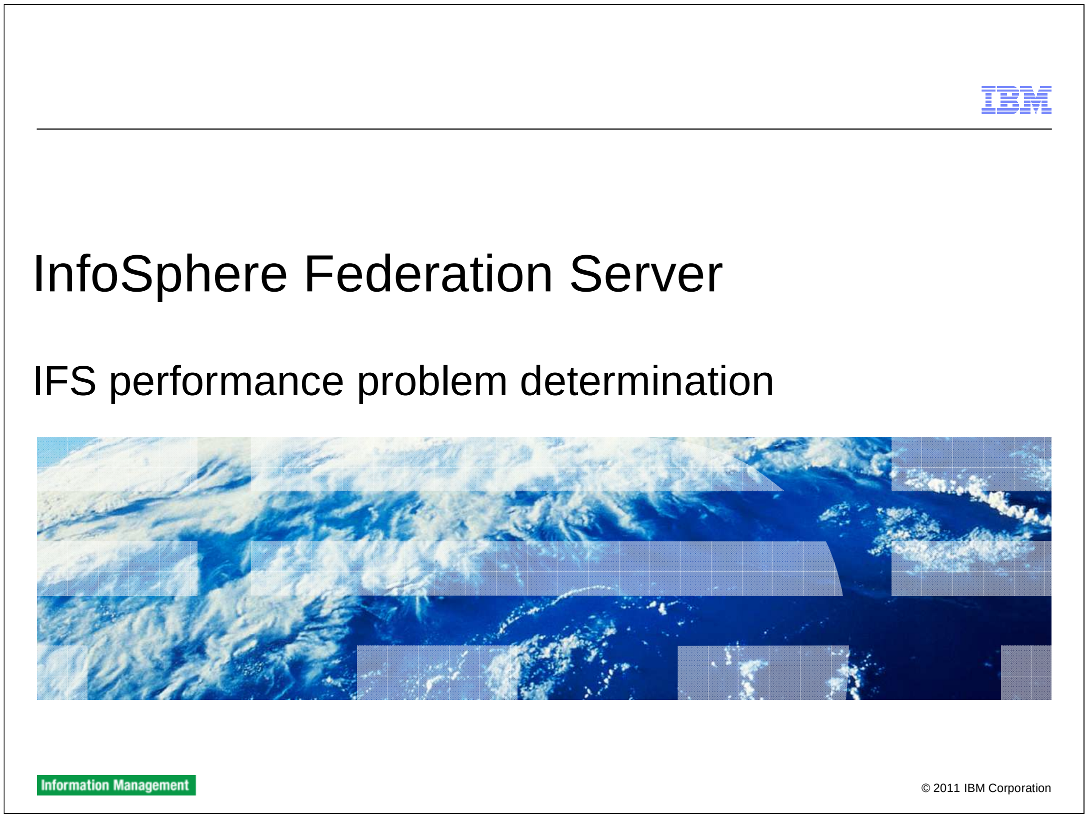


InfoSphere Federation Server

IFS performance problem determination

Information Management

This module presents the best practices for problem determination when a performance problem is encountered while running the InfoSphere™ Federation Server, referred to as IFS. The audience includes DBAs, users and Application Developers of IFS.

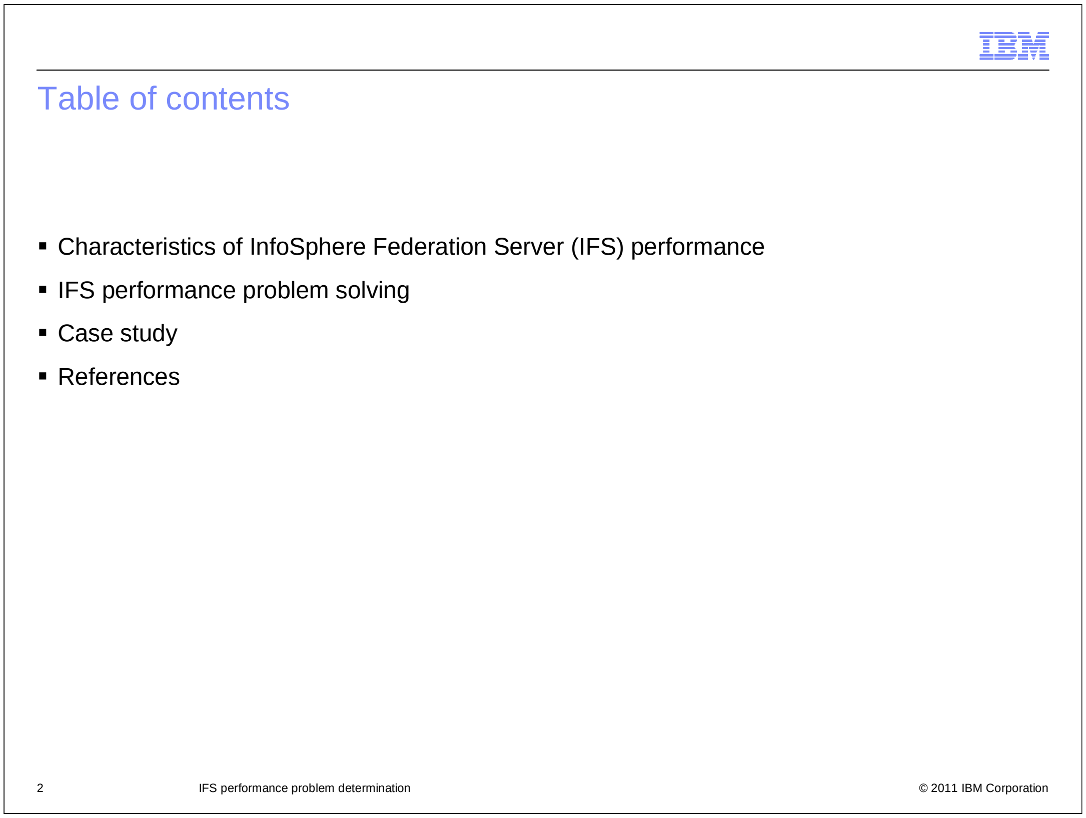
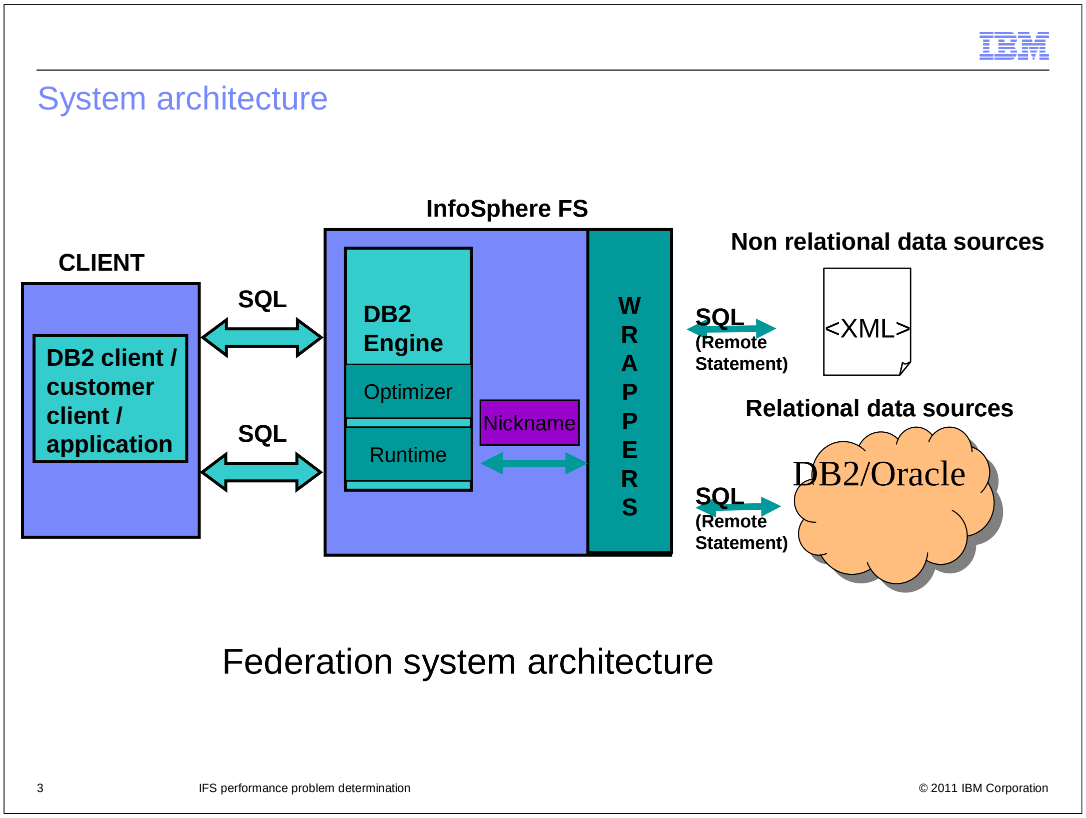


Table of contents

- Characteristics of InfoSphere Federation Server (IFS) performance
- IFS performance problem solving
- Case study
- References

2 IFS performance problem determination © 2011 IBM Corporation

This module first explains the architecture of a typical IFS deployment and the general characteristics of IFS performance problems. Next, there is a general problem determination roadmap. You will study the roadmap and steps to attack a problem in detail and in the end, you will study a case. A list of useful references is included at the end of this presentation.



System architecture

CLIENT

DB2 client / customer client / application

SQL

SQL

InfoSphere FS

DB2 Engine

Optimizer

Runtime

Nickname

WRAPPERS

Non relational data sources

SQL (Remote Statement)

<XML>

Relational data sources

DB2/Oracle

SQL (Remote Statement)

Federation system architecture

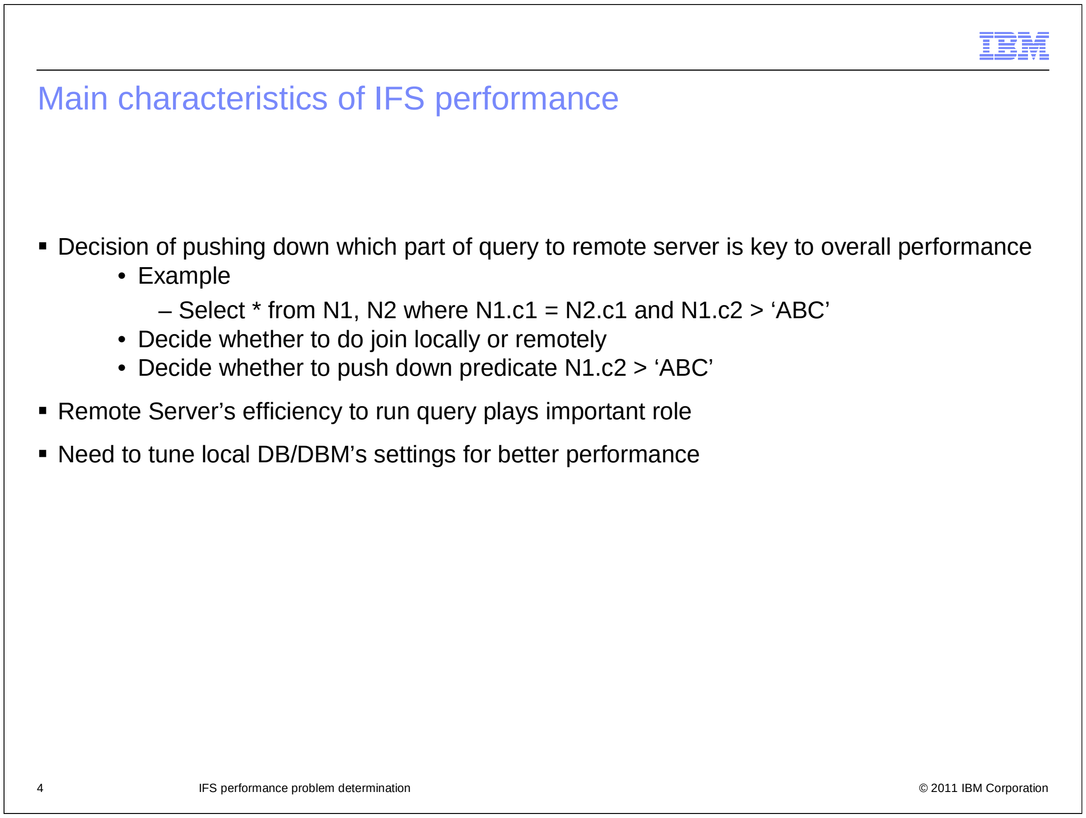
3 IFS performance problem determination © 2011 IBM Corporation

This slide displays a picture of the architecture of IFS and how the dataflow goes inside. To understand a performance problem, you need to know the whole picture for the architecture and data flow.

You have client, IFS and remote servers in a Federation deployment. First, you might see some query, namely an SQL statement issued against IFS from the client or applications. When receiving the SQL statement, IFS will parse it, generate a query execution plan, and send the plan to runtime for execution.

Runtime will access the objects being referenced by the SQL statement according to the query plan. When nicknames are referenced, it will call wrapper components to access remote server and remote tables. The remote tables are accessed by issuing SQLs against remote servers. These SQLs are called “Remote Statements”.

After the remote statements are started by the remote server, result data is fetched to the IFS. IFS will then process this data along with the data from local tables. Then, the final result data is returned to client or application.



Main characteristics of IFS performance

- Decision of pushing down which part of query to remote server is key to overall performance
    - Example
        - Select * from N1, N2 where N1.c1 = N2.c1 and N1.c2 > ‘ABC’
    - Decide whether to do join locally or remotely
    - Decide whether to push down predicate N1.c2 > ‘ABC’
- Remote Server’s efficiency to run query plays important role
- Need to tune local DB/DBM’s settings for better performance

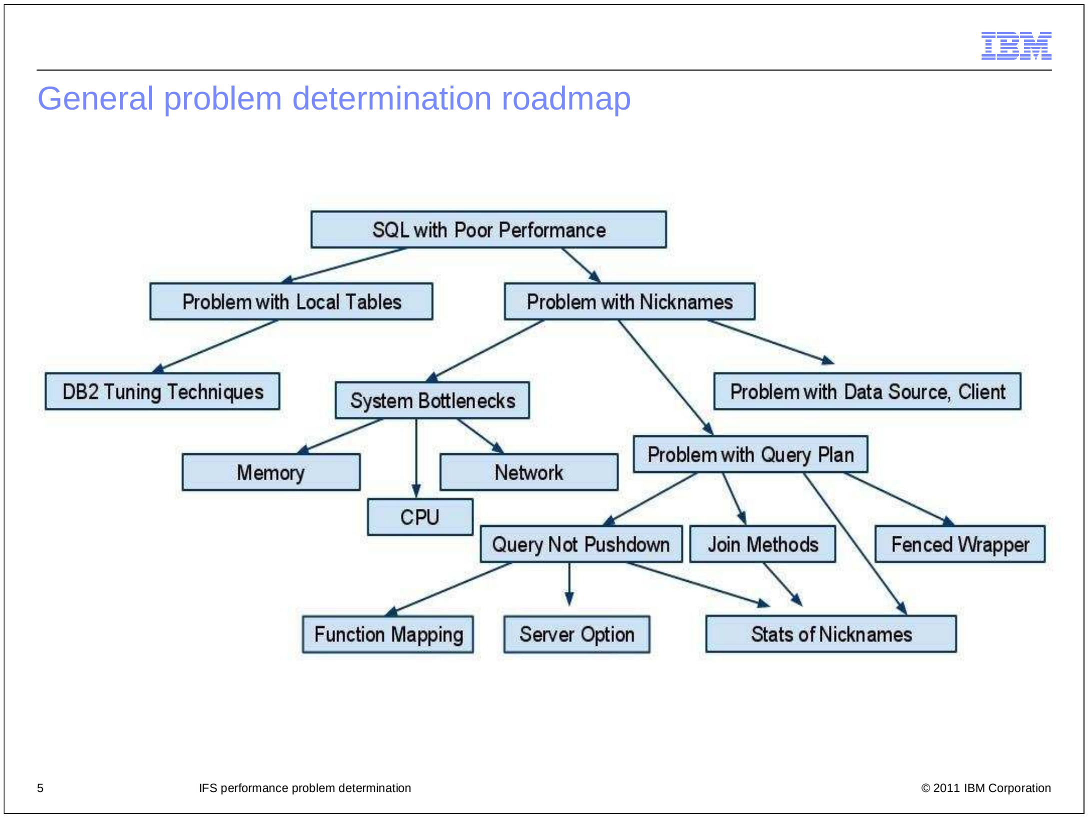
For a query, IFS may issue the remote statements and fetch data from remote server, then process the fetched data locally. So the decision of issuing what remote statements, that is, pushing down which part of the query to remote server, is key to the overall performance.

