Information Management software

Redpaper

James Pickel

DB2 9 for z/OS Data Sharing: Distributed Load Balancing and Fault Tolerant Configuration

Introduction

This IBM® Redpaper provides information about exploiting client load balancing and fail-over capabilities across a DB2® data sharing group or a subset of the group members.

The information provided will help network specialists and database administrators in appropriately configuring data sharing groups as TCP/IP servers.

We show how to use the functions of DB2 sysplex workload balancing functions, supported by DB2 Connect™ Server and the IBM DB2 Driver for JDBC™ and SQLJ, in conjunction with the functions of TCP/IP Sysplex Distributor, configured with Dynamic Virtual IP address (Dynamic VIPA or DVIPA) and automatic VIPA takeover, to provide superior load distribution and fail-over capability for transactions and connections across the members of a data sharing group. The two functions work together to ensure the highest possible availability for *remote* applications accessing a DB2 location in a sysplex.

This paper contains the following sections:

- Exploiting client-load balancing across a DB2 group
- Enabling sysplex workload balancing in IBM data server clients
- Configuring sysplex workload balancing for a subset of members

Disclaimer

This paper is current as of July 2008 and contains recommendations from results recorded in several customer environments. Individual results may vary depending on factors such as network topology, system capacity, type of SQL, virtual storage utilization, connection pooling, number of active connections, transactions per connection, maintenance level, and other factors.

Exploiting client-load balancing across a DB2 group

DB2 sysplex workload balancing functions provided by DB2 Connect Server and the IBM DB2 Driver for JDBC and SQLJ, and Connection Concentrator support are critical components of the DB2 data-sharing fault-tolerant environment. DB2 sysplex workload balances work across the DB2 group using connection concentration. The connection concentrator balances application connections by multiplexing transactions across a set of managed server connections to DB2. To balance transactions (individual units of work delimited by a commit) across individual members of the DB2 group, unique member IP addresses are configured for each member using a DVIPA called the member-specific DVIPA. Member IP addresses are not used by remote applications since they route connections to a specific member. Setting up a member IP address as a DVIPA allows routing to work even if a member fails over to another LPAR using VIPA takeover. As soon as the member is started after a failure, any distributed in-doubt threads can be quickly recovered.

The TCP/IP sysplex distributor configured with DVIPA and automatic VIPA is the other critical component of the DB2 data-sharing fault-tolerant environment. The functionality of TCP/IP sysplex distributor is that one IP entity advertises ownership of an IP address by which a particular service is known. In this fashion, the single system image of DB2 by remote applications is the use of a special IP address enabled for connection distribution across the group. This IP address is called a *distributed DVIPA*. Further, in sysplex distributor, the IP entity advertising the distributed DVIPA and dispatching connections destined for it is itself a system image within the sysplex, referred to as the distributing stack. To support a distributed DB2 single system image, a group IP address is configured and used by all application servers to access the DB2 group. All DB2 members are configured with the same group DVIPA which is distributed across the group using a distributed DVIPA supported by the sysplex distributor. DB2 calls the distributed DVIPA the DB2 location or group dynamic VIPA.

General guidelines for configurations

Configuring a fault tolerant environment for applications accessing a DB2 data sharing group using TCP/IP requires expertise in z/OS V1R8 Communications Server, DB2 9 for z/OS and the DB2 Version 9.5 for LUW data server clients. This paper was written to provide recommendations when configuring the network, configuring the DB2 subsystems and configuring the DB2 clients. Once configured correctly, the system autonomically manages work to the DB2 group and ensures the highest availability without requiring application knowledge of the parallel sysplex.

Configuring the TCP/IP network

DB2 provides an initial contact port and a resync port. The initial contact port is normally DB2's registered well-known port 446 and is used to process SQL requests. The resync port is used by a database client in two situations. One is when the SQL connection fails leaving in-doubt threads, and the client and server need to resynchronize after the error. The other one is for other connections used to interrupt SQL processing on a different application connection. Obviously, resynchronization needs to occur with the specific DB2 instance with which the client was in session, so this instance must be reachable via a specific IP address (the member-specific DVIPA in this case). To address the case when the DB2 itself terminates and is restarted on another image, possibly with another DB2 instance, each DB2 instance configures its resync address with a port number unique to that instance. In Figure 1 on page 3, these resync addresses are represented by ports 5001, 5002, and 5003, for the three DB2 instances DB2A, DB2B, and DB2C, respectively.

The initial contact point to the DB2 group is with the DB2 port 446 using the distributed DVIPA bound to each member DVIPA, with the BIND parameter on the PORT reservation statement in TCP/IP configuration. Example 1 on page 3 shows the port and VIPA definitions for the three DB2 instances. Similar TCP/IP configuration for each DB2 instance may be configured on all TCP/IP stacks where the DB2 member may be executed, whether normally or through restart.

In DB2 9 for z/OS® the specific and group IP addresses can be configured in the DB2 boot strap dataset allowing DB2 to listen to any IP address, thereby eliminating the need to use the BIND parameter on the PORT reservation statement in TCP/IP configuration. The location of a particular DB2 instance at any point in time is thus transparent to the database application servers.

