality – applies only to DML

Workloads can map directly to a service subclass. Doing so bypasses evaluation via superclass level work action sets.

Example: work action set has a rule to map all LOAD activities to the LOAD service subclass.

However, any activities from auditors are always mapped directly to the low priority service class

In this presentation, I focus on using work action sets in service superclasses to map activities to service subclasses by type and or cost.

Work action sets can also appear in other contexts for other purposes.

Step 1 – Define Workloads

- Define a workload for each interesting source of work
  - Application
    - Allows you to treat some applications differently
    - LOAD is not an application
  - Individual user, group or role
    - Treat activities differently based on **who** submitted them
  - DB2 client information fields
    - Some middleware can tag work by filling in client info fields
  - Combination of any of the above
    - Queries from SAS treated differently if submitted by analyst versus other users

The first step in configuring workload management is to define a workload for each interesting source of work.

A workload defines a category of work, based on specifying connection attributes.

The most relevant of these are the application name and the user ID, group or role.

Later, we will talk about how to apply additional criteria for finer grained control.

Workloads categorize **connections**, they do not directly categorize **activities**.

The only available criteria are attributes of connections, such as user or application.

A classic beginner error is to try to define a workload for LOAD activities.

LOAD is not an application – you connect via an application, then invoke LOAD.

You must use work actions to identify LOAD activities, not workloads.

There are some additional fields for defining workloads, categorized as “client information”. Some client software can use these fields to tag activities

in various ways. For example, Cognos can be configured to pass in the Cognos user ID.

A typical approach might be to create a workload for each combination of user group and significant applications used by that group.

This lets you set policies for a group, such as a line of business in your organization, and also treat important applications differently

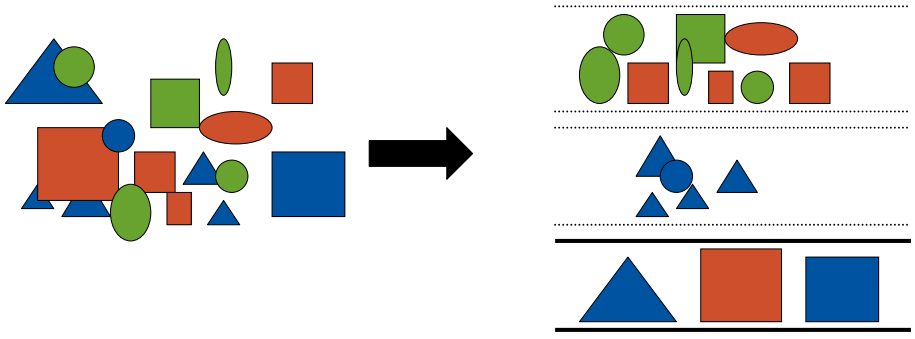
within each group.

For example, creating a workload that distinguishes the use of SAS by trusted analysts lets you prioritize their work differently from

other work from the same group, or other people also using SAS.

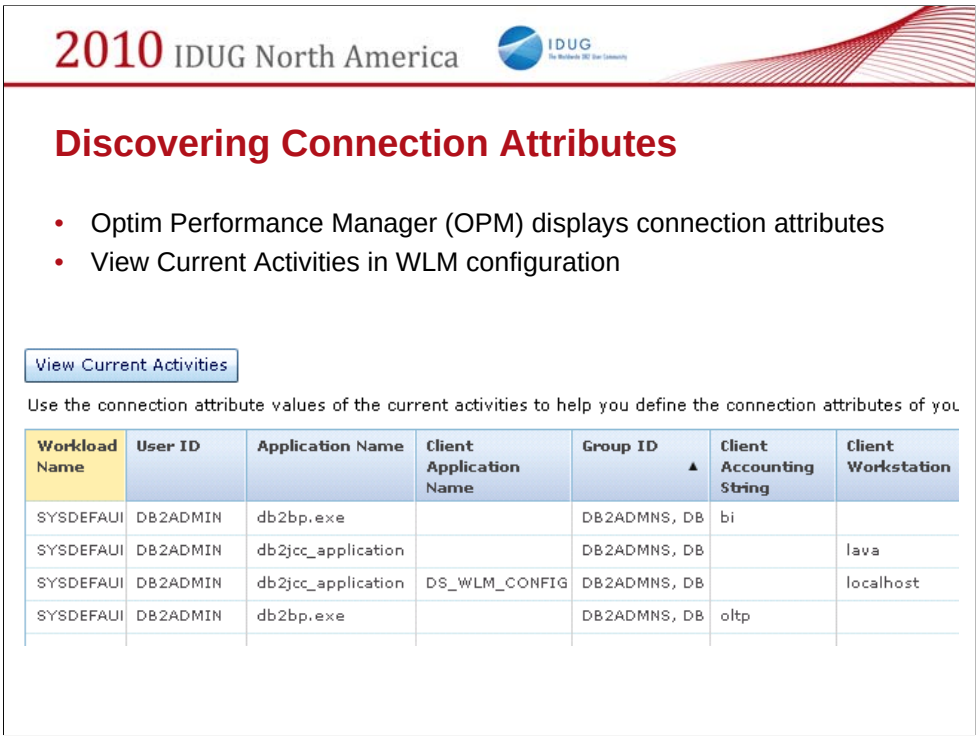
Reasons to define a workload

- Categorize work (map to a service class)
- Monitor by workload
- Label activities for troubleshooting



Discovering Connection Attributes

- Optim Performance Manager (OPM) displays connection attributes
- View Current Activities in WLM configuration



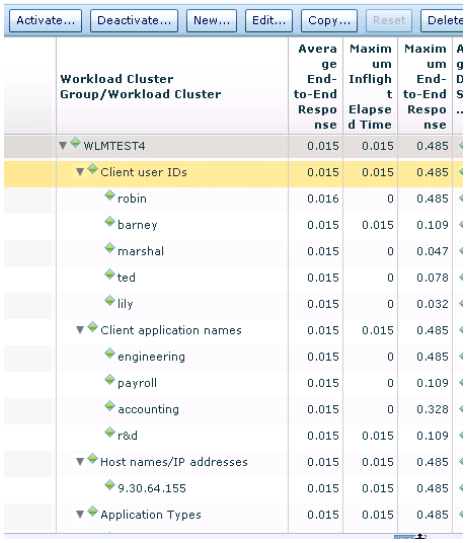
Screen shot: View Current Activities in the workloads tab of WLM configuration.

This shows all connection attributes of all currently active connections.

It is aggregated, so that every combination appears only once in the report.