For a specific query, IFS can optionally issue different remote statement alternatives to remote server. For example, select * from N1, N2 where N1.c1 = N2.c1 and N1.c2 > ‘ABC’, where N1 and N2 are nicknames from the same server. IFS has two options to run this query. The first option is to fetch the two nickname’s data and then do the join between them locally. The second option is to push the whole query to remote server and do the join remotely. IFS will decide where to do the join here. This is often up to cost estimation by optimizer based on statistics information of the nicknames.

An additional decision is whether to push down predicate N1.c2 > ‘ABC’. It is up to semantics correctness. For example, if remote server has the same collating sequence as IFS, IFS will push down the predicate since IFS knows that pushing down the query will generate the same result as when keeping it not pushed down.

Additionally, for an IFS performance problem, remote server efficiency to run the remote statements will play an important role. Sometimes, you might find that IFS has made the right decision of pushing down the query but remote server does not perform well in running the remote queries.

You may also need to tune the local DB and DBM’s settings for better performance. When you fetched massive data from remote server, the next stage is to process it locally to generate the final result. The DB and DBM settings have a great impact on performance of the local processing so you need to tune them for the best performance.



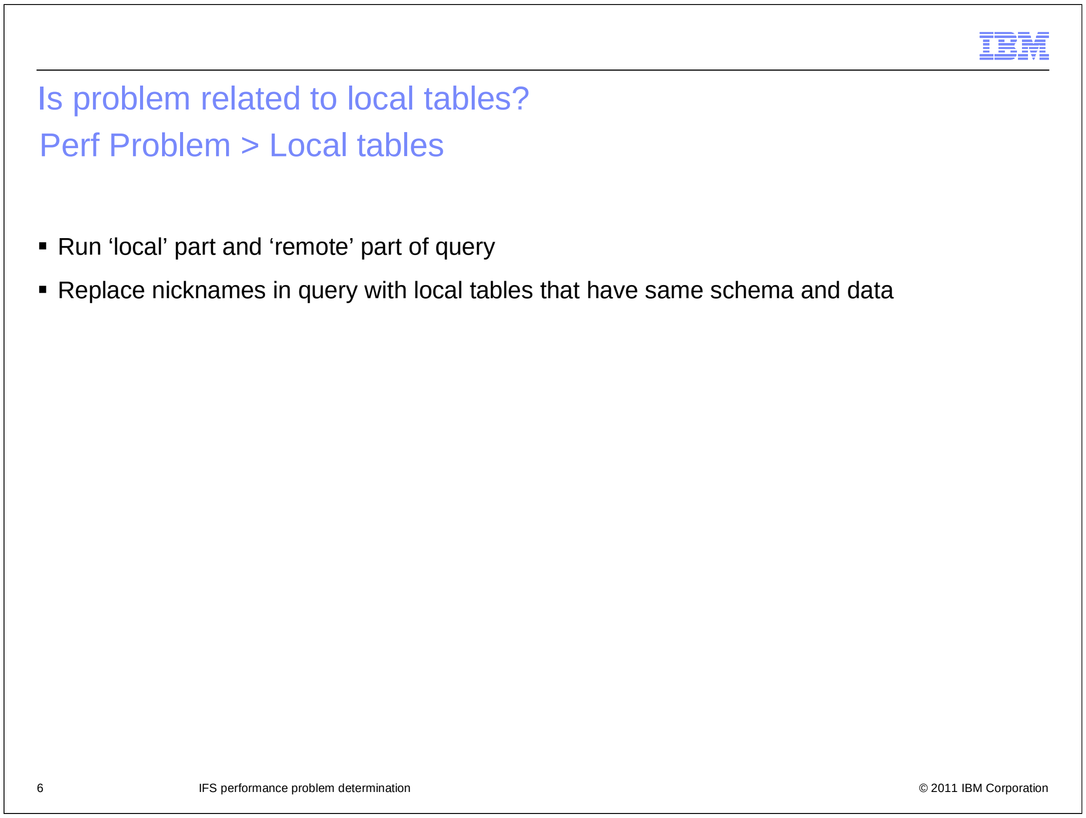
General problem determination roadmap

SQL with Poor Performance
Problem with Local Tables
Problem with Nicknames
DB2 Tuning Techniques
System Bottlenecks
Problem with Data Source, Client
Memory
CPU
Network
Problem with Query Plan
Query Not Pushdown
Join Methods
Fenced Wrapper
Function Mapping
Server Option
Stats of Nicknames

5 IFS performance problem determination

This slide displays the general roadmap for problem determination. When a performance problem is observed and the specific SQL, namely a query with poor performance has been identified, you can follow this roadmap to drill down and find out the root cause. When you are at one level or one point of the roadmap, you need to check, monitor and analyze the problem to determine which step to do next. Finally you will narrow down the root cause and reach the bottom. The following slides will go through each nodes in the roadmap one by one.

First, if the query involves both local tables and nicknames, you will have to figure out if the problem exists only for local tables.



Is problem related to local tables?

Perf Problem > Local tables

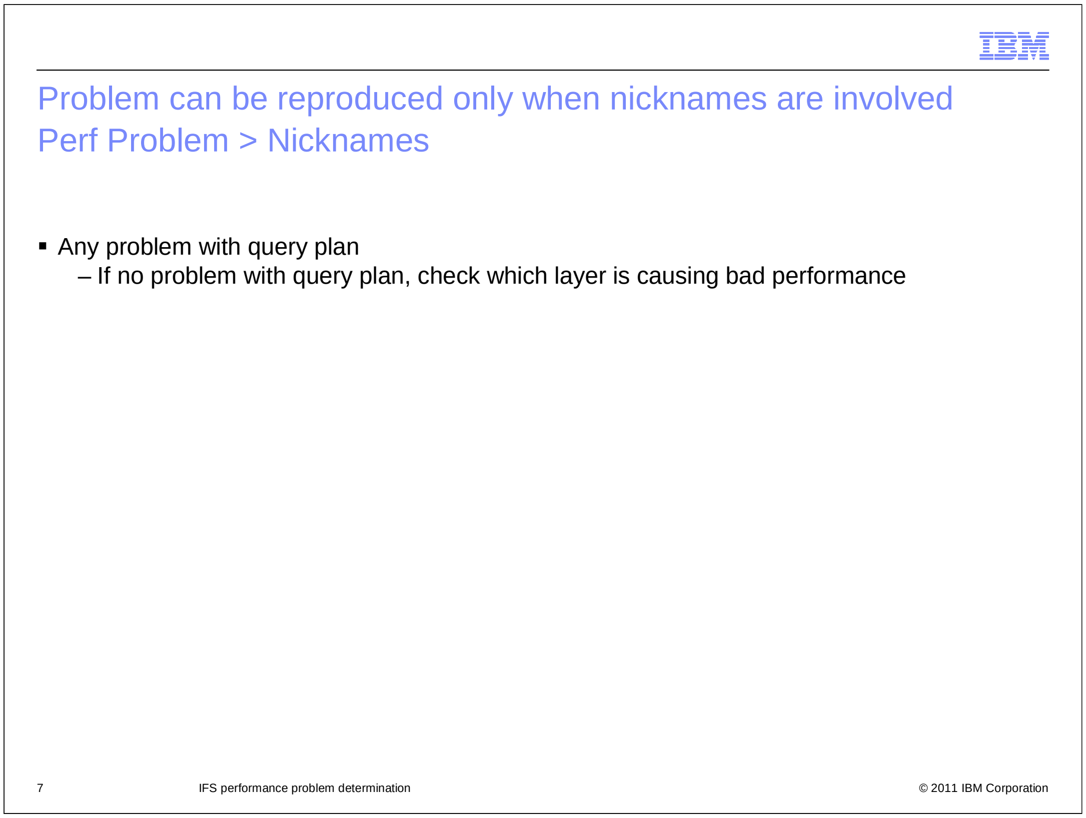
- Run ‘local’ part and ‘remote’ part of query
- Replace nicknames in query with local tables that have same schema and data

Divide the query into two parts. One part is the local part, which only references the local tables. The second part is the remote part, which references the nicknames.

Then run and time them to see which part is consuming the most elapsed time of the query.

Next, if possible, you can replace all the nicknames with local tables, populate them with the same data, re-run the query and check if the performance problem still exists. If so, you can suspect it’s the local processing of data that causes the performance problem instead of anything with remote servers.

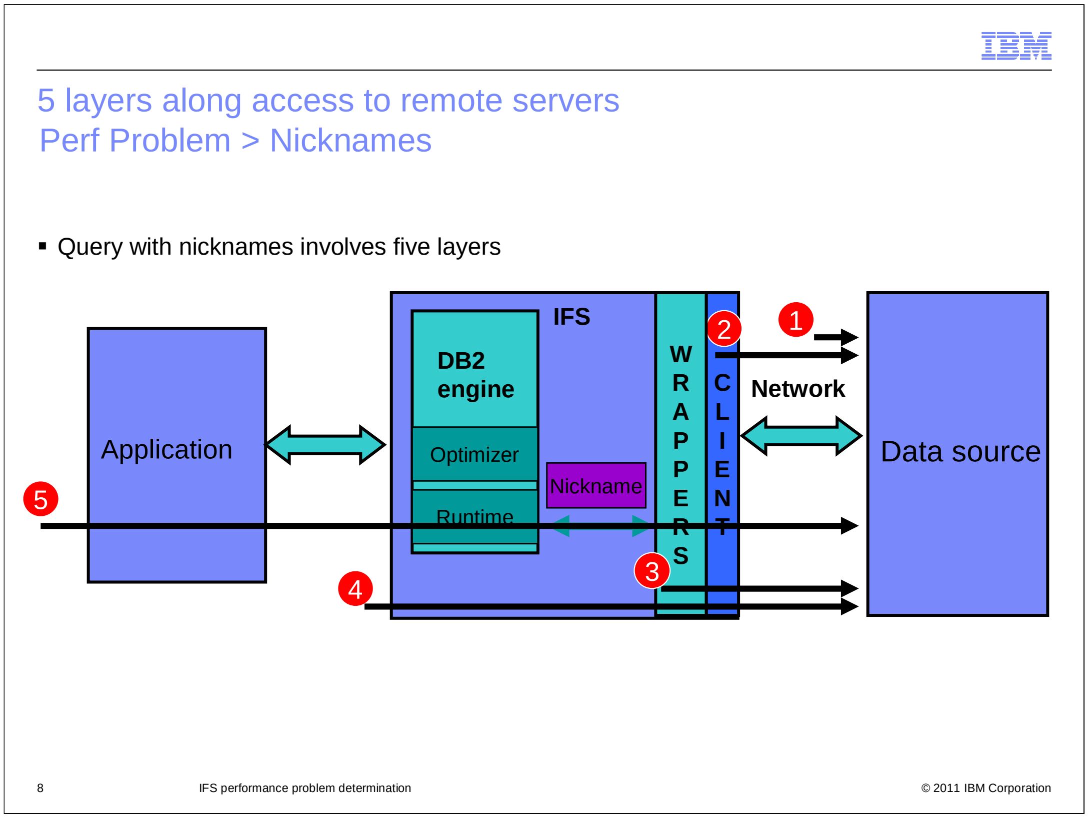
If by the previous two steps you have determined there is a problem with processing of local data, you need to tune IFS with regular DB2® performance tuning techniques. The techniques required here are the same as what you do with local tables.



Problem can be reproduced only when nicknames are involved
Perf Problem > Nicknames

- Any problem with query plan
  - If no problem with query plan, check which layer is causing bad performance

If the problem can be reproduced only when nicknames are involved, you will check first if there are any problems with the query plan. This is discussed more later in the presentation. If there is no problem with the query plan, you want to know which layer is causing the bad performance. A query involving access to a nickname will go through five layers. You have to figure out in which layer lies the root cause of the problem.

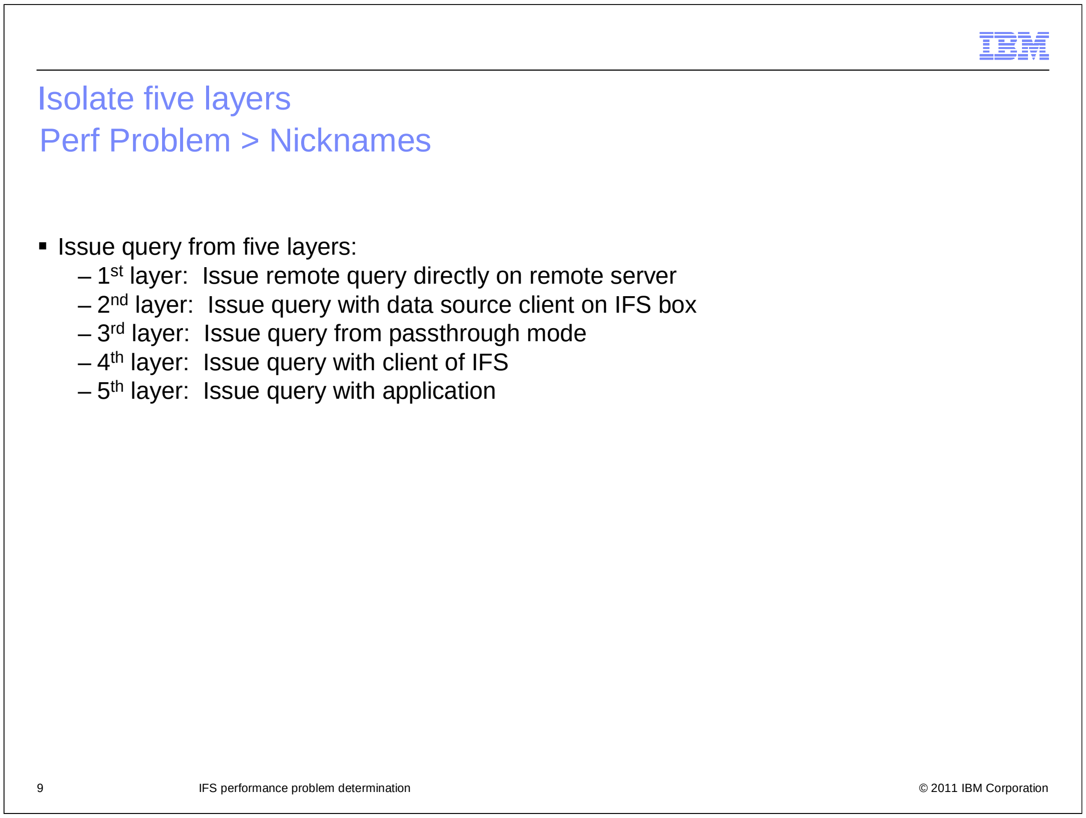


5 layers along access to remote servers
Perf Problem > Nicknames

- Query with nicknames involves five layers

8 IFS performance problem determination

For a query over nickname, you will have to fetch data from remote server back to IFS. The data going back to the IFS will go through five layers. The five layers are the remote data source server, the client code for the data source along with the network wrapper component of the IFS, the Federation engine and the application or client. Each of the layers can be the bottleneck and root cause of the slow performance. You have to isolate them by issuing the query from each of the layers.