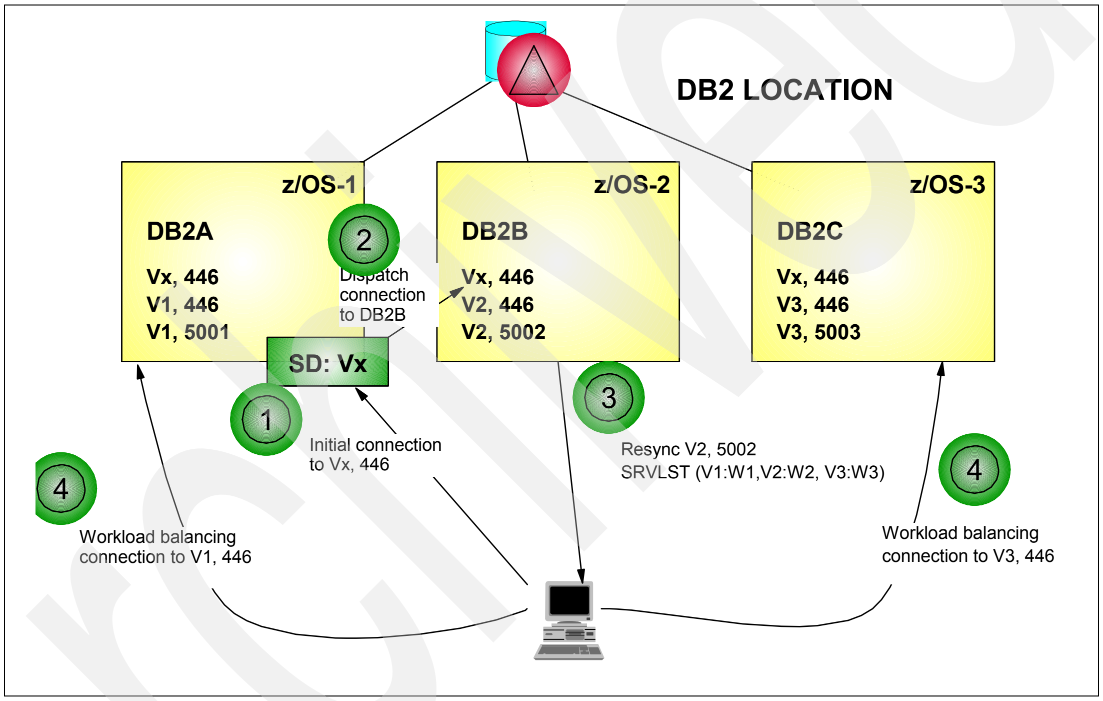
DB2 LOCATION
z/OS-1
DB2A
Vx, 446
V1, 446
V1, 5001
z/OS-2
DB2B
Vx, 446
V2, 446
V2, 5002
z/OS-3
DB2C
Vx, 446
V3, 446
V3, 5003
SD: Vx
2
Dispatch connection to DB2B
1
Initial connection to Vx, 446
3
Resync V2, 5002
SRVLST (V1:W1,V2:W2, V3:W3)
4
Workload balancing connection to V1, 446
4
Workload balancing connection to V3, 446

Figure 1   DB2 instances and configured resync addresses with unique port numbers

In Figure 1, the first socket is bound to the distributed DVIPA, Vx, on port 446, and the resync port is bound to a unique IP address and port for each DB2 instance (V1:5001 for DB2A, V2:5002 for DB2B, and V3:5003 for DB2C).

Example 1   Port and DVIPA definitions for DB2 instances

DB2A

```
Port
  446  tcp db2adist shareport bind Vx
  5001 tcp db2adist bind V1
  446  tcp db2bdist shareport bind Vx
  5002 tcp db2bdist bind V2
  446  tcp db2cdist shareport bind Vx
  5003 tcp db2cdist bind V3
VipaDynamic
  VipaRange Define 255.255.255.255 V1
  VipaRange Define 255.255.255.255 V2
  VipaRange Define 255.255.255.255 V3
```

DB2B

```
Port
  446  tcp db2adist shareport bind Vx
  5001 tcp db2adist bind V1
  446  tcp db2bdist shareport bind Vx
  5002 tcp db2bdist bind V2
  446  tcp db2cdist shareport bind Vx
  5003 tcp db2cdist bind V3
VipaDynamic
  VipaRange Define 255.255.255.255 V1
  VipaRange Define 255.255.255.255 V2
  VipaRange Define 255.255.255.255 V3
  VipaDefine 255.255.255.255 Vx        VipaBackup 1 Vx
  VipaDistribute Define Vx Port 446   VipaDistribute Define Vx Port 446
       DestIP all                           DestIP all
EndVipaDynamic                       EndVipaDynamic
```

DB2C

```
Port
  446  tcp db2adist shareport bind Vx
  5001 tcp db2adist bind V1
  446  tcp db2bdist shareport bind Vx
  5002 tcp db2bdist bind V2
  446  tcp db2cdist shareport bind Vx
  5003 tcp db2cdist bind V3
VipaDynamic
  VipaRange Define 255.255.255.255 V1
  VipaRange Define 255.255.255.255 V2
  VipaRange Define 255.255.255.255 V3
  VipaBackup 2 Vx
  VipaDistribute Define Vx Port 446
       DestIP all
 EndVipaDynamic
```

