Expert Sizing & Methods of Sizing Validation

Performance & Scalability, SAP
December, 2020

PUBLIC
Goals of this Presentation

Sometimes customers and consultants develop their own application or strongly modify existing coding, so that standard guidelines often cannot apply.

This presentation will

- Explain principles of expert sizing, that is, sizing beyond the scope of tools
- Show you how to **create your own sizing guidelines**
- Provide a small **check list** for test cases and test systems
- Explain methods of sizing validation
Agenda

Expert Sizing
- Guidelines
- Expert Usage of Quick Sizer
- Creating Your Own Sizing Guideline

Methods of Sizing Validation
Interface Integration – Example of Complex Data Flow

1. **CRM → SCM (ATP check):** sRFC (synchronous RFC, proprietary protocol)
2. **CRM → ECC (transfer unchecked delivery):** qRFC (CRM middleware)
3. **ECC ← SCM (transfer unchecked delivery):** qRFC (CRM middleware)
4. **PI → ECC (transfer ASN):** http / XML
5. **ECC → PI → SCM (transfer Advanced Shipping Notification (ASN)):** http / XML

---

**ECC Server**
- Send orders to SCM via qRFC
- Send Advanced shipping notification (ASN) to SCM via PI

**CRM Server**
- Perform ATP check in SCM
- Send unchecked delivery to ECC

**SCM Server**
- Receive ASN from PI
- Receive orders from ECC
- Send orders to ECC

**User 1**
- Send orders to SCM via qRFC
- Send Advanced shipping notification (ASN) to SCM via PI

**PI Server**
- Send ASN to ECC
- From ECC route ASN to SCM
Sizing Integrated Scenarios: Custom Coding/3rd Party/Cloud

General Statements

- Effects of custom coding/3rd party often not predictable
  - A "small" modification may have a large impact

Method

- Analysis: To what extend are bread-and-butter processes affected?
  - Small backend transactions without business impact may be omitted
  - If necessary, measure performance of test cases
  - SAP delivers the toolset (CCMS, eCATT)
  - For quality assurance, performance measurements ought to be done, anyhow

- Use modular approach
  - You may use SAP standard sizing tools for the SAP-related sizing
  - Add the requirements of 3rd party, custom coding on top
  - Note: some ISVs (Independent Software Vendors) are required to provide standard sizing guidelines

<table>
<thead>
<tr>
<th>Scenario</th>
<th>SAPS</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000 SAPS</td>
<td>12 GB</td>
<td></td>
</tr>
<tr>
<td>800 SAPS</td>
<td>3 GB</td>
<td></td>
</tr>
<tr>
<td>2000 SAPS</td>
<td>6 GB</td>
<td></td>
</tr>
<tr>
<td>1500 SAPS</td>
<td>6 GB</td>
<td></td>
</tr>
<tr>
<td>4000 SAPS</td>
<td>12 GB</td>
<td></td>
</tr>
<tr>
<td>11,300 SAPS</td>
<td>39 GB</td>
<td></td>
</tr>
</tbody>
</table>
Interface Integration – Load Caused by Communication

**Procedure**

- Size user and quantity structure individually for each system
- Assume roughly 10% overhead for communication in sending and receiving system (IDoc, RFC)
  - Depends on size of the file (small files produce big CPU/network overhead, big files need much memory)
  - Error handling
  - Monitoring
  - Transactional behavior
- Map to GUI transactions
- Can be synchronous (⇒ impact on response time!) or asynchronous
How to Size Initial Data Load

Initial data load
- In general, even very high volumes of master data are small compared to document-type data
- Quick Sizer only has limited master data sizing (SAP for Utilities, Banking, SAP CRM)
- SAP NetWeaver Master Data Management → Sizing guideline

Rule of thumb for initial data upload
- Precondition: ratio of DB:App = 1:5
- 1s for each object on one CPU core
  - In some cases, where resource consumption for the data upload might be critical, such as material master, SAP notes show how to improve throughput for initial load
Agenda

Expert Sizing
- Guidelines
- Expert Usage of Quick Sizer
- Creating Your Own Sizing Guideline