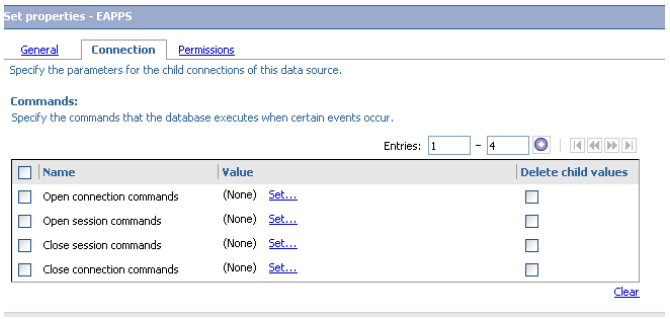
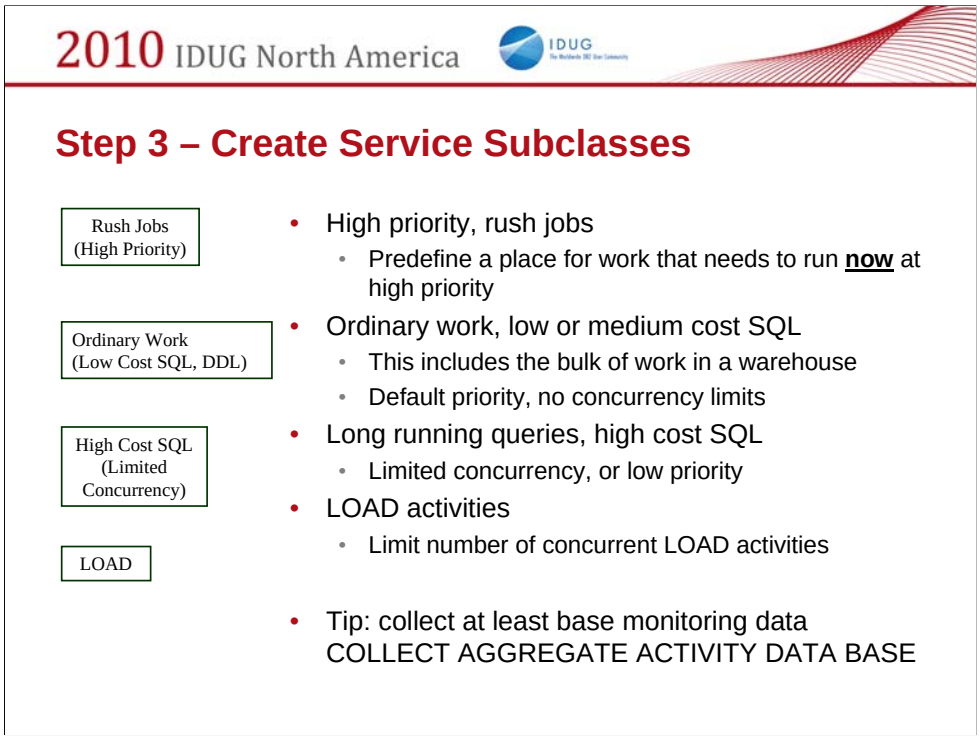
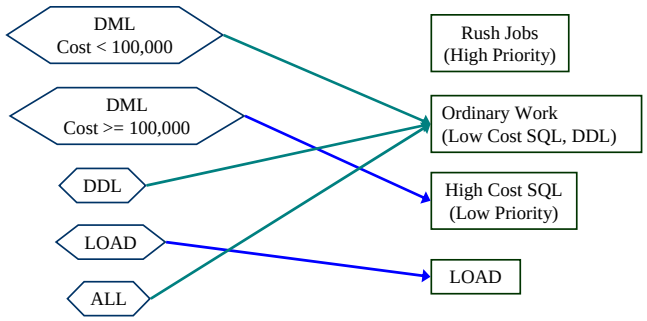
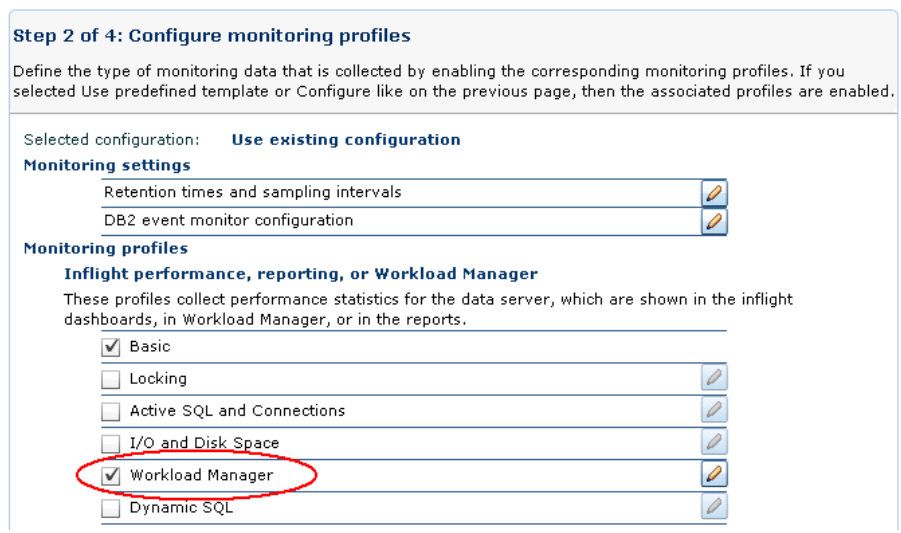
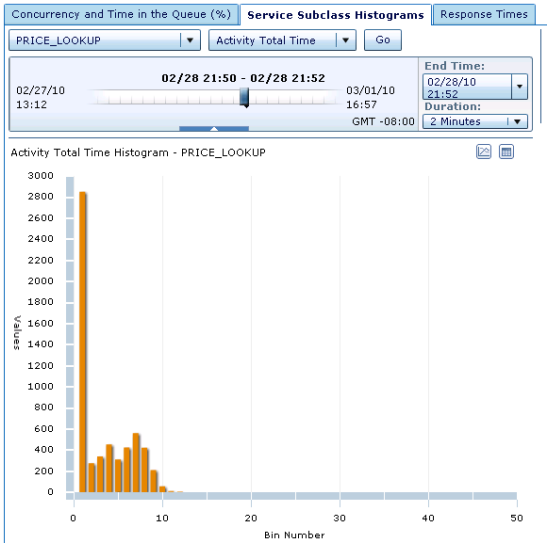
Connection Attributes via Extended Insight

- Extended Insight clusters activities by connection attributes
- Browse history of activities



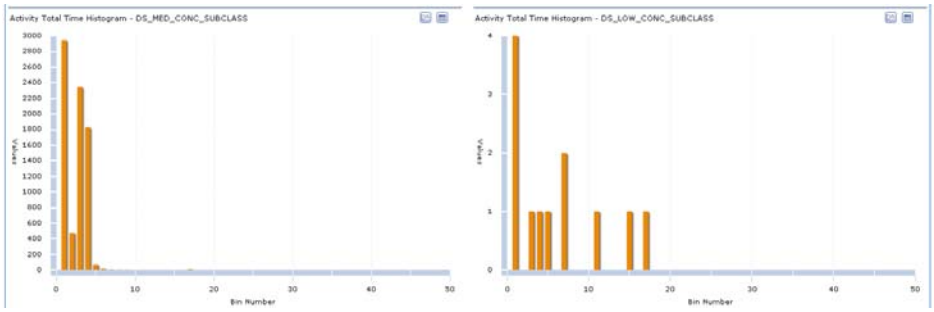
Workload Cluster Group/Workload Cluster	Average End-to-End Response	Maximum Inflight Elapsed Time	Maximum End-to-End Response
WLMTEST4	0.015	0.015	0.485
Client user IDs	0.015	0.015	0.485
robin	0.016	0	0.485
barney	0.015	0.015	0.109
marshal	0.015	0	0.047
ted	0.015	0	0.078
lily	0.015	0	0.032
Client application names	0.015	0.015	0.485
engineering	0.015	0	0.485
payroll	0.015	0	0.109
accounting	0.015	0	0.328
r&d	0.015	0.015	0.109
Host names/IP addresses	0.015	0.015	0.485
9.30.64.155	0.015	0.015	0.485
Application Types	0.015	0.015	0.485