The initial contact request from any client may go to any of the DB2 instances in the data sharing group. Once initial connection is established, however, application servers in addition may perform their own load balancing for subsequent work among the available instances. This is accomplished by DB2 sending a list of IP addresses to the DRDA® client, one for each DB2 instance. Because the initial contact listening socket is listening on the location/group distributed DVIPA, DB2 opens an additional listening socket on the same port as the location listening socket, but bound to the member-specific DVIPA. This is illustrated in Figure 1 on page 3 as the middle socket pair (V1:446, V2:446, and V3:446 for the three DB2 instances) which leverage z/OS TCP/IP Dynamic VIPAs and sysplex distributor for higher availability.

To show how this works in Figure 1 on page 3, the DRDA client connects to access DB2 location using the group DVIPA and port 446. Sysplex distributor (running on the same image as DB2A) decides where to route the request based on available capacity, and in this scenario, the request actually goes to DB2B. DB2B returns its resync info (V2:5002) to the client. For database application servers where the connection concentrator is not configured, this is all that is needed. Each unit of work results in a new connection to the DB2 location DVIPA, which is distributed to the DB2 on the operating system image with the most apparent capacity and other information considered by sysplex distributor in making the decision.

DB2 returns a list of server addresses and corresponding server's capacity on a connection boundary, transaction boundary, or when a connection is reused by another application, termed the server list (SRVLST) in flow number 3 in Figure 1 on page 3. Once the database client receives the server list, it balances subsequent connections among the available servers using the info in the server list. This is illustrated as the two flows numbered 4 in the diagram. DB2 is thus a full exploiter of both DVIPA and sysplex distributor, through specific adaptation by DB2 for z/OS.

Configuring the DB2 subsystems

The following are the general guidelines for setting DB2 installation parameters that impact the utilization of the DB2 connection concentrators:

- DSN6FAC CMTSTAT INACTIVE: required if using concentrator
- DSN6SYSP MAXDBAT 500: if mostly dynamic SQL (JDBC or CLI), reduce to 350
- DSN6SYSP CONDBAT 5000: max number of pooled connections per member

- DSN6FAC TCPKPALV - 1 to 65534: time to execute longest SQL statement

DB2 needs to enable threads to be pooled by setting the CMTSTAT to INACTIVE. An inactive connection uses less storage and frees up DB2 resources associated with the transaction when a thread commits a transaction. When connections are disassociated from the thread, the thread is allowed to be pooled and reused for other connections. This provides better resource utilization because there are typically a small number of threads that can be used to service a large number of connections. You can allow threads to be pooled to improve performance.

MAXDBAT constrains the total number of threads available to process remote SQL requests. If a request for a new connection to DB2 is received and MAXDBAT has been reached, the request is queued, waiting for a thread to become available to process the request. MAXDBAT generally should not be set higher than 500.[1] If virtual storage is constrained, do not set higher than 350.

Specify the maximum number of concurrent remote connections by setting the CONDBAT install parameter. This value must be greater than or equal to MAXDBAT. When a request to allocate a new connection to DB2 is received, and CONDBAT has been reached, the connection request is rejected. The value should be the largest number of pooled connections that would connect to the member at any point in time.

Active threads that have not committed their work in a timely fashion are canceled after IDTHTOIN expires; locks and cursors are released. Inactive connections and in-doubt threads are not subject to time-out. Threads are checked every two minutes to see if they have exceeded the time-out value. If the timeout value is less than two minutes, the thread might not be canceled if it has been inactive for more than the time-out value but less than two minutes.

The quicker DB2 can detect the communication error and return the thread to the pool, the lower the chance to reach MAXDBAT. In cases where the z/OS TCP/IP KeepAlive value in the TCP/IP configuration is not appropriate for the DB2 subsystem, you can use the TCPKPALV as an override.

In addition to defining the IP addresses to TCP/IP, the member and group DVIPA corresponding host names are required to be defined prior to starting DDF. DDF recovery processing may require the use of these names during in-doubt resolution after a subsystem failure. You define the host names by configuring the hlq.HOSTS.LOCAL data set, the /etc/hosts file in the hierarchical file system (HFS), or the domain name server (DNS).

Enabling sysplex workload balancing in IBM data server clients

In this section we examine the sysplex routing functions of DB2 Driver for JDBC and SQLJ and those of DB2 Connect.

WLM-based sysplex balancing

DB2 sysplex routing is performed by the JDBC and SQLJ Universal Driver and DB2 Connect Server gateway. Workload balancing is based on each target server's capacity to do work. There are no DB2 server configuration changes to enable WLM-based routing. When DDF is

[1] The total amount of storage must fit within the virtual private region. The primary limit for storage is the total number of threads (CTHREAD+MAXDBAT). Performance monitors like IBM Tivoli® Omegamon for DB2 Performance Expert calculate the amount of storage used per thread and thus the maximum number of threads. Configure to queue threads, rather than to abend the DB2 subsystem.

started, it registers the subsystem to WLM. When MAXDBAT is set to 0 or DDF is stopped, DDF deregisters from WLM to remove DB2 from the list of available servers. This routing method is based on z/OS Workload Manager (WLM) and its ability to gauge server load and provide a WLM recommendation. In this paradigm, WLM provides DB2 with a WLM recommendation for each target member called a WLM system weight. The WLM system weight is an indication of the target system's capacity for additional work. DB2 returns this list of available members and their corresponding WLM server weight on new connections or when an existing connection is reused by the concentrator. When a member has no capacity, WLM drops the member out of the server list.