Isolate five layers
Perf Problem > Nicknames

- Issue query from five layers:
  - 1st layer: Issue remote query directly on remote server
  - 2nd layer: Issue query with data source client on IFS box
  - 3rd layer: Issue query from passthrough mode
  - 4th layer: Issue query with client of IFS
  - 5th layer: Issue query with application

Check how much time is being consumed by each of the layers by issuing the query from each layer. First, you need to get the remote statement from the query plan. You can get the query plan with the db2exfmt tool after explaining the query. The first layer is the data source server. On this layer, run the remote statement directly on the remote server box with the data source's tool. Meanwhile, record the time the query runs.

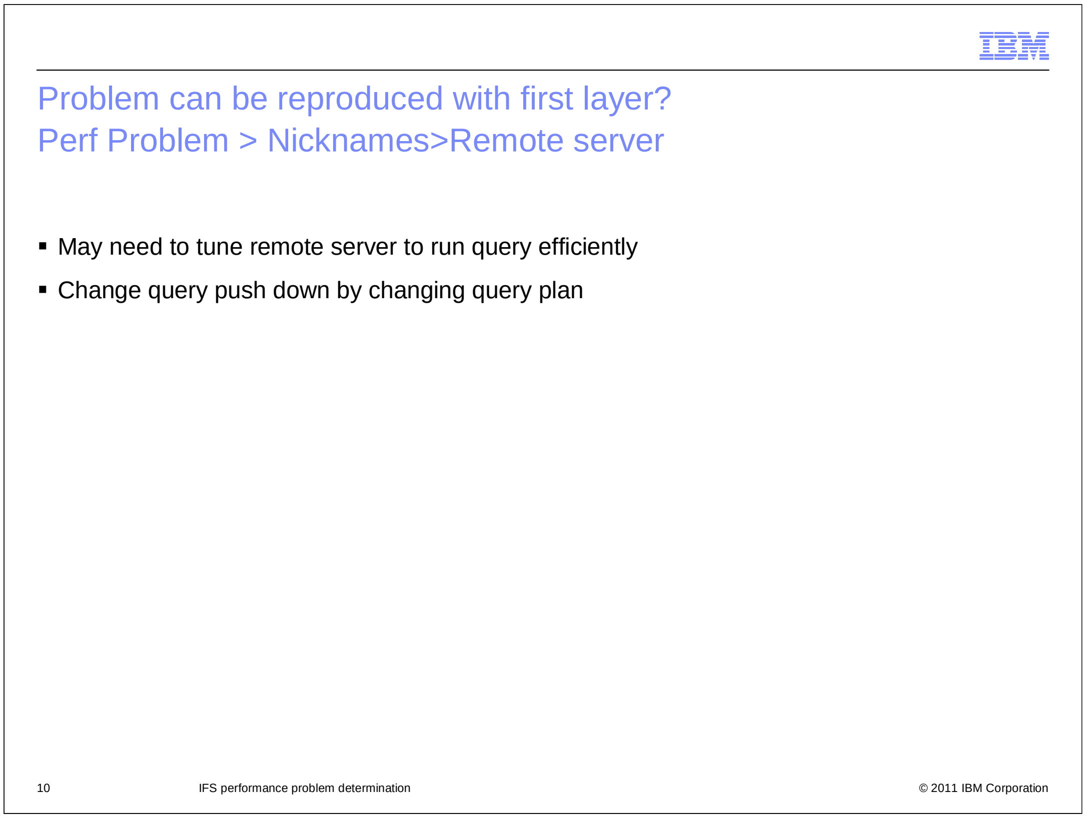
The second layer is the data source client layer. Comparing with the first layer, you will include network communication cost in this layer. On the second layer, you can issue the remote statement with the data source client tool installed in the same box as IFS.

The third layer is the wrapper layer. This layer involves wrapper code of Federation. You can issue the remote statement from the passthrough mode. This way, you only touch the wrapper code and some runtime code to run the query.

The fourth layer is the IFS layer. On this layer, you can run original SQL with IFS client. If the original SQL from application contains parameter markers, you can replace the parameter markers with literals.

The fifth layer is the application layer. On this layer, you can run the query with application, you do not have to do this since you already know the performance problem exists when running the application.

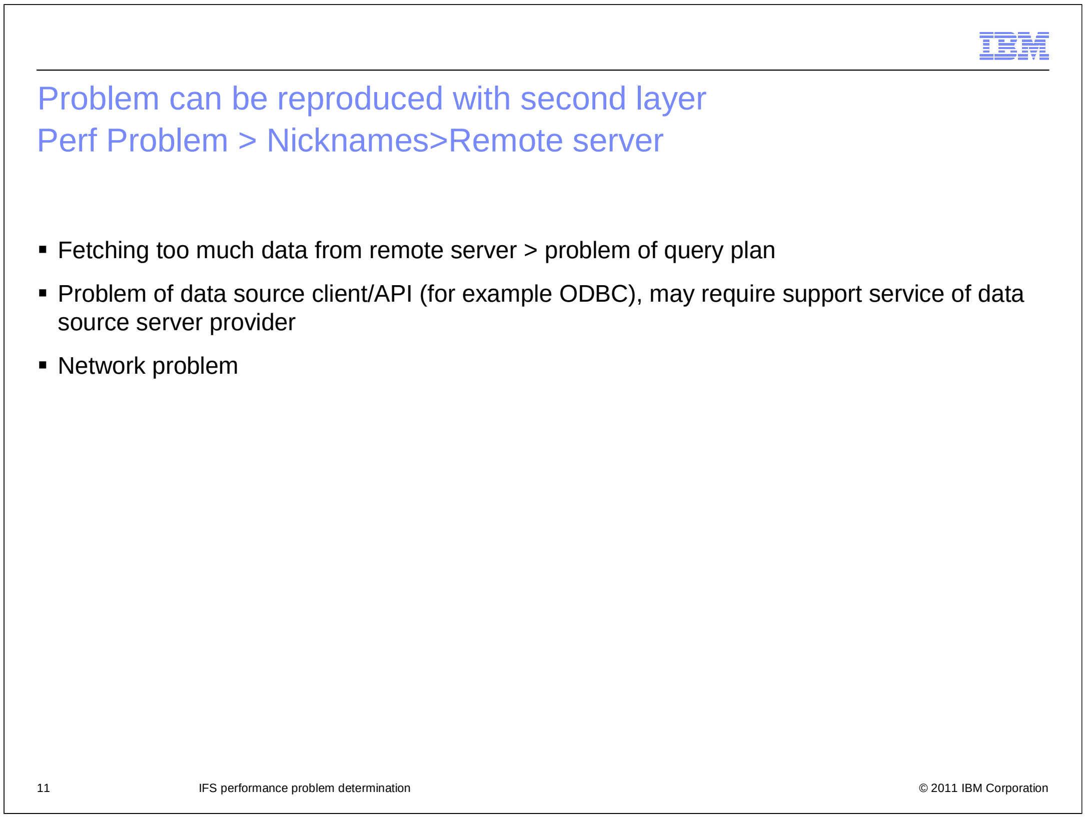
After running queries on each layer, you can compare times of each layer. If the performance problem can be reproduced on layer N but not on N-1, you know that the root cause may reside in layer N.



Problem can be reproduced with first layer?
Perf Problem > Nicknames>Remote server

- May need to tune remote server to run query efficiently
- Change query push down by changing query plan

If the problem can be reproduced with first layer, it may be that the remote server needs to be tuned to run the query efficiently. Another solution might be changing the query, namely the remote statement pushed down to remote server. Changing it to a more efficient query can improve the performance.



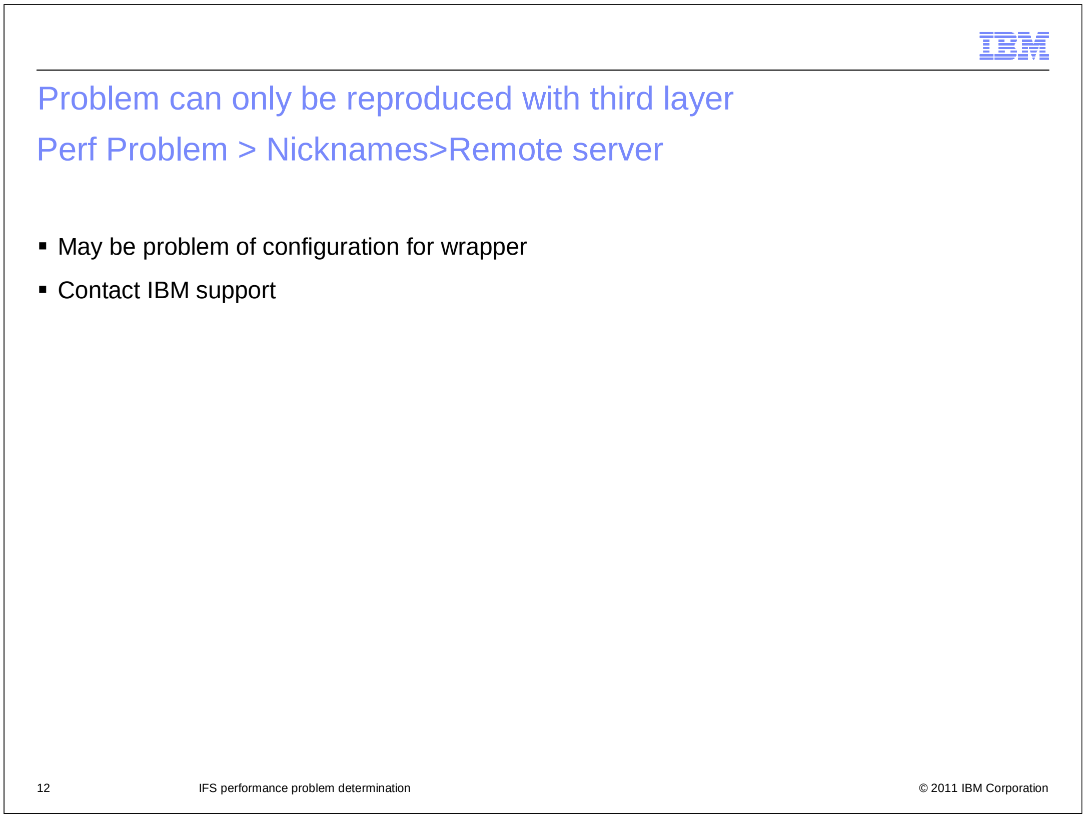
Problem can be reproduced with second layer
Perf Problem > Nicknames>Remote server

- Fetching too much data from remote server > problem of query plan
- Problem of data source client/API (for example ODBC), may require support service of data source server provider
- Network problem

If the problem can be reproduced with second layer but cannot with first layer, that is, as long as network layer is involved, the problem will occur.

If you suspect IFS might be fetching too much data from remote server, this can be caused by a problem with the query plan. The plan you are using might not be optimal. Possibly there can be more of the query that can be pushed down and data volume to be fetched from remote side could be reduced.

If you suspect this may be a problem of data source client or API, the data source client may not be well-configured. The client along with the remote server are not working well for the query. This may require support service of the data source server provider. Also, it may be that the network does not perform well and network communication rate is very low.

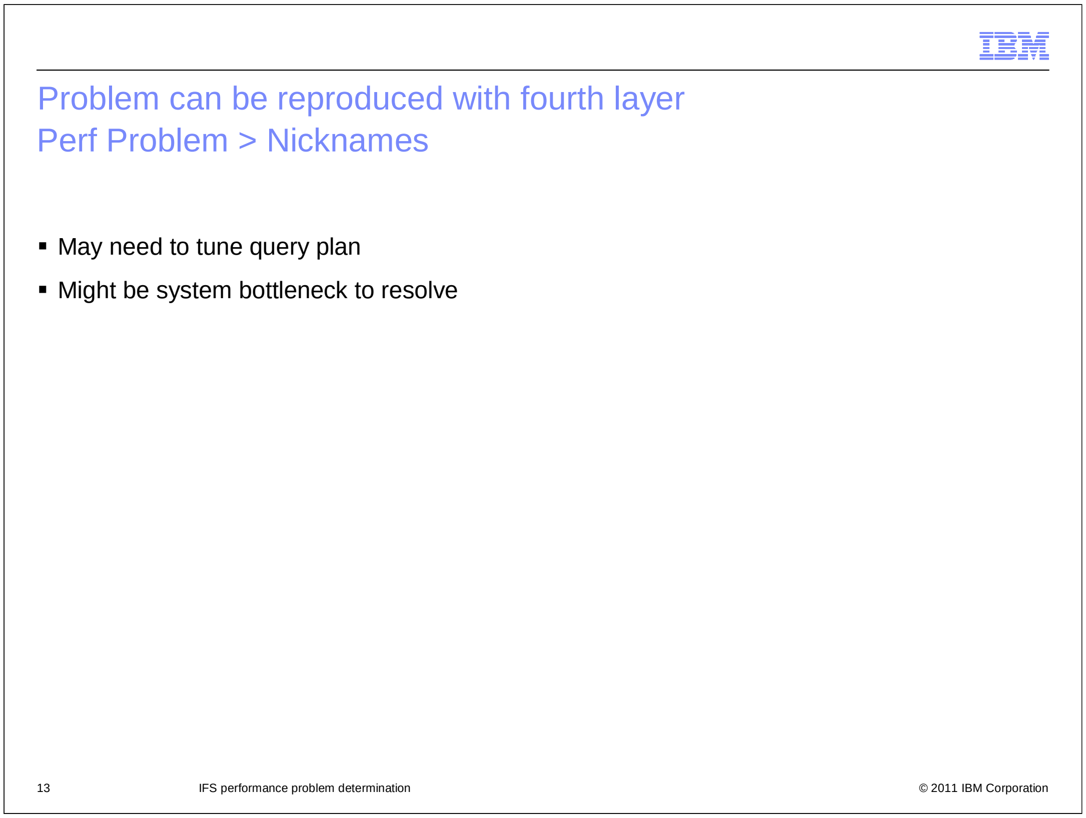


Problem can only be reproduced with third layer

Perf Problem > Nicknames>Remote server

- May be problem of configuration for wrapper
- Contact IBM support

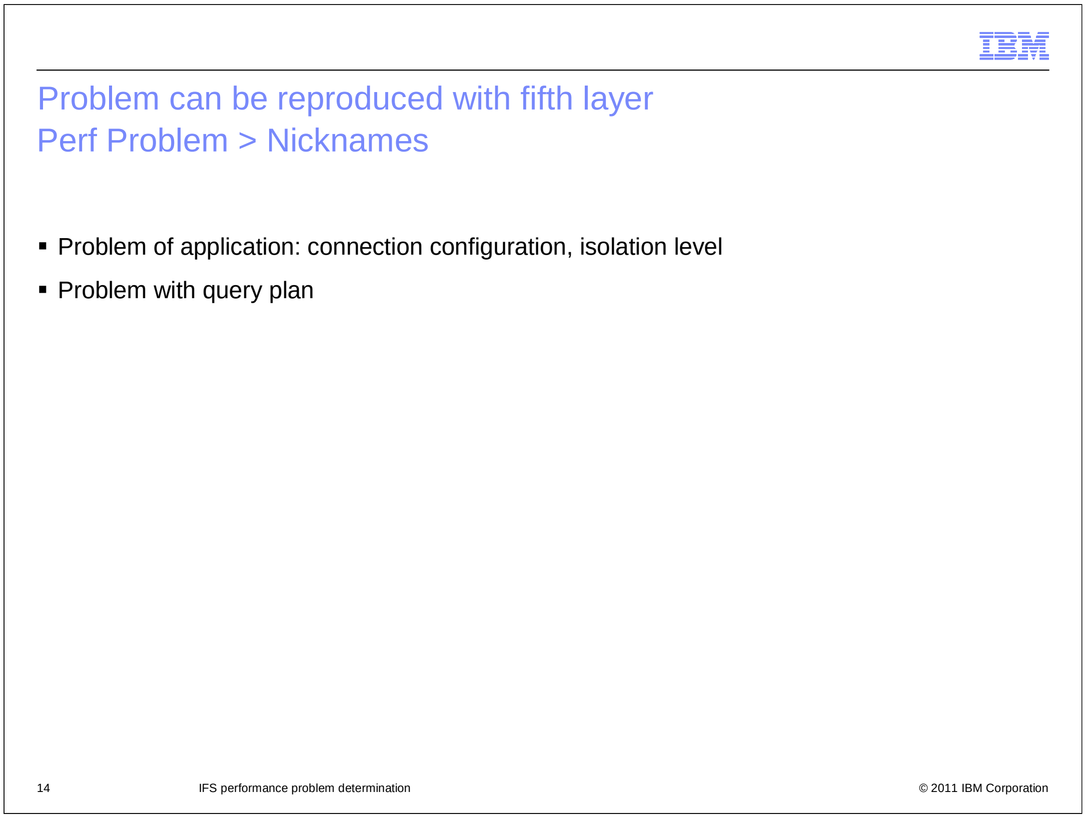
If the problem can be reproduced with third layer but not with the second layer, this may be a problem with the configuration for the wrapper. The way the wrapper code is connecting to the remote server may need to be optimized and you may need to contact IBM support to solve the problem.



