



Explore the IBM Cognos Dispatcher

IBM Cognos BI 10.2.2

Business Analytics software

© 2015 IBM Corporation



Objectives

- At the end of this module, you should be able to:
 - describe IBM Cognos Dispatcher
 - describe request routing and the routing process
 - describe Content Manager Cache Service




Business Analytics software

IBM

What is a Thread?

- unit of processing time scheduled by the operating system

© 2015 IBM Corporation



Threads, or threads of execution, are the smallest unit of processing time that is scheduled by the operating system (OS). You can have more than one thread of execution in a process. Multiple threads can share resources, as subsets of a process, but processes handle resources differently than threads and are typically independent.

Multi-threading can be handled by the OS distributing to different Central Processing Units (CPU). Programs must be carefully designed in such a way that all the threads can run at the same time without interfering with each other.

What is a Servlet?

- multi-threaded program
- handles HTTP requests sent to it in one or multiple threads
- responds by HTTP

© 2015 IBM Corporation



Servlets will use threads within the application server process scope, proportional to the number of requests sent to it. Servlets are the application tier equivalent to CGI programs in the Web tier, as they typically do not implement business logic.

A servlet is a Java program which runs in a specific environment, the servlet container as defined by the Java EE framework. The Java EE framework is typically implemented by a Java application server like WebSphere, or JBOSS.

A servlet cannot be run stand-alone but rather runs within the process of the container, and serves the single purpose to accept a request received by the servlet container and passed to it.

What is a Process?

- an instance of a computer program that is being executed
- executes instructions

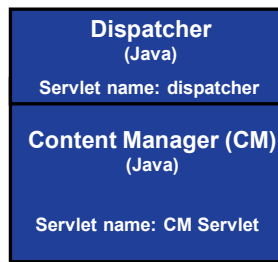
© 2015 IBM Corporation



While a program itself is a passive collection of instructions, a process is something which executes those instructions. Several processes can be associated with the same program, although each would execute independently. A single process is always assigned to a single physical CPU.

What is the IBM Cognos Dispatcher?

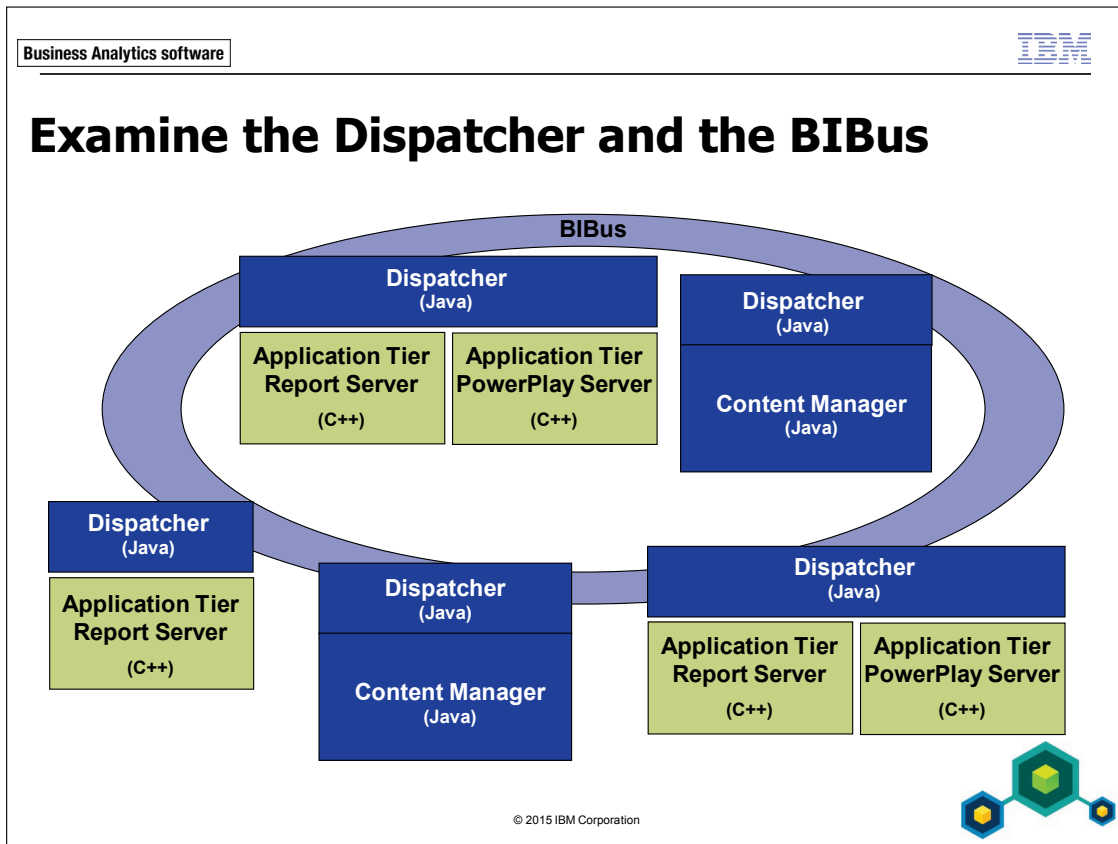
- Java servlet
- multi-threaded
- available in Application Tier and Content Manager install components
- interface to BIBus for local IBM Cognos services