Methods of Sizing Validation
Example: Quick Sizer, SAP’s Online Sizing Tool

Characteristics
- Structured sizing questionnaires
- Input for
  - Greenfield sizing
  - GoingLive Check
- Hardware vendor contact list

Facts and Figures
- Available online since 1996
- SAP HANA Quick Sizer version available since 09/2014
- SAP HANA Cloud Quick Sizer version available since 2018
- Free of charge
- As of 2016: avg. 35,000 new projects per year

Scope
- SAP Key applications
  - SAP S/4HANA
  - SAP HANA Standalone
  - SAP BW/4HANA
  - etc.
- Sizing by users and/or by throughput
Batch Processing

Specific batch jobs in batch part of Quick Sizer

- Make only sense in batch mode

Check batch screen

If respective online process is contained in Quick Sizer

⇒ Use that one, overhead negligible
Additional Guidelines for Quick Sizer

**Example**
- **Legacy**
  - File
  - RFC
  - BAPIs
  - XML
  - SOAP...
- **SAP Business Suite**

**Procedure**
- Size user and quantity structure individually for each system
- Assume roughly 10% overhead for communication in sending and receiving system (IDocs, RFC)
  - Depends on size of the file → each file has an administrative overhead
  - Error handling
  - Monitoring
  - Transactional behavior
- Can be synchronous (impact on response time) or asynchronous

**Advantages**
- More detailed level

**Disadvantages**
- Have to be mapped to Quick Sizer
- Limited transparency

### Table 3: Throughput - Standard Sizing

<table>
<thead>
<tr>
<th>countries/standard</th>
<th>objects</th>
<th>items</th>
<th>% avg</th>
<th>% std</th>
<th>index</th>
<th>size</th>
<th>size</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS-DAO</td>
<td>A</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>CS-DAO</td>
<td>F</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>SE-CUST</td>
<td>A</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>SE-CUST</td>
<td>F</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>SE-DLIL</td>
<td>A</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>SE-DLIL</td>
<td>F</td>
<td>P</td>
<td>30,000</td>
<td>30</td>
<td></td>
<td>12</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>SE-POS</td>
<td>A</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>SE-POS</td>
<td>F</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>SE-CUST</td>
<td>A</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>SE-CUST</td>
<td>F</td>
<td>P</td>
<td>30,000</td>
<td>30</td>
<td></td>
<td>10</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>SE-POS</td>
<td>A</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>SE-POS</td>
<td>F</td>
<td>P</td>
<td>30,000</td>
<td>30</td>
<td></td>
<td>10</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

- Importing orders from CRM
- RFC overhead
Expert Use of Quick Sizer: Multiple Averages

Multiple averages are rarely required, e.g. with Retail

- Per default, Quick Sizer checks if average CPU is smaller than peak
- ID to determine which average(s) is/are checked against which peak(s)
Additional Guidelines

Expert sizing is usually performed for comprehensive or high-volume sizings
- Using Quick Sizer functions for system integration (e.g. RFC)
- Creating a custom sizing guideline
- Continuous validation of assumptions and improvements
- The Quick Sizer can help to avoid potential time zone clashes

For expert sizing we recommend the Lego approach
- Use the Quick Sizer as the basis to collect the key data
- For special data constellations or requirements perform separate sizings
- To record your custom sizings you may use the Quick Sizer functions
Agenda

Expert Sizing
  ▪ Guidelines
  ▪ Expert Usage of Quick Sizer
  ▪ Creating Your Own Sizing Guideline

Methods of Sizing Validation
Assumptions

- Performance and sizing are influenced by manifold factors that cannot be accounted for in such a presentation.
- This approach is straightforward and must always be adopted to the current situation.
- The measuring guidelines described here are geared to yield sizing results, for a performance analysis more information is required.
- Performance optimizations have already taken place:
  - The application adheres to the general performance standards
  - Measurements alone don't speed up the application, i.e.: the solution is scalable
- Most applications still run on ABAP, so the focus here is on NW-ABAP tools. You can use other tools, too, as long as they deliver reproducible results.
How To Make Sizing Measurements – A Bird’s Eye View