Connection Attributes via WLM Table Functions

- ```
  SELECT workload_name, application_name, …
  FROM table(wlm_get_service_class_workload_occurrences('', '', -2));
  ```
- As shown, displays connections attributes for all current connections
- Filter as necessary to find what you are looking for
  ```

Real Life Example: Cognos




```
<commandBlock>
  <commands>
    <sqlCommand>
      <sql> CALL SYSPROC.WLM_SET_CLIENT_INFO(
        #$account.personalInfo.userName#, ‘MachineName',
        #$account.parameters.var1#, ‘ApplicationName', 'AUTOMATIC')
      </sql>
    </sqlCommand>
  </commands>
</commandBlock>
```

Cognos provides a facility to run some SQL every time it connects to DB2

Use this to set connection attributes

There are some macros provided by Cognos for the real user ID who invoked the Cognos application, so you can pass that through to WLM

Step 2 - Create a Service Superclass

- A service superclass is a prerequisite for:
  - A work action set to map activities by type and estimated cost
  - Service subclasses
- One service superclass is sufficient for basic use cases
- Redirect default workload to new service superclass
- SQL syntax
  - CREATE SERVICE CLASS main;
  - ALTER WORKLOAD SYSDEFAULTUSERWORKLOAD SERVICE CLASS main;

There are restrictions on what you can add to the default service superclass.

Hence, at least one user defined service superclass must be created as a sort of infrastructure step.

You can get a long ways with a single service superclass. Many customers will never need to create more than one.

Towards the end of the presentation, I will describe a couple of scenarios that require additional service superclasses.



These are the initial service subclasses OPM will create for you.

They cover common categories of activities that benefit from being broken out separately.

For all subclasses, you should specify to collect at least base aggregate monitoring data

COLLECT AGGREGATE ACTIVITY DATA { BASE | EXTENDED }

Collecting BASE monitoring data has negligible overhead and provides several very useful metrics

EXTENDED aggregate monitoring data is also relatively inexpensive and provides additional metrics

Service Subclasses in OPM Template

- Service subclasses created by WLM configuration in OPM

When you configure WLM using OPM, these four service subclasses are created for you.

You can add additional service subclasses as needed.

Step 4 – Create a Work Action Set

- Map activities to subclasses based on type or cost
- Anything not mapped falls into SYSDEFAULTSUBCLASS
- Evaluation order matters
- Template mappings:



Estimated costs apply only for DML

Timerons are floating point numbers.

If you do something of the form “0 < x < 1000” followed by “1001 < y < 2000” estimated costs of 1000.5 will fall through the gap.

The template uses a work action of type ALL at the end to ensure nothing falls through to SYSDEFAULTSUBCLASS.

The service class for high priority / rush jobs is intentionally unreachable via the work action set.

Only activities from workloads specifically mapped to it will run there.

Template DDL for Work Class Set + Work Action Set

```
CREATE WORK CLASS SET "DS_AUTOMGMTSU_WORK_CLASS_SET" (
    WORK CLASS "DS_LOW_COST_DML_WC" WORK TYPE DML
      FOR TIMERONCOST FROM 0.0 TO 100000.0 POSITION AT 1,
    WORK CLASS "DS_HIGH_COST_DML_WC" WORK TYPE DML
      FOR TIMERONCOST FROM 100000.0 TO UNBOUNDED POSITION AT 2,
    WORK CLASS "DS_DDL_WC" WORK TYPE DDL POSITION AT 3,
    WORK CLASS "DS_LOAD_WC" WORK TYPE LOAD POSITION AT 4,
    WORK CLASS "DS_OTHER_WC" WORK TYPE ALL POSITION AT 5);
```

```
CREATE WORK ACTION SET "DS_AUTOMGMTSU_WORK_ACTION_SET“
  FOR SERVICE CLASS "DS_AUTO_MGMT_SUPER“
  USING WORK CLASS SET "DS_AUTOMGMTSU_WORK_CLASS_SET" (
    WORK ACTION "DS_MAP_LOW_COST_DML_WA" ON WORK CLASS "DS_LOW_COST_DML_WC“
      MAP ACTIVITY WITHOUT NESTED TO "DS_MED_CONC_SUBCLASS",
    WORK ACTION "DS_MAP_HIGH_COST_DML_WA" ON WORK CLASS "DS_HIGH_COST_DML_WC“
      MAP ACTIVITY WITHOUT NESTED TO "DS_LOW_CONC_SUBCLASS",
    WORK ACTION "DS_MAP_DDL_WA" ON WORK CLASS "DS_DDL_WC“
      MAP ACTIVITY WITHOUT NESTED TO "DS_MED_CONC_SUBCLASS",
    WORK ACTION "DS_MAP_LOAD_WA" ON WORK CLASS "DS_LOAD_WC“
      MAP ACTIVITY WITHOUT NESTED TO "DS_LOAD_SUBCLASS",
    WORK ACTION "DS_MAP_OTHER_WA" ON WORK CLASS "DS_OTHER_WC“
      MAP ACTIVITY WITHOUT NESTED TO "DS_LOW_CONC_SUBCLASS" );
```

For reference only.

This is the template DDL created by WLM configuration in OPM.

If you choose to do this by hand, work actions must be defined in terms of work classes.

Work classes are defined separately in a work class set.

Use the Tooling to Save Work

