How to Pinpoint and Fix Sources of Performance Problems in Your SAP BusinessObjects BI Reports and Dashboards

Pravin Gupta
Global Practice Lead, SAP HANA & BW
pravin.Gupta@teklink.com
TekLink International

In This Session

- Determine where time is spent during various workflows and get to the bottom of performance complaints and improve the experience for end users
- See which tools which can be used to collect and analyze data and how to extract the information needed, as well as how to interpret the information to determine where most time is spent during workflows
- Delays can have unexpected root causes; see how to identify performance bottlenecks with real-life examples, as well as how different components in the architectural workflow can influence execution times
- Pinpoint the cause of poor response times in reports and dashboards – whether it’s front-end report design, server resources/configuration, back-end data execution, or specific product code calls that require optimization

What We'll Cover

- Best practice when measuring performance – controlled and consistent testing
- Performance analysis for some example workflows
- Tools to use to interpret the data, and accounting for time spent in each processing layer
- Wrap-up

Best Practice When Measuring Performance

- Architectural workflows within SAP BusinessObjects Enterprise can be complex
- When a performance concern is raised, it can be a challenge to identify the root cause
  - Front end – inefficient report design?
  - Are you using the best application for the job at hand?
  - SAP BusinessObjects and database – insufficient capacity, incorrect configuration?
  - Other factors

Best Practice When Measuring Performance

- SAP BusinessObjects and database – insufficient capacity, incorrect configuration?
  - Inefficient distribution of servers across cluster
  - Inadequate or inefficient heap assignment
  - Suboptimal configuration of services; missing SAP Notes and fixes
- Other factors
  - Room to improve your database query?
  - Network latency?
  - End-user laptop configuration causing a bottleneck?

Best Practice When Measuring Performance (cont.)

- Front-end application (e.g., WebI) is frequently blamed since this is the only part users interact with …

- Another frequently cited concern is lack of server resources
- These concerns can be easily investigated, verified, and addressed or put to rest

Best Practice When Measuring Performance (cont.)

- Start by analyzing a workflow in isolation
  - Where possible, take measurements in an environment where you can control the load
  - If production-like resources are not available, be prepared to extrapolate
  - If using tools to generate load, be sure to set realistic active concurrency

- Take multiple measurements at different times during the day
  - Time may vary even in the same workflow – always take an average
  - Don't forget to account for system/database/client cache

Best Practice

Best Practice When Measuring Performance (cont.)

- Do some sites suffer more than others?
  - Is number of users at that site a factor? Bandwidth issues?
  - Are jobs running overnight which affect different time zones?
  - How far is that location from your data? Network round trip/latency?

Best Practice

Best Practice When Measuring Performance (cont.)

- Once you have consistent measurements, check all processing layers involved and determine time spent within each layer
  - How much time is spent in the database running the query?
  - How much time is spent bringing data back from the data source?
  - How much time is spent building each page and delivering it to the user?

- This will help identify which areas you may need to revisit:
  - Content design: Are you trying to fit too much into one report or database query?
  - Environment sizing or configuration: Are additional resources required?
  - Jobs and data loading: Can the schedule be adjusted?
  - Customization or call stack: Is it time to raise a Support case?

What We'll Cover

- Best practice when measuring performance – controlled and consistent testing
- Performance analysis for some example workflows
- Tools to use to interpret the data, and accounting for time spent in each processing layer
- Wrap-up

Analyzing Performance for Some Example Workflows

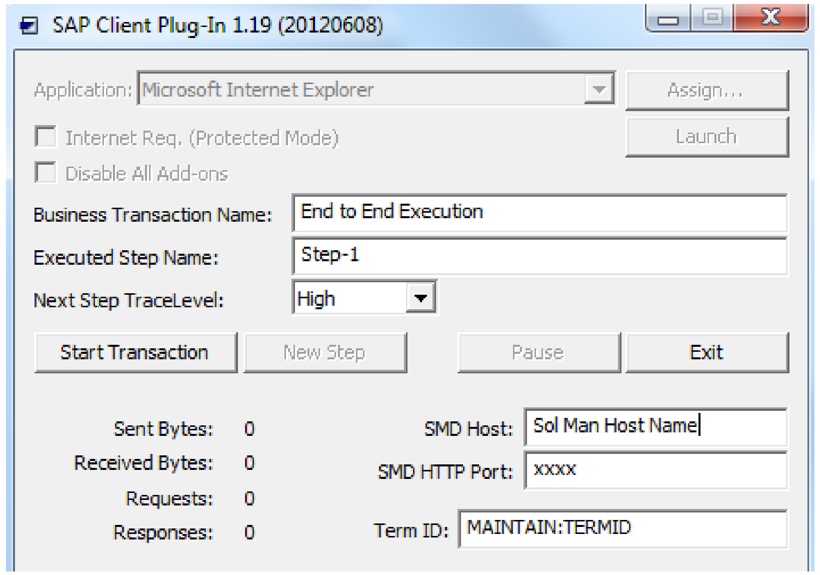
- Web Intelligence refresh via OLAP BICS to BW on SAP HANA
- Web Intelligence refresh via JDBC to SAP HANA sidecar
- SAP BusinessObjects Design Studio refresh via JDBC BICS to SAP HANA
- AD or SAP logon into BI launchpad
- OLAP refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA

Web Intelligence Refresh via OLAP BICS to BW on SAP HANA

- Web Intelligence (WebI) is an ad hoc reporting application that allows end users to define queries and design reports, or to modify existing reports depending on requirements

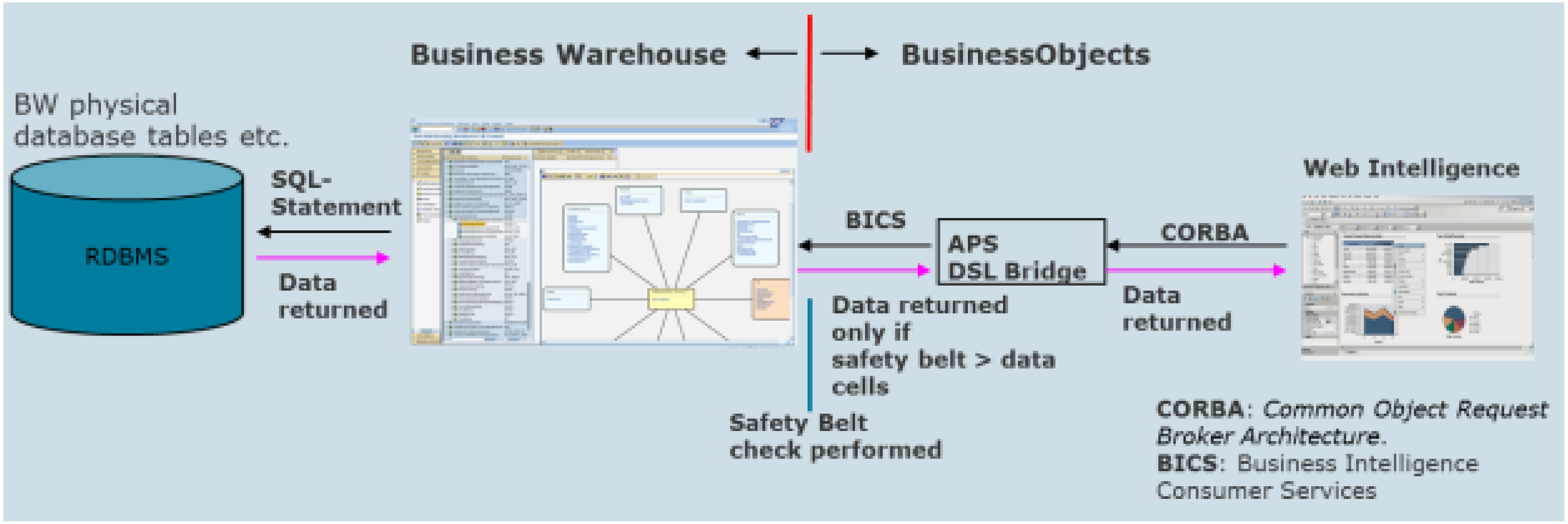
- To optimize report performance or find potential bottlenecks:
  - Capture SAP BusinessObjects traces
    - Use *SAP Client Plug-in* (see SAP Note 1861180)
    - This tags each step with a unique correlationID
  - Measure front-end execution using, e.g., HttpWatch
  - Record BW execution time using ST12
  - Review BW stats with ST13 BIIPTOOLS

Tool


SAP Client Plug-In 1.19 (20120608)

Application:	Microsoft Internet Explorer
Internet Req. (Protected Mode)	
Disable All Add-ons	
Business Transaction Name:	End to End Execution
Executed Step Name:	Step-1
Next Step TraceLevel:	High
Sent Bytes:	0
Received Bytes:	0
Requests:	0
Responses:	0
SMD Host:	Sol Man Host Name
SMD HTTP Port:	xxxx
Term ID:	MAINTAIN:TERMID

WebI Refresh via OLAP BICS to BW on SAP HANA : Flow



Source: SAP

WebI Refresh via OLAP BICS to BW on SAP HANA

- Workflow starts when user clicks button in browser
  - HttpWatch, Fiddler, etc. will accurately record start (and end) of execution

- Take several measurements without traces to establish a baseline
- With a consistent measurement, now perform same workflow but with traces enabled
  - Be aware of overhead introduced by the traces – overhead can be significant!
  - Consider overhead in your final assessment

- Gather and filter the traces based on tag generated by the SAP Client Plug-in
  - This will enable you to follow the workflow through the traces
  - Several trace analysis tools are provided by SAP
    - E.g., FlexiLogReader (refer to SAP Knowledge Base Article 2203047)

Tool

WebI Refresh via OLAP BICS to BW on SAP HANA (cont.)