<table>
<thead>
<tr>
<th>Business Application, Scenario, Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test Case</strong></td>
</tr>
<tr>
<td>▪ 80/20 rule, simulate user behavior or background processing</td>
</tr>
<tr>
<td>▪ <strong>Representative</strong> (switch on the typically needed functions, typical amount of data, typical data distribution), reproducible</td>
</tr>
<tr>
<td><strong>Test System</strong></td>
</tr>
<tr>
<td>▪ Stable, no transports, no virtualization mechanisms</td>
</tr>
<tr>
<td>▪ Power saving and performance optimization of the CPUs must be switched off</td>
</tr>
<tr>
<td><strong>Single-User Tests</strong></td>
</tr>
<tr>
<td>▪ Adherence to basic performance requirements (e.g. accesses to DB, linear dependency,...)</td>
</tr>
<tr>
<td>▪ <strong>Scalability with the size of the input parameter(s), e.g. size of the object.</strong> Scalability is referred to the resource requirements, such as CPU time, memory and DB disk size</td>
</tr>
<tr>
<td><strong>Load tests are also possible</strong></td>
</tr>
<tr>
<td><strong>Multi-User Load Tests</strong></td>
</tr>
<tr>
<td>▪ Especially for Java-based applications, or high-throughput</td>
</tr>
<tr>
<td>▪ To check locking mechanisms, stability, etc.</td>
</tr>
<tr>
<td>▪ Hardware scalability, sizing verification</td>
</tr>
</tbody>
</table>
How to Define Sizing-Relevant Business Processes and Test Cases?
Sizing-Relevant Business Processes

Business processes that are enabled for parallel processing
- Parallel execution by concurrent users
- Split into packages that can run in parallel to reduce processing time

Business processes with significant usage
- Frequently-used business processes, i.e. "bread and butter" functionality
- Business processes that must run in a tight timeframe

Sizing is relevant for platforms
- Minimal hardware requirements for installation and configuration
- Sanity minimum hardware with a Reference scenario
Sizing Test Cases

Test cases should be defined for the sizing-relevant business processes

Reproducible test cases
- The test must always produce the same measurement results
- Ideally, a business process is represented by the same test cases in different releases

Fully-documented test cases
- Document the required settings, such as customization parameters and user roles
- Document the type and volume of master data; ideally maintain data generation procedure
- Describe the flow of actions; ideally take screenshots of all process steps
Example: Reproducible Test Cases

**Good Example**

Create a new sales order
- Choose “New sales order”
- Choose “Add item” and enter the product ID and quantity
- Choose “Save order”

**Bad Example**

My personal inbox
- Open Inbox
- Create Task
- Close Inbox

“Open Inbox” will consume more and more CPU, memory, and so on, because it will display more and more tasks.

To make the test case reproducible, add the processing or deletion of the task to the test case definition.

**Special Example**

Create a billing document for a sales order
- Select sales order
- Start billing job

The sales order becomes "unusable" after the first execution of the test case.

Several customer orders with the same characteristics (number of items, type of product, and so on) must be available.
Example: Test Case Documentation (1/3)
Poorly-Documented Test Case

Bad Example
1. Main screen
2. Call /nVA01 (Create customer order)
3. 1st screen
4. 2nd screen (with 5 items)
5. Choose Save

Problems with Test Case Definition
- Input values are not specified, for example, what are the input values for the items?
- Navigation steps are not specified, for example, how do you switch between the first and second screen?
- Was “input help” screen measured in previous release?
Example: Test Case Documentation (2/3)