© 2015 IBM Corporation

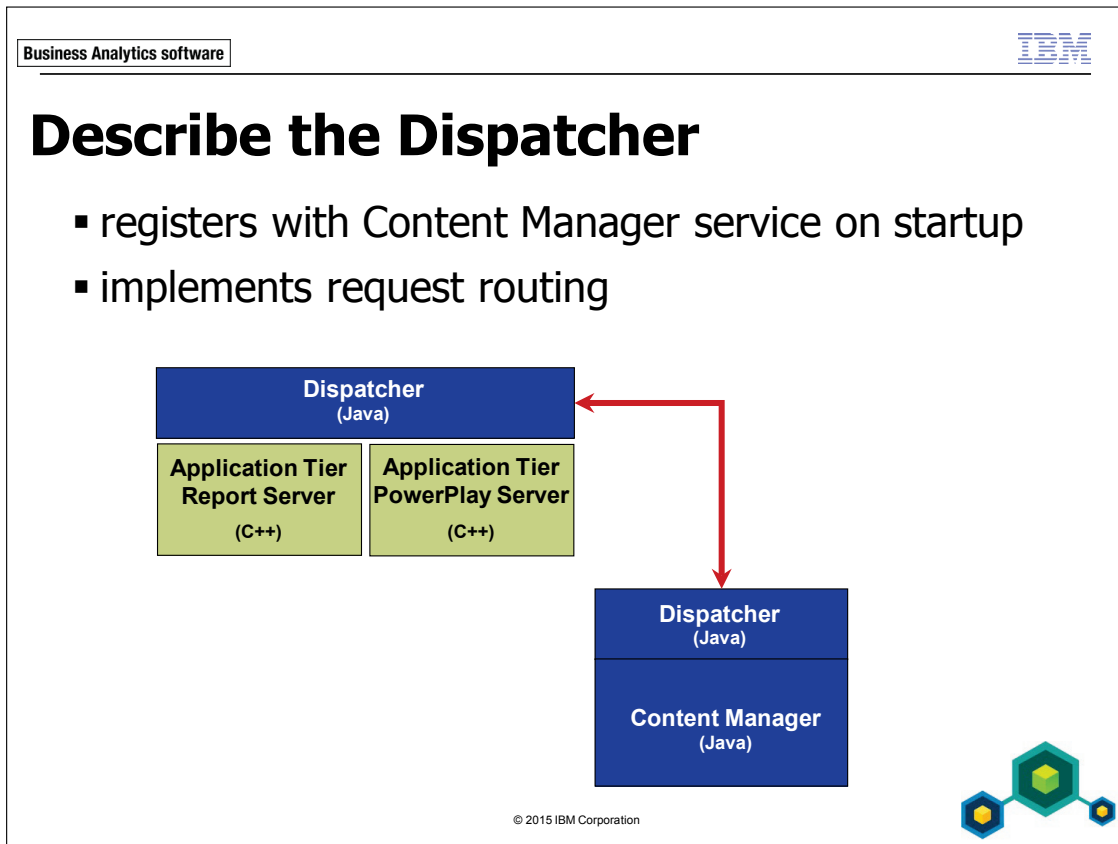


The IBM Cognos dispatcher hosts IBM Cognos services exposed by the components of the installed service. The dispatcher starts all local services as configured in IBM Cognos Configuration, with the exception of the Content Manager service, which is separate.