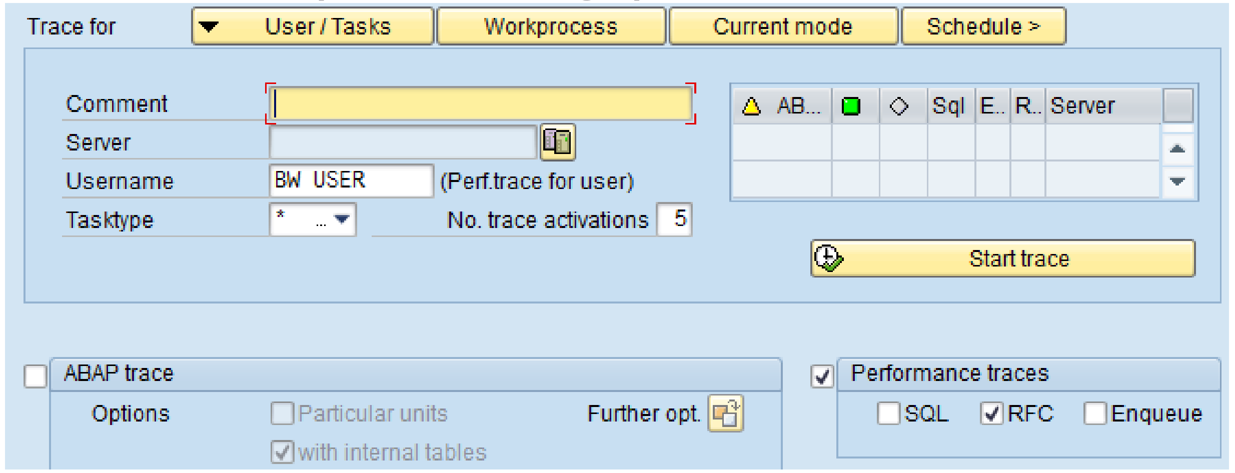
- In this example workflow, several servers will be called
  - WebI Processing Server sees incoming call from HttpServletRequest
  - We can follow the function call stack which will invoke other servers as required
    - Calls to CMS (for security and core processing queries)
    - Calls to Security Token Server (STS) to resolve single sign-on to BW
    - DSLBridge (hosted in an APS) to fetch data from BW and to build transient universe

WebI Refresh via OLAP BICS to BW on SAP HANA (cont.)

- Traces show the architecture involved – investigate time spent in each processing layer
- Add up time spent for all "HttpServletRequest" calls
  - Does this match time measured by, e.g., HttpWatch, Fiddler, etc.?
  - Small discrepancies to be expected
  - Time recorded in front end, but not traced, is activity outside of SAP BusinessObjects
    - E.g., time spent processing by client browser

WebI Refresh via OLAP BICS to BW on SAP HANA (cont.)

- Use the following tools to capture information during the workflow:
  - SAP Client Plug-in (set trace level to high)
    - Can also activate traces via CMC, though transactions won’t be tagged
  - HttpWatch or Fiddler (or equivalent) can measure front-end execution time
    - Will also show the HTTP requests involved
  - Use ST12 transaction in BW for user ID executing the query
    - Only capture RFC calls (at this stage)



Comment
Server
Username: BW USER (Perf.trace for user)
Tasktype: *
No. trace activations: 5

[ ] ABAP trace
Options: [ ] Particular units; [✓] with internal tables

[✓] Performance traces
[ ] SQL [✓] RFC [ ] Enqueue

WebI Refresh via OLAP BICS to BW on SAP HANA (cont.)

- After workflow has completed:
  - Save the HttpWatch output as a .hwl file
  - Stop recording with SAP Client Plug-in and collect relevant trace files
    - BI launchpad
    - WIPS
    - DSLBridge
    - CMS
    - STS
  - Stop ST12 and collect output, pick up statistics in ST13/BIIPTOOLS
  - Both ST12 and BIIPTOOLS output can be saved to Excel for offline analysis

Tip

Webl Refresh via OLAP BICS to BW on SAP HANA (cont.)

- For comprehensive traces, use SAP Client Plug-in on “high” setting
- But this creates a problem – activity will be logged on virtually all trace files (even those not relevant to our workflow)
  - “High” will pick up unimportant communication between internal components
  - This background noise is not important to us

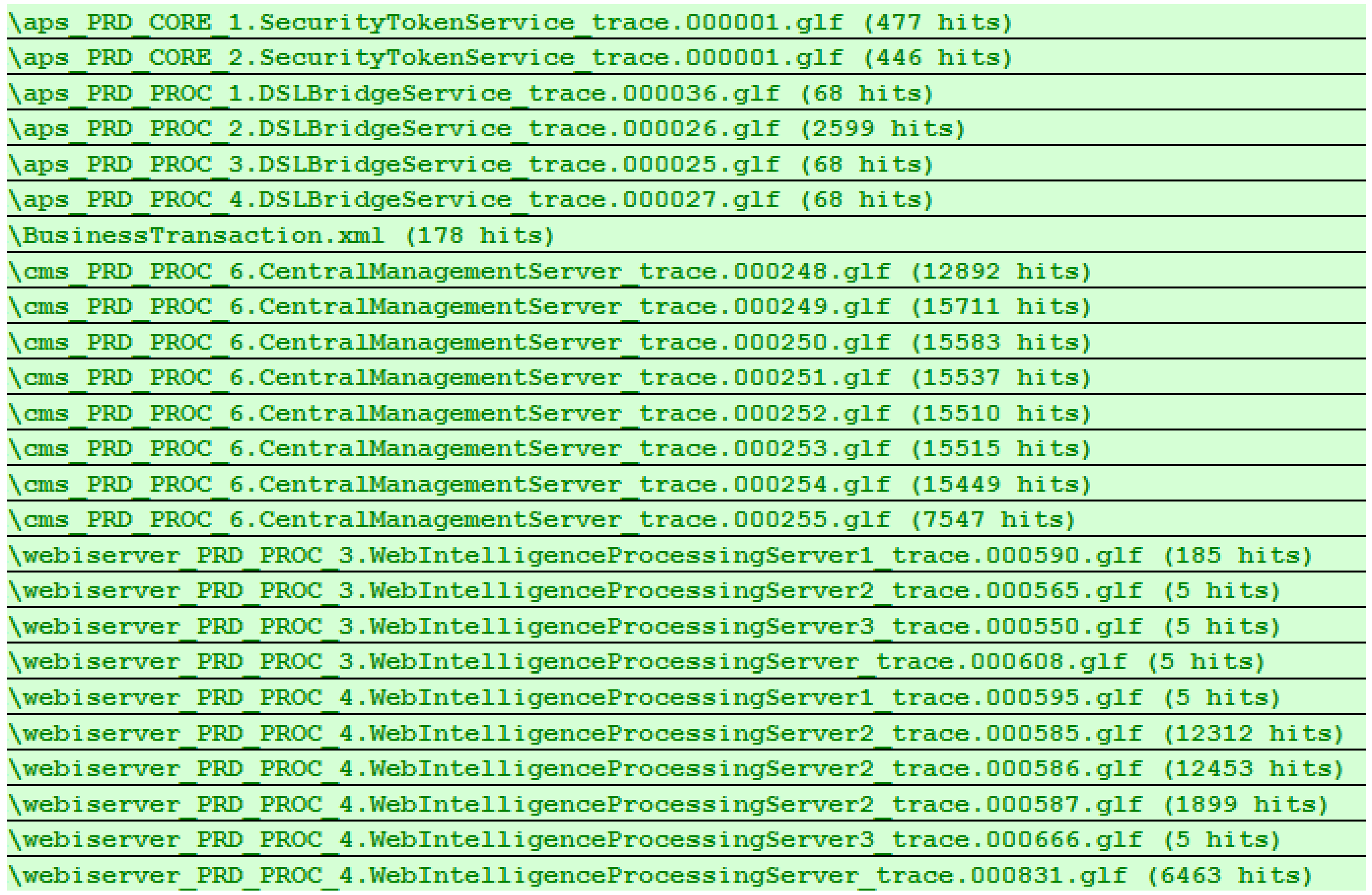
- To see which components are of interest in any workflow, first run the traces on “low”
  - Then gather ALL trace files with .glf file extension
  - Search all .glf files for correlationID tag from businesstransaction.xml
  - There is less “noise” – we only find hits in the files that are important to us

- Now run traces again on “high” setting to record your workflow
  - Gather only trace files you identified as important, and ignore the rest

Tip

WebI Refresh via OLAP BICS to BW on SAP HANA (cont.)