Fully-Documented Test Case

Good Example

<table>
<thead>
<tr>
<th>Dialog Step</th>
<th>Description</th>
<th>Input Field</th>
<th>Input Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation step: Log-on (not measured)</td>
<td>Log-on to the system</td>
<td>User Password</td>
<td>Testuser1 abc12345</td>
</tr>
<tr>
<td>1_CreateOrderView</td>
<td>Start the transaction for creating orders</td>
<td>Transaction code field</td>
<td>/nVA01</td>
</tr>
<tr>
<td>2_ChOOSETypeOfOrder</td>
<td>Specify the type of order</td>
<td>Order type</td>
<td>CMR</td>
</tr>
<tr>
<td>3_ClickOnItemsOverview</td>
<td>Open the screen for entering order items</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4_EnterItemsDetails</td>
<td>Enter five items</td>
<td>Item</td>
<td>10,20,30,40,50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Material</td>
<td>100-100,100-200,100-300,100-400,100-500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quantity</td>
<td>1,1,1,1,1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unit</td>
<td>ST,ST,ST,ST,ST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Description</td>
<td>Oranges, apples, bananas, kiwi, grapes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Customer material number</td>
<td>239489273,370089286,239481383,239467923,249485243</td>
</tr>
<tr>
<td>5_SaveOrder</td>
<td>Choose Sales Document — Save.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Test Case Documentation (3/3)
Fully-Documented Test Case

Another Good Example

1. [Image of SAP System login screen]
2. [Image of SAP Easy Access]
3. [Image of Create Sales Order: Initial Screen]
4. [Image of Create Standard Order: Overview]
5. [Image of Additional SAP interface elements]
Testing for Linear Resource Consumption

Because scalability is a fundamental prerequisite for sizing

- we have to test whether the solution is scalable
- and perform “linearity tests“

Scalability tests are basically a repeatable pattern for (example):

- Scalability with an increasing number of objects
  - 1 object, five line items
  - 10 objects, five line items
  - 100 objects, five line items

- Scalability with an increasing number of line items
  - 1 object, 1 line item
  - 1 object, 10 line items
  - 1 object, 100 line items

The results of these controlled tests can be analyzed to determine if there are any performance bottlenecks

- Tests like these enable more detailed questionnaires, where you do not rely on particular test scenarios but can ask for variations
Evaluating Measurement Results

Make sure you define at least three measurement points per scenario – the more, the better.

Example: Scalability with the number of line items – three tests

Application CPU time and DB time grow linearly.
Testing Linear Resource Consumption – Measurement Example

Database time is growing logarithmically.
Application CPU time is growing exponentially.

Cannot continue with the creation of Sizing Guidance
Handling Dialog and Batch Test Cases

Dialog or online usage test cases
- Repeat the test cases five times to have the buffers properly filled before you perform the measurements
- If you run the test case once only to get measurement results, this will blur the result
- For scalability testing, vary some parameters

Batch test cases
- Sometimes difficult to test
  - Sometimes they cannot be repeated
  - Usually of long duration
- For sizing measurements runtime of about 15 minutes are convenient to handle
- For scalability tests, you should have at least two runs of e.g. 5 min. and 50 min., respectively
- If the batch runs are parallelized by RFC, you must also consider the RFC resources (CPU, DB calls,…)
General Statements on the Test System

Try to be alone, the more concurrent processes the less accurate the results will be.

It is easier to analyze on a central system (app server and DB on one machine) or a system with one application server only

CPU measurements are extremely platform-dependent

- Hyperthreading (Intel) or Hardware-Multithreading (IBM) or similar technologies that simulate several logical CPUs on one physical CPU
  - For reliable measurements: turn it off
  - If you can't, you must make sure the load on the system is kept at a minimum, below 10% in peak
  - Power saving settings like “CPU Frequency Throttling” that control the current speed of the CPU should be turned off
- The less CPUs you have available, the more difficult it is, because of "interfering" users, batch jobs, etc...
  - Optimally, no other processes, systems or DBs should be active on the server
- Use the Measurement Environment Check tool (MEC) to see if system settings are ok
Ensuring an Optimal Environment

Keep the number of other users in the system at a minimum
- It would be better to be entirely on your own
- If this cannot be helped, make sure these users do not change coding, DDIC objects and customizing data or settings