Problem can be reproduced with fourth layer
Perf Problem > Nicknames

- May need to tune query plan
- Might be system bottleneck to resolve

If the problem can be reproduced with fourth layer, you may need to tune the query plan or, there might be a system bottleneck you need to resolve.

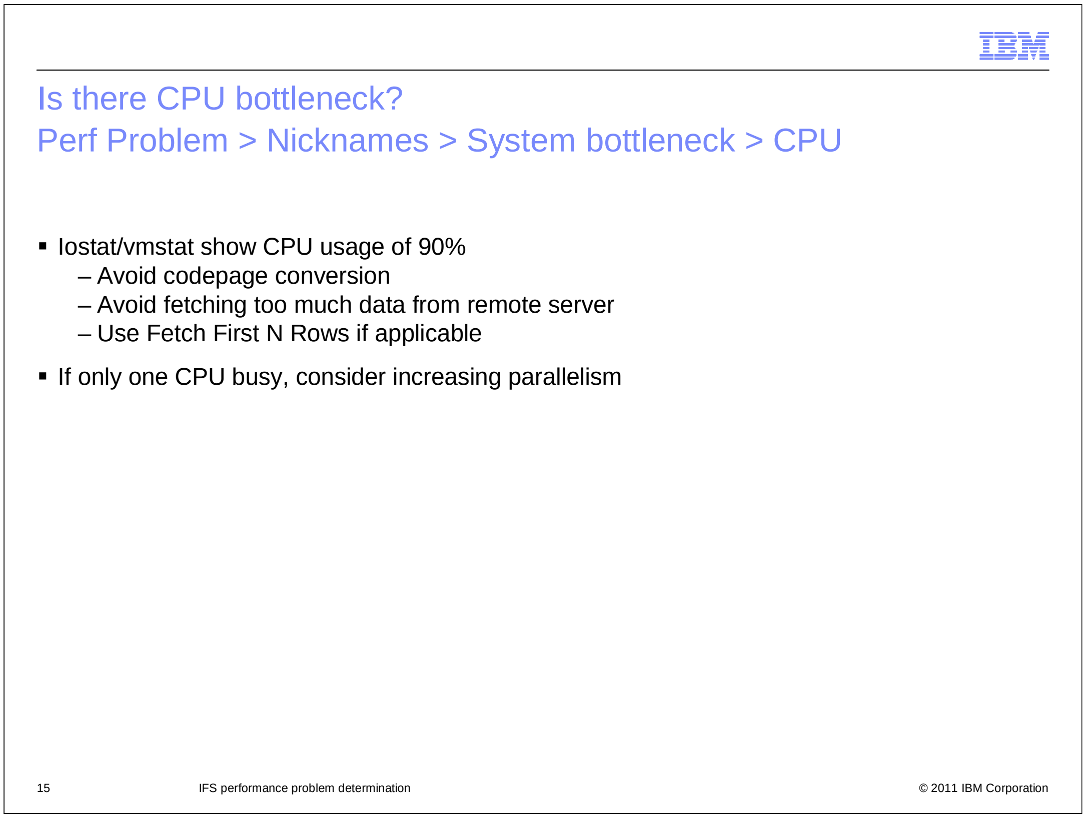


Problem can be reproduced with fifth layer
Perf Problem > Nicknames

- Problem of application: connection configuration, isolation level
- Problem with query plan

If the problem can only be reproduced with fifth layer, this may be a problem with the application, such as the connection configuration, isolation level and so on.

If the query has a parameter marker or host variable but does not have a performance problem when running with literal, there should be a problem with the query plan. It can be that the query plan with the parameter marker has a problem of query push down and is not efficient. You need to tune the problematic plan with parameter markers.



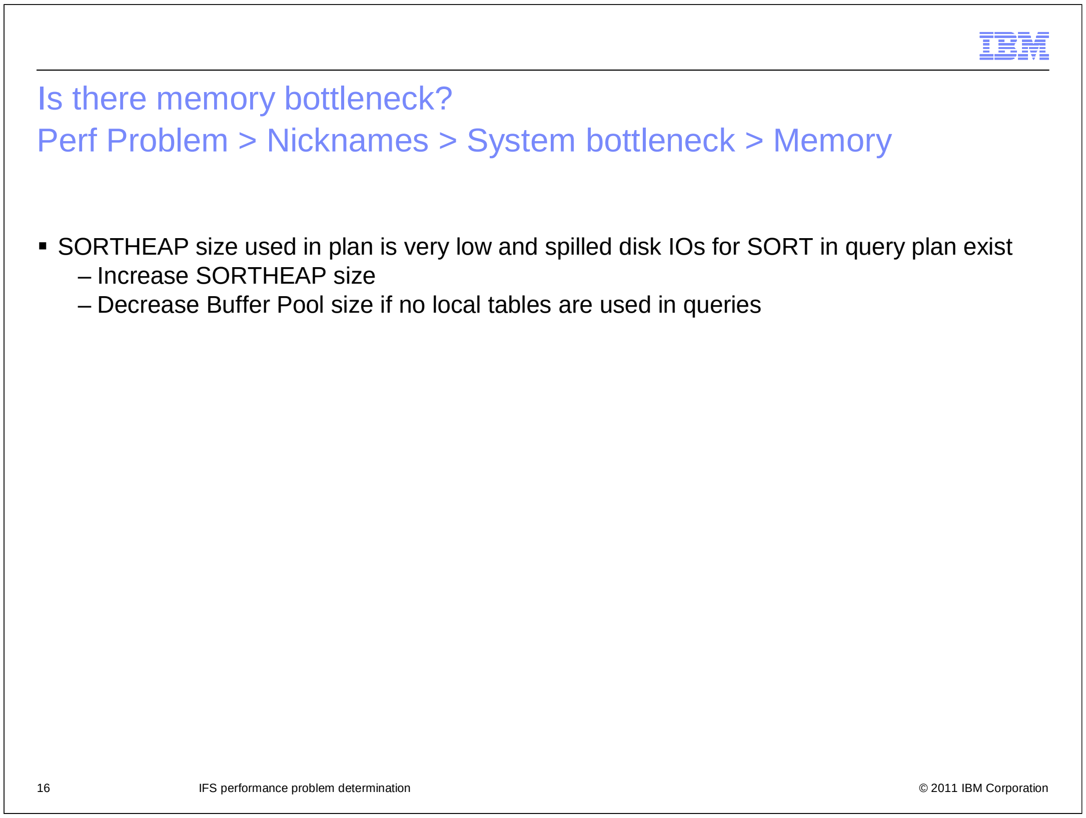
Is there CPU bottleneck?
Perf Problem > Nicknames > System bottleneck > CPU

- Iostat/vmstat show CPU usage of 90%
  - Avoid codepage conversion
  - Avoid fetching too much data from remote server
  - Use Fetch First N Rows if applicable
- If only one CPU busy, consider increasing parallelism

This presentation will provide details regarding tuning the query plan later. Before that, it will cover the system bottleneck issue. Looking at the CPU usage, if Iostat/vmstat show CPU usage of 90% or above, there may be a CPU bottleneck. To lower the CPU usage, check if there is codepage conversion. Codepage conversion will consume lots of CPU usage. Try to create Federation database with the same codepage as remote server to avoid codepage conversion.

Avoid fetching too much data from remote server. If IFS fetches too much data, there is lots of network interrupts that require CPU attention. To reduce the data fetched from remote server, consider changing the query plan to push down more predicates to remote server. Use Fetch First N Rows if applicable. If you only need the first N rows of the final result, you should add the FETCH FIRST N ROWS ONLY clause. By doing so, this will limit the query to the rows you are to fetch and process and reduce CPU usage.

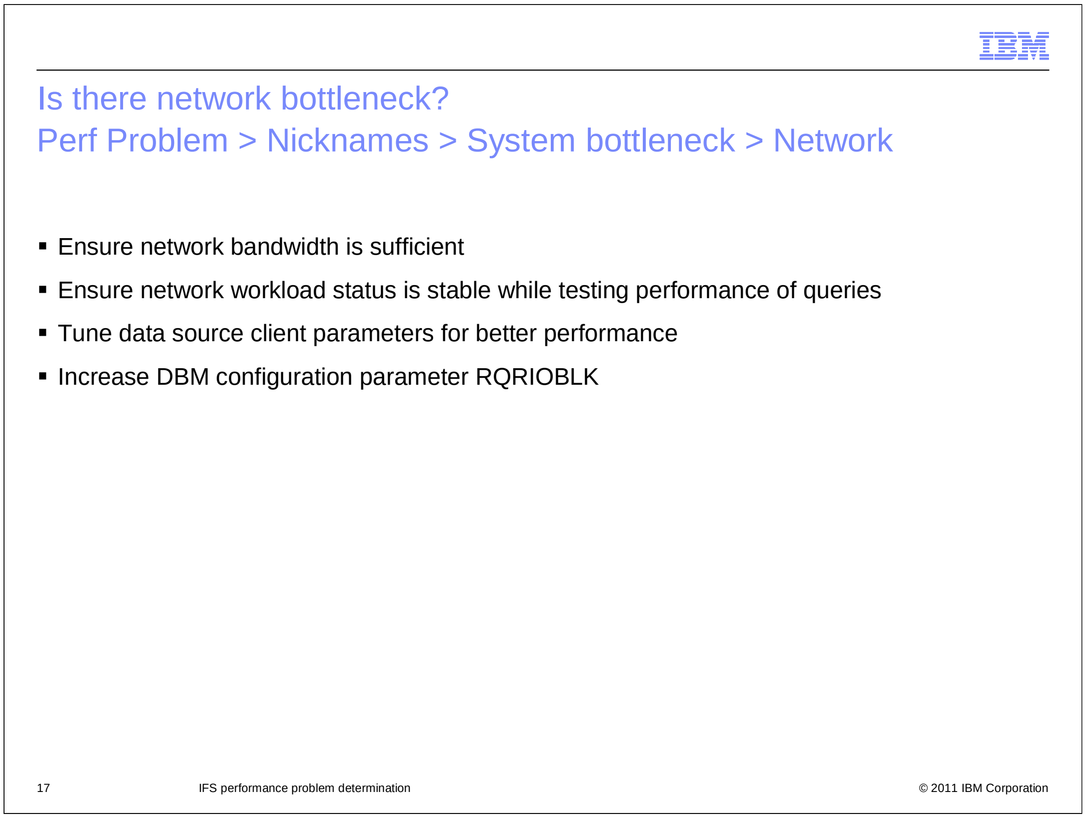
If only one CPU is busy, others are idle, it may be that this CPU is busy fetching data or processing data from remote server. Consider increasing parallelism to keep other CPUs busy fetching and processing data in parallel. You can add more concurrent connections to the IFS, each of which queries different nicknames.



Is there memory bottleneck?
Perf Problem > Nicknames > System bottleneck > Memory

- SORTHEAP size used in plan is very low and spilled disk IOs for SORT in query plan exist
  - Increase SORTHEAP size
  - Decrease Buffer Pool size if no local tables are used in queries

Consider if there is a memory configuration problem. From the query plan, if you see the SORTHEAP size used in the plan is very low and there are spilled disk IOs for SORT in the plan, the SORTHEAP size may need to be adjusted. You can increase SORTHEAP size each time by 250M. If there is no available memory, consider decreasing the Buffer Pool size if you are not accessing local tables in queries. The Buffer Pool is not quite useful for access to nicknames.

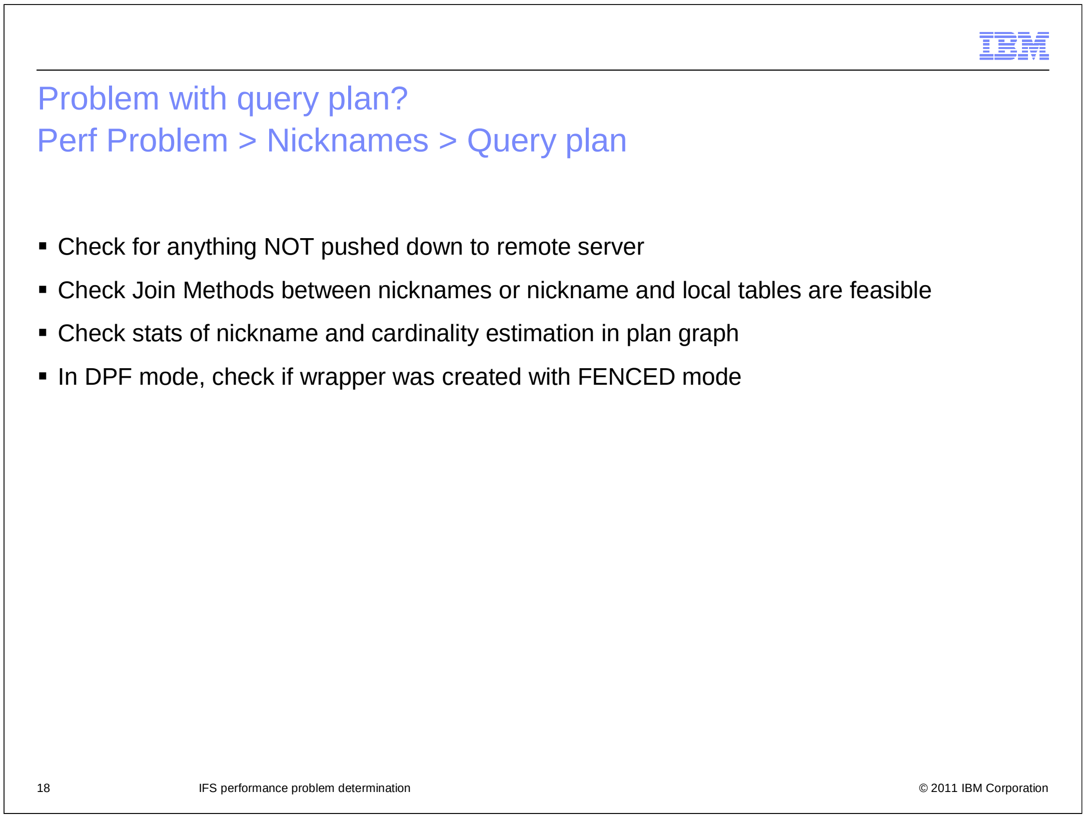


Is there network bottleneck?
Perf Problem > Nicknames > System bottleneck > Network

- Ensure network bandwidth is sufficient
- Ensure network workload status is stable while testing performance of queries
- Tune data source client parameters for better performance
- Increase DBM configuration parameter RQRIOBLK

Next, check the network. First ensure the network bandwidth is sufficient and the network status is stable while you are collecting the performance data of the queries. You can then tune the data source client parameters for better performance. For a different data source client, such as different ODBC, JDBC drivers and so on, there may exist different parameters that can be tuned for performance. Refer to the data source client configuration guide for details.