Only dispatchers sit on the BIBus. All inter-system communication travels this bus, although services require a dispatcher to gain access to the BIBus. In other words, all of the installations use the BIBus to communicate with each other, and all communication is done through the dispatchers.

The diagram shows multiple installed instances with different sets of install components.



As the dispatcher registers with the Content Manager service, it enables built-in failover handling, and the dispatcher is assigned a unique ID. The registration can be revoked through IBM Cognos Administration, as in the case of a server name change during migration.

The dispatcher implements request routing as an internal service, and then forwards the requests to locally hosted service instances. The dispatcher can also forward requests to other dispatchers on the BIBus hosting remote service instances.

For the dispatcher to route requests to remote dispatchers through the BIBus, each dispatcher requires knowledge about other dispatchers in the system, the services hosted by the other dispatchers, and the status of the services on the other dispatchers. You will explore this in Demo 1: Review the Output of p2plbDiag.

If a dispatcher fails or is unavailable, requests for that dispatcher are routed to the next available dispatcher until the failed dispatcher updates its status to active again. For this to work dispatchers require knowledge about other dispatchers in the system, and a mechanism to determine whether a dispatcher is available or down; this is done by registration in the content store through the Content Manager service.

What is the Purpose of Request Routing?

- delivers client requests to the correct IBM Cognos service as quickly as possible
- influenced by:
 - load balancing
 - external factors
 - configuration items

© 2015 IBM Corporation



Request routing is the task of delivering client requests, and is influenced by many factors and underlying concepts.

Load balancing may be affected by the targeted service or the number of available instances of that service. External factors such as system load, dispatcher hardware resources, or external routing software or hardware, may influence the request routing of a dispatcher. Advanced routing configuration based on user identity or packages used can impact request routing also.

Describe Request Routing Concepts

- role of gateway
- cluster information
- front dispatchers
- request queue
- conversations
- request affinity
- customizations

© 2015 IBM Corporation



Over the next few pages, the request routing concepts listed in the slide will be presented.

Describe the Role of the Gateway

- relays request to a single dispatcher
- does not load balance or route
- list of dispatchers in gateway configuration is for fail-over only
- pings all configured dispatchers every 30 seconds or on forward failure

© 2015 IBM Corporation



The dispatcher list for a gateway is specified in IBM Cognos Configuration. The gateway will work down the list in the order specified, so if the first dispatcher is not available, the gateway tries the second dispatcher, and so on.

If the gateway pings a dispatcher, and encounters a second failed attempt to connect, that dispatcher is marked as unavailable for this cycle.

Describe Dispatcher Cluster Information

- dispatcher cluster information includes:
 - list of dispatchers registered to the system
 - server groups they are assigned to
 - IBM Cognos services hosted by each dispatcher and the running state of those services
- the dispatcher cluster information provides each dispatcher with a landscape of the system

© 2015 IBM Corporation



As part of their start-up, each dispatcher registers itself with Content Manager by querying the Content Manager service for dispatcher cluster information, and then the dispatcher joins the system. With the dispatcher cluster information, a dispatcher can eventually forward requests to remote dispatchers and services that are based on it.

The Content Manager service is the keeper and creator of the dispatcher cluster information.

Cluster refers to an entity where several servers are joined together. When a dispatcher registers with Content Manager, they are essentially reporting their part that Content Manager will put into the larger system.

Server group is a property of a dispatcher, and will be mentioned later in this module.

Explore Dispatcher Cluster Information Details

- updated automatically
- dispatchers query CM service for the cluster information
- can take up to one minute to be propagated to all dispatchers in the IBM Cognos system
- accessible through:
 - http://<INTERNAL_DISP_URI>/p2plbDiag
 - any dispatcher on the IBM Cognos system

© 2015 IBM Corporation



The dispatcher cluster information updates automatically when a new dispatcher is added to the system or removed from the system. Dispatchers will query the CM service at regular intervals to signal that they are active and to update their local cluster information.

INTERNAL_DISP_URI refers to the internal dispatcher URI from IBM Cognos Configuration. In a default install this will be <http://localhost:9300/p2pd/servlet/dispatch/p2plbDiag>.

p2plbDiag shows information about a specific dispatcher including GUID and server group, all known dispatchers, and load balancing statistics.