There should be no influences of other systems or transports
- Transports to the test system are not allowed
  - They may change coding, DDIC objects and customizing settings
- If there are other SAP or non-SAP systems or DBs on the same server, they should not incur extra load
- Rule of thumb: There should be at least two Cores for you, three or four are even better because they may handle possible regular background processes
- Memory must not become a bottleneck
  - SAP memory (Extended Memory, Roll, Paging)
  - Physical memory (avoid OS paging by all means)
Create Test Cases

The test cases have been thoroughly documented

- eCATTs are preferable
- For manual tests: The goal is that an inexperienced reader with basic knowledge should be able to follow the instructions
  - For this case: include the exact data to be entered, for example
    - Order type, customer number
    - The flow of actions, for example buttons and menu paths
    - Ideally, you have a screenshot of all steps
- Test descriptions must include information about implicit settings, such as user-specific parameters, for example
Sizing-Relevant KPIs (Key Performance Indicators)

- **CPU**
  - Processing times of business transactions or tasks
  - Cost factor: Number and processing power of servers

- **Memory**
  - Allocated to a user or background process
  - Garbage collection, acceleration, planning capabilities, buffers, caches
  - Cost factor: Physical memory slots

- **Disk size**
  - Data that resides on the database
  - File read and write activity to storage
  - Cost factors: Backup/recovery depends on size of database

- **Front-end Network Load**
  - Transferred amount of data
  - Network time and roundtrips
  - Cost factor: Leasing bandwidth
## Which KPIs to Collect

<table>
<thead>
<tr>
<th>CPU / Processing Power</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time per scenario or per interaction step</td>
<td>Memory is usually dominated by user contexts methods different for ABAP and Java</td>
</tr>
<tr>
<td>Detailed information</td>
<td></td>
</tr>
<tr>
<td>▪ On application</td>
<td>▪ ABAP</td>
</tr>
<tr>
<td>▪ On database</td>
<td>– User context (maximum over test case)</td>
</tr>
<tr>
<td>▪ Browser rendering time, network time (only for response-time critical applications consider synchronous traffic only)</td>
<td>– Memory in background jobs usually no issue</td>
</tr>
<tr>
<td>▪ [DB independent] multiply entries</td>
<td>▪ Java</td>
</tr>
<tr>
<td></td>
<td>– User session space, processing space, framework space</td>
</tr>
<tr>
<td></td>
<td>HANA: see disk</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database Disk Space and Disk I/O</th>
<th>Frontend Network Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk space</td>
<td>Transferred amount of data</td>
</tr>
<tr>
<td>▪ Fine granular:</td>
<td>▪ Average number of KB sent and received per interaction step</td>
</tr>
<tr>
<td>– Number of inserts into tables during test</td>
<td>▪ Calculate bandwidth</td>
</tr>
<tr>
<td>– DDIC information on table and index width (DB independent) multiply entries</td>
<td>Number of (synchronous) roundtrips between client and server</td>
</tr>
<tr>
<td>– On HANA: compression of table</td>
<td>▪ Impact on response time</td>
</tr>
<tr>
<td>▪ Coarse</td>
<td></td>
</tr>
<tr>
<td>– DB snapshot before and after the test</td>
<td></td>
</tr>
<tr>
<td>Disk I/O: Monitor activity on ST04</td>
<td></td>
</tr>
</tbody>
</table>
Example: Using STAD for CPU, Memory & Frontend

This guideline is valid for applications using the ABAP application server

**Indicators**

1. CPU time is APP server CPU (ms)
   - non-HANA: DB req. time can be approximated as DB server CPU (ms)
   - HANA: new releases: HANA CPU time; older releases: see next slide
2. Dialog: Max. ext. mem. In Step (KB)
   - Batch (only Windows): Max. ext. mem. in Step (KB) + Max. Mem in roll (KB)
   - Batch (Unix): Try to run in dialog mode
3. Terminal: in-message
   - Terminal: out-message

**Procedure**

1. Add the entries for CPU
2. Take the maximum value for memory (across all dialog steps)
3. Calculate network traffic per dialog step
DB - CPU Measurements on HANA
before HANA 2 SP4 or before ABAP Kernel 7.77

CPU Time on process level in monitoring view M_SERVICE_STATISTICS:

- Read before and after the test case, take the difference
- All components are included (indexserver, XS Engine, …)
- Requires exclusive system access
- Requires certain user authorizations
- Very accurate, limited by the granularity of the underlying OS/kernel frequency
- Background noise like save points, delta merges, background jobs might also be measured

→ repeat measurements, also at other points of time and be alone on the system

Alternatively, the data of the expensive statement trace can be used:

- The cheapest statements are not included
- If you try to include also the cheapest statements, the measurement overhead might become a problem
Obtaining the Values for CPU/Memory and take Care!