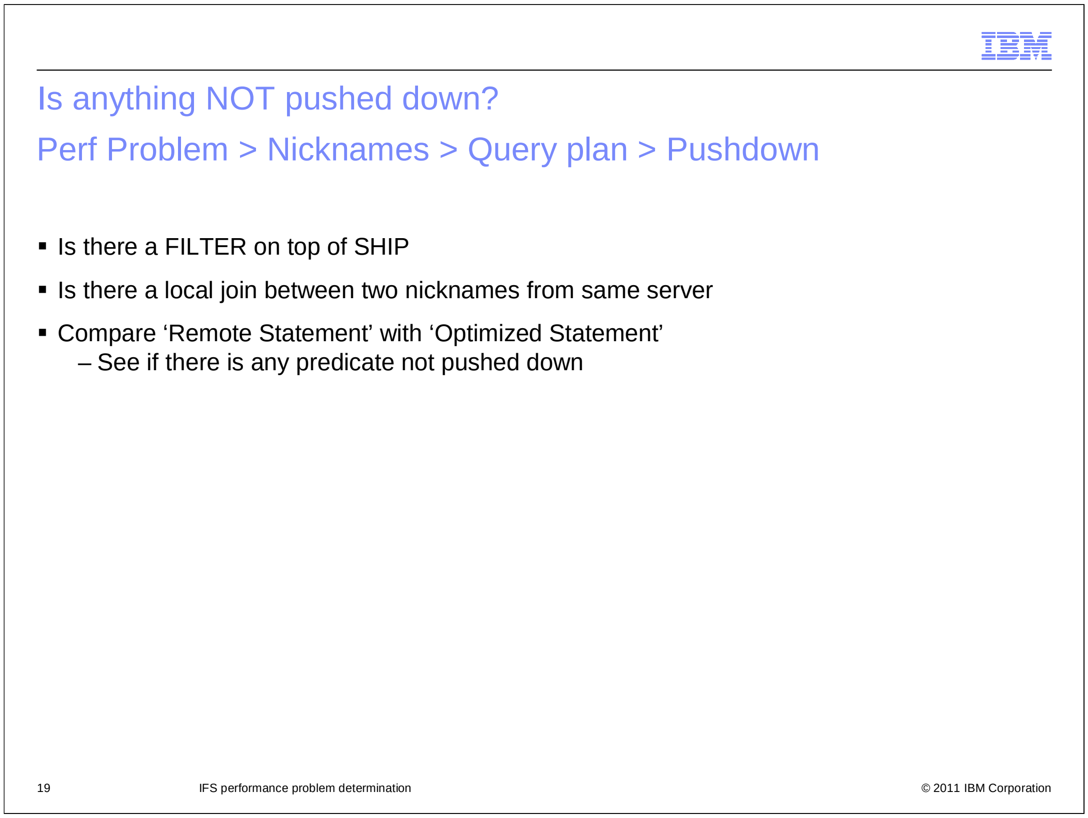
Another thing to consider is to increase the DBM configuration parameter RQRIOBLK, it is the buffer size used by IFS to fetch remote data. If you are using queries that fetch massive data from remote side, consider increasing this parameter.



Problem with query plan?
Perf Problem > Nicknames > Query plan

- Check for anything NOT pushed down to remote server
- Check Join Methods between nicknames or nickname and local tables are feasible
- Check stats of nickname and cardinality estimation in plan graph
- In DPF mode, check if wrapper was created with FENCED mode

If the system bottlenecks are handled, pay attention to the query plan. In fact, a large portion of the performance problems are due to poor query plans. First, get a query plan using db2exfmt (DB2 E-X-F-M-T). Refer to the reference page at the end of this presentation for reference of how to generate a query plan. When reviewing the query plan, check if there is anything that is NOT pushed down to remote server and if Join Methods between nicknames or nickname and local table are feasible. Also check the stats of nickname and cardinality estimation in the plan graph. In DPF mode, check if wrapper was created with FENCED mode.



Is anything NOT pushed down?

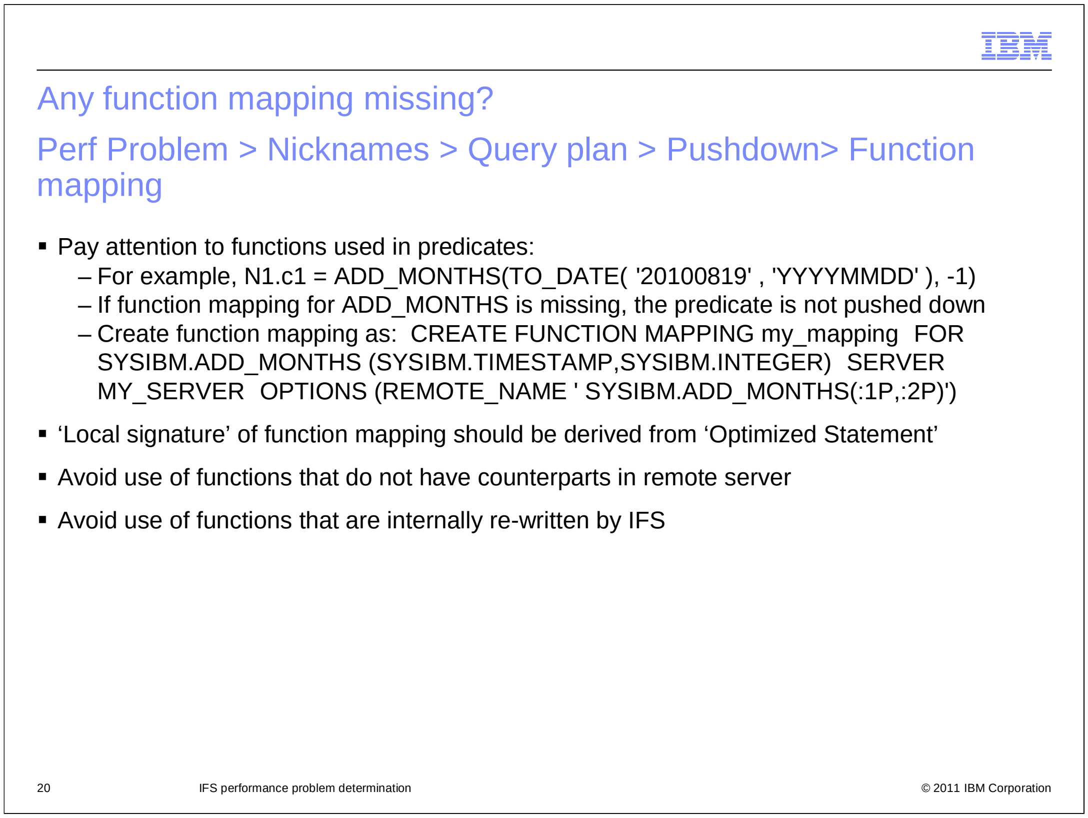
Perf Problem > Nicknames > Query plan > Pushdown

- Is there a FILTER on top of SHIP
- Is there a local join between two nicknames from same server
- Compare ‘Remote Statement’ with ‘Optimized Statement’
  - See if there is any predicate not pushed down

To check if there is anything NOT pushed down to remote server, check from the plan if there is a FILTER just on top of SHIP. If there is a FILTER right above SHIP, you know that the FILTER is basically for a predicate that is not deemed able to be pushed down to remote server. This may be due to lack of function mappings or server option settings, or both.

Is there a local join between two nicknames from the same server? If so, it may be due to the join predicate not being deemed able to push down or the optimizer decided not to push down according to cost estimation, or both. If this is the optimizer’s decision, update nickname stats or set server option db2_maximal_pushdown. This may make a different plan.

Compare ‘Remote Statement’ with ‘Optimized Statement’, see if there is any predicate not pushed down. The optimized statement in the plan generated by db2exfmt indicates the internal format of the SQL statement. It is also the statement based on which IFS makes the push down decisions. The remote statements are among the information of each SHIP operator in the plan. It indicates the actual statement pushed down to remote server. Comparing these two statements, you can tell how much of the query, including the predicates, are actually pushed down by IFS.



Any function mapping missing?

Perf Problem > Nicknames > Query plan > Pushdown> Function mapping

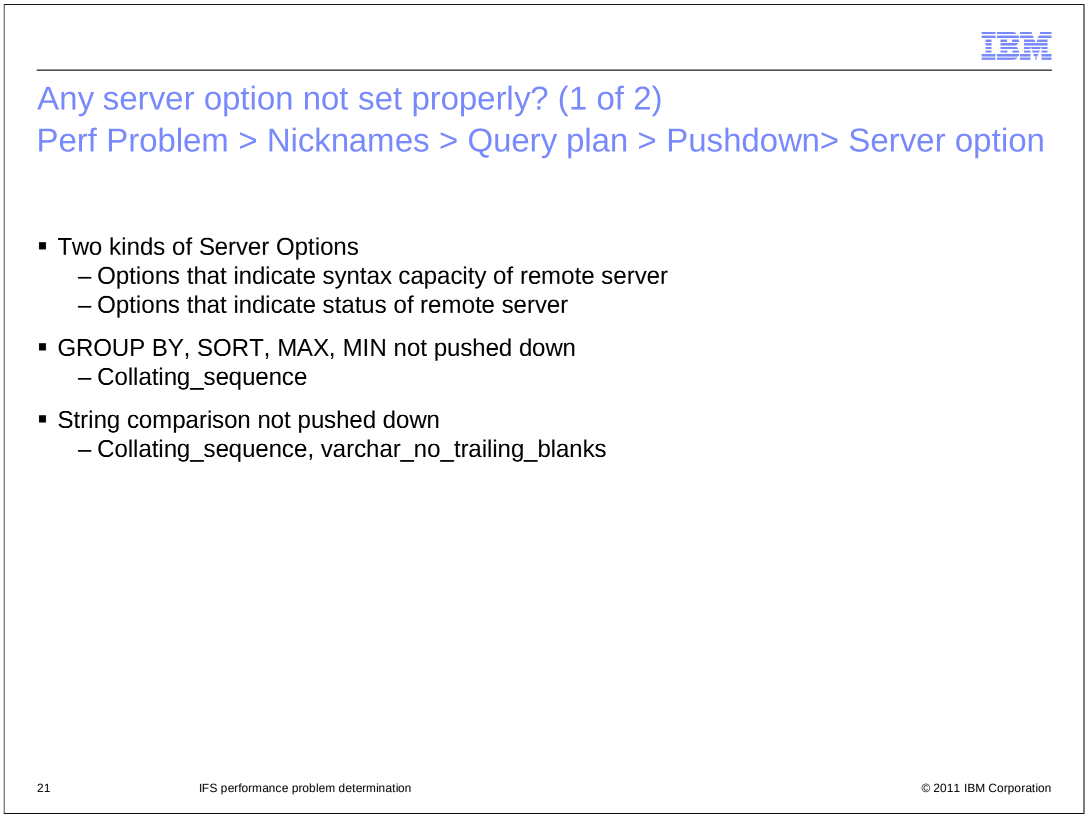
- Pay attention to functions used in predicates:
  - For example, N1.c1 = ADD_MONTHS(TO_DATE('20100819' , 'YYYYMMDD'), -1)
  - If function mapping for ADD_MONTHS is missing, the predicate is not pushed down
  - Create function mapping as: CREATE FUNCTION MAPPING my_mapping FOR SYSIBM.ADD_MONTHS (SYSIBM.TIMESTAMP,SYSIBM.INTEGER) SERVER MY_SERVER OPTIONS (REMOTE_NAME ' SYSIBM.ADD_MONTHS(:1P,:2P)')
- ‘Local signature’ of function mapping should be derived from ‘Optimized Statement’
- Avoid use of functions that do not have counterparts in remote server
- Avoid use of functions that are internally re-written by IFS

If you confirm that, there is something not pushed down, you need to drill down and see if it is caused by the missing function mappings. Missing of a function mapping, part of the query, including predicates, joins, and select list, cannot be pushed down. Pay attention to functions used in predicates. See the example displayed on this slide.

Note that the ‘local signature’ you use to create the function mapping should be derived from the ‘Optimized Statement’ from the plan. This is because when IFS is searching for a function mapping match, it will use signature from the internal SQL, that is the Optimized Statement of the plan.

Generally, try to avoid use of functions that do not have counterparts in remote server since function mappings are not possible for them and you will not be able to push down them.

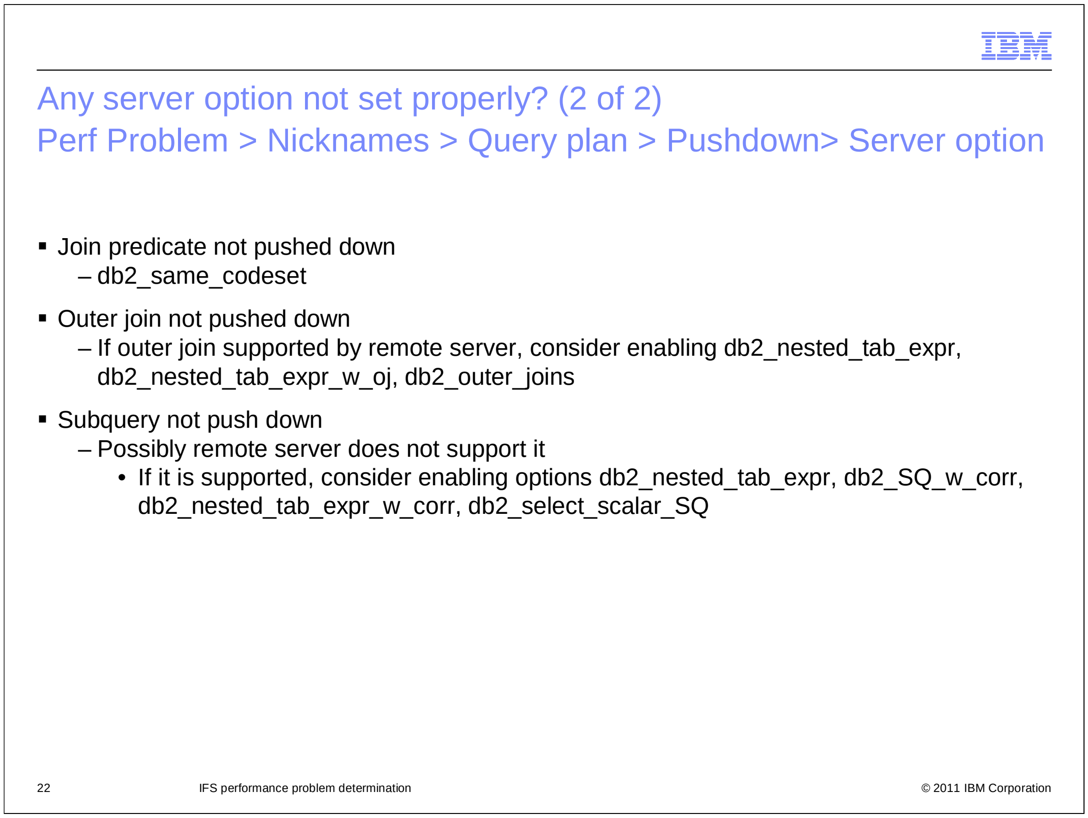
Also, avoid use of functions that are internally re-written to a format that are not usable by external users. For example, ROWNUM, since you will not be able to create function mappings for their internal format.



