SAP HANA Platform 2.0 SPS03
Introduction to workload management

Lucas Kiesow, SAP HANA Product Management
April 2018
Disclaimer

- The information in this presentation is confidential and proprietary to SAP and may not be disclosed without the permission of SAP. Except for your obligation to protect confidential information, this presentation is not subject to your license agreement or any other service or subscription agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or any related document, or to develop or release any functionality mentioned therein.

- This presentation, or any related document and SAP’s strategy and possible future developments, products and or platforms directions and functionality are all subject to change and may be changed by SAP at any time for any reason without notice. The information in this presentation is not a commitment, promise or legal obligation to deliver any material, code or functionality. This presentation is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This presentation is for informational purposes and may not be incorporated into a contract. SAP assumes no responsibility for errors or omissions in this presentation, except if such damages were caused by SAP’s intentional or gross negligence.

- All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.
Agenda

1. NUMA-awareness in workload management
2. What is workload in the context of SAP HANA?
3. Options for managing workload
4. Dynamic Admission Control for workload management
5. SAP HANA Cockpit 2.0 integration for workload management
NUMA-awareness in SAP HANA
Scale out versus scale up

Why scale up instead of scaling out?
- If no obvious and consistent partitioning criterion available
- If business transactions cover a mesh of different business objects
- If normalized data model includes OLAP and OLTP workload
NUMA-awareness in SAP HANA
Topology of a multi-socket system

1-hop latency = medium

Local latency = low

max-hop latency = high

Blade #1
Blade #2
Blade #3
Blade #n
NUMA-aware Data Placement

- Randomized hashing scheme to distribute memory allocation for tables uniformly

NUMA-aware Job Scheduling

- Preferred NUMA location can be passed to job
- With SAP HANA 2 SPS0 jobs are bound to target NUMA node (no stealing allowed)
- If no target NUMA node is defined, topology-aware smart stealing (from NUMA neighbors) is allowed
What is workload in the context of SAP HANA?
What is workload in the context of SAP HANA?

General definition

- **Workload is a set of requests with common characteristics, e.g.**
  - Source of request (user, application, etc.)
  - Type of query (analytical, transactional, etc.)
  - Business importance (high, low, etc.)
  - System Level Objectives (SLOs) (response time, throughput, etc.)

- **Workload has certain resource demands to the system, e.g.**
  - CPU
  - Memory
  - Disk I/O
  - Network bandwidth

- **Mixed workloads compete for shared system resources...**
  - If a request consumes a significant amount of system resources, other requests are subject to queuing effect
  - Waiting workloads might fail to meet their expected performance
  - Overall system performance degradation may occur
What is workload in the context of SAP HANA?
Robust performance

Run the database in an optimal resource state and maximize overall system performance. Ensure that requests meet their required SLOs.

Map your Workload to System Resources
- CPU
- Memory
- Priority
- Degree of Parallelism
- Workload Classes

Analyze System Performance
- CPU utilization
- Memory consumption
- Expensive Statements (Memory)
- Parallelization

Understand your Workload
- Business Importance
- Application Type
- Query Types
- Resource Demand
- Users
- Conflicts between Workloads
Options for managing workload
Options for managing workload
Different levels of granularity

- **static settings**
  - Settings on OS level
  - SAP HANA global settings (.ini parameters)

- **dynamic settings**
  - SAP HANA workload classes

- **low granularity**
  - Operating system level

- **high granularity**
  - SAP HANA system level
  - SAP HANA session level
Options for managing workload
Settings on operating system level

CPU-binding is a resource pooling technique on OS-level, outside the view of the SAP HANA database. We recommend to use SAP HANA parameter settings first.

Features

- **Partitioning CPU resources on process-level**
  - within single SAP HANA instance
  - between SAP HANA instances
  - between SAP HANA tenants

- **Binding CPU resources to processes**
  - Guaranteed CPU resources on process-level
  - Ensures data locality per CPU-socket
  - Keeps specific resources dedicated to specific applications or use cases

CPU-binding is a resource pooling technique on OS-level, outside the view of the SAP HANA database. We recommend to use SAP HANA parameter settings first.

**Features**

- **Partitioning CPU resources on process-level**
  - within single SAP HANA instance
  - between SAP HANA instances
  - between SAP HANA tenants

- **Binding CPU resources to processes**
  - Guaranteed CPU resources on process-level
  - Ensures data locality per CPU-socket
  - Keeps specific resources dedicated to specific applications or use cases

CPU-binding is a resource pooling technique on OS-level, outside the view of the SAP HANA database. We recommend to use SAP HANA parameter settings first.
Options for managing workload
Settings on SAP HANA system level

SAP HANA global settings can be done as parameter changes in the database .ini files. These parameters affect the complete SAP HANA system.