Download data: Go to: Workload → Download → Spreadsheet

- 780 ms app server CPU
- 537 s DB server CPU (separator bug) ; 3025 ms DB server CPU (performance bug, that should be corrected before sizing measurement is made !! We do not want to size bugs !!)
- 3025 ms have to be reduced, and are not 3025/1000 ms
- 5082 kB (maximum) consumed during this business process
- Bug in Header: Max. ext. mem.: Bytes are wrong, kB are correct → Take care, sometimes the units are wrong!

<table>
<thead>
<tr>
<th>User</th>
<th>Response time (ms)</th>
<th>CPU time (ms)</th>
<th>DB req. time (ms)</th>
<th>Memory used (kB)</th>
<th>kBytes transferred</th>
<th>Max. ext. mem. in TA (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td>144</td>
<td>40</td>
<td>95</td>
<td>1.12</td>
<td>10.6</td>
<td>3.554</td>
</tr>
<tr>
<td>USER</td>
<td>586</td>
<td>40</td>
<td>16</td>
<td>1.12</td>
<td>4.3</td>
<td>2.538</td>
</tr>
<tr>
<td>USER</td>
<td>506</td>
<td>70</td>
<td>346</td>
<td>1.12</td>
<td>551.4</td>
<td>3.558</td>
</tr>
<tr>
<td>USER</td>
<td>162</td>
<td>10</td>
<td>77</td>
<td>1.628</td>
<td>79.7</td>
<td>4.066</td>
</tr>
<tr>
<td>USER</td>
<td>4,444</td>
<td>620</td>
<td>3,025</td>
<td>2.643</td>
<td>3,892.3</td>
<td>5.082</td>
</tr>
<tr>
<td></td>
<td>780</td>
<td>537.025</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Unit bug**

**Separator bug**

**Performance bug**
Using ST05 for Disk Sizing – The Detailed Approach

The objective is to estimate the amount of data written on the database during the process by analyzing inserted database rows.

- Activate the trace (SQL TRACE)
- Conduct the test cases
- Deactivate the trace
- Start analyzing

Note: Depending on the amount of actions, the traces can be overwritten easily, so either analyze quickly, or save the trace list to file.
Aggregate Trace According to SQL Statements
Sort the Aggregated List According to SQL Statements
Determine the Right Entries

We're looking only at the netto INSERT statements

- INSERTED minus DELETED
- For example:
  - You'll find VBMOD & VBDATA in both insert and delete, so they are not relevant.

To obtain the size of the tables & indexes mark the object name & choose:
- DDIC information and choose table view (press button) or call the standard transaction SE12
Obtaining Table and Index Length

In the DDIC table dictionary
- Choose Extras -> Table length
- Take the structure width of the dictionary length

- Choose Indexes and for every index
- Add the lengths of the index fields

Note: This number is only valid, if the table contains no fields of variable lengths e.g. no strings and no LRAWs

If it is the case, two scenarios are possible:
1. Make a reasonable assumption about the average length
2. Evaluate the length in a test system with realistic data. Use DB tools or write an ABAP report

Note: In Unicode systems, 1 character is equal to 1 - 4 Bytes
Disk Sizing Using ST05 and DDIC

1. Take the lengths of the row of a table and all indexes
2. Multiply this value with the number of inserted records
3. Find the algorithm for the number of inserted records
4. Add the figure for all tables
   - That's your disk growth per test case – independent of DB compression algorithms, etc.
   - **New since 7.30**: Hidden transaction code ‘SIZING’ in the aggregated ST05 – Trace provides automatically some Information:
     - Netto – Inserts
     - DDIC - information
Disk Sizing Using DB02 – The Coarse Approach (especially for BW, also for HANA with good testcase)
Choose Tables and Indexes