- Various tools can be used to search multiple trace files for the correlationID tag
  - E.g., Notepad++, UltraEdit, etc.
  - Relevant traces shown here:
    - BI launchpad
    - WIPS
    - DSLBridge
    - CMS
    - STS



```
\aps_PRD_CORE_1.SecurityTokenService_trace.000001.glf (477 hits)
\aps_PRD_CORE_2.SecurityTokenService_trace.000001.glf (446 hits)
\aps_PRD_PROC_1.DSLBridgeService_trace.000036.glf (68 hits)
\aps_PRD_PROC_2.DSLBridgeService_trace.000026.glf (2599 hits)
\aps_PRD_PROC_3.DSLBridgeService_trace.000025.glf (68 hits)
\aps_PRD_PROC_4.DSLBridgeService_trace.000027.glf (68 hits)
\BusinessTransaction.xml (178 hits)
\cms_PRD_PROC_6.CentralManagementServer_trace.000248.glf (12892 hits)
\cms_PRD_PROC_6.CentralManagementServer_trace.000249.glf (15711 hits)
\cms_PRD_PROC_6.CentralManagementServer_trace.000250.glf (15583 hits)
\cms_PRD_PROC_6.CentralManagementServer_trace.000251.glf (15537 hits)
\cms_PRD_PROC_6.CentralManagementServer_trace.000252.glf (15510 hits)
\cms_PRD_PROC_6.CentralManagementServer_trace.000253.glf (15515 hits)
\cms_PRD_PROC_6.CentralManagementServer_trace.000254.glf (15449 hits)
\cms_PRD_PROC_6.CentralManagementServer_trace.000255.glf (7547 hits)
\webiserver_PRD_PROC_3.WebIntelligenceProcessingServer1_trace.000590.glf (185 hits)
\webiserver_PRD_PROC_3.WebIntelligenceProcessingServer2_trace.000565.glf (5 hits)
\webiserver_PRD_PROC_3.WebIntelligenceProcessingServer3_trace.000550.glf (5 hits)
\webiserver_PRD_PROC_3.WebIntelligenceProcessingServer_trace.000608.glf (5 hits)
\webiserver_PRD_PROC_4.WebIntelligenceProcessingServer1_trace.000595.glf (5 hits)
\webiserver_PRD_PROC_4.WebIntelligenceProcessingServer2_trace.000585.glf (12312 hits)
\webiserver_PRD_PROC_4.WebIntelligenceProcessingServer2_trace.000586.glf (12453 hits)
\webiserver_PRD_PROC_4.WebIntelligenceProcessingServer2_trace.000587.glf (1899 hits)
\webiserver_PRD_PROC_4.WebIntelligenceProcessingServer3_trace.000666.glf (5 hits)
\webiserver_PRD_PROC_4.WebIntelligenceProcessingServer_trace.000831.glf (6463 hits)
```

Analysis of Trace Files

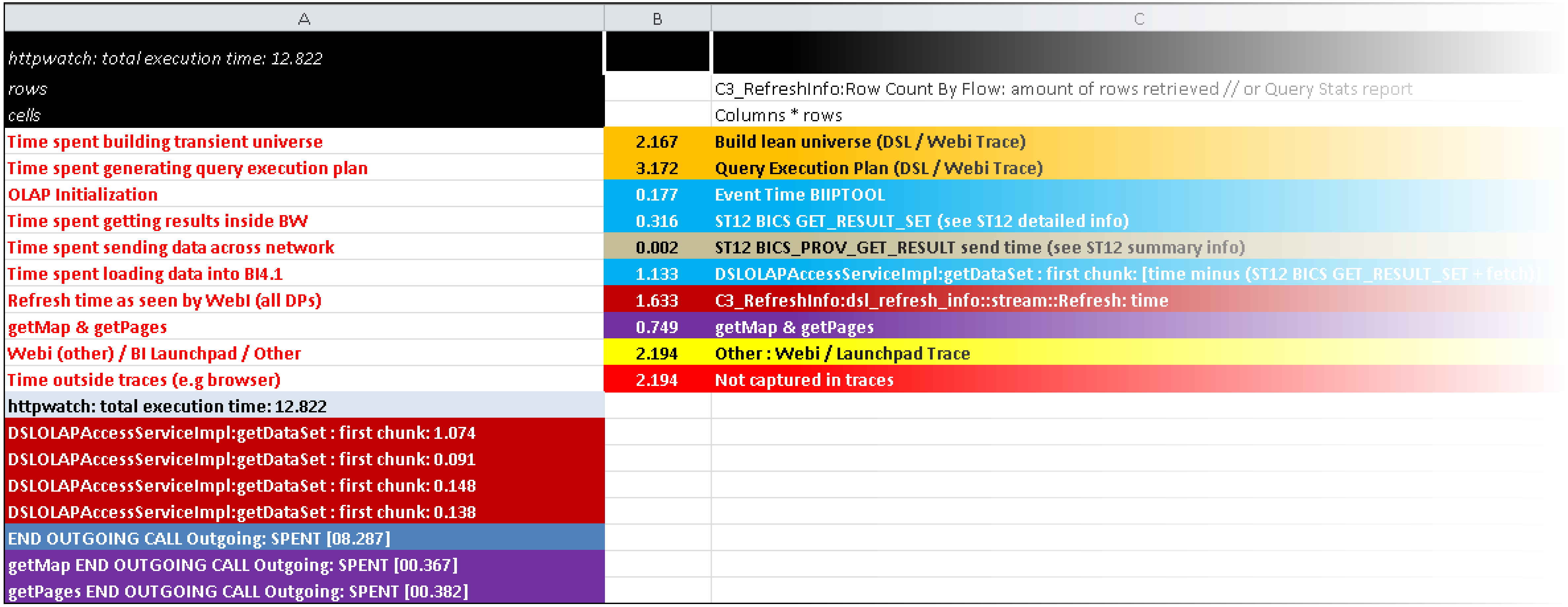
- Use SAP tools (e.g., GLF Viewer, FlexiLogReader) to review trace files
- Filter by correlationID from businesstransaction.xml
  - This removes “background noise”
  - You can now focus on activity from your workflow
- Use indents and/or color to highlight each function call
  - Can now easily find start/end of function calls
  - Makes it easy to see how long each step takes
- Add threadID column
  - Can now see and filter by individual threads
- Scan transaction times, and look for “jumps”
  - This is when activity is happening outside the traces
  - E.g., when activity is happening in data source

Analysis of Trace Files (cont.)

- When bringing the results together, you'll see time spent on operations like:
  - DPCommandsEx
  - answerPrompts
  - openDocumentMDP
  - getSessionInfosEx
  - getMap, getPages, getBlobInfo, etc.
- Follow incoming/outgoing function calls from Tomcat layer to WIPS, and from WIPS to other services
- For activity outside SAP BusinessObjects (e.g., in BW), details of time spent can be found using tools provided by that particular module/interface (e.g., ST12, ST13)
  - SAP BusinessObjects traces can only record overall time for database activity
  - Traces contain no detail/granularity on transactions inside the data source

Analysis of Trace Files (cont.)