DB2 uses the WLM IWMSRSRS service to determine where to route work using the SPECIFIC function. When the WLM SPECIFIC function is issued, the list of servers in the sysplex is associated with the DB2 location are returned along with a relative weighting for each member.

With the SPECIFIC function the following three additional factors are taken into account:

- The performance index that indicates the achievement of the WLM-defined goals of the work executed at a server. A server that permits the goals of its work requests to be achieved is preferred over one that does not or does not as effectively.
- The queue delays waiting for a database access thread are taken into account that the work is subject to, due to the queue times of the DB2-owned enclaves. A member with less-than-average queue times for its enclaves is preferred over one with higher-than-average queue times.
- In DB2 9 for z/OS, the health factor of each member is also taken into consideration. DB2 reports the health of a member using the WLM IWM4HLTH service.

DB2 Connect Server sysplex support for use with IBM Data Server Driver for ODBC, CLI, and .NET

While connection pooling and DB2 Connect Server sysplex workload balancing seem to have similarities, they differ in their implementation and address different issues. Connection pooling helps reduce the overhead of connections and handle connection volume. The DB2 connection concentrator helps increase the availability and scalability of your DB2 group by optimizing the use of connections in your data-sharing environment ensuring the highest availability possible.

When using connection pooling, the connection is only available for reuse after the application owning the connection issues a disconnect request. In many 2-tier client-server applications users do not disconnect for the duration of the workday. Likewise, most application servers in multi-tier applications establish database connections at server start-up and do not release these connections until the application server is shut down. In these environments, connection pooling will have little, if any, benefit. In Web and client-server environments, however, where the frequency of connections and disconnections is higher, connection pooling will produce significant performance benefits.

The connection concentrator uses connections to the DB2 group only for the duration of an SQL transaction while keeping user application's connections active. This allows for configurations where the number of DB2 threads and the resources they consume can be much smaller than if every application connection had its own active DB2 thread. When it comes to fail-safe operation and load balancing of workload, connection concentrator is clearly the only choice as it allows reallocation of work with every new transaction. Alternatively, connection pooling can only offer limited balancing and only at connect time.

The connection concentrator reduces the resources required by servers to support large numbers of workstation and web users. This function can increase the scalability of your DB2 group while also providing for fail-safe operation and transaction-level load balancing in a DB2 data-sharing environment.

The connection concentrator allows applications to stay connected without any resources being consumed on the DB2 host server. You can have thousands of users active in applications and only have a few threads active on the DB2 host server.

Connection concentrator takes the concept of an agent and splits it into two entities:

- Logical agent, which represents an application connection.
- Coordinating agent, which owns the connection to a DB2 subsystem.

When a new application attempts a connection to a DB2 location, it is assigned a logical agent. To send an SQL request to DB2, a coordinating agent is assigned which initiates a new transaction using an existing connection to the DB2 subsystem. DB2 Connect Server implements a scheduling algorithm that uses the WLM weight information. This information is provided by DB2 on the start of each new transaction. This information is used to distribute workload across members of a data-sharing group according to criteria set up in WLM. This allows DB2 Connect Server to transparently relocate work away from failed or overloaded members to members that are up and under-utilized.

DB2 Connect Server connection concentrator is activated when you set:

number of maximum logical agents (max_connections) higher than number of coordinating agents (max_coordagents)

The connection concentrator allows DB2 Connect Server to make a connection available to an application as soon as the application has finished the transaction and has not created any global resources that are required to persist across transactions. For example:

- Any WITH HOLD cursors not closed
- Any declared global temp tables not dropped
- Any application using packages bound with the KEEPDYNAMIC(YES) option

These features prevent the DB2 connection and associated DB2 thread from being used by other applications reducing the efficiency of the DB2 subsystem. If the transaction completes and no global resources are open, the DB2-associated connection and DB2 thread are available for use by any other application that needs to have a transaction executed.

There are a number of important restrictions to the use of the DB2 Connect Server concentrator. Refer to the DB2 Connect Server publication for the latest information.

Only inbound connections to the DB2 Connect Server using TCP/IP or Local (IPC) can take advantage of the connection concentrator. For XA tightly-coupled transaction support, all applications that participate in the same XA transaction must use the same DB2 Connect Server to connect to the DB2 group.

The configuration suggestions given below focus on reducing response time to application servers, avoiding extra workload when the system is busy and making better use of DB2 Connect Server (db2agent) resources. The concentrator and client reroute are suggested to be enabled on DB2 Connect Server.