```
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD;
CREATE SERVICE CLASS "DS_AUTO_MGMT_SUPER";
CREATE SERVICE CLASS "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" COLLECT AGGREGATE ACTIVITY DATA BASE;
CREATE SERVICE CLASS "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" COLLECT AGGREGATE ACTIVITY DATA BASE;
CREATE SERVICE CLASS "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" COLLECT AGGREGATE ACTIVITY DATA BASE;
CREATE SERVICE CLASS "DS_LOAD_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" COLLECT AGGREGATE ACTIVITY DATA BASE;
CREATE WORK CLASS SET "DS_AUTOMGMTSU_1263546069031_WORK_CLASS_SET" ( WORK CLASS "DS_LOW_COST_DML_WC" WORK
      TYPE DML FOR TIMERONCOST FROM 0.0 TO 100000.0 POSITION AT 1, WORK CLASS "DS_HIGH_COST_DML_WC" WORK TYPE DML
      FOR TIMERONCOST FROM 100000.0 TO UNBOUNDED POSITION AT 2, WORK CLASS "DS_DDL_WC" WORK TYPE DDL POSITION AT 3,
      WORK CLASS "DS_LOAD_WC" WORK TYPE LOAD POSITION AT 4, WORK CLASS "DS_OTHER_WC" WORK TYPE ALL POSITION AT 5);
        .
        .
        .
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_PRI_CONC_DB_TH" FOR SERVICE CLASS "DS_LOW_CONC_SUBCLASS" UNDER
      "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE  WHEN CONCURRENTDBCOORDACTIVITIES > 4 AND
      QUEUEDACTIVITIES UNBOUNDED  CONTINUE;
ALTER WORKLOAD "SYSDEFAULTUSERWORKLOAD" SERVICE CLASS "DS_AUTO_MGMT_SUPER"  COLLECT AGGREGATE ACTIVITY DATA
      BASE ;
CREATE WORK ACTION SET "DS_AUTOMGMTSU_1263546069031_WORK_ACTION_SET" FOR SERVICE CLASS "DS_AUTO_MGMT_SUPER"
      USING WORK CLASS SET "DS_AUTOMGMTSU_1263546069031_WORK_CLASS_SET" ( WORK ACTION "DS_MAP_LOW_COST_DML_WA"
      ON WORK CLASS "DS_LOW_COST_DML_WC" MAP ACTIVITY WITHOUT NESTED TO "DS_MED_CONC_SUBCLASS", WORK ACTION
      "DS_MAP_HIGH_COST_DML_WA" ON WORK CLASS "DS_HIGH_COST_DML_WC" MAP ACTIVITY WITHOUT NESTED TO
      "DS_LOW_CONC_SUBCLASS", WORK ACTION "DS_MAP_DDL_WA" ON WORK CLASS "DS_DDL_WC" MAP ACTIVITY WITHOUT NESTED
      TO "DS_MED_CONC_SUBCLASS", WORK ACTION "DS_MAP_LOAD_WA" ON WORK CLASS "DS_LOAD_WC" MAP ACTIVITY WITHOUT
      NESTED TO "DS_LOAD_SUBCLASS", WORK ACTION "DS_MAP_OTHER_WA" ON WORK CLASS "DS_OTHER_WC" MAP ACTIVITY
      WITHOUT NESTED TO "DS_LOW_CONC_SUBCLASS" ) ;
```

Even a rather minimal WLM configuration generates a large amount of arcane DDL

Let the WLM configuration tool do the heavy lifting for you.

Actual Result from OPM