- The illustration below brings together HttpWatch, traces, ST12, and ST13 data. Total execution was measured as 12.822s, and 12.109s was accounted for in the breakdown:



A	B	C
httpwatch: total execution time: 12.822		
rows		C3_RefreshInfo:Row Count By Flow: amount of rows retrieved // or Query Stats report
cells		Columns * rows
Time spent building transient universe	2.167	Build lean universe (DSL / Webi Trace)
Time spent generating query execution plan	3.172	Query Execution Plan (DSL / Webi Trace)
OLAP Initialization	0.177	Event Time BIIPTOOL
Time spent getting results inside BW	0.316	ST12 BICS GET_RESULT_SET (see ST12 detailed info)
Time spent sending data across network	0.002	ST12 BICS_PROV_GET_RESULT send time (see ST12 summary info)
Time spent loading data into BI4.1	1.133	DSLOLAPAccessServiceImpl:getDataSet : first chunk: [time minus (ST12 BICS GET_RESULT_SET + fetch)]
Refresh time as seen by WebI (all DPs)	1.633	C3_RefreshInfo:dsl_refresh_info::stream::Refresh: time
getMap & getPages	0.749	getMap & getPages
Webi (other) / BI Launchpad / Other	2.194	Other : Webi / Launchpad Trace
Time outside traces (e.g browser)	2.194	Not captured in traces
httpwatch: total execution time: 12.822		
DSLOLAPAccessServiceImpl:getDataSet : first chunk: 1.074		
DSLOLAPAccessServiceImpl:getDataSet : first chunk: 0.091		
DSLOLAPAccessServiceImpl:getDataSet : first chunk: 0.148		
DSLOLAPAccessServiceImpl:getDataSet : first chunk: 0.138		
END OUTGOING CALL Outgoing: SPENT [08.287]		
getMap END OUTGOING CALL Outgoing: SPENT [00.367]		
getPages END OUTGOING CALL Outgoing: SPENT [00.382]		

Summarize Log in Excel

- Fiddler
- SAP End to End client tool
- ST12 - Performance Trace
- ST13 –WIIP tool trace

Analysis of Trace Files — Breakdown of Findings

- Total execution time as seen by user was recorded using HttpWatch

  - This comprises:
    - BI4 Platform time
      - *CMS authorization (STS was used, because of SSO to BW)*
      - *DSLBridge*
      - *WebI Processing Server (WIPS)*
    - Data source time (BW on SAP HANA)
      - *Query execution time*
      - *Calculation in database*
    - Transfer time between database and BI4 platform
    - Time outside BI4 traces – e.g., processing time in client browser or laptop

Analysis of Trace Files — Breakdown of Findings (cont.)

- Here is where most time was spent at each stage:
  - BI4 Platform time – captured in BI4 trace files
    - CMS time (and STS time, since we are using SSO to BW)
      - *Security/authorization checks can be expensive*
        - Can we simplify security (e.g., reduce group membership) to improve response times?
    - DSLBridge
      - *Query execution plan (QXP) and transient/lean universe generation*
        - More objects in the BEx query means more expensive QXP
        - Challenge the business users – Do they really need/use ALL those objects?
        - Can you satisfy business requirements using several smaller, leaner queries/reports?
    - WebI processing
      - *Microcube population, rendering report for display, calculation of variables, etc.*
        - Check for expensive pagination functions – e.g., *GetPages* (for "page number x of y")
        - Can any report-level calculations be push down to the database?

Analysis of Trace Files — Breakdown of Findings (cont.)

- Here is where most time was spent at each stage: (cont.)
  - Data source time (in our case BW on SAP HANA)
  - Not captured in BI4 traces – this is found in ST12 and ST13 BIIPTOOLS output
    - Query execution time
      - *Can you add more mandatory prompts to restrict the dataset?*
    - OLAP calculations
      - *Can any more calculations be pushed down to HANA?*
    - OLAP processing: data is sorted and formatted according to query design
    - Receive data from OLAP processor and put into transfer structure in BICS interface
  - Data transfer from BW to BI Layer
    - DSL (WebI) BICS time: data transfer and preparation for WebI usage

Analysis of Trace Files — Breakdown of Findings (cont.)

- Don't forget to account for time <u>not recorded</u> in BI4 traces
  - Execution time in HttpWatch always adds up to more than recorded in traces
  - A small discrepancy is to be expected
    - Includes initial/final communication between client and BI4 platform
    - Also includes any client-side processing time
      - *E.g., work done by browser, Java, etc.*
    - Anything more than a couple of seconds warrants further investigation

Analysis of Trace Files — Points to Consider

- Is the poor performance an intermittent problem?
  - Data loading can affect data manager time
    - Taking measurements throughout the day may help you pinpoint this
    - Refer to earlier section on best practice in testing methodology

- Are clients making full use of all available caching mechanisms?
  - Check content design for functions which cause cache to be ignored!
  - Refer to http://scn.sap.com/docs/DOC-58532
    - See section “Do not (accidentally) disable the cache mechanism of Web Intelligence”

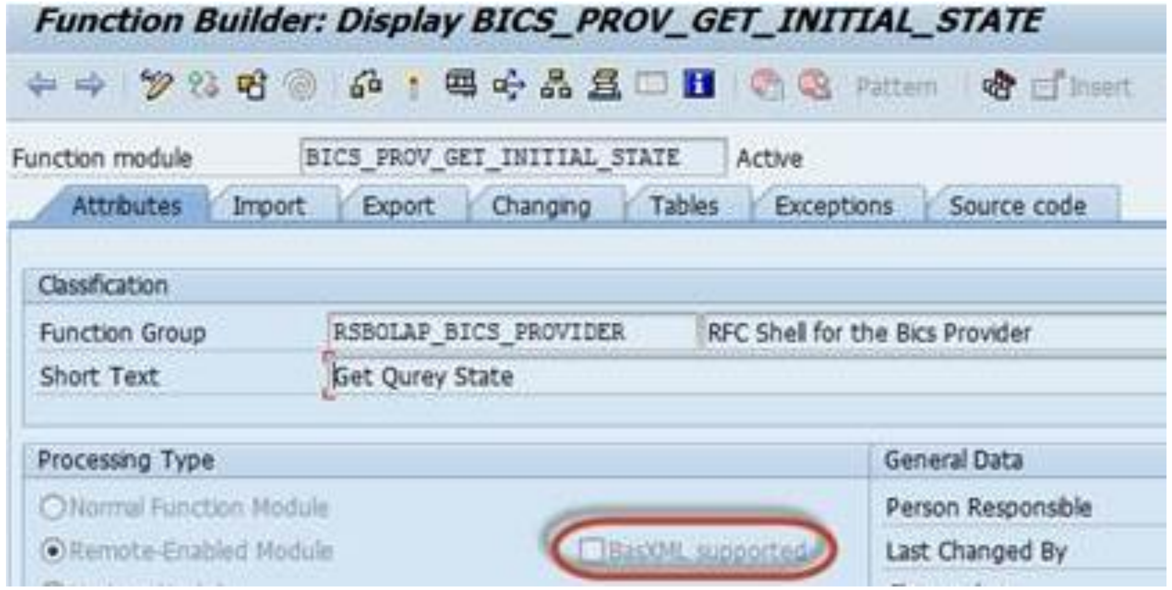
Analysis of Trace Files — Points to Consider (cont.)

- Metadata transmission can be expensive
  - This may be influenced by content design
    - E.g., transient/lean universe generation – Can you reduce number of key fields and/or characteristics?
    - Too many prompt responses can result in high send time in the RFC record

Tip

Analysis of Trace Files — Points to Consider (cont.)

- This may also be influenced by BW configuration
  - Activate compression via BASXML flag for the relevant BICS calls affected (e.g., BICS_PROV_GET_INITIAL_STATE)
    - *See SAP Note 1733726*
    - *Available since:*
      - BW 7.30 SP09
      - BW 7.31 SP06
      - BW 7.40 SP11


Function Builder: Display BICS_PROV_GET_INITIAL_STATE

Function module: BICS_PROV_GET_INITIAL_STATE Active

Classification
Function Group: RSBOLAP_BICS_PROVIDER RFC Shell for the Bics Provider
Short Text: Get Qurey State

Processing Type
○ Normal Function Module
● Remote-Enabled Module
☐ BasXML supported

General Data
Person Responsible
Last Changed By

Tip

Analyzing Performance for Some Example Workflows

- Web Intelligence refresh via OLAP BICS to BW on SAP HANA
- Web Intelligence refresh via JDBC to SAP HANA sidecar
- SAP BusinessObjects Design Studio refresh via JDBC BICS to SAP HANA
- AD or SAP logon into BI launchpad
- OLAP refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA

Web Intelligence Refresh via JDBC to SAP HANA Sidecar

- The methodology we previously described applies here too

  - However, data fetch for JDBC is now done by the WIPS itself (INPROC)
  - WIPS doesn’t outsource to a “third party” to communicate with the data source
    - In previous workflow, data fetch was handled by DSLBridge
    - Connection Server (CS) would be used for relational data sources
  - Communication with data source is handled by the CS JNI Engine inside the WIPS
  - To trace activity in the JNI call stack we must update the cs.cfg file

Web Intelligence Refresh via JDBC to SAP HANA Sidecar (cont.)