Do a REORGCHK (IBM, in other DBs it's a refresh, for HANA it’s DELTA MERGE)

Choose the infoCubes you want: /BIC/F*

Write down the size of the table and the indexes

Run the test scripts

When finished, repeat the procedure above, do a reorg and choose the infoCubes you want: /BIC/F*
To create a sizing guideline from the test results apply linear extrapolation

Data (Example: refer to slide number 28)
- 780 ms application, 537 ms DB CPU, total: 1317 ms
- 5.082 MB consumed during this business process
- Server SAPS: 1250 SAPS per CPU (example)
- Sum of generated data in the database: Assume 70 KB per process execution

CPU
- Formula: \( \text{SAPS\_required\_for\_one\_process\_per\_hour} = \frac{\text{server\_saps}}{\frac{3600}{\text{measured\_CPU\_sec}}} \)
  - Example: \( \frac{1250}{\frac{3600}{1.317}} = 0.26 \) SAPS
    - This is the actual SAPS required, assuming the CPU will be utilized to 100%
    - We assume the CPU should run at 65% average system load of a productive system, therefore: divide \( \text{SAPS\_required\_for\_one\_process\_per\_hour} \) by 0.65
      - Example: \( 0.26 \) SAPS/0.65 = 0.4 SAPS
- The final sizing will then be: Number of business processes per hour * 0.4 SAPS = required SAPS
Turn Measurements into Sizing Guideline

Memory
- General: number of active users in the system * Size of user context
- Required memory = parallel users * 5 MB + 2 GB buffer offset
  - 720 users * 5 MB ≈ 6 GB
- Batch: take number of concurrent batch jobs instead of number of active users
- Add the memory for active users and batch

Disk
- Total disk size: size per object * objects per year * retention period in years
- Example:
  - Size per object: 70 KB
  - Objects per year: 400,000
  - Retention period: 2 years
  - Total disk size = 0.07 MB * 400,000 * 2 = 56,000 MB (~56 GB)

Front-end
- Add terminal-in and terminal-out numbers and divide by the number of dialog (!) steps → front-end load per dialog step
- General formula is provided in document front-end network sizing
# Examples of Sizing Calculations Based on Measurement Results

## Example 1 – CPU time per scenario
- You measure 4s for a scenario of 10 interaction steps.
  - Deductions:
    - 900 (3600s/4s) scenarios per hour, evenly distributed at 100% server/core/thread utilization
    - 590 (900*0.65) at 65% utilization (SAP standard)
    - 590 scenarios (e.g. one per user per hour) → net time, no think time included

## Example 2 – CPU time per interaction step
- In a 10-step scenario, each step takes 0.4s on average.
  - Deductions:
    - Assume think time of 30s between steps incl. system response time
    - 120 interaction steps per user per hour (3600s/30s) → 12 scenarios at 100% utilization
    - 48 CPU s per user per hour (12*4s)
    - 75 users (3600/48) at 100% → 49 at 65% (continuous work)

## Example 3 – Throughput
- A payroll run of 100,000 payroll periods takes 20 minutes on a single-processor, quad-core, dual-thread server. Deductions:
  - On a quad-processor, quad-core, dual-thread server with the same processors, the run should take 5 minutes
  - On a quad-processor, quad-core, dual-thread server with the same processors, 400,000 payroll periods could be processed in 20 minutes

## Example 4 – Lock on central object
- A central object (e.g. material in goods issue) is locked for 200 ms
- Maximum throughput is 5 goods issues per second or 18,000 goods issues per hour.
- Poss. considerations if you need higher throughput:
  - faster processor
  - optimization of coding
  - changing the business process
  - ...
Agenda