Any server option not set properly? (1 of 2)
Perf Problem > Nicknames > Query plan > Pushdown> Server option

- Two kinds of Server Options
  - Options that indicate syntax capacity of remote server
  - Options that indicate status of remote server
- GROUP BY, SORT, MAX, MIN not pushed down
  - Collating_sequence
- String comparison not pushed down
  - Collating_sequence, varchar_no_trailing_blanks

When there is something not pushed down, you need to also check if the necessary server options are set properly. There are two kinds of Server Options. Options that indicate syntax capacity of a remote server, for example, option “pushdown”, and Options that indicate status of remote server, for example, collating_sequence. From the plan graph, check if plan operators GROUP BY，SORT and functions MIN, MAX are pushed down. If not, you might need to consider setting server option collating_sequence. This server option indicates whether collating sequence of remote server is the same as Federation. If so, set it to ‘Y’ to facilitate push down. Check if there is any string comparison predicate not pushed down. If so, consider setting collating_sequence, varchar_no_trailing_blanks.

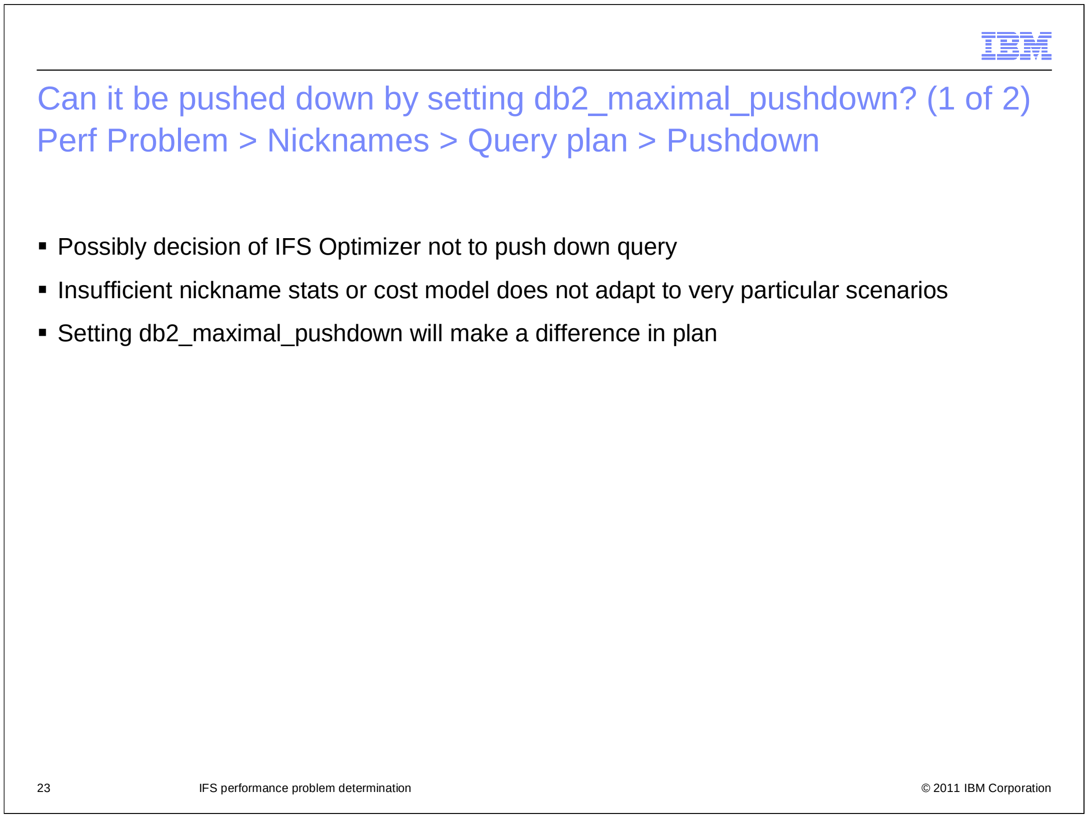


Any server option not set properly? (2 of 2)
Perf Problem > Nicknames > Query plan > Pushdown> Server option

- Join predicate not pushed down
  - db2_same_codeset
- Outer join not pushed down
  - If outer join supported by remote server, consider enabling db2_nested_tab_expr, db2_nested_tab_expr_w_oj, db2_outer_joins
- Subquery not push down
  - Possibly remote server does not support it
    - If it is supported, consider enabling options db2_nested_tab_expr, db2_SQ_w_corr, db2_nested_tab_expr_w_corr, db2_select_scalar_SQ

If there is any join not pushed down, you suspect that the join predicate is blocking push down. Consider setting server option db2_same_codeset to ‘Y’ if remote server share the same code page as Federation. Check if there is any outer join not pushed down. If outer join is supported by remote server, consider enabling db2_nested_tab_expr (db2 nested table expression), db2_nested_tab_expr_w_oj (db2 nested table expression with outer join), and db2_outer_joins. This will enable push down of outer join for this server.

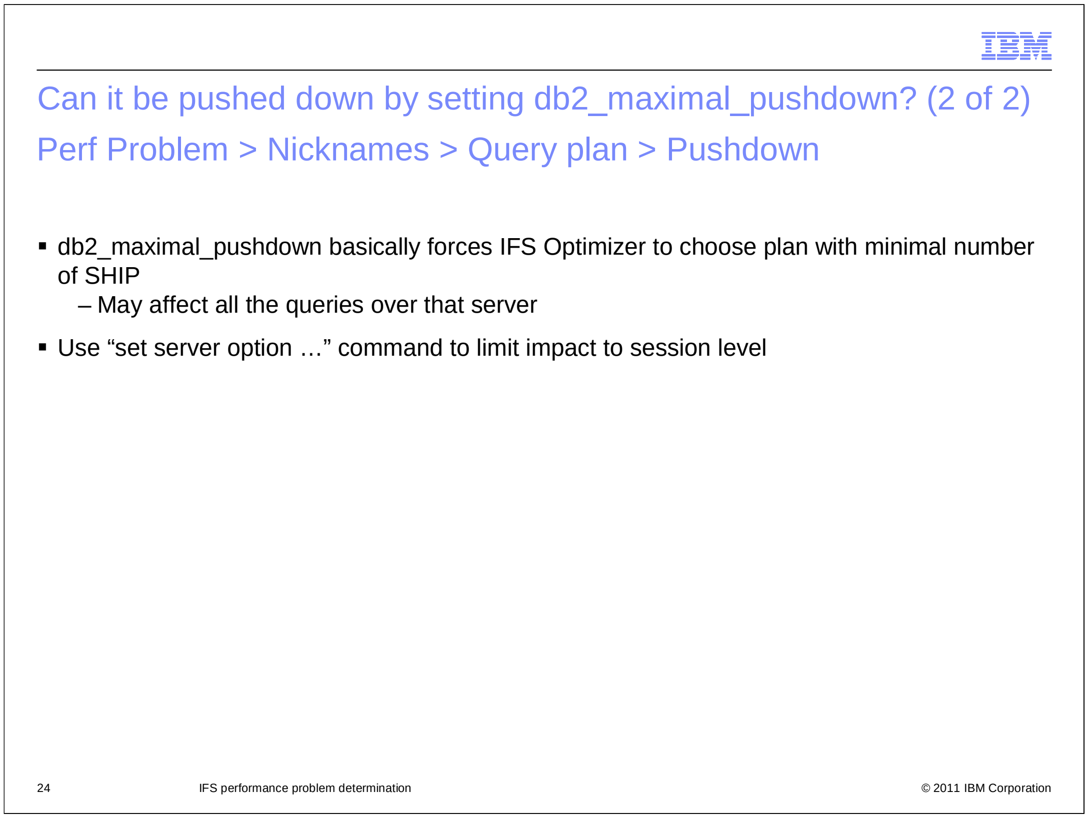
Check if there is any subquery not pushed down. It can be that the remote server does not support subquery. If it is supported, consider enabling options ((db2 nested table expression), db2_nested_tab_expr (db2 sub-query with correlation), db2_SQ_w_corr (db2 nested table expression with correlation), db2_nested_tab_expr_w_corr, and (db2 select scalar sub-query) db2_select_scalar_SQ all to ‘Y’ to push down it.



Can it be pushed down by setting db2_maximal_pushdown? (1 of 2)
Perf Problem > Nicknames > Query plan > Pushdown

- Possibly decision of IFS Optimizer not to push down query
- Insufficient nickname stats or cost model does not adapt to very particular scenarios
- Setting db2_maximal_pushdown will make a difference in plan

It may be the decision of IFS Optimizer not to push down the query. This can be caused by insufficient nickname stats or cost model does not adapt to very particular scenarios. If this is the case, setting db2_maximal_pushdown to ‘Y’ will make a difference in the plan. If it is set, Optimizer will try to seek a maximally pushed down plan.

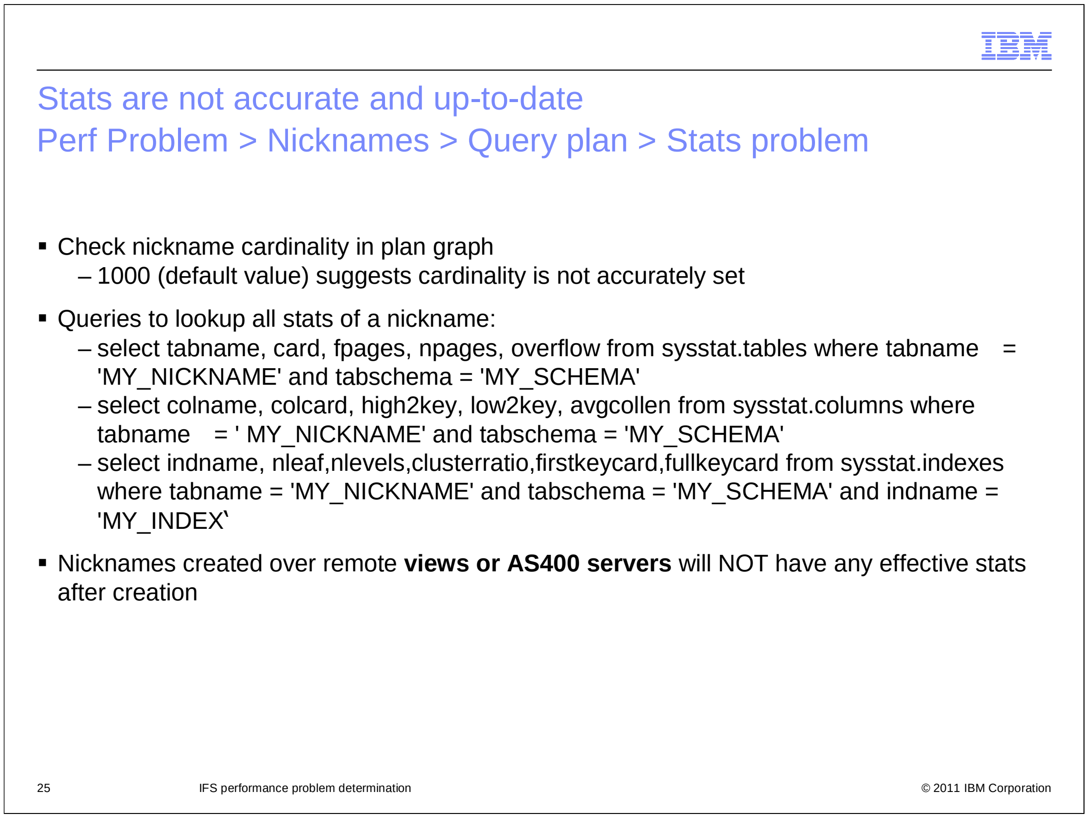


Can it be pushed down by setting db2_maximal_pushdown? (2 of 2)

Perf Problem > Nicknames > Query plan > Pushdown

- db2_maximal_pushdown basically forces IFS Optimizer to choose plan with minimal number of SHIP
  - May affect all the queries over that server
- Use “set server option …” command to limit impact to session level

Note that db2_maximal_pushdown basically forces IFS Optimizer to choose the plan with the minimal number of SHIPs. Setting it may affect all the queries over that server. To reduce the impact, use the “set server option db2_maximal_pushdown to 'Y' for server <server_name>” command instead of an “Alter server” command to limit its impact to session level.

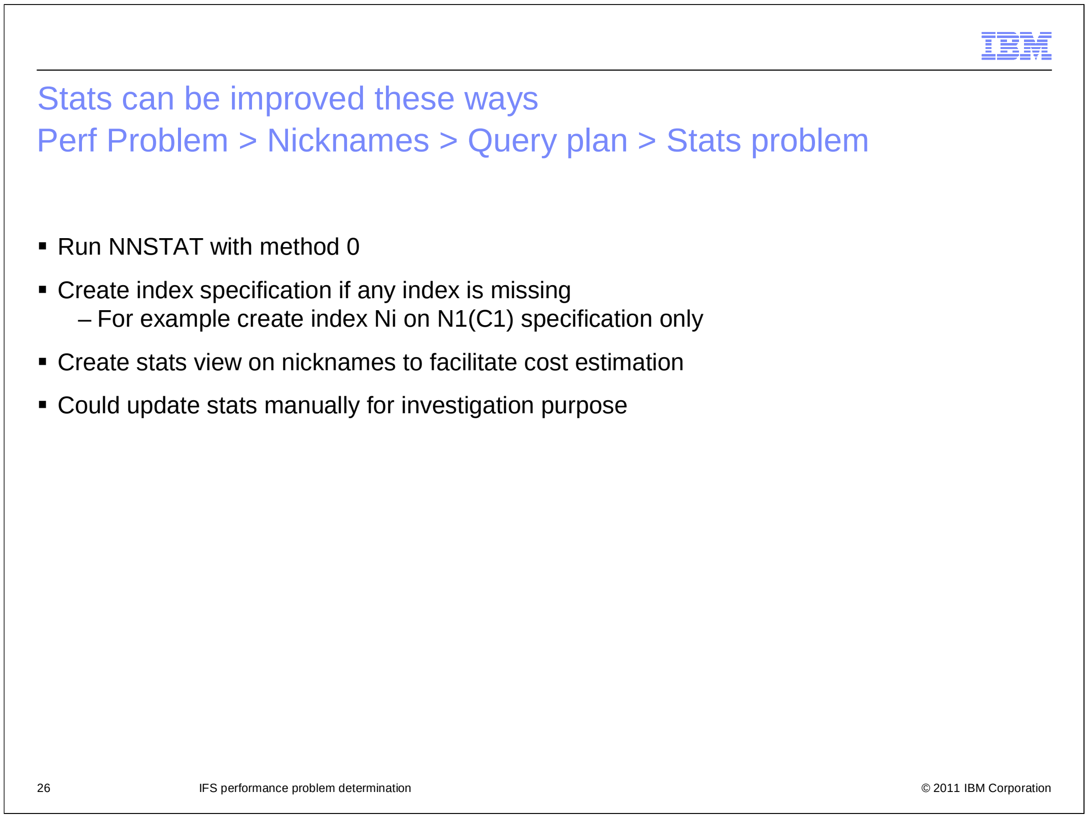


Stats are not accurate and up-to-date
Perf Problem > Nicknames > Query plan > Stats problem

- Check nickname cardinality in plan graph
  - 1000 (default value) suggests cardinality is not accurately set
- Queries to lookup all stats of a nickname:
  - select tabname, card, fpages, npages, overflow from sysstat.tables where tabname = 'MY_NICKNAME' and tabschema = 'MY_SCHEMA'
  - select colname, colcard, high2key, low2key, avgcollen from sysstat.columns where tabname = ' MY_NICKNAME' and tabschema = 'MY_SCHEMA'
  - select indname, nleaf,nlevels,clusterratio,firstkeycard,fullkeycard from sysstat.indexes where tabname = 'MY_NICKNAME' and tabschema = 'MY_SCHEMA' and indname = 'MY_INDEX`
- Nicknames created over remote **views or AS400 servers** will NOT have any effective stats after creation

Stats is key for the optimizer to generate the optimal plan. Check the nickname cardinality in the plan graph. 1000, which is the default value, suggests the cardinality is not accurately set.