- cs.cfg file resides at following location:

  <boeinstalldir>/dataAccess/connectionServer

  - Update the following to enable tracing:

```
<JavaVM>
        <Options>
                <Option>-Dtracelog.configfile=<yourFolder>/BO_trace.ini</Option>
                <Option>-Dtracelog.logdir=<yourFolder></Option>
                <Option>-Dtracelog.name=CSJNIEngine</Option>
        </Options>
```

- If SSO is used, then JDBC will either use XML-based SAML, or Kerberos AD
- Time spent will show in the WIPS and STS output

Web Intelligence Refresh via JDBC to SAP HANA Sidecar (cont.)

- The output will be captured in the CSJNIEngine_trace file
- Output looks something like this:

```
13:46:01.479|+0000|Information| |==| | |CSJNIEngine|14160|1471|Thread-1459 |{|431|4|3|4|BIlaunchpad.WebApp|BOESERVER|webiserver_ACC_PROC_2.WebIntelligenceProcessingServer.processDPCommandsEx|localhost:14160:13844.107885:1|CSJNI.JNIbeforeIncomingCall| BOESERVER START OUTGOING CALL execute: FROM [CSJNI.JNIbeforeIncomingCall# BOESERVER]-

13:46:01.503|+0000|Information| |==| | |CSJNIEngine|14160|1471|Thread-1459     |}|431|2|3|2|BIlaunchpad.WebApp| BOESERVER |webiserver_ACC_PROC_2.WebIntelligenceProcessingServer.processDPCommandsEx| BOESERVER |CSJNI.JNIbeforeIncomingCall| BOESERVER ||CS::JAVA::fetch: 00.025-

13:46:01.505|+0000|Error| |==|E| |CSJNIEngine|14160|1471|Thread-1459     |}|431|3|3|1|BIlaunchpad.WebApp| BOESERVER |webiserver_ACC_PROC_2.WebIntelligenceProcessingServer.processDPCommandsEx| BOESERVER |CSJNI.JNIbeforeIncomingCall| BOESERVER ||END INCOMING CALL  SPENT [04.029] FROM [webiserver_ACC_PROC_2.WebIntelligenceProcessingServer.processDPCommandsEx#] TO [CSJNI.JNIbeforeIncomingCall# BOESERVER]
```

Analyzing Performance for Some Example Workflows

- Web Intelligence refresh via OLAP BICS to BW on SAP HANA
- Web Intelligence refresh via JDBC to SAP HANA sidecar
- SAP BusinessObjects Design Studio refresh via JDBC BICS to SAP HANA
- AD or SAP logon into BI launchpad
- OLAP refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA

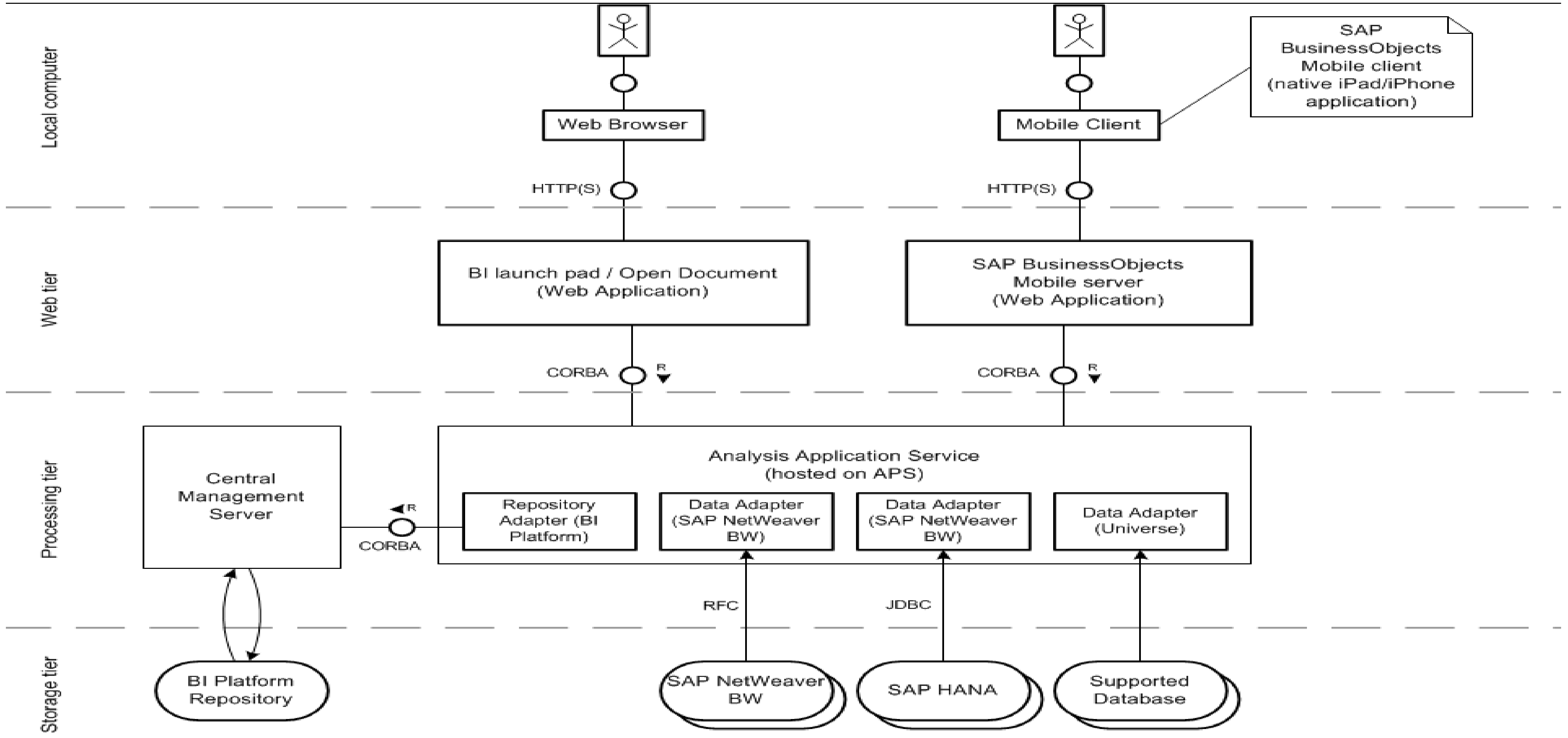
- Scenario: User executes Design Studio (DS) content via OpenDocument syntax from HANA XS portal
  - Review content design for complexity, and simplify where possible
    - Do separate components sit within one iframe?
    - Are any custom (SDK) components being used?

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA: Key Areas

- Key areas for this workflow:
  - Enterprise session generation (logon) initiated via OpenDocument call
  - JavaScript and other static content processing and cache validation on client side
  - Session generation for Analysis Application Service (AAS) on BI4 Platform
  - Processing of Design Studio content components including:
    - Validate security, retrieve report from File Repository
    - Establish JDBC connection to HANA, resolve trust to HANA if SSO being used
    - BICS script execution on HANA, data fetching

SAPinsider

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)



Local computer
Web tier
Processing tier
Storage tier
Web Browser
Mobile Client
SAP BusinessObjects Mobile client (native iPad/iPhone application)
HTTP(S)
HTTP(S)
BI launch pad / Open Document (Web Application)
SAP BusinessObjects Mobile server (Web Application)
CORBA
R
CORBA
R
Central Management Server
R
CORBA
Analysis Application Service (hosted on APS)
Repository Adapter (BI Platform)
Data Adapter (SAP NetWeaver BW)
Data Adapter (SAP NetWeaver BW)
Data Adapter (Universe)
RFC
JDBC
BI Platform Repository
SAP NetWeaver BW
SAP HANA
Supported Database

Source: SAP

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- Use HttpWatch, Fiddler, etc.
  - Measures overall execution time as experienced by user
  - Importantly, also records the static content downloaded on the client
  - Compare initial execution to subsequent executions
    - Static content should be cached and re-used after initial load

- Check for consistent behavior with different browsers
  - Is performance better/worse in IE vs. Chrome?
  - Is performance better/worse in different versions of same browser?