The most critical factor is to make sure the DB2 Connect Server is large enough and has enough memory to handle the expected workload. The suggested configuration is provided as an illustration only and is workload dependent. The example is based on using an AIX® system with 4 GB of memory which is expected to handle 1000–1400 concurrent connections to the DB2 group. The number of db2agent processes needed on DB2 Connect Server depends on how many new connection requests are opened per minute and how soon the transactions are committed. In a typical 3-tier system, the expectation is 300–600 new connection requests per minute and 1000–1400 db2agent processes. To handle heavier-than-usual workload or the situation of one of z/OS member being down, MAXAGENTS must be set properly.

If the database connection fails, then error message SQL30081N is issued, and the connection dropped. If a further connection request is received for the same location, DB2 Connect Server does the following:

- The DB2 Connect Server tries the highest priority server from the cached list of addresses based on the WLM priority information last returned by the DB2 server. If this connection attempt fails, then the other addresses in the list are tried, in descending order of priority.
- If all other attempts to connect fail, then DB2 Connect Server retries the connection to location using the address contained in the cataloged node directory for the location.
- The **`db2pd`** command with the sysplex parameter (db2pd -sysplex) can be used for retrieving information about servers associated with a Sysplex environment.

General guidelines for configuring the ODBC, CLI, and .NET Drivers

Turn off query time-out (for CLI/ODBC set **`QueryTimeoutInterval = 0`** in db2cli.ini) or set it to an amount expected to for the largest SQL statement to process on DB2. As the Microsoft® ADO.NET applications have CommandTimout set to 30 seconds by default, and on DB2 client QueryTimeoutInterval is 5 seconds by default, DB2 client interrupts the running of a SQL statement which cannot be finished within 30 - 35 seconds.

When a subsystems system becomes slow or under stress, the interrupt request could cause DB2 Connect Server to create an additional db2agent process and may further degrade the DB2 Connect performance and lead to MAX_CONNECTIONS limit being hit.

A value of 0 for AutoCommit means that the application is responsible for issuing commits. A value of 1 means that each transaction has only one SQL statement if the connection is type 1.

If AutoCommit is enabled, DB2 Connect Server performs pooling and work load management on every transaction. If the concentrator is used, AutoCommit should be off. It is critical that applications explicitly commit the transactions or threads will remain active and the threads cannot be pooled for use by another connection.

If applications exist which do not issue commits, then the recommendation is to set AutoCommit to 1. For applications that manage their transactions, it is recommended the application turn off AutoCommit as a connection attribute. It is preferred that the application explicitly commits or rollback transactions and do not rely on AutoCommit setting.

General guidelines for configuring the DB2 Connect Server

To enable SYSPLEX routing:

1. Set the 6th positional parameter of the DCS directory entry representing the DB2 group to `,,,,,SYSPLEX`.
2. Catalog the node directory representing the DB2 group to a host name that resolves to the group distributed DVIPA or set it to the group distributed DVIPA.
3. Enable the DB2 Connect Server to kill the connection and not just rollback the executing SQL statement when an application interrupts an SQL request. The settings for the DB2

registry variable DB2CONNECT_DISCONNECT_ON_INTERRUPT takes precedence over the db2cli.ini Interrupt keyword setting of 1.

With the setting:

```
DB2CONNECT_DISCONNECT_ON_INTERRUPT=TRUE
```

DB2 Connect Server closes the connection and aborts the DB2 thread on the DB2 group if an interrupt request is received from a CLI client instead of interrupting the currently executing SQL statement.

For a busy system, the quicker DB2 Connect Server can detect a communication error and return the exception to the application, the lower the chance to reach max_coordagents which is the maximum number of coordinating agents that can exist at one time on a server node. There is one coordinating agent acquired for each local or remote application that connects to DB2 group.

The DB2TCP_CLIENT parameter settings should be reviewed since they control the time-out for both the TCP/IP RECV and CONNECT functions. The lower the value the faster the client will respond to connection or member failures:

- DB2TCP_CLIENT_RCVTIMEOUT and DB2TCP_CLIENT_CONTIMEOUT parameters set the timeout for TCP/IP RECV and CONNECT functions respectively.
- DB2TCP_CLIENT_CONTIMEOUT works only when client reroute is turned on. By default client reroute is enabled even if DB2_MAX_CLIENT_CONNRETRIES is not set.
- DB2TCP_CLIENT_RCVTIMEOUT should be no smaller than the largest expected time required for an SQL query to complete.

The recommended parameters in this case are:

```
DB2TCP_CLIENT_RCVTIMEOUT=60 (seconds)
DB2TCP_CLIENT_CONTIMEOUT=10 (seconds)
DB2_MAX_CLIENT_CONNRETRIES    - don't set it
DB2_CONNRETRIES_INTERVAL      - don't set it
```

DB2TCP_CLIENT_RCVTIMEOUT should be set to the largest time is expected for all SQL request to complete.

DB2TCP_CLIENT_CONTIMEOUT should be set based on the longest time needed to establish socket connection from DB2 Connect Server to the DB2 group. The default algorithm for client reroute, when DB2_MAX_CLIENT_CONNRETRIES is not set, is to try for 10 minutes before giving up. If client reroute is enabled then the timeout value is applied over the entire logical connection and not the individual TCP/IP CONNECT request. The DB2_MAX_CLIENT_CONNRETRIES should be set to 0 if applications can not tolerate the -30108 message.