```
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD;
CREATE SERVICE CLASS "DS_AUTO_MGMT_SUPER" DISABLE;
CREATE SERVICE CLASS "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" COLLECT AGGREGATE ACTIVITY
        DATA BASE DISABLE;
CREATE SERVICE CLASS "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" COLLECT AGGREGATE
        ACTIVITY DATA BASE DISABLE;
CREATE SERVICE CLASS "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" COLLECT AGGREGATE
        ACTIVITY DATA BASE DISABLE;
CREATE SERVICE CLASS "DS_LOAD_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" COLLECT AGGREGATE ACTIVITY
        DATA BASE DISABLE;
CREATE WORK CLASS SET "DS_AUTOMGMTSU_1263546069031_WORK_CLASS_SET" ( WORK CLASS
        "DS_LOW_COST_DML_WC" WORK TYPE DML FOR TIMERONCOST FROM 0.0 TO 100000.0 POSITION AT 1, WORK
        CLASS "DS_HIGH_COST_DML_WC" WORK TYPE DML FOR TIMERONCOST FROM 100000.0 TO UNBOUNDED
        POSITION AT 2, WORK CLASS "DS_DDL_WC" WORK TYPE DDL POSITION AT 3, WORK CLASS "DS_LOAD_WC"
        WORK TYPE LOAD POSITION AT 4, WORK CLASS "DS_OTHER_WC" WORK TYPE ALL POSITION AT 5);
CREATE THRESHOLD "DS_AUTOMGMTSU_HIGHPRI_1263546069032_EST_COST_TH" FOR SERVICE CLASS
        "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE DISABLE
        WHEN ESTIMATEDSQLCOST > 1000000  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_HIGHPRI_1263546069032_ROWS_RET_TH" FOR SERVICE CLASS
        "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE DISABLE
        WHEN SQLROWSRETURNED > 100000  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_HIGHPRI_1263546069032_TIMEOUT_TH" FOR SERVICE CLASS
        "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE DISABLE
        WHEN ACTIVITYTOTALTIME > 60 MINUTES  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_HIGHPRI_1263546069032_TEMP_TH" FOR SERVICE CLASS
        "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        PARTITION DISABLE  WHEN SQLTEMPSPACE > 102400 K  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_HIGHPRI_1263546069032_ROWS_READ_TH" FOR SERVICE CLASS
        "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        PARTITION DISABLE  WHEN SQLROWSREAD > 1000000 CHECKING EVERY 60 SECONDS  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_HIGHPRI_1263546069032_CPU_TIME_TH" FOR SERVICE CLASS
        "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        PARTITION DISABLE  WHEN CPUTIME > 300 SECONDS CHECKING EVERY 60 SECONDS  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_MED_1263546069033_PRI_CONC_DB_TH" FOR SERVICE CLASS
        "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        DISABLE  WHEN CONCURRENTDBCOORDACTIVITIES > 15 AND QUEUEDACTIVITIES UNBOUNDED  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_MED_1263546069033_EST_COST_TH" FOR SERVICE CLASS
        "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        DISABLE  WHEN ESTIMATEDSQLCOST > 1000000  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_MED_1263546069033_ROWS_RET_TH" FOR SERVICE CLASS
        "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        DISABLE  WHEN SQLROWSRETURNED > 200000  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_MED_1263546069033_TIMEOUT_TH" FOR SERVICE CLASS
        "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        DISABLE  WHEN ACTIVITYTOTALTIME > 60 MINUTES  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_MED_1263546069033_TEMP_TH" FOR SERVICE CLASS
        "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        PARTITION DISABLE  WHEN SQLTEMPSPACE > 102400 K  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_MED_1263546069033_ROWS_READ_TH" FOR SERVICE CLASS
        "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        PARTITION DISABLE  WHEN SQLROWSREAD > 1000000 CHECKING EVERY 60 SECONDS  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_MED_1263546069033_CPU_TIME_TH" FOR SERVICE CLASS
        "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT
DATABASE PARTITION DISABLE  WHEN CPUTIME > 300 SECONDS CHECKING EVERY 60 SECONDS  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_PRI_CONC_DB_TH" FOR SERVICE CLASS
        "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        DISABLE  WHEN CONCURRENTDBCOORDACTIVITIES > 4 AND QUEUEDACTIVITIES UNBOUNDED  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_EST_COST_TH" FOR SERVICE CLASS
        "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        DISABLE  WHEN ESTIMATEDSQLCOST > 1000000  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_ROWS_RET_TH" FOR SERVICE CLASS
        "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        DISABLE  WHEN SQLROWSRETURNED > 200000  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_TIMEOUT_TH" FOR SERVICE CLASS
        "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        DISABLE  WHEN ACTIVITYTOTALTIME > 60 MINUTES  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_TEMP_TH" FOR SERVICE CLASS
        "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        PARTITION DISABLE  WHEN SQLTEMPSPACE > 102400 K  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_CPU_TIME_TH" FOR SERVICE CLASS
        "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        PARTITION DISABLE  WHEN CPUTIME > 300 SECONDS CHECKING EVERY 60 SECONDS  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_PRI_CONC_DB_TH" FOR SERVICE CLASS
        "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE
        DISABLE  WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES UNBOUNDED  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_EST_COST_TH" FOR SERVICE CLASS
        "DS_LOAD_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE DISABLE
        WHEN ESTIMATEDSQLCOST > 1000000  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_ROWS_RET_TH" FOR SERVICE CLASS
        "DS_LOAD_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE DISABLE
        WHEN SQLROWSRETURNED > 100000  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOW_1263546069034_TIMEOUT_TH" FOR SERVICE CLASS
        "DS_LOAD_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE DISABLE
        WHEN ACTIVITYTOTALTIME > 60 MINUTES  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOAD_1263546069035_TEMP_TH" FOR SERVICE CLASS "DS_LOAD_SUBCLASS"
        UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE PARTITION DISABLE  WHEN
        SQLTEMPSPACE > 102400 K  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOAD_1263546069035_ROWS_READ_TH" FOR SERVICE CLASS
        "DS_LOAD_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE PARTITION
        DISABLE  WHEN SQLROWSREAD > 1000000 CHECKING EVERY 60 SECONDS  CONTINUE;
CREATE THRESHOLD "DS_AUTOMGMTSU_LOAD_1263546069035_CPU_TIME_TH" FOR SERVICE CLASS
        "DS_LOAD_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ACTIVITIES ENFORCEMENT DATABASE PARTITION
        DISABLE  WHEN CPUTIME > 300 SECONDS CHECKING EVERY 60 SECONDS  CONTINUE;
ALTER WORKLOAD "SYSDEFAULTUSERWORKLOAD" SERVICE CLASS "DS_AUTO_MGMT_SUPER"  COLLECT AGGREGATE
        ACTIVITY DATA BASE ;
CREATE WORK ACTION SET "DS_AUTOMGMTSU_1263546069033_WORK_ACTION_SET" FOR SERVICE CLASS
        "DS_AUTO_MGMT_SUPER" USING WORK CLASS SET "DS_AUTOMGMTSU_1263546069031_WORK_CLASS_SET" (
        WORK ACTION "DS_MAP_LOW_COST_DML_WA" ON WORK CLASS "DS_LOW_COST_DML_WC" MAP ACTIVITY
        WITHOUT NESTED TO "DS_MED_CONC_SUBCLASS", WORK ACTION "DS_MAP_HIGH_COST_DML_WA" ON WORK
        CLASS "DS_HIGH_COST_DML_WC" MAP ACTIVITY WITHOUT NESTED TO "DS_HIGH_PRI_SUBCLASS", WORK
        ACTION "DS_MAP_LOAD_WA" ON WORK CLASS "DS_LOAD_WC" MAP ACTIVITY WITHOUT NESTED TO
        "DS_MED_CONC_SUBCLASS", WORK ACTION "DS_MAP_LOAD_WA" ON WORK CLASS "DS_LOAD_WC" MAP
        ACTIVITY WITHOUT NESTED TO "DS_LOAD_SUBCLASS", WORK ACTION "DS_MAP_OTHER_WA" ON WORK
        CLASS "DS_OTHER_WC" MAP ACTIVITY WITHOUT NESTED TO "DS_LOW_CONC_SUBCLASS" )  DISABLE;
ALTER SERVICE CLASS "DS_AUTO_MGMT_SUPER" ENABLE;
ALTER SERVICE CLASS "DS_HIGH_PRI_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ENABLE;
ALTER SERVICE CLASS "DS_MED_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ENABLE;
ALTER SERVICE CLASS "DS_LOW_CONC_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ENABLE;
ALTER SERVICE CLASS "DS_LOAD_SUBCLASS" UNDER "DS_AUTO_MGMT_SUPER" ENABLE;
ALTER WORK ACTION SET "DS_AUTOMGMTSU_1263546069031_WORK_ACTION_SET" ENABLE;
```

No queries were harmed during the making of this movie.

- No controls imposed yet
  - All configuration to this point only ***categorizes*** work
  - These changes are safe in a production DB
- Next steps
  - Monitor to validate work is properly categorized

Working Iteratively

- Configure WLM iteratively
  - Categorize work
  - Monitor to validate categories
  - Apply controls
  - Monitor to validate controls

- Make one change at a time
- Keep monitoring data for future comparison

Step 5 - Baseline Monitoring

- Create all WLM related event monitors
  - Activity Event Monitor
    - Allows capture of details about activities in a workload or service class
  - Statistics Event Monitor
    - Captures histograms, counts and high water marks
  - Threshold Event Monitor
    - Allows capture or details about threshold violations
- Choose an appropriate tablespace
  - Spans all partitions
  - Suitable for heavy IO activity
- No overhead for unused WLM event monitors
  - No events captured by these unless requested
  - Configure individual workloads, service classes, work actions to capture only events of interest

Event monitors write data on all partitions, so use a table space that is present on all DB partitions

Event monitors can potentially be the targets of heavy IO activity.

Choose a table space for them that won't interfere with other work.

Turn on WLM Related Monitoring in OPM



If you are using Optim Performance Manager, most of the monitoring setup is taken care of for you.

For WLM, you do still need to turn on the collection of some of the WLM specific monitoring.

There are also some useful event monitors that are not yet managed by OPM.

You can make use of these independently from OPM.

Creating WLM Event Monitors

- Look at **sqllib/misc/wlmevmon.ddl**
  - Modify the script to specify an appropriate tablespace
    event monitors can consume substantial space
- Activate the WLM event monitors
  - `SET EVENT MONITOR <name> STATE 1;`
- Set the collection interval
  - OPM default is 5 minutes
  - Syntax for configuring interval manually
    `UPDATE DATABASE CONFIGURATION FOR <dbname> USING WLM_COLLECT_INT 5;`

Turn on Low Overhead Monitoring

- Service subclasses
  - Aggregate activity (basic)
  - Low overhead
  - RECOMMENDATION: turn this on permanently
  - `ALTER SERVICE CLASS <subclass-name> UNDER <superclass-name> COLLECT AGGREGATE ACTIVITY DATA BASE ;`
- Workloads
  - High water marks for workloads are always collected if statistics event monitor is enabled

Aggregate activity statistics are collected only for subclasses, workloads, work actions.

You must aggregate yourself to find values for superclass or database.

Viewing Histograms in OPM

- Shown in OPM
  - ActivityTotalTime
  - QueueTime
- Aggregated for selected period



Viewing Histograms via SQL

- You can use SQL to view histograms
- SQL for viewing a histogram
  - ```
    SELECT TOP/1000 AS TIME_seconds,
    SUM(NUMBER_IN_BIN) AS #EXECUTIONS FROM
    HISTOGRAMBIN_DB2STATISTICS WHERE HISTOGRAM_TYPE
    = 'CoordActLifetime' GROUP BY TOP/1000 order by
    TOP/1000;
    ```

In the case of OPM, the table is HISTOGRAMBIN in the monitoring repository DB maintained by OPM.