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- Enable of profiling parameter in the URL which opens your Design Studio content
  - Create OpenDoc link with your DS content, add “**&profiling=X**” to end of OpenDoc URL
  - This generates DS statistics on client rendering and Java server time

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

SAPinsider

- In addition, use the SAP Client Plug-in (low setting) to generate trace output – alternatively, enable Tomcat and Analysis Application Server traces
  - Take several measurements to ensure consistent readings and establish baseline/average
  - Compare response times for:
    - Initial load without client cache:
      - *New Enterprise session generation for logon to BOE platform (AD SSO into application layer)*
      - *All static content (JavaScript/CSS) must be loaded by the client*
      - *New CMS queries for security validation*
    - Initial load <u>with</u> client cache:
      - *New Enterprise session generation for BOE platform (AD SSO into application layer)*
      - *All static content should now be read from local cache*
      - *New CMS queries for security validation*
    - Subsequent reload with client cache:
      - *Enterprise session already exists and all static content should now be read from cache*

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- Find the start and end for your execution in the Tomcat web app output

  - 12:00:18.950|+0100|Information| |==| | |BIlaunchpad| 4540|1552|http-apr-8443-exec-13| |1|0|1|0|BIlaunchpad.WebApp| BOESERVER |BIlaunchpad.WebApp| BOESERVER |BIPSDK.EnterpriseSession:getService| BOESERVER |com.crystaldecisions.sdk.framework.internal.EnterpriseSession||getService(): service=InfoStore, server=-

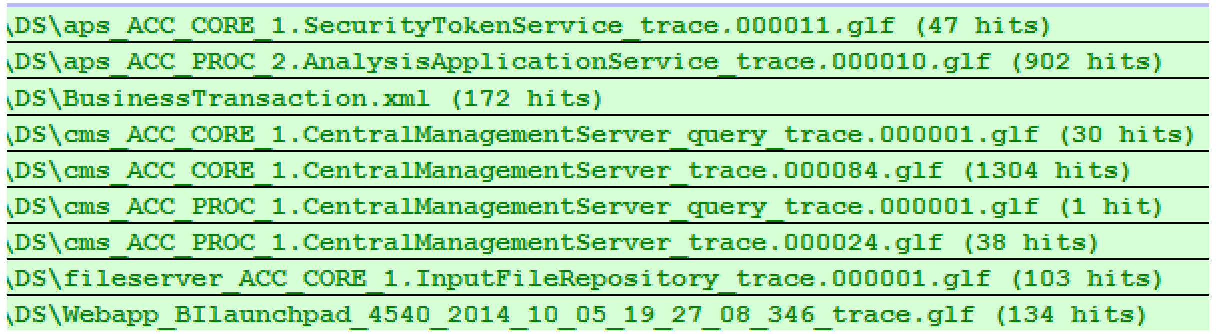
  - 12:00:49.954|+0100|Information| |==| | |BIlaunchpad| 4540| 63|http-apr-8443-exec-5|}|24|1|0|1|BIlaunchpad.WebApp| BOESERVER |-|-|BIlaunchpad.WebApp| BOESERVER |HttpServletRequest: 00.046

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- This time should match the HttpWatch total time
  - Difference between start of first request and end of final request should match time recorded in HttpWatch
  - Very easy to find this in traces if using SAP Client Plug-in, as transactions are tagged
  - Remember – a small discrepancy to be expected

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- Now bring results together to consider end-to-end workflow
  - AAS from BI4 Platform
  - Tomcat web app trace from SAP HANA XS
  - BICS profiling output (Java statistics)
  - HttpWatch capture
- If using the SAP Client Plug-in, use the correlationID (in businesstransaction.xml) to filter out background noise and see only relevant activity inside BI4 Platform traces
- When you're looking for potential bottleneck, start by zooming into the most time-consuming calls



```
DS\aps_ACC_CORE_1.SecurityTokenService_trace.000011.glf (47 hits)
DS\aps_ACC_PROC_2.AnalysisApplicationService_trace.000010.glf (902 hits)
DS\BusinessTransaction.xml (172 hits)
DS\cms_ACC_CORE_1.CentralManagementServer_query_trace.000001.glf (30 hits)
DS\cms_ACC_CORE_1.CentralManagementServer_trace.000084.glf (1304 hits)
DS\cms_ACC_PROC_1.CentralManagementServer_query_trace.000001.glf (1 hit)
DS\cms_ACC_PROC_1.CentralManagementServer_trace.000024.glf (38 hits)
DS\fileserver_ACC_CORE_1.InputFileRepository_trace.000001.glf (103 hits)
DS\Webapp_BIlaunchpad_4540_2014_10_05_19_27_08_346_trace.glf (134 hits)
```

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- "Int.do" in HttpWatch trigger BICS script execution/data fetch on SAP HANA
  - 15.12s recorded in HttpWatch

```
+ 23.094    15.120    3892    6442    POST    200    html    https:// BOESERVER
/BOE/portal/1502261134/zenwebclient/int.do
```

- This triggers main *executeBIRequest* function – which can be seen in AAS traces
  - 14.83s recorded in trace files (remember: small discrepancy always expected)

```
12:00:28.121|+0100|Debug| |<<| | | BOESERVER.AnalysisApplicationServer|13296|157637|Transport:Shared-1439/56|
|23|1|1|1|BIlaunchpad.WebApp| BOESERVER |BIlaunchpad.WebApp| BOESERVER|.executeBIRequest|
BOESERVER|||||interface com.sap.ip.bi.base.application.IApplication|BI-RA-AD|[BICS HANA]Fetch members for
characteristic EVENT_DESCRIPTION-


12:00:42.954|+0100|Debug| |<<| | | BOESERVER|13296|157637|Transport:Shared-1439/56|
|23|1|1|1|BIlaunchpad.WebApp| BOESERVER|BIlaunchpad.WebApp| BOESERVER |.executeBIRequest|
BOESERVER|0|||||interface com.sap.ip.bi.base.application.IApplication|BI-RA-AD|[BICS HANA](14833ms) Get the
crosstab data (GET_CROSSTAB_DATA)-
```

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- Most transactions occur within the executeBIRequest call. Here is the start of the main executeBIRequest:

  12:00:21.847|+0100|Information| |==| | |aps_ACC_PROC_2.AnalysisApplicationService| 3744|184296|Transport:Shared-1170/56|{|383|1|1|2|BIlaunchpad.WebApp| BOESERVER |BIlaunchpad.WebApp| BOESERVER |.START OUTGOING CALL Outgoing: FROM [.executeBIRequest# BOESERVER

- Here is a call which is triggered by executeBIRequest, and has been "outsourced" by the AAS to the Security Token Server (STS) – this call resolves SSO (SAML, in this example):

  12:00:21.972|+0100|Information| |==| | |aps_ACC_PROC_2.AnalysisApplicationService| 3744|184296|Transport:Shared-1170/56|}|411|0|1|2|BIlaunchpad.WebApp| BOESERVER |BIlaunchpad.WebApp| BOESERVER |.executeBIRequest| BOESERVER END OUTGOING CALL Outgoing: SPENT [00.255] FROM [.executeBIRequest# BOESERVER] TO [.getSAMLSSOResponseByHostAndPort# BOESERVER]-

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- There are many smaller operations (e.g., creation of controller session, fetching of BIAPP through CORBA layer, creation of components), all accounting for tiny amounts of time
  - Only look for expensive calls when looking for bottlenecks
  - Adding up expensive calls will account for vast majority of measured execution time
  - The sum of all the smaller calls makes up for any “missing” time

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- HttpWatch recorded total execution time of 30.97s
  - This comprises
    - Enterprise session generation (logon) via BI launchpad web app: 4.2s
    - Client-side JavaScript loading, cache validation, load supporting files: 6.8s
    - Session generation for AAS (including security calls/InfoStore and CMS time): ~1s
    - Processing of Design Studio report components: 18.6s
      - *Processing DS report components can be broken down as follows:*
      - *Validate security: 0.5 sec*
      - *Set Template: 1.5 sec*
      - *Retrieve report from File Repository: 0.3 sec*
      - *Establish JDBC connection to HANA: 0.25 sec*
      - *Resolve trust to HANA for SSO: 0.25 sec*
      - *Process command before rendering: 1 sec*
      - *BICS script execution on HANA and data fetching: 14.8 sec*

Design Studio Profiling & Trace Demo

- Enable DS Profiling in local client
- Enable DS trace in BOE server / Opendoc
- Measure the DS component load time

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- Compression can have a significant impact on initial load time
  - Be aware of existing product defect ADAPT01704289
    - Refer to SAP Note 1906557 – Tomcat crash due to EXCEPTION_ACCESS_VIOLATION in the zip.dll with Java_java_util_zip_ZipEntry_initFields *
  - SAP have identified root cause as a bug in JVM/Tomcat
    - To overcome this issue, use latest SAP JVM 6.1 Patch Collection 65 (build 6.1.070)
    - Confirm that this SAP JVM is compatible with your version of BI4

Caution

*** Requires login credentials to the SAP Service Marketplace**

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- Parallel processing for Design Studio
  - To prevent deadlocks, Design Studio uses a Java-based version of BICS
  - Prior to version 1.5, DS could only process sequentially
  - Parallel processing now available from version 1.5
    - Apply SP2 to resolve parallel processing bug when using SDK components

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- Be aware of:
  - Cache validation has a significant impact on initial load time
    - Cache-Control is configured to 10 years and used in combination with a “Last-modified” property
    - After AAS restart, last modified time gets updated
    - In a clustered BI4 environment (i.e., with AAS on different nodes), last modified property won’t be the same across all nodes, due to different AAS restart timestamps
    - This results in client browser cache being ignored
      - *Every time the content is opened it is treated as an initial load*
    - SAP improved Design Studio at 1.3 SP2 to improve the way it handles static content

SAP BusinessObjects Design Studio Refresh via JDBC BICS to SAP HANA (cont.)