When the DB2TCP_CLIENT_RCVTIMEOUT and DB2TCP_CLIENT_CONTIMEOUT parameters are set as recommended, the client reroute feature tries each server only once.

To illustrate a DB2 Connect Server system that usually has 1000–1400 concurrent connections to the DB2 group, NUM_POOLAGENTS being 1000 is good since it matches the workload. 1000 db2agent processes are pooled on DB2 Connect Server; therefore, most of time DB2 Connect Server doesn't need to create a new db2agent process to serve the new connect request.

Here are the recommended parameters for this example:

```
MAX_COORDAGENTS = MAXAGENTS (DB2 Connect pre V9.5) = 2000
(number of connections possible in application server's connection pool)
```

```
NUM_POOLAGENTS = 1000                        (Half of MAX_COORDAGENTS)
MAX_CONNECTIONS = 2001                       (MAX_COORDAGENTS+1)
```

The application support layer heap and client I/O block size configuration parameter configure the communication buffer size between the DB2 Connect Server pooled agent and the DB2 server thread. The heap buffer is allocated as shared memory by each database manager agent started. The I/O block size controls the amount of data returned on each query reply from DB2 for z/OS. To optimize the storage on DB2 for z/OS, set the ASLHEAPSZ and RQRIOBLKSZ as following:

```
ASLHEAPSZ=15
RQRIOBLKSZ=65536
```

IBM Data Server for JDBC and SQLJ sysplex support

Java applications that use the IBM Data Server Driver for JDBC and SQLJ (type 4 connectivity) to access a DB2 data sharing group using TCP/IP can exploit sysplex workload balancing built in the driver without the need to access DB2 through a DB2 Connect Server. The sysplex support and connection concentrator built into the Java driver provide the same availability and balancing features as in the DB2 Connect Server. This provides the simplest configuration and best performance because it eliminates the need to access the DB2 group through a gateway.

The driver uses the same WLM information provided to DB2 Connect Server to determine the data sharing member to which the next transaction should be routed. With sysplex workload balancing, DB2 and WLM ensures that work is distributed efficiently among members of the data sharing group and that work is transferred to another member of a data sharing group if one member has a failure.

The JDBC driver uses transport objects and a global transport objects pool to support the connection concentrator and sysplex workload balancing. There is one transport object for each physical connection to the database source. When you enable the connection concentrator and sysplex workload balancing, you set the maximum number of physical connections to the database source at any point in time by setting the maximum number of transport objects.

At the driver level, you set limits on the number of transport objects using the JDBC configuration properties.

At the connection level, you can use DataSource properties to enable and disable the JDBC connection concentrator and sysplex workload balancing and set limits on the number of transport objects. You can set these properties when you obtain a connection using the DataSource interface or the DriverManager interface.

You can monitor the global transport objects pool in either of the following ways:

- Using traces that you start using the JDBC configuration properties
- Using an application programming interface

General guidelines configuring JDBC and SQLJ

Set the JDBC driver configuration properties to enable the connection concentrator and sysplex workload balancing for all DataSource or Connection instances that are created under the driver. Set the configuration properties in a DB2JccConfiguration.properties file by following these steps:

- Create a DB2JccConfiguration.properties file or edit the existing DB2JccConfiguration.properties file.
- Using the above illustration (described in the “General guidelines for configuring DB2 Connect ServerDB2 Connect Server”), set the following configuration properties:

  ```
  db2.jcc.maxTransportObjects=max connections allowed
  db2.jcc.maxTransportObjectWaitTime=-1
  ```

  maxTransportObjects controls the number of connections to the server. Set the value to the total number of connections you want managed across the DB2 group.

  The number of connections per member will be managed by the driver using the WLM weights being returned from DB2.

  Set the following JDBC data source properties to enable sysplex workload balancing with the connection concentrator and sysplex:

  ```
  enableSysplexWLB = true
  enableConnectionConcentrator = true
  maxTransportObjects = max connections allowed
  ```

  If enableSysplexWLB is set to true, enableConnectionConcentrator defaults to true. In the WebSphere® Application Server administrative console, set the properties for the data source that your application uses to connect to the data source.

Configuring sysplex workload balancing for a subset of members

The use of the DB2 connection concentrator in conjunction with the sysplex distributor configured with DVIPA and automatic VIPA takeover can be isolated to a subset of members in a group. Load distribution for both transactions and connections are performed across the subset. If all members of the subset are down, connection failures will occur even if other members not in the subset are started and capable of processing work. By designating subsets of members, you can:

- Limit the members to which application servers can connect. System and database administrators might find this useful for any number of purposes.
- Ensure that initial connections are established only with members that belong to the specified subset. Without subsets, requesters can make initial and subsequent connections to any member of the data sharing group.
- Provide requesters with information about only those members in the subset. With subsets, a member that receives an initial connection request can return to the requester a list of members that are currently active, able to perform work, and represented by the location alias.

Each DB2 member configures its alias port number which is unique across all members participating in the subset.

In Figure 2 on page 12, the location and location alias is represented by ports 446 and 5446 for instances DB2A and DB2B. DB2C is not part of the subset, so no connections accessing DB2 using the alias will route to it.