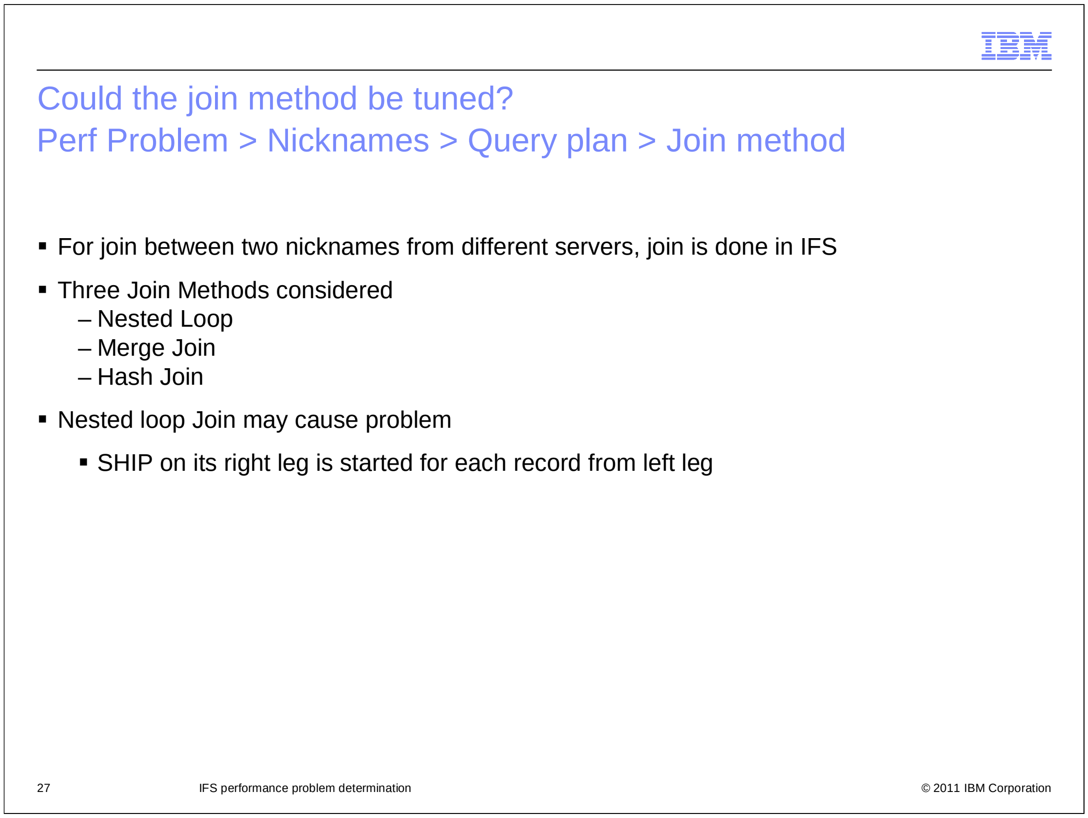
The other way to check stats is using the queries displayed on this slide to lookup all stats of a nickname. Note that nicknames created over remote **views or AS400 servers** will NOT have any effective stats after creation. Avoid creating nicknames based on remote views.



Stats can be improved these ways
Perf Problem > Nicknames > Query plan > Stats problem

- Run NNSTAT with method 0
- Create index specification if any index is missing
  - For example create index Ni on N1(C1) specification only
- Create stats view on nicknames to facilitate cost estimation
- Could update stats manually for investigation purpose

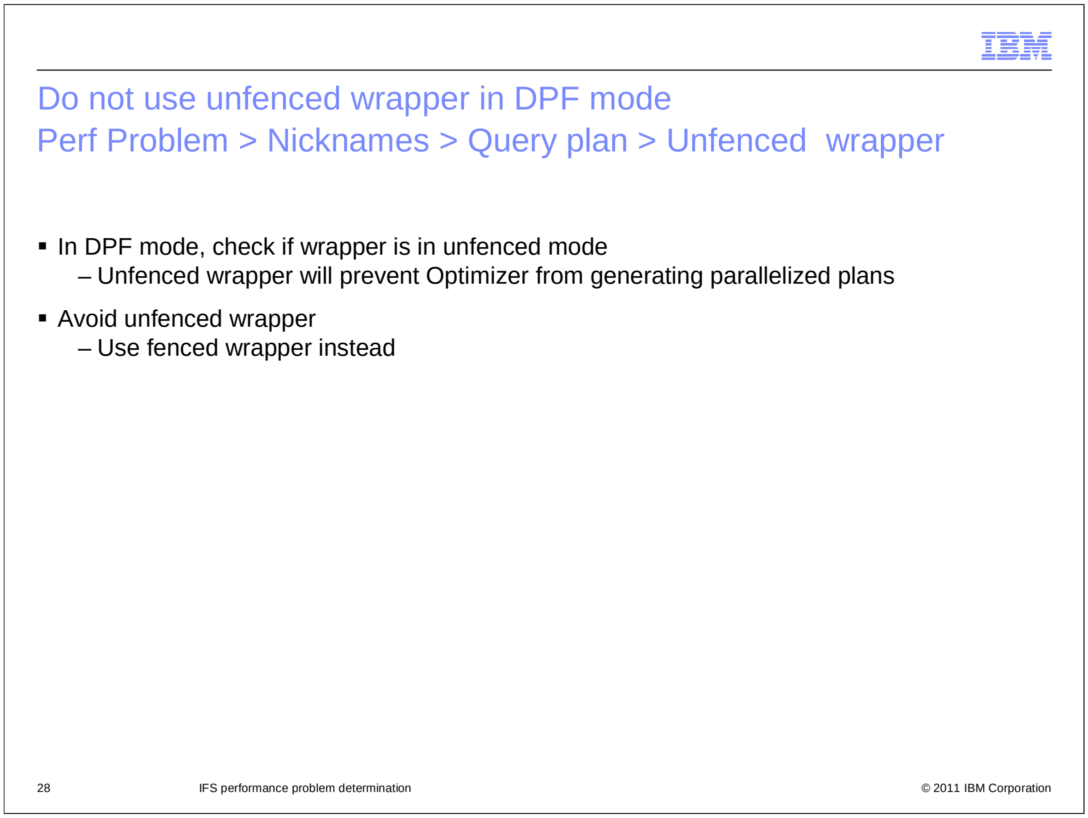
To fix stats problem, run NNSTAT with method 0. If there is an index on remote table, but no local index specification for the corresponding nickname, create index specification manually. For example, create index Ni on N1(C1) specification only. If the cardinality estimation from the plan graph is not accurate, consider creating stats view on nicknames to facilitate cost estimation. If you suspect the poor stats is causing a bad plan, try updating the stats manually for investigation purpose.



Could the join method be tuned?
Perf Problem > Nicknames > Query plan > Join method

- For join between two nicknames from different servers, join is done in IFS
- Three Join Methods considered
  - Nested Loop
  - Merge Join
  - Hash Join
- Nested loop Join may cause problem
  - SHIP on its right leg is started for each record from left leg

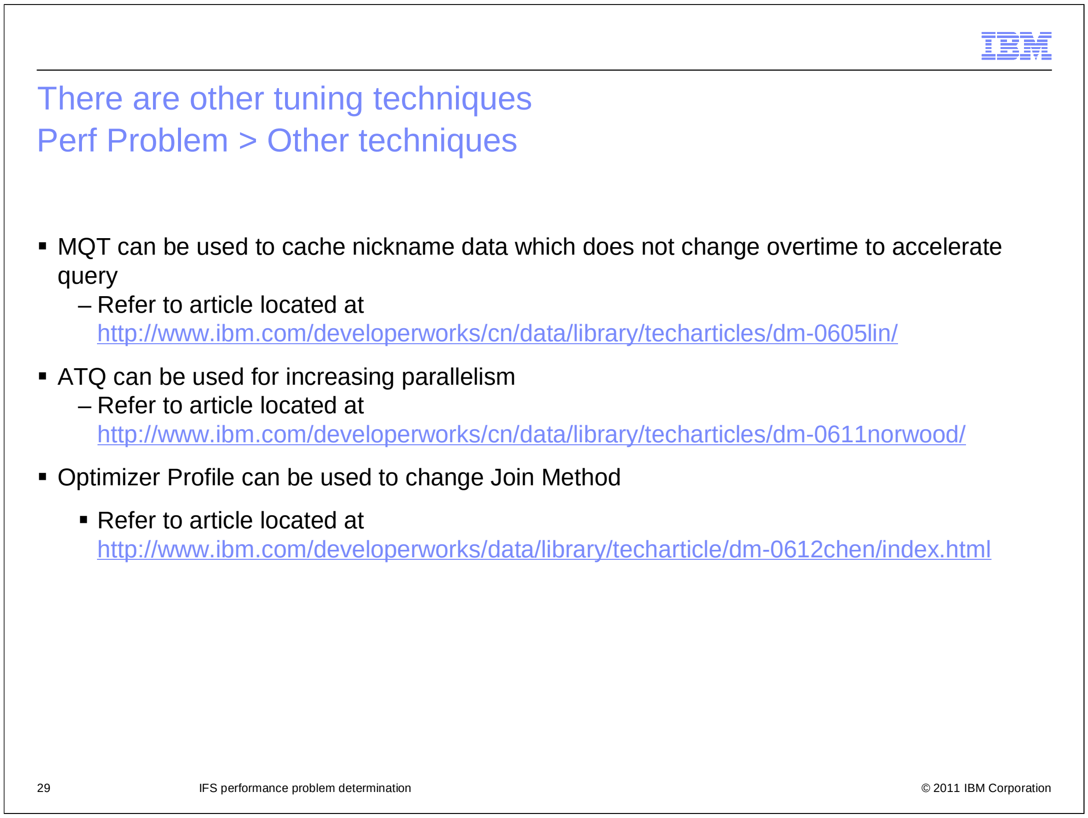
For a join between two nicknames from different servers, the join is done in IFS. When generating plans, three Join Methods are considered by IFS: Nested Loop Join, Merge Join and Hash Join. Nested Loop Join may cause a problem since the SHIP on its right leg is started for each record from the left leg.



Do not use unfenced wrapper in DPF mode
Perf Problem > Nicknames > Query plan > Unfenced wrapper

- In DPF mode, check if wrapper is in unfenced mode
  - Unfenced wrapper will prevent Optimizer from generating parallelized plans
- Avoid unfenced wrapper
  - Use fenced wrapper instead

If you are in DPF mode, check if wrapper is in unfenced mode. Unfenced wrapper will prevent Optimizer from generating parallelized plans. Avoid unfenced wrapper if possible and use fenced wrapper instead.



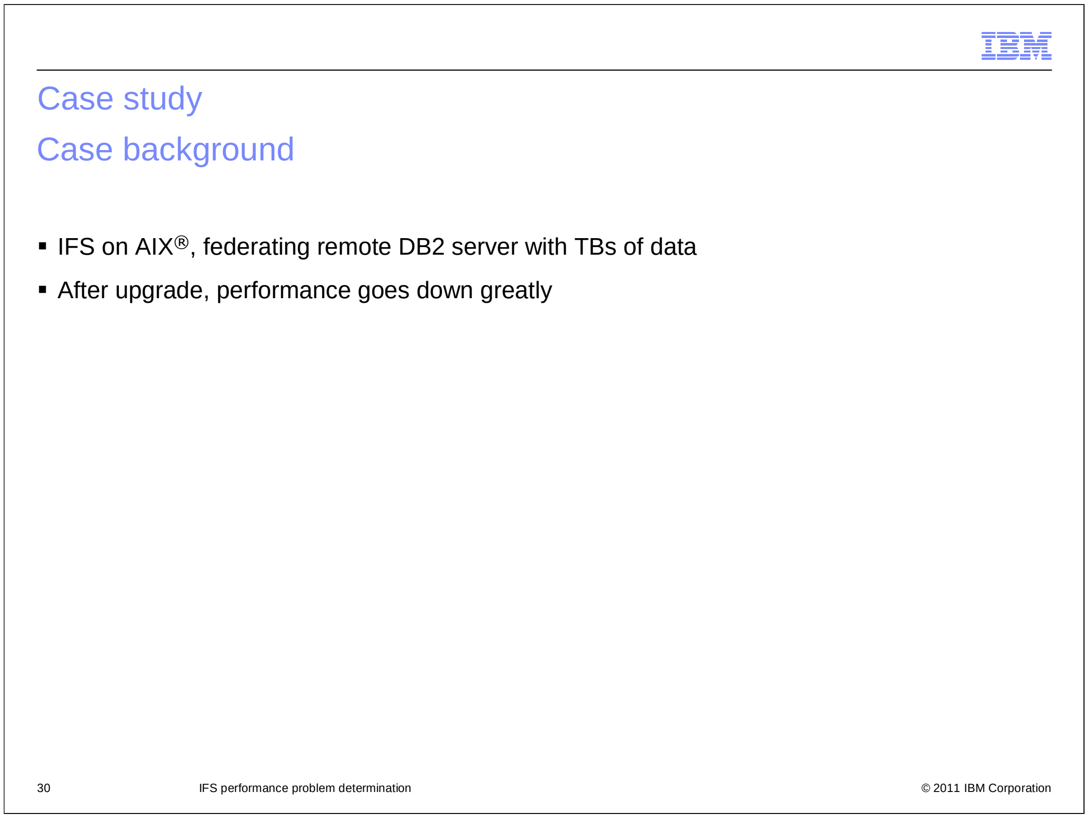
There are other tuning techniques
Perf Problem > Other techniques

- MQT can be used to cache nickname data which does not change overtime to accelerate query
  - Refer to article located at http://www.ibm.com/developerworks/cn/data/library/techarticles/dm-0605lin/
- ATQ can be used for increasing parallelism
  - Refer to article located at http://www.ibm.com/developerworks/cn/data/library/techarticles/dm-0611norwood/
- Optimizer Profile can be used to change Join Method
  - Refer to article located at http://www.ibm.com/developerworks/data/library/techarticle/dm-0612chen/index.html

Most of the time a federated query is typically spent in fetching remote data. MQT can be used to cache nickname data which does not change overtime to accelerate query. Refer to the link to the article displayed on this slide.

ATQ can be used for increasing parallelism. Refer to the link to the ATQ article displayed on this slide.

Optimizer Profile can be used to change Join Method. If you want to force a certain join method between nicknames or tables, consider using Optimizer Profile. Refer to the link to the Optimizer Profile article displayed on this slide.