SAP HANA supports NUMA-aware data placement, hash-distributing tables and partitions across NUMA nodes, this ensures faster processing between memory and CPU. SAP HANA also supports NUMA-aware job scheduling, taking into account the NUMA location of tables when scheduling jobs.

- **CPU**
  - thread pool size for complex, long-running statements (e.g. OLAP) and SAP HANA system tasks and for short-running requests (e.g. OLTP)

- **Degree of Parallelism**
  - statement thread limit for the SAP HANA system

- **Memory**
  - global allocation memory limit for the SAP HANA system
  - statement memory limit for the SAP HANA system
Options for managing workload
Settings on SAP HANA system level – dynamic resource management

- SAP HANA dynamically balance thread-pool sizes depending on workload
  - SQL executor
    - in peak load situations or certain scenarios, statement may exceed its thread limit and create additional threads
    - SAP HANA retires these additional threads after peak load is gone
  - Job executor
    - may shrink its available pool size (temporarily) to provide additional resources to the SQLexecutor
    - may exceed its available pool size (temporarily), e.g. due to blocked jobs (waiting for other jobs, IO wait)

SAP HANA auto-determines the degree of parallel statement execution

- Default: unlimited use of available threads for parallel statement execution
Options for managing workload
Settings on SAP HANA session level

SAP HANA workload classes allow the dynamic mapping of SAP HANA resources to SAP HANA DB clients based on workloads. Workload classes define resource properties on statement level granularity.

- Create workload class with provided options for definition
- Define mapping rules based on session variables
- Monitor system for irregular behavior
Options for managing workload
Settings on SAP HANA session level

Configurable limits:
• statement thread limit: max # of parallel threads to execute a statement
• statement memory limit: max # of memory allocated per statement (unit is GB)
• total statement thread limit (new with SAP HANA SPS02): max # of parallel threads to execute statements belonging to a workload class
• total statement memory limit (new with SAP HANA SPS02): max # of memory allocated to a workload class (unit is GB)
• priority: execution priority for a statement (0 is lowest, 9 is highest priority, and 5 is default)
• statement timeout (new with SAP HANA SPS03): timeout value (unit is seconds, disabled by default)

Usable session variables:
• APPLICATION USER NAME: Name of application user logged into application
• CLIENT: ABAP client number
• APPLICATION COMPONENT NAME: Name of application component
• APPLICATION COMPONENT TYPE: Type of application component
• APPLICATION NAME: Name of application
• USER NAME: Name of DB user logged into database
Options for managing workload
Settings on SAP HANA session level – session variables

Any SAP HANA client-application that wants to benefit from SAP HANA workload classes needs to pass the SAP HANA session variables with their client-specific values to SAP HANA.

- With SAP HANA 1.0 SPS12, all SAP-clients providing session variables are listed in SAP note 2331857.

### Setting of SAP HANA session variables

- **First option:** Get meaningful default values from OS → This is always possible and ensures that some variables are set.
- **Second option:** Optional passing of variables via **connection string** → These overwrite first option, can be set in connection string (JDBC / SQLDBC).
- **Third option:** Explicit setting of **session variables** via setClientInfo() → Developer/vendor can call setClientInfo() function in application, overwrites all of the above.

### Setting session variables

- Setting session variables in the client info object of the SAP HANA interface is documented in the **SAP HANA developer guide** (≥SAP HANA 1.0 SPS12)

<table>
<thead>
<tr>
<th>SAP client application</th>
<th>Supported session variables (key = value)</th>
<th>Min. Release Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>HANA WebIDE (IO Client)</td>
<td>APPLICATION = &quot;sap.hana.ide&quot; APPLICATIONUSER = &lt;OS-username&gt; CLIENT = &lt;client&gt; USER = &lt;DB-username&gt;</td>
<td>HANA rev 102</td>
</tr>
<tr>
<td>HANA Studio</td>
<td>APPLICATION = &quot;HDBstudio&quot; APPLICATIONUSER = &lt;OS-username&gt; CLIENT = &lt;client&gt; USER = &lt;DB-username&gt;</td>
<td>HANA rev 102</td>
</tr>
<tr>
<td>ABAP applications</td>
<td>APPLICATION = &quot;ABAP&quot;&lt;ABAP-HD&gt; APPLICATIONUSER = &lt;ABAP-username&gt; CLIENT = &lt;ABAP-Client&gt; USER = &lt;DB-username&gt;</td>
<td>ABAP Server 7.x HANA rev 102</td>
</tr>
</tbody>
</table>
Options for managing workload
Limit resources per application, user or workload class

Options to limit CPU/memory prior to SAP HANA 2.0 SPS02:

• Options on system/global level:
  • Statement thread limit
  • Statement memory limit

• Options on session level:
  • Workload classes offer statement thread limit, statement memory limit, statement priority
  • Mapping is done via sessions variables
  → Only works on statement level; multiple statements from one application, user or workload class can not exceed their limits, but parallel statements cannot be limited
Options for managing workload
Limit resources per application, user or workload class

Prior to SAP HANA 2.0 SPS02
Statement #1: 20GB Memory limit
Statement #2: 20GB Memory limit
Statement #3: ...