- Be aware of: (cont.)
  - If using custom SDK components, initial response time may seem excessive
    - Gathering resources from SDK components takes longer than normal
    - This is because SDK components are stored in the BI4 as repository objects (BLOBS)
    - Resultant processing in the web tier represents an additional overhead
      - *Resources only need to be fetched once – and then can be cached*
      - *They are subsequently read from cache, improving response times for users*

Analyzing Performance for Some Example Workflows

- Web Intelligence refresh via OLAP BICS to BW on SAP HANA
- Web Intelligence refresh via JDBC to SAP HANA sidecar
- SAP BusinessObjects Design Studio refresh via JDBC BICS to SAP HANA
- AD or SAP logon into BI launchpad
- OLAP refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA

AD or SAP Logon into BI Launchpad

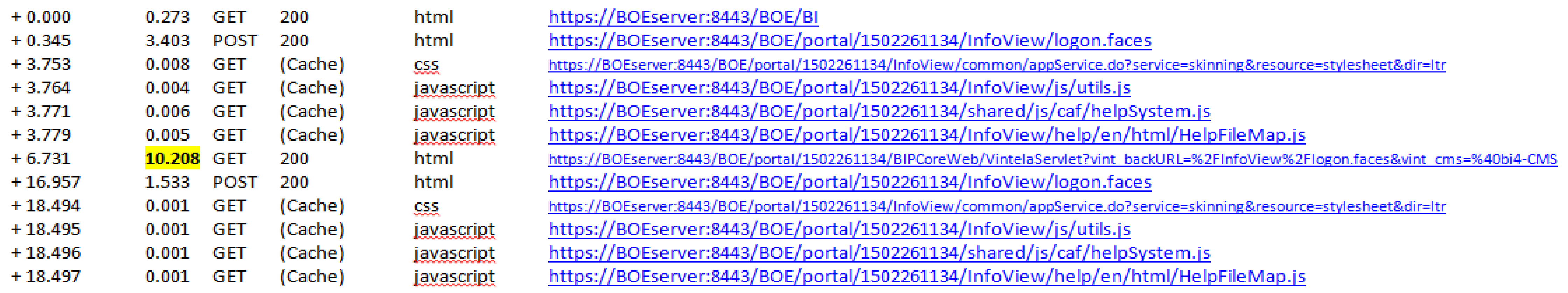
- Each authentication type has its own plug-in
  - There are client-side SDK plug-ins and server-side (CMS) plug-ins
  - Client tools may also use web services, etc.
- User logon is initiated by client-side plug-in
  - Handshake occurs between the client-side and the CMS plug-ins
  - On successful handshake, the CMS permits logon
- Plug-ins may need to perform different functions, depending on authentication type
  - Regardless of authentication type, CMS security sub-system performs the following:
    - Ensures user count does not exceed the maximum allowed for current license key
    - Creates a session InfoObject and increments the license count
    - Creates the logon token InfoObject (a unique string instead of username/password)
    - Optionally creates the user InfoObject if it does not already exist
      - *This depends on settings in Authentication tab in CMC*

AD or SAP Logon into BI Launchpad (cont.)

- Enterprise authentication logon requires the secEnterprise client-side plug-in
  - Plug-in encrypts the user name and password
  - Plug-in stores this in a security buffer, and hands the security buffer to CMS
  - CMS security sub-system calls the object sub-system to verify user InfoObject exists and pwd matches
    - If these criteria are satisfied and security allows, the user is allowed to log on
- AD authentication logon requires the secWinAD client-side plug-in
    - Queries the Domain Controller (DC) to authenticate the user
    - On successful authentication, a token from DC gets passed to the server-side secWinAD plug-in on CMS
    - Server-side plug-in verifies group membership by searching through group graph
      - *Group graph includes information about User Groups and how they relate to each other*

AD or SAP Logon into BI Launchpad (cont.)

- Use case scenario: users report very slow into BI Platform
  - With use of a viewer like HttpWatch, monitor HTTP(S) traffic on the client



```
+ 0.000    0.273   GET   200      html        https://BOEserver:8443/BOE/BI
+ 0.345    3.403   POST  200      html        https://BOEserver:8443/BOE/portal/1502261134/InfoView/logon.faces
+ 3.753    0.008   GET   (Cache)  css         https://BOEserver:8443/BOE/portal/1502261134/InfoView/common/appService.do?service=skinning&resource=stylesheet&dir=ltr
+ 3.764    0.004   GET   (Cache)  javascript  https://BOEserver:8443/BOE/portal/1502261134/InfoView/js/utils.js
+ 3.771    0.006   GET   (Cache)  javascript  https://BOEserver:8443/BOE/portal/1502261134/shared/js/caf/helpSystem.js
+ 3.779    0.005   GET   (Cache)  javascript  https://BOEserver:8443/BOE/portal/1502261134/InfoView/help/en/html/HelpFileMap.js
+ 6.731    10.208  GET   200      html        https://BOEserver:8443/BOE/portal/1502261134/BIPCoreWeb/VintelaServlet?vint_backURL=%2FInfoView%2Flogon.faces&vint_cms=%40bi4-CMS
+ 16.957   1.533   POST  200      html        https://BOEserver:8443/BOE/portal/1502261134/InfoView/logon.faces
+ 18.494   0.001   GET   (Cache)  css         https://BOEserver:8443/BOE/portal/1502261134/InfoView/common/appService.do?service=skinning&resource=stylesheet&dir=ltr
+ 18.495   0.001   GET   (Cache)  javascript  https://BOEserver:8443/BOE/portal/1502261134/InfoView/js/utils.js
+ 18.496   0.001   GET   (Cache)  javascript  https://BOEserver:8443/BOE/portal/1502261134/shared/js/caf/helpSystem.js
+ 18.497   0.001   GET   (Cache)  javascript  https://BOEserver:8443/BOE/portal/1502261134/InfoView/help/en/html/HelpFileMap.js
```

  - For each call you can see execution time in seconds, which protocol used, bytes received, and if static content is read from the cache
  - In the example above, a particular call stands out: Vintela request
    - https://BOEserver:8443/BOE/portal/1502261134/BIPCoreWeb/VintelaServlet?vint_backURL=%2FInfoView%2Flogon.faces&vint_cms=%40bi4-CMS

Analyzing Performance for Some Example Workflows

- Web Intelligence refresh via OLAP BICS to BW on SAP HANA
- Web Intelligence refresh via JDBC to SAP HANA sidecar
- SAP BusinessObjects Design Studio refresh via JDBC BICS to SAP HANA
- AD or SAP logon into BI launchpad
- OLAP refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA

OLAP Refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA

- In this section, we investigate performance of Analysis, edition for OLAP (aOLAP)

- aOLAP is designed for query and analysis rather than reporting
  - Works against multi-dimensional data sources
  - Limited formatting capability – normally used for generating tabular “workspaces”
- The multi-dimensional analysis server (MDAS) handles aOLAP workspaces
  - MDAS is hosted by an Adaptive Processing Server (APS)

OLAP Refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA (contd.)

-
  For a typical aOLAP refresh via BICS to BW on SAP HANA, we see interaction between various layers and servers using same methodology as previously discussed
  - For process operations outside the BI4 platform, use BW statistics, ST12, etc.
    - BW statistics and ST12 provide extra detail that can’t be recorded in the BI4 traces

OLAP Refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA (cont.)

- Total execution time (e.g., from HttpWatch) will comprise:
  - CMS time (and STS time, if using SSO to BW)
  - MDAS function calls for query execution (BICS calls)
  - Database processing:
    - HANA DB retrieval times (depending on how much data the query asks for)
    - OLAP Calculations (most calculations will be processed in OLAP)
      - *Check possibility to push down to HANA*
    - OLAP processing time
    - BICS interface (BI/BW)
    - Data transfer from BW to BI Layer
  - Display in BI launchpad and client rendering time

OLAP Refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA (cont.)