Demo 1: Review the Output of p2plbDiag

At the beginning of this workshop, only the IBM Cognos Full:9315 dispatcher needs to be running. Stop the IBM Cognos DispCM:9320 dispatcher service if it is running, before you begin this demo.

Purpose:

You want to use p2plbDiag to review information about a specific dispatcher including GUID and server group, all known dispatchers, and load balancing statistics.

Task 1. Access p2plbDiag and log on.

1. Launch **Internet Explorer**, in the **Address** box, type **http://vclassbase:9315/p2pd/servlet/dispatch/p2plbDiag**, and then press **Enter**.
2. Log in to the **LDAP_Dev** namespace with **admin/Education1** credentials. If you are prompted to turn AutoComplete on, click **No**.

Task 2. Review the results of p2plbDiag and the current system.

1. Use the results to answer the following:
 - Dispatcher:
 - GUID:
 - Content Manager:
 - Server Group:
 - Load Balancing Mode:
 - Disabled Services:
 - Are there multiple dispatchers available?

A section of the result appears similar to the following:

```
This dispatcher is: vclassbase:9315/p2pd/servlet/dispatch
GUID=2015-03-16-20.22.44.255878
Using CM: vclassbase:9315/p2pd/servlet
Current time: Mar 18, 2015 3:36:31 PM EDT
Configured dispatchers and services:
this dispatcher is : "/configuration/dispatcher[@name='http://vclassbase:9315/p2pd']"
this dispatcher is in serverGroup : "Group 64"

All known dispatchers:
  Dispatcher: /configuration/dispatcher[@name='http://vclassbase:9315/p2pd']
    name: vclassbase:9315/p2pd
    dispatcherID: 2015-03-16-20.22.44.255878
    capacity: 1.0
    SSL: false
    serverGroup: Group 64
    loadBalancingMode: weightedRoundRobin
    edition: 10.2.6100.68
    Services:
      Service name: repositoryService disabled? false
      Service name: dispatcher disabled? false
      Service name: contentManagerCacheService disabled? false
      Service name: reportDataService disabled? false
      Service name: dispatcherCacheService disabled? false
      Service name: eventManagementService disabled? false
      Service name: dimensionManagementService disabled? false
```

The information here should match up with your configuration settings in IBM Cognos Administration. You can use this as a quick way to find out if the Dispatcher is available; you can also determine this in IBM Cognos Administration.

If you see two dispatchers (http://vclassbase:9315/p2pd and http://vclassbase:9320/p2pd) listed on this page, you may need to:

- a) Close the Web browser.
- b) Shut down the IBM Cognos DispCM:9320 service if it is not already shut down.
- c) Restart the IBM Cognos Full:9315 service.
- d) Wait 60 seconds.
- e) Reopen the Web browser and browse to **http://vclassbase:9315/p2pd/servlet/dispatch/p2plbDiag** again (logging on as admin/Education1 when prompted).

Now that you can see one Dispatcher is registered and available in the environment, what will happen if another Dispatcher is running? Will the registered Dispatcher display in p2plbDiag?

2. Close the browser window.

Task 3. Start the second dispatcher and observe the result.

1. In the **Taskbar**, click the **Services** icon, and then start the **IBM Cognos DispCM:9320** service.

It will take a few minutes to start the service, and you may get a message indicating that the service did not start in a timely fashion (click OK to dismiss this message). Wait until this instance has fully started (you may do a refresh of the display every two minutes) before proceeding to the next step.

2. When the service has started successfully, in **Internet Explorer**, go to **http://vclassbase:9315/p2pd/servlet/dispatch/p2plbDiag**, and log in (if prompted) to the **LDAP_Dev** namespace using **admin/Education1** credentials.