Otherwise, you can directly query the event monitor table
    ```

Step 6 - Validate Mapping of Large Queries

- Compare histograms of ActivityTotalTime for subclasses
- Verify that longer running queries are routed as desired



Determining Cost Boundaries for Work Actions

- Adjust cost boundaries to distribute long running activities to the intended service subclass
- Look at estimated cost histograms to get a sense of estimated costs for queries running in a service subclass
- Estimated cost histograms are collected only if you specify COLLECT AGGREGATE ACTIVITY DATA EXTENDED on the service subclass

Step 7 – Limit Concurrency

- Use the ConcurrentDBCoordActivities threshold
  - Restricts the number of concurrent activities in a service subclass
  - Provides fine grained control
- Start by limiting concurrency for LOAD, large queries

Concurrent **activities** is slightly different from concurrent **queries**.

One query can spawn multiple activities

- Cursors
- Stored procedure calls

Step 8 – Set Priorities for Subclasses (Optional)

- Recommended: adjust agent priority for “by cost” service subclasses
  - Lower priority for long running queries
  - Higher agent priority for rush jobs
  - Agent priority for SYSDEFAULTSYSTEMCLASS must be at least as high as any user defined service subclass

- For data warehouse type workloads, consider adjusting prefetch priority in subclasses
  - Applicable only for scans
  - Has no effect on page reads, such as index lookups
- For OLTP workloads, consider adjusting buffer pool priority
  - This is a v9.7 only feature

DB2 provides 3 predefined service superclasses.

SYSDEFAULTMAINTENANCECLASS

SYSDEFAULTSYSTEMCLASS

SYSDEFAULTUSERCLASS

These are created for you when the database is created and cannot be dropped.

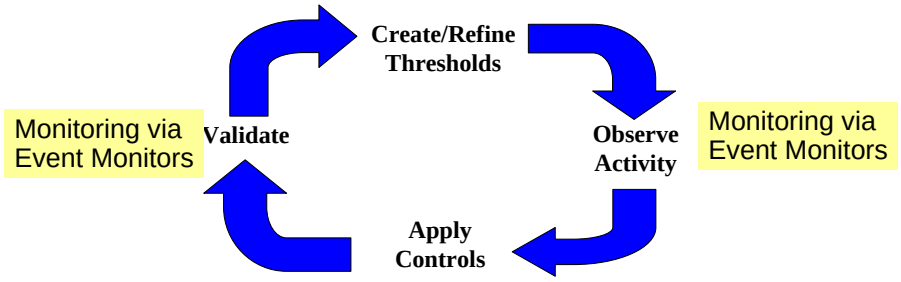
Many internal DB2 activities run in SYSDEFAULTSYSTEMCLASS. Setting the CPU priority of other service classes higher than this can cause significant performance problems.

Step 9 - Protect Against Rogue Queries

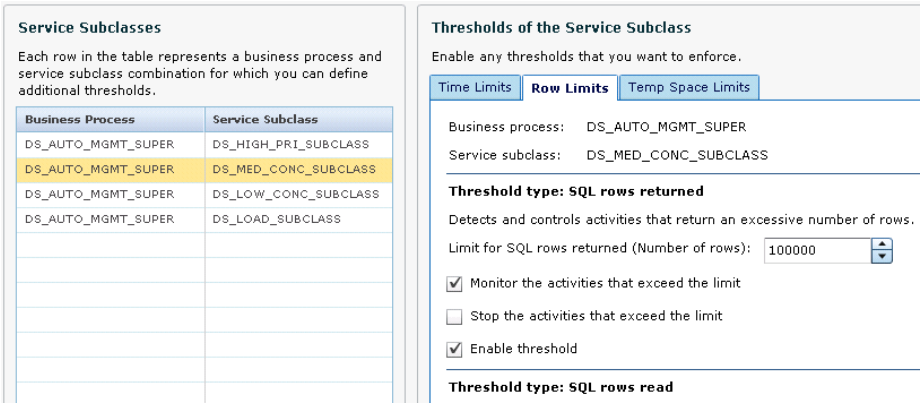
- Reactive activity thresholds
  - ActivityTotalTime
  - CPUTime **(New in v9.7)**
  - SQLRowsReturned
  - SQLRowsRead **(New in v9.7)**
  - SQLTempSpace
  - ConnectionIdleTime
- Predictive activity threshold
  - EstimatedSQLCost

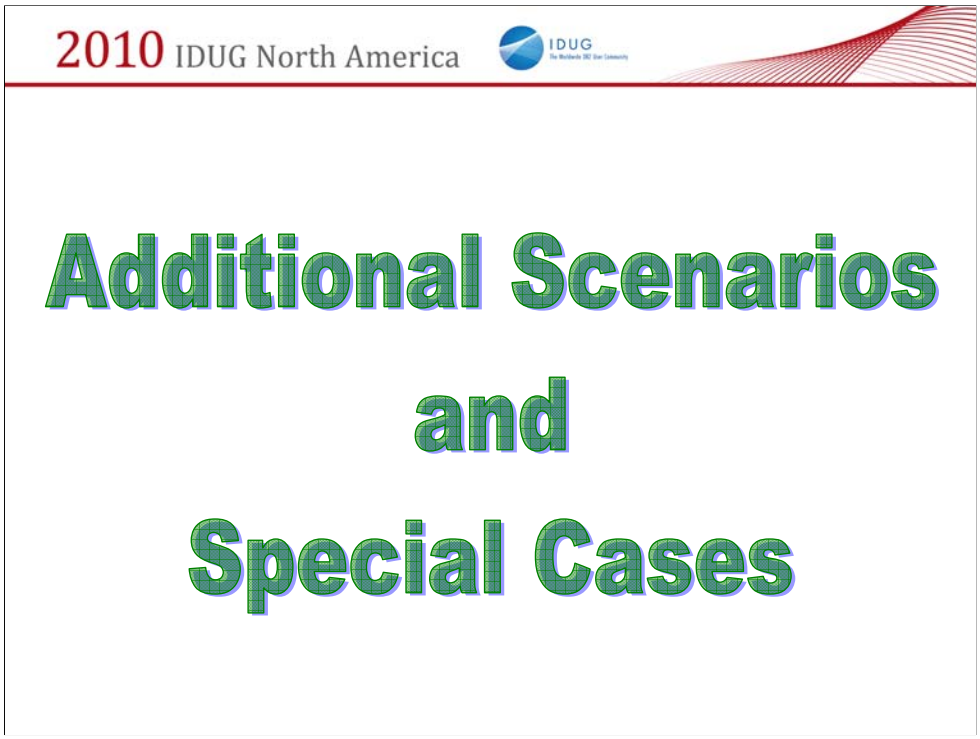
Configure Thresholds Iteratively

- WLM policies cannot be verified outside of production DB
  - Verify by deploying monitoring only controls
  - After reviewing monitoring data, alter to enforced controls
- WLM policies must be informed by baseline monitoring
  - Browse history of how long queries run
  - Create thresholds based on historical trends and data



Threshold Configuration in OPM



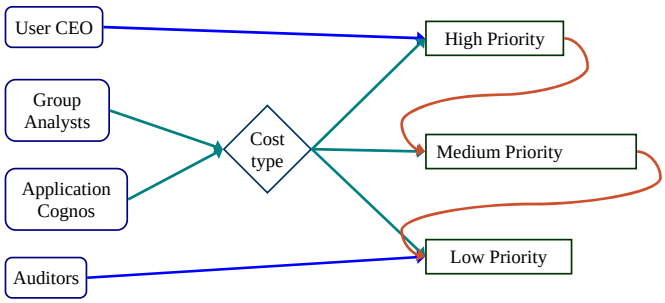


This concludes coverage of the basic WLM constructs and recommended practices that apply to most customers.