- BW back-end statistics for aOLAP workflows can be generated by enabling BICS profiling
  - This is controlled via a properties file on the BOE server running the MDAS service:

```
<BOE install dir>\java\pjs\services\MDAS\resources\com\businessobjects\multidimensional\services\mdas.properties
```

- If multidimensional.services.bics.profiling.enabled is set to true:
  - OLAP statistics in table RSDDSTAT_OLAP are generated
  - Data Manager statistics in table RSDDSTAT_DM are generated
- This means a RSBOLAP_BICS_STATISTIC_INFO function for almost every activity
  - This can have a significant impact on aOLAP performance as experienced by end users!
    - E.g., time taken to display a prompt to the user: we measured 11.5s with and 5s without profiling
    - E.g., time taken to execute a data fetch: we measured 45s with and 32s without profiling

Heads-Up

aOLAP Demo

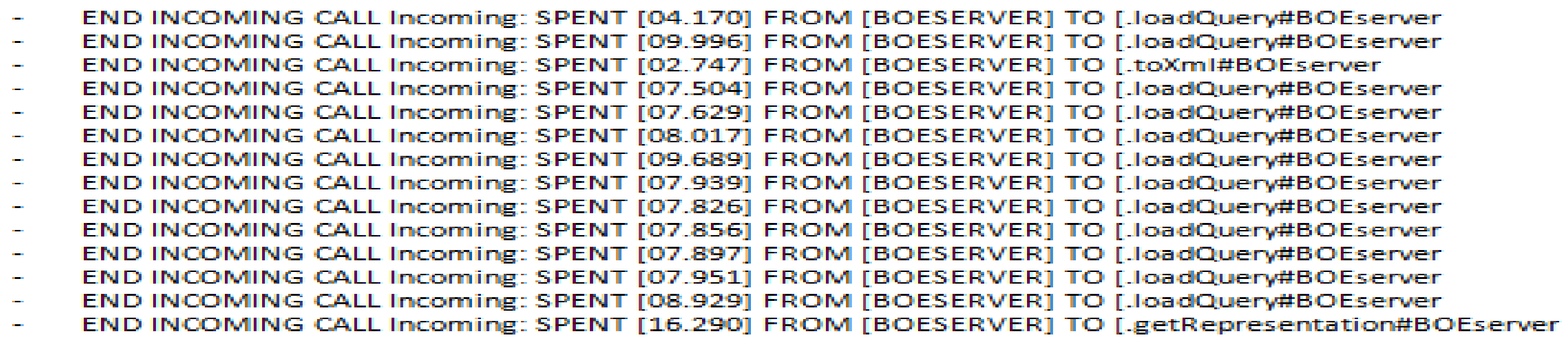
- Change MDAS properties on BOBJ server
- Analyze OLAP statistics in RSDDSTAT_OLAP
- Data Manager statistics in table RSDDSTAT_DM
- Other MDAS Properties

OLAP Refresh via OLAP BICS in SAP BusinessObjects BI 4.1 to BW on SAP HANA (cont.)

SAPinsider

- When a BEx query is connected to multiple sheets in an aOLAP workspace, check that aOLAP is only loading the active tab
- MDAS may perform the loadQuery for ALL tabs, having huge impact on response times
- Below is an example of a workspace containing 12 sheets:
  - There is only one getRepresentation call
  - But the MDAS issues a loadQuery for all 11 (inactive) sheets too!

Heads-Up



```
-     END INCOMING CALL Incoming: SPENT [04.170] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [09.996] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [02.747] FROM [BOESERVER] TO [.toXml#BOEserver
-     END INCOMING CALL Incoming: SPENT [07.504] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [07.629] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [08.017] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [09.689] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [07.939] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [07.826] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [07.856] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [07.897] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [07.951] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [08.929] FROM [BOESERVER] TO [.loadQuery#BOEserver
-     END INCOMING CALL Incoming: SPENT [16.290] FROM [BOESERVER] TO [.getRepresentation#BOEserver
```

  - SAP Development Group provided code optimization via ADAPT01730767 to address the overhead in the call stack when using multi-tabs

Performance Impact of “Lazy Loading” #1

- aOLAP can be configured to always load queries when workspace is opened, or only load queries when explicitly requested by user interaction
  - Only loading when asked is referred to by SAP as “lazy loading”
- To enable lazy loading, make a configuration change in the mdaclient.properties file on the server running Tomcat:
  - #Configure whether queries are lazy loaded when first accessed to retrieve data or preloaded when the workspace is opened.#query.lazyload=false

Performance Impact of “Lazy Loading” #2

- There is another configuration setting on the MDAS stack that SAP refers to as “lazy loading”
  - Lazy loading also means to prevent pre-load of metadata from BW with MDAS
  - Controls whether metadata hierarchies and attributes are pre-loaded all at once or are lazy loaded when their dimension is expanded by the user
  - To enable pre-load of metadata, make a configuration change in mdas.properties
    - multidimensional.services.preload.metadata=true

- Both lazy loading options should be assessed for potential performance gains

Other Performance Considerations

- If you take an aOLAP workspace from BI4.0 and open it in BI4.1, an in-situ upgrade of the internal workspace takes place to allow for new capabilities delivered with BI4.1
  - Upgrade causes several seconds delay when first opening the workspace in BI4.1
  - Happens each time the workspace is opened in BI4.1 until upgraded version is saved
    - After an upgrade, ask users to open and then save the workspace
    - The in-situ upgrade will not happen again

Other Performance Considerations (cont.)

- *Maximum Member Selector Size* and *Member Selector Cache Limit* tuning

- *Maximum Member Selector Size*
  - Maximum Member Selector Size is set to 100,000 by default and acts as a safety belt limit
    - Regardless of how many members exist, we never fetch more than this number of members
    - E.g., if set to 500, then in the first SQL statement you will find out if there are more than 500 entries
- *Member Selector Cache Limit*
  - If *Maximum Member Selector Size* is less than or equal to *Member Selector Cache Limit* then members will be cached in MDAS for faster access on future queries
  - This is only applicable in flattened hierarchies

What We'll Cover

- Best practice when measuring performance – controlled and consistent testing
- Performance analysis for some example workflows
- Tools to use to interpret the data, and accounting for time spent in each processing layer
- Wrap-up

Tools to Use to Interpret the Data, and Accounting for Time Spent in Each Processing Layer

- The following tools were used to aid the analysis exercise for the examples outlined in this presentation:
  - SAP Client Plug-in: a BI4 trace utility provided by SAP (SAP article 1861180)
  - Wireshark: freely available network analyzer
  - HttpWatch, Fiddler: freely available HTTP viewer and web debugger
  - GLF Viewer: a log file viewer provided by SAP
    - This has been superseded by FlexiLogReader (SAP KB article 2203047)
  - Notepad++: a freely available editor that supports several programming languages
  - nbtstat: diagnostic tool for NetBIOS over TCP/IP (part of the Windows OS)

Tool

What We'll Cover

- Best practice when measuring performance – controlled and consistent testing
- Performance analysis for some example workflows
- Tools to use to interpret the data, and accounting for time spent in each processing layer
- Wrap-up

Where to Find More Information

- Toby Johnston, "BI Platform E2E tracing with Solution Manager E2E Trace Analysis" (SCN, September 2013).
  - http://wiki.scn.sap.com/wiki/display/BOBJ/BI+Platform+E2E+tracing+with+Solution+Manager+E2E+Trace+Analysis
- Matthew Shaw, "Standard BI Platform log tracing" (SCN, January 2015).
  - http://wiki.scn.sap.com/wiki/display/BOBJ/Standard+BI+Platform+log+tracing
- SAP, "2103024 - *** MASTER NOTE *** How To Trace Business Objects 4.0 and 4.1 (Servers and Clients)" (SAP, December 2015).
  - http://service.sap.com/sap/support/notes/2103024 *
- Toby Johnston, "Wily Introscope for BI Platform 4.0: Why it is important and how to get started" (SCN, August 2012).
  - http://scn.sap.com/community/bi-platform/remote-supportability/blog/2012/08/06/an-administrators-match-made-in-heaven-extend-your-bi-platform-40-monitoring-capabilities-with-wily-introscope
- "Official Product Tutorials – SAP BusinessObjects Business Intelligence Platform 4.x" (SCN, July 2015).
  - http://scn.sap.com/docs/DOC-8292
    - Process flows for SAP BusinessObjects 4.x

* Requires login credentials to the SAP Service Marketplace

7 Key Points to Take Home

- Simplify and isolate your workflow, and get a consistent baseline measurement
- Examine which processing layers are involved, and understand how the various layers interact with each other
- Use the best tools to gather traces for the problem workflow and simplify your analysis
- Break down the total runtime into different processing layers to see where the hold-up is
- Don't forget that delays can occur outside the traces – e.g., in end-user laptop!
- Verify if active concurrency, data volumes, or core processing play a factor in performance degradation
- If appropriate, then embark on a tuning exercise based on your findings
  - Challenge user requirements – outline/demonstrate benefit of a smaller BEx query!
  - Review server resources, consider distribution and/or configuration changes
  - If necessary, raise a Support case to highlight inefficient/incorrect product behavior

Your Turn!

Questions?

How to contact me:
Pravin Gupta
pravin.gupta@teklink.com

Please remember to complete your session evaluation

Disclaimer