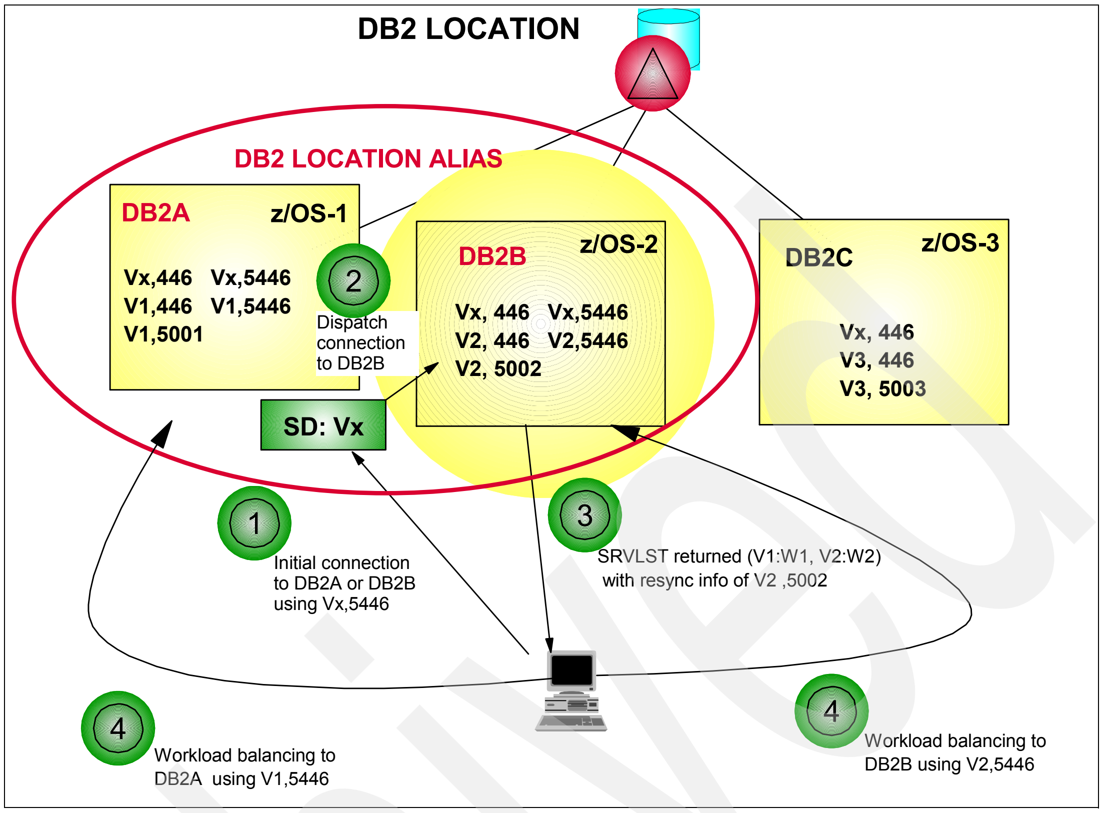
DB2 LOCATION
DB2 LOCATION ALIAS
DB2A
z/OS-1
Vx,446 Vx,5446
V1,446 V1,5446
V1,5001
2
Dispatch connection to DB2B
DB2B
z/OS-2
Vx, 446 Vx,5446
V2, 446 V2,5446
V2, 5002
DB2C
z/OS-3
Vx, 446
V3, 446
V3, 5003
SD: Vx
1
Initial connection to DB2A or DB2B using Vx,5446
3
SRVLST returned (V1:W1, V2:W2) with resync info of V2 ,5002
4
Workload balancing to DB2A using V1,5446
4
Workload balancing to DB2B using V2,5446

Figure 2 DB2 instances and addresses for subset members DB2A and DB2B

Example 2 shows the pertinent port and VIPA definitions for the three DB2 instances.

The initial contact point to the DB2 subset group is with the location alias port and the distributing DVIPA bound to each member DVIPA in the subset. Location alias and its port are configured in the DB2 boot strap data set. The following DSNJU003 control statement is used to configure an alias:

```
DDF ALIAS=DB2LOCATIONALIAS:5446
```

The particular DB2 member of the subset at any point in time is thus transparent to the database application servers.

Example 2 Port and DVIPA definitions for DB2 instances subset

```
DB2A                                  DB2B

Port                                  Port
  446  tcp db2adist shareport bind Vx   446  tcp db2adist shareport bind Vx
  5446 tcp db2adist shareport bind Vx   5446 tcp db2adist shareport bind Vx
  5001 tcp db2adist bind V1             5001 tcp db2adist bind V1
  446  tcp db2bdist shareport bind Vx   446  tcp db2bdist shareport bind Vx
  5446 tcp db2bdist shareport bind Vx   5446 tcp db2bdist shareport bind Vx
  5002 tcp db2bdist bind V2              5002 tcp db2bdist bind V2
  446  tcp db2cdist shareport bind Vx    446  tcp db2cdist shareport bind Vx
  5003 tcp db2cdist bind V3              5003 tcp db2cdist bind V3
VipaDynamic                           VipaDynamic
  VipaRange Define 255.255.255.255 V1   VipaRange Define 255.255.255.255 V1
  VipaRange Define 255.255.255.255 V2   VipaRange Define 255.255.255.255 V2
  VipaRange Define 255.255.255.255 V3   VipaRange Define 255.255.255.255 V3
  VipaDefine 255.255.255.255 Vx          VipaBackup 1 Vx
  VipaDistribute Define Vx               VipaDistribute Define Vx
       Port 446 5446 DestIP all               Port 446 5446 DestIP all
```

```
EndVipaDynamic                       EndVipaDynamic
```