Expert Sizing
- Guidelines
- Creation of a Sizing Guideline

Methods of Sizing Validation
Methods for Sizing Validation

Logical checks on inputs

Phased rollout
- 10% of the planned volume to go live → relate actual usage to planned volume
- Very good approach for HANA Greenfield, because compression hard to predict

Performance load tests
- Baseline tests
- Full-blown volume tests

Use the Quick Sizer for greenfield sizing only
- Do not use the Quick Sizer for retro calculation. Only in some cases it makes sense to calculate upgrade factors with retro calculations in the QS.
- Once you have finalized Customizing, your own data is the best basis for a sizing
Input Analysis in Quick Sizer: Check Meaningfulness of Inputs

In many sizing projects, the discrepancy between the result of user sizing and throughput sizing is extremely large.
Input Analysis: Check Meaningfulness of Inputs

On result level "Line results and inputs", you can make checks on result versus input values – Do 1000 users of medium activity only create 10,000 activities in 10 hours?

These logical checks can be quite helpful

- Exceptions: batch-driven processing 😊
Different Peaks

Remember, there may be different peaks at different times of the day – Database peak may be different from application peak.
Optimizing Batch Processing Times

Moving processing times may decrease overall load

– Overall result: 19,300 SAPS (project level)

<table>
<thead>
<tr>
<th>Throughput</th>
<th>CPU</th>
<th>SAPS (total)</th>
<th>Memory (MB)</th>
<th>Disk</th>
<th>Disk (total, MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XL</td>
<td></td>
<td>19,300</td>
<td>11,779</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

– More detailed analysis reveals overlapping processes (solution level)

Individual results and chart reveal optimization potential
Overview of Fastest Growing Tables
Validating Quick Sizer Results (1/3)

Situation
- Quick Sizer result was turned into physical hardware
- Phased rollout, several org-units have gone productive
- In the next couple of months, additional org structures (i.e. users) will go live

Goal
- Validate Quick Sizer projection against actual usage

Suggestion
- Reverse engineering
  - Relate actual usage to QS result and apply on QS calculations
Validating Quick Sizer Results (2/3)

Procedure

▪ Establish hardware SAPS rating of each production server (allocated, from hardware vendor)
▪ SAP Quick Sizer uses a rule-of-thumb for sizing a server to run at 65% CPU load to provide optimal performance
▪ To establish the true SAPS capacity required by application
  – At the intervals where we gather total number of logged on users and also gather the CPU load
  – Based on actual CPU load, revalidate actual SAPS workload requirement

Continue to use Quick Sizer, but …

▪ Apply "usage" factor on result to calculate the net requirement for future rollouts

Note: this procedure should not be used for batch-driven installations
Validating Quick Sizer Results (3/3)

For applications where user workload has some bearing on database size (via transaction throughput, OLTP systems only),

Procedure
- From DB02 establish weekly and monthly database growth by system
- From SM04
  - Establish number of users
  - Also take down the memory value in MB
    - Sum of all the user's external sessions + roll, page, extended memory, and heap memory
- By dividing “DB growth by system” by “active users”, establish a metric for “Database Growth by user”.
- Derive formula:
  User numbers * DB growth rating = Total DB growth requirements

Note: Data archiving is ignored here
Reflections on High Volume / Load Tests

Opportunities
Verification of scalable software and system infrastructure

- Load balancing
- Parallel processing
- Locking behavior
- System tuning and parameterization
- Bottleneck analysis for infrastructure
- Fail-over, back-up and disaster recovery strategies
- Monitoring and system administration, calibration of thresholds

Robustness

- No deadlocks
- Data consistency
- Good Test coverage
- Possible performance degradation in overload situations

Verification of sizing models
Identification of memory leaks (especially with Java applications)

Threats

Projects

- Across organizational structures
- Get the right people on the project (knowledge and expertise)
- High planning effort to simulate proper system behavior

Additional expenses for infrastructure / hardware

- Dedicated test systems, the more complex, the more servers

Single analysis including tracing and profiling still has to be done

High degree of discipline required

- Complexity of test cases and their setup
- Changes to test case / process

No linear correlation between utilization and percentage of throughput in systems with CPUs, that have more than 1 thread per core