The remainder of the presentation covers some additional scenarios and alternative techniques

Alternate Solution - Priority Aging

- "Easy button"
  - Work started at appropriate priority based on cost or workload
  - Automatically remap long running work to lower priority
  - Customization is possible, but not necessary



Benefits

•Easy to implement
Works right out of the box with no customization necessary

•Low risk
Mis-configuration is very unlikely to cause any serious problem

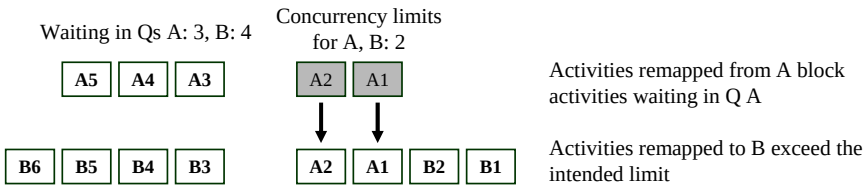
•Very tolerant
You will get some benefit even for poorly chosen values

Drawbacks

•Minimal benefit

Mixing Priority Aging with Concurrency

- Recommendation: choose one approach, don't mix
- WLM configuration tooling in OPM forces you to choose
  - Presented as separate solutions
  - You can change your mind later, but tooling treats it as fresh start



Consequences of concurrency thresholds + priority aging

- Mapping activities out of a service class does not free any concurrency slots, so incoming work can be blocked, waiting on activities previously mapped out
- Mapping activities into a service class can easily exceed the intended concurrency limit for the destination service class

Mixing Priority Aging with Concurrency

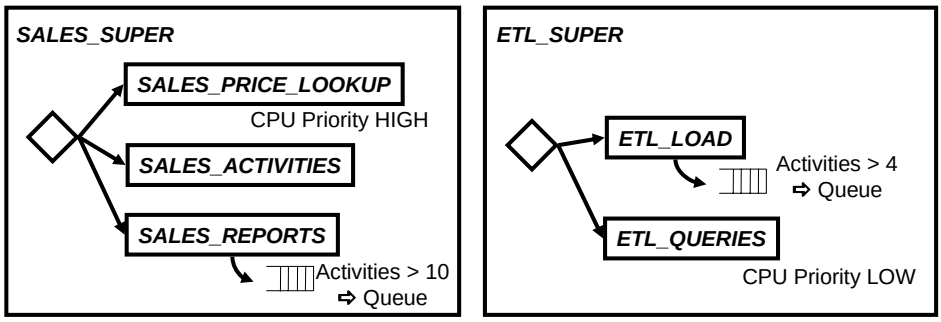
- Recommendation: choose one approach, don't mix
- Potential pitfalls
  - Activities continue to hold concurrency tickets from originating service class after they are mapped
  - Activities mapped into a service class do not queue for entry
- WLM configuration tooling in OPM forces you to choose
  - Presented as separate solutions
  - You can change your mind later, but tooling treats it as fresh start

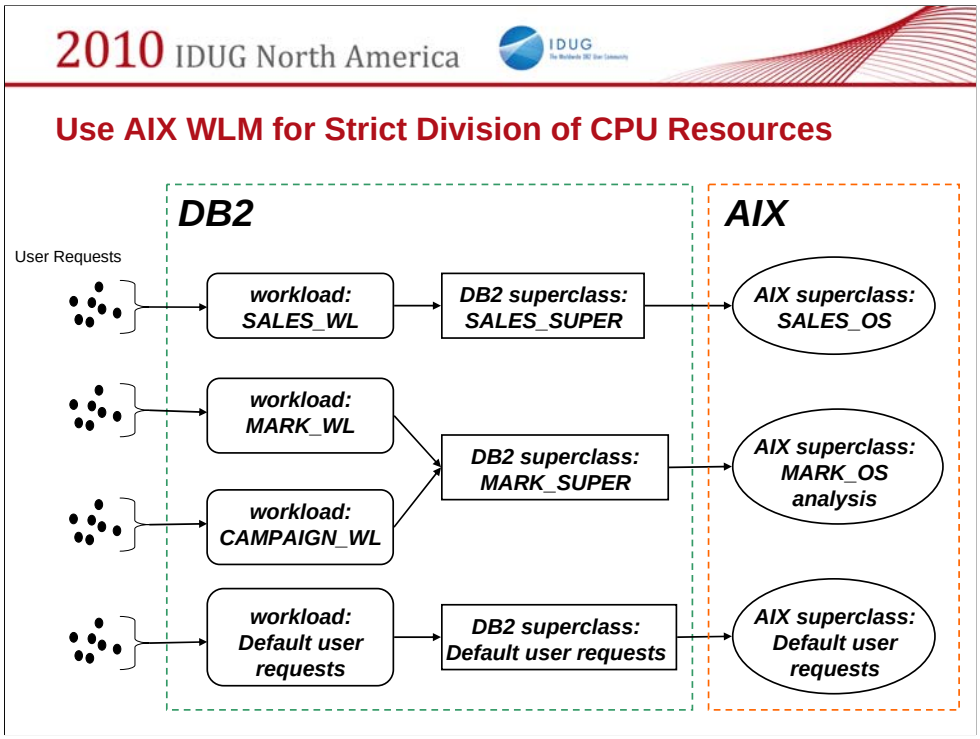
Consequences of concurrency thresholds + priority aging

- Mapping activities out of a service class does not free any concurrency slots, so incoming work can be blocked, waiting on activities previously mapped out
- Mapping activities into a service class can easily exceed the intended concurrency limit for the destination service class

Scenarios for Multiple Service Superclasses

- Divide resources
  - Separate superclasses for departments or applications
  - Divide total available system resources among superclasses
- Different rule sets
  - Separate superclass for ETL
    - Up to 4 concurrent LOAD activities
    - Huge queries allowed, but at low priority





In this example the system has been divided between two departments: sales and marketing.