DB2C

```
Port
  446  tcp db2adist shareport bind Vx
  5446 tcp db2adist shareport bind Vx
  5001 tcp db2adist bind V1
  446  tcp db2bdist shareport bind Vx
  5446 tcp db2bdist shareport bind Vx
  5002 tcp db2bdist bind V2
  446  tcp db2cdist shareport bind Vx
  5003 tcp db2cdist bind V3
VipaDynamic
  VipaRange Define 255.255.255.255 V1
  VipaRange Define 255.255.255.255 V2
  VipaRange Define 255.255.255.255 V3
  VipaBackup 2 Vx
  VipaDistribute Define Vx
       Port 446 5446 DestIP all
EndVipaDynamic
```

In Figure 2 on page 12, the first socket representing the location is bound to the distributed DVIPA, Vx, on port 446, the next socket representing the subset is bound to the distributed DVIPA, Vx, on alias port 5446 and then the alias and resync ports are bound to each unique IP address for each DB2 member in the subset (address V1:port 5001 and address V1:port 5446 for DB2A, address V2:port 5002 and addressV2:port 5446 and for DB2B, and address V3:port 5003 for DB2C).

The initial connect using the distributed DVIPA and alias port may go to any of the DB2 instances in the subset. Once the initial connection is established, DB2 returns to application server a list of IP addresses, one for each DB2 instance participating in the subset and its WLM weight. Because the initial contact listening socket is listening on the location/group distributed DVIPA, DB2 opens an additional listening socket on the same port as the location listening socket, but bound to the member-specific DVIPA. This is illustrated in the figure as the middle socket pair (address V1:port 5446 and address V2:port 5446) for the two DB2 instances participating in the subset) to allow concentrator to route to specific members in subset leveraging z/OS TCP/IP Dynamic VIPAs and sysplex distributor for both the subset and group.

To show how this works in Figure 2 on page 12, the DRDA client will connect to DB2 using the alias distributed DVIPA and port 5446. Sysplex distributor (running on the same image as DB2A) will decide where to route the request based on available capacity, and in this scenario, the request actually goes to DB2B. DB2B will return to the client its resync info (address V2: port 5002). For database application servers where the connection concentrator is not configured, this is all that is needed. Each unit of work results in a new connection to the DB2 location DVIPA, which is distributed to the DB2 on the operating system image with the most apparent capacity and other information considered by sysplex distributor in making the decision.

DB2 returns a list of server addresses and associated server weight on a connection boundary or when a connection is reused by another application, termed the server list (SRVLST) in flow number 3 in Figure 2 on page 12. Once the database client receives the server list, it balances subsequent connections and transaction among the available subset of servers using the WLM weights returned for each server in the list, illustrated as the two flows numbered 4 in the diagram.

These are general guidelines. For more information about how to configure DB2 and TCP/IP in this data sharing environment, see *DB2 Version 9.1 for z/OS Data Sharing: Planning and Administration,* SC18-9845.

To configure a subset for member-specific access, use the alias-port parameter of the ALIAS option of the DSNJU003 (change log inventory) utility. By following the location alias with a TCP/IP port number (a decimal number between 1 and 65534), you indicate that the DB2 subsystem is a member of the specified data sharing group subset. When DDF starts, it performs the following tasks:

- Registers any subset location aliases with z/OS Workload Manager. The list of members in the subset is managed automatically by z/OS.
- Adds the TCP/IP port numbers of the subset location aliases to a TCP/IP SELECT socket call for the SQL request listener. Doing so enables the sysplex distributor to send requests that are intended for subset members to only those members that belong to the subset. It also enables members of the subset to respond to those requests.

The team that wrote this IBM Redpaper

This Redpaper was produced by a DB2 developer from Silicon Valley Laboratory working with the International Technical Support Organization, San Jose Center.

James Pickel is an IBM Senior Technical Staff Member, DB2 for z/OS Development, Silicon Valley Lab. Jim received his BA in Computer Science from UC Berkeley and has been working in DB2 for z/OS Development since Version 1. He is one of the original designer of the Distributed Data Facility, an architect of DRDA, and a member of the Open Group Database Interoperability Group. Jim's primary focus is distributed database and security technologies.

Acknowledgements

Many thanks are owed to the following people for their contributions to this project:

Paolo Bruni
Roger Miller
IBM Silicon Valley Lab

References

- *DB2 Version 9.1 for z/OS Data Sharing: Planning and Administration,* SC18-9845
- *TCP/IP in a Sysplex*, SG24-5235
- *Distributed Functions of DB2 for z/OS and OS/390*, SG24-6952

Notices

This document REDP-4449-00 was created or updated on May 12, 2010.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Redpaper™

Trademarks