With SAP HANA 2.0 SPS02
Statement #1
Statement #2 20 GB Memory Limit
Statement #3
Options for managing workload
Limit resources per application, user or workload class

- Can have either statement-level or application-level concurrency limit, but not both
- Application context (limitation on application-level) always precedes request context (limitation on statement-level) if set
Options for managing workload

Workload class wildcards

Wildcards for workload class mappings

- **Problem:** Mapping workload classes with multiple values for an application property
- **Solution:**
  - Add wildcard functionality for workload class mapping
  - Wildcard should be placed at the end of the value (i.e., prefix wildcard only like HANA%)
  - Only 1-byte ASCII character is supported for wildcard and it should be printable (i.e., does not support special ASCII code like 0x00 (NUL), 0x06 (ACK))
  - “USER NAME” property does not support wildcard as it should be a user name existing in the database
Options for managing workload

Workload class hint

Hint to execute given statement with the resource configurations maintained for a certain Workload Class. The hint overrules normal dispatching of Workload Classes, should there be multiple workload classes assigned to the statement.

Use case #1: Re-run certain SQL statement with different Workload Class attached to test impact during execution for possible optimization potential or as a potential emergency fix in case there appear to be issues with Workload Classes.

<SQL statement> WITH HINT (USE WORKLOAD CLASS '<Workload Class name>')</n

Use case #2: The new hint can be used with Plan Pinning, making it possible to execute cached statements including the pinned hint, without changing the application code related to the statement.

ALTER SYSTEM PIN SQL PLAN CACHE ENTRY LIKE <plan_id> WITH HINT (<hint_items>)
Options for managing workload

Query timeout

Query timeout
• Currently supported for
  • SQLDBC
  • JDBC/ODBC
  • ADO.Net
  • HDBSQL
• Configurable through client-side API
• Configurable via .ini parameter
  • indexserver.ini [session] statement_timeout
• Configurable via workload class definition
  • STATEMENT TIMEOUT property
Dynamic Admission Control for workload management
Dynamic Admission Control
Overview

- **Motivation:**
  - Goal: Handle short peak-load situations
  - Assumption: correctly sized hardware
  - Avoid or remedy contention issues by queueing or rejecting incoming request

- **Idea:**
  - Keep track of recent CPU and memory consumption
  - Handles requests at **Session Layer** only
  - Request via **HTTP to XSC, TrexNet, and other internal calls are not considered** (will be passed through)
  - Only affects newly-invoked statements, no impact on running statements
  - By default disabled, configuration via ini-parameters in indexserver.ini → [admission_control]

2 public monitoring view are available
- M_ADMISSION_CONTROL_STATISTICS
- M_ADMISSION_CONTROL_QUEUES
Extended M_CONNECTIONS.CONNECTION_STATUS for queueing status
With SAP HANA 2.0 SPS02 the feature is turned on by default.
Default mode is set to queuing only, no rejections from the queue.
SAP HANA Cockpit 2.0 integration for workload management
Managing workload classes
Integration into HANA Cockpit

- Full integration into HANA Cockpit 2.0
- Includes overview of workload classes, option to create new workload classes, set of global limits, define user-specific parameters, manage admission control

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Execution Priority</th>
<th>Statement Memory Limit (GB)</th>
<th>Statement Thread Limit</th>
<th>Mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limit memory</td>
<td>Enabled</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Higher memory limit</td>
<td>Enabled</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Managing workload classes
Workload class management

- Interface to create workload class
  - Define limits and priority
  - Define mapping details
- Interface for mapping workload classes
- New UI for importing and exporting workload classes
Managing workload classes
User-specific parameters, Global limits and DAC

- Interface to define user-specific parameters related to workload management
- Interface for defining global (system-wide) limits related to workload management
- New UI for configuring Dynamic Admission Control
Current roadmap for workload management
Summary
Key Takeaways

- **Robust Performance via**
  - Scenario-based system configuration
  - Workload Classes
  - Admission Control
  → robust response times and throughput
  → avoid contention on 1000+ cores on systems with many cores

- **Scalable Performance via**
  - NUMA-aware data placement
  - NUMA-aware job scheduling
  - Numerous scalability improvements
  → OLAP workload highly NUMA-optimized
  → good scalability on 1000+ cores on systems with many cores
Thank you.

Contact information:

Lucas Kiesow
SAP HANA Product Management
+49 160 3603467
lucas.kiesow@sap.com