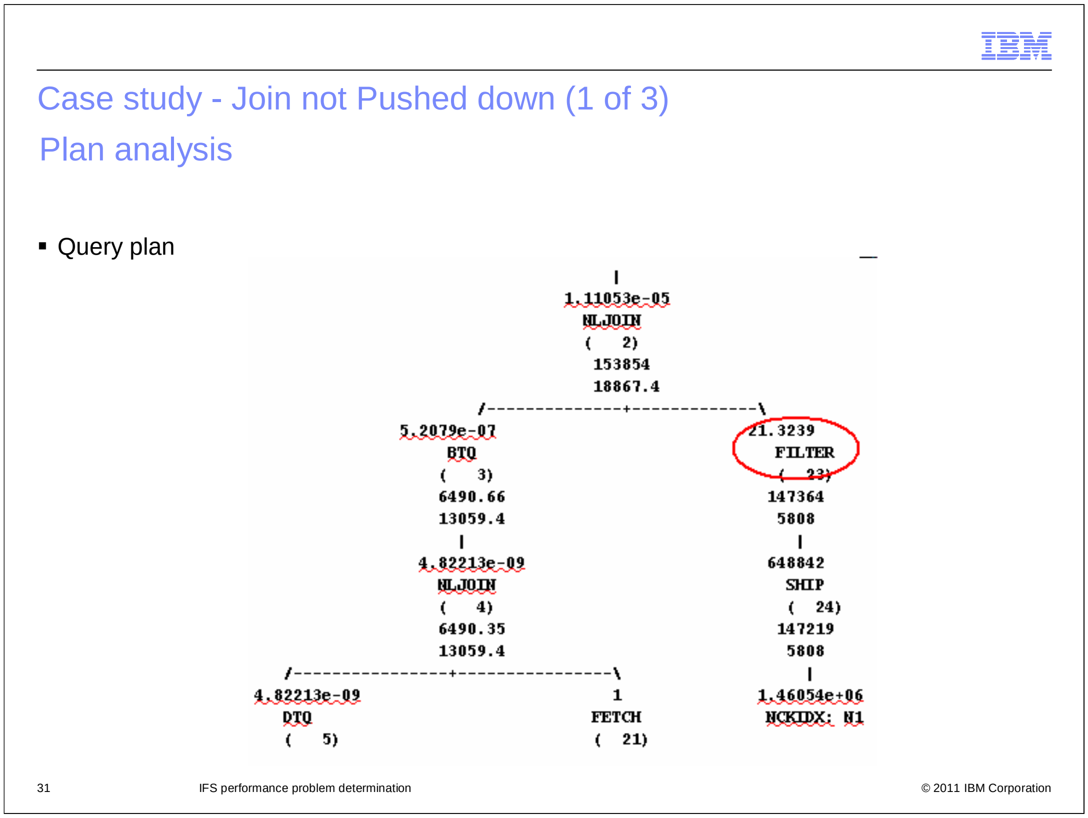


Case study

Case background

- IFS on AIX®, federating remote DB2 server with TBs of data
- After upgrade, performance goes down greatly

Consider the case of IFS running on AIX, federating remote DB2 server with TBs of data and after an upgrade the performance goes down greatly. See the next slide for the Plan Analysis.



Case study - Join not Pushed down (1 of 3)

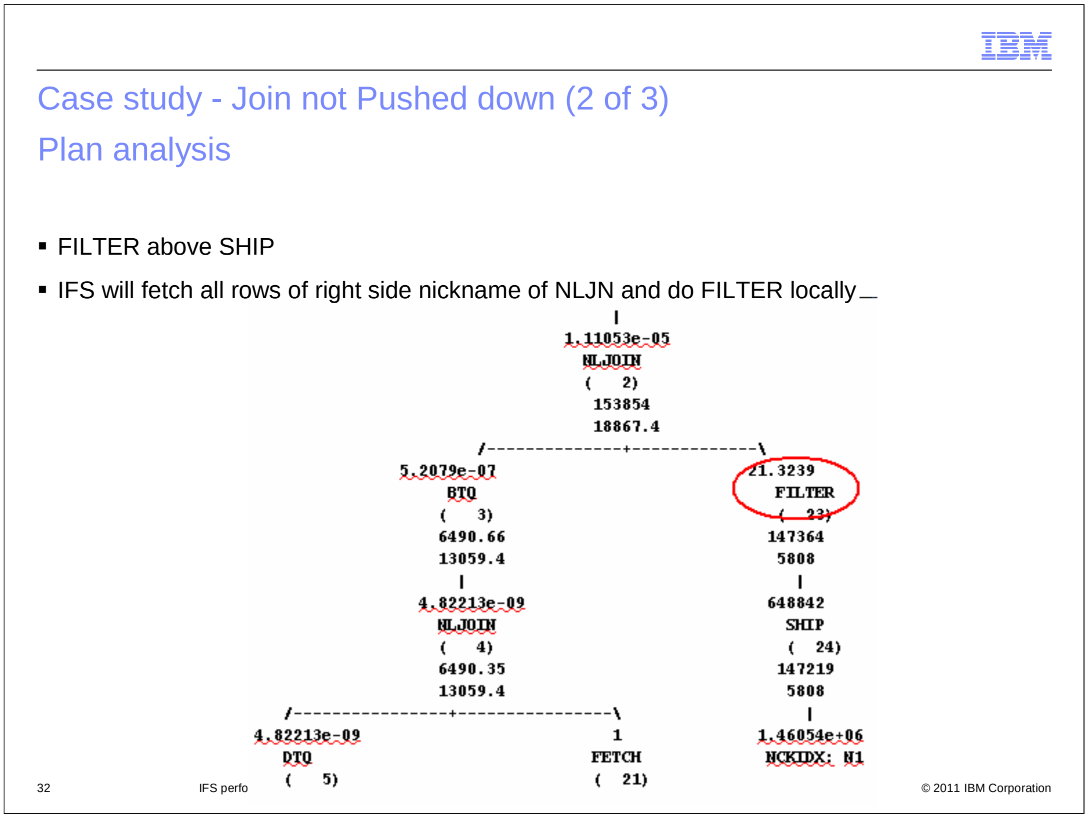
Plan analysis

- Query plan

```
                                     |
                                1.11053e-05
                                  NLJOIN
                                  (   2)
                                   153854
                                   18867.4
                       /-------------+-------------\
               5.2079e-07                            21.3239
                    BTQ                                FILTER
                   (   3)                              (  23)
                   6490.66                           147364
                   13059.4                            5808
                     |                                  |
                 4.82213e-09                         648842
                   NLJOIN                              SHIP
                   (   4)                              (  24)
                   6490.35                            147219
                   13059.4                             5808
   /---------------+---------------\                   |
4.82213e-09                          1              1.46054e+06
   DTQ                             FETCH             NCKIDX: N1
   (   5)                          (  21)
```

31 IFS performance problem determination © 2011 IBM Corporation

This slide displays the query plan graph from db2exfmt output.



Case study - Join not Pushed down (2 of 3)

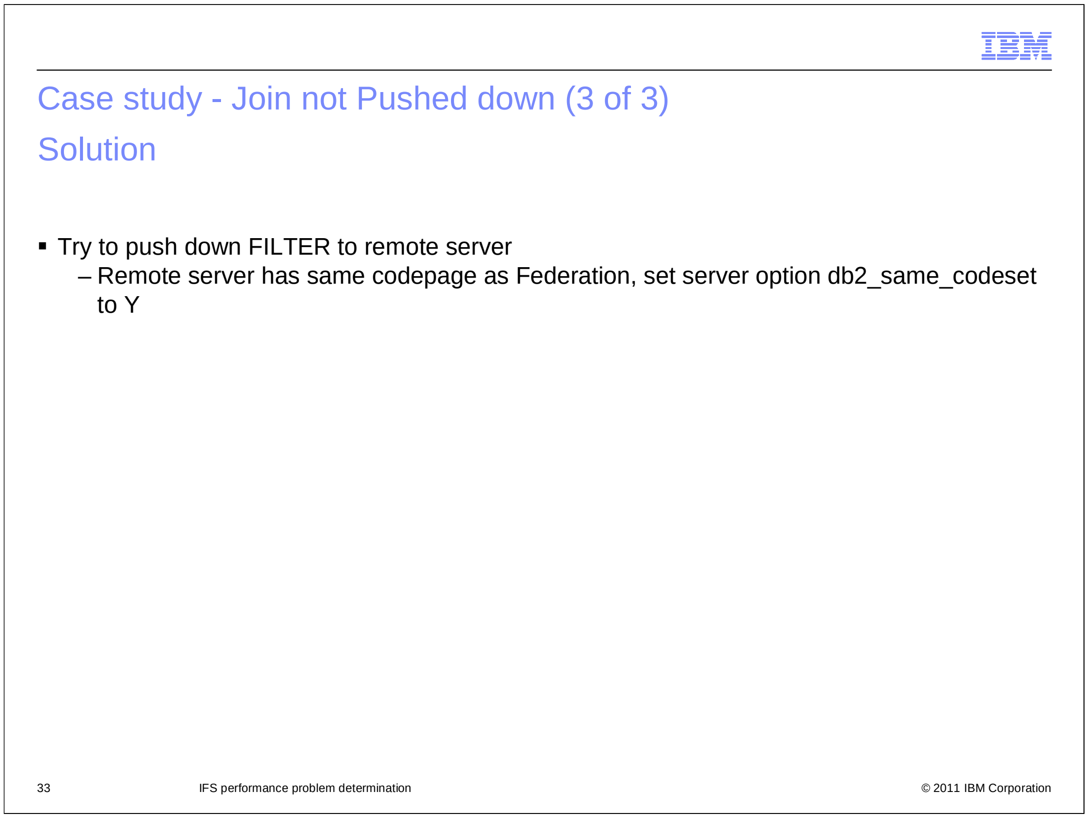
Plan analysis

- FILTER above SHIP
- IFS will fetch all rows of right side nickname of NLJN and do FILTER locally

```
                                      |
                                1.11053e-05
                                  NLJOIN
                                   (    2)
                                    153854
                                    18867.4
                        /--------------+-------------\
               5.2079e-07                          21.3239
                    BTQ                               FILTER
                    (    3)                            (   23)
                    6490.66                          147364
                    13059.4                           5808
                      |                                  |
                 4.82213e-09                         648842
                   NLJOIN                              SHIP
                    (    4)                             (   24)
                    6490.35                           147219
                    13059.4                            5808
    /----------------+----------------\                   |
4.82213e-09                           1             1.46054e+06
   DTQ                             FETCH             NCKIDX: N1
    (    5)                         (   21)
```

From the plan, you can see that the cardinality of all the nicknames are correct and it seems there is no stats problem. However, there is a FILTER above SHIP, suggesting there might be predicate not pushed down. Looking up information of the FILTER, you can see that it represents the join predicate T1.c1 = N1.c1. With this FILTER not pushed down, IFS will have to fetch all rows of the right side nickname, and do the FILTER locally.

Why is it not pushed down? All functions used in the query have their function mappings, so no missing function mappings. It might be the server options that cause the predicate not to be pushed down.

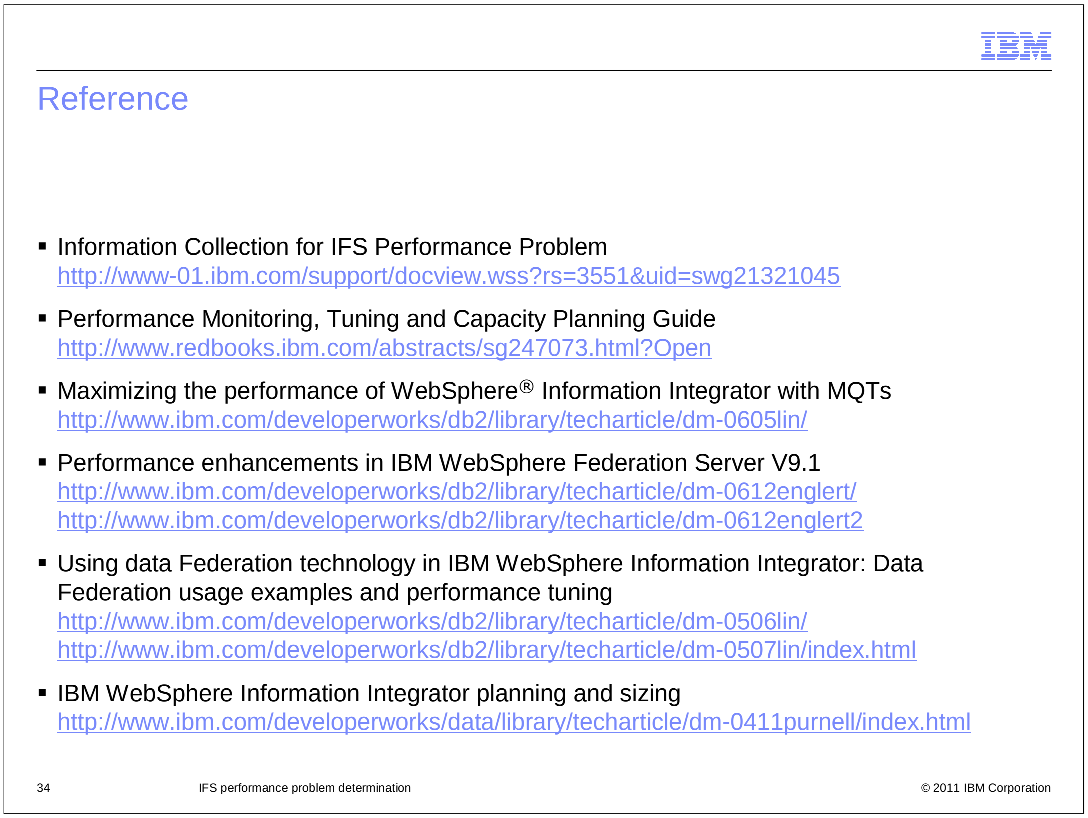


Case study - Join not Pushed down (3 of 3)

Solution

- Try to push down FILTER to remote server
  - Remote server has same codepage as Federation, set server option db2_same_codeset to Y

To push down the FILTER to remote server, and given that the remote server has the same codepage as Federation, set server option db2_same_codeset to Y. This server option setting can push down the FILTER because due to code page conversion, only if db2_same_codeset is set to ‘Y’, the join predicate T1.c1 = N1.c1 which is a string comparison with different column lengths, is pushed down. By setting server option db2_same_codeset to Y, you will have the join predicate pushed down, and index was then used in remote server to access N1’s remote table. Hence, the performance problem is resolved.



Reference

- Information Collection for IFS Performance Problem
  http://www-01.ibm.com/support/docview.wss?rs=3551&uid=swg21321045
- Performance Monitoring, Tuning and Capacity Planning Guide
  http://www.redbooks.ibm.com/abstracts/sg247073.html?Open
- Maximizing the performance of WebSphere® Information Integrator with MQTs
  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0605lin/
- Performance enhancements in IBM WebSphere Federation Server V9.1
  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0612englert/
  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0612englert2
- Using data Federation technology in IBM WebSphere Information Integrator: Data Federation usage examples and performance tuning
  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0506lin/
  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0507lin/index.html
- IBM WebSphere Information Integrator planning and sizing
  http://www.ibm.com/developerworks/data/library/techarticle/dm-0411purnell/index.html

This slide displays links to websites referenced in this presentation. The first link is important in that it contains the commands used to collect db2exfmt output and other diagnose information.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_EduProbDeterminiation.ppt

This module is also available in PDF format at: ../EduProbDeterminiation.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information