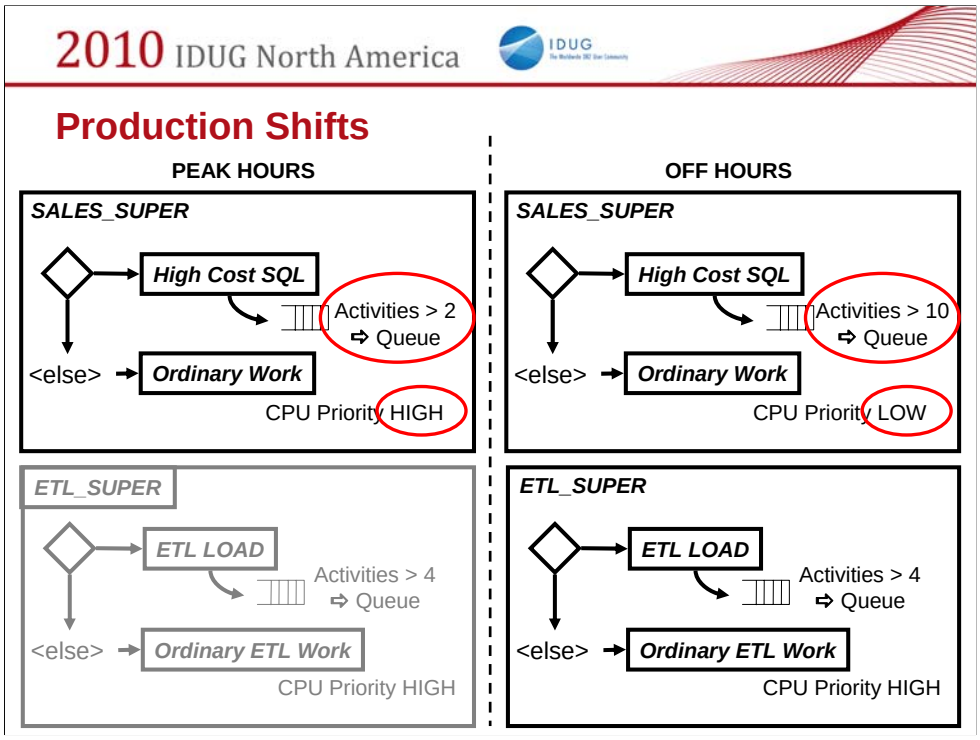
There is a service superclass for each department.

Workloads map connections to the appropriate superclass, based on user group.

Each DB2 service superclass is associated with an AIX service class that limits its share of CPU time.

Integrating DB2 WLM with AIX WLM

- HARDMAX caps CPU usage of a service class
  - Applicable in scenarios where concurrency limits are insufficient in reducing priority of a service class
- AIX WLM
  - Use HARDMAX to cap CPU usage of a service class
  - AIX WLM provides additional statistics
  - Other features of AIX WLM applicable only for systems which are heavily resource constrained
- Refer to WLM Best Practices white paper for details
  - Example script for dynamically adjusting HARDMAX



At different times of the day, different WLM policies are appropriate.

Best practice is to have the same service classes, thresholds etc. in place at all times.

At shift changes:

- Enable/Disable service classes to allow/prevent the corresponding class of activities
- Alter values of thresholds, CPU and prefetch priorities to adjust priorities at different times

In this simplified example, concurrency for sales reports is limited to 2 during peak hours, but 10 for off hours.

ETL processing is not allowed at all during the peak hours. This is accomplished by disabling the service class.

Different WLM entities behave slightly differently when disabled

- Disabling a service class causes work routed to that service class to be rejected
- Disabling a workload effectively removes it from the order of evaluation.
  When evaluating a connection to match it to a workload, DB2 will skip over the disabled ones looking for another to match it to.
  Recall that the default workload always exists, always matches everything and is always last in the list, so work will never be rejected due to disabling workloads.
- Disabling a threshold or a work action set effectively removes it.

Recommendations for Production Shifts

- Avoid drop / create of workloads and service classes
  - Drop requires you first disable, then wait for activities to drain
- To block a workload from running
  - ALTER WORKLOAD … DISALLOW DB ACCESS;
  - Matching connections will error out
- Alter is robust and online
  - Alter priority of a service class
  - Alter mapping of a workload
  - Alter limits for a threshold
  - Alter a threshold to disable

Migration from QP / Governor

- QP can provide insights about activities in your DB
- WLM in DB2 offers additional controls
  - Thresholds
  - Controls for activities other than DML
  - CPU and prefetch priorities
  - Can control other activities such as DDL, LOAD
- Concurrency controls in WLM are slightly different

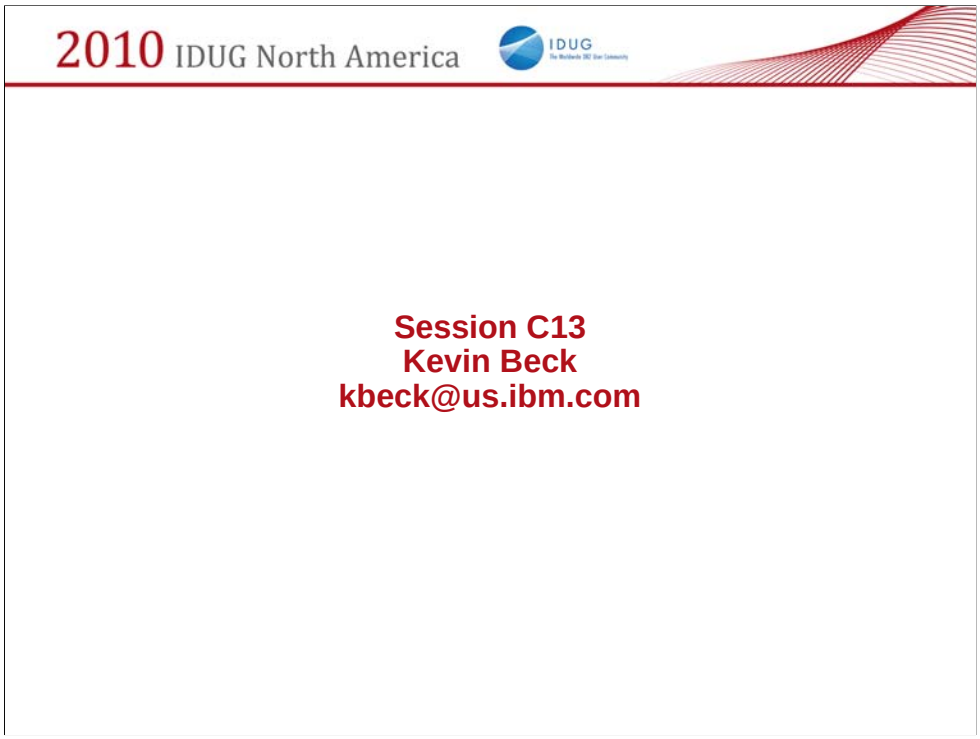
- Migration script available to partly automate migration
  - sqllib/samples/perl/qpwlmmig.pl
  - Recommend revisiting policies during migration
  - OPM recognizes output of this script

WLM provides additional capabilities beyond what was possible in QP.

If you are migrating from QP, do take it as an opportunity to revisit your existing policies and rules.

Be aware that there are some differences in how WLM works, compared to QP.

The migration script produces a WLM configuration that emulates QP very closely, but there are few slight, unavoidable differences.



Kevin is the architect for tooling to support workload management features in DB2 LUW. His special interests include business intelligence, data mining, and data warehouses. He has been a member of the DB2 development team at IBM since 2001, and prior to that, a member of the Informix data server development team. He has contributed to benchmarks and performance work, and he has deep knowledge of how the Informix and IBM data servers operate at the query processing level. Kevin has many years of experience delivering education and presentations on data server topics.