If you are prompted to turn AutoComplete on, click No. A section of the result appears similar to the following:

```
This dispatcher is: vclassbase:9315/p2pd/servlet/dispatch
GUID=2015-04-01-21.08.14.757907
Using CM: vclassbase:9320/p2pd/servlet
Current time: Apr 7, 2015 11:14:50 AM EDT
Configured dispatchers and services:
this dispatcher is : "/configuration/dispatcher[@name='http://vclassbase:9315/p2pd']"
this dispatcher is in serverGroup : "Group 64"

All known dispatchers:
  Dispatcher: /configuration/dispatcher[@name='http://vclassbase:9320/p2pd']
    name: vclassbase:9320/p2pd
    dispatcherID: 2015-04-02-20.26.05.342844
    capacity: 1.0
    SSL: false
    serverGroup: Group 32
    loadBalancingMode: weightedRoundRobin
    edition: 10.2.6100.68
    Services:
      Service name: repositoryService disabled? false
      Service name: dispatcher disabled? false
      Service name: contentManagerCacheService disabled? false
```

3. Use the result to answer the following:
 - What is the result of the number of available dispatchers?
 - What does this say about the output of p2plbDiag and registered dispatchers?

The dispatchers displayed in p2plbDiag are only active dispatchers. Dispatchers that are registered to the Content Manager service are not displayed in p2plbDiag unless they are currently available.

4. Close the browser window, and the **Services** window.

Results:

You used p2plbDiag to review information about a specific dispatcher, including GUID and server group, all known dispatchers, and load balancing statistics.

Describe Front Dispatchers

- first dispatcher to touch on a request
- may require specific IBM Cognos services to be available to handle some requests
- multiple front dispatchers can exist if there are multiple gateways
- start request routing

© 2015 IBM Corporation



The list of dispatcher URIs for a gateway configuration should only contain those dispatchers that are capable of acting as a front dispatcher that can offer the required services. All gateways could have the same front dispatcher, but that would always funnel requests to a single dispatcher and hinder efficient failover as well as introduce a single point of failure. A front dispatcher is a dispatcher that is referred to by a gateway and one that interacts directly with external clients such as SDK apps.

The theoretical reason that this is important is that certain services must be enabled on a Front Dispatcher. In practice this is not an issue because all services are turned on by default after an install. However, one of these required services is the `DispatcherService()` and a Content Manager only install does not enable the `DispatcherService()` so what this means is do not have a Gateway use a dispatcher from a CM-only install. This is intentional, and should not be changed under normal circumstances. A front dispatcher will need to run the Presentation service for SSO (Single Signon).

Describe the Dispatcher Request Queue

- each dispatcher manages a queue for requests
- request queues develop due to:
 - demand being greater than resource provided
 - time intensive requests
 - inappropriate tuning of capacity and process allocation across dispatchers

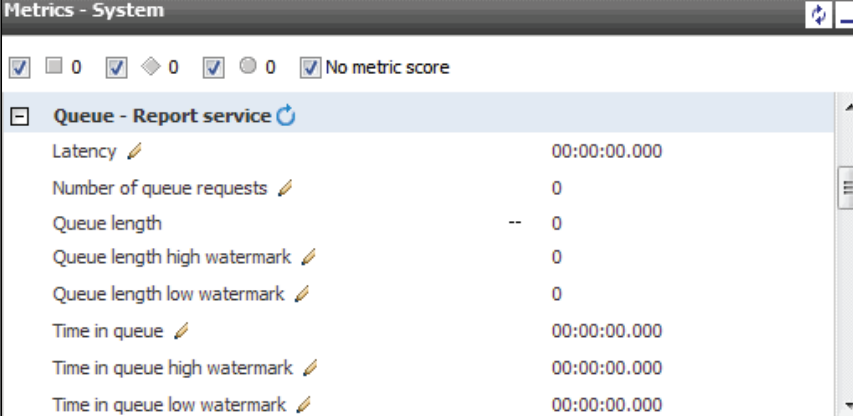
© 2015 IBM Corporation



If the number of execution type requests for a certain service is greater than the number of requests which can be handled at a time by that service at the given dispatcher, then a dispatcher request queue is developed. This is also the case for long running database or cube queries that are time intensive, or if capacity tuning is not done properly.

Examine the Details of the Request Queue

- latency
- monitored in IBM Cognos Administration tool



Queue - Report service	
Latency	00:00:00.000
Number of queue requests	0
Queue length	0
Queue length high watermark	0
Queue length low watermark	0
Time in queue	00:00:00.000
Time in queue high watermark	00:00:00.000
Time in queue low watermark	00:00:00.000

© 2015 IBM Corporation

The amount of time a request has spent in the queue is called latency.

Note: The values displayed on the service monitoring page in IBM Cognos Administration are 30 to 60 seconds behind the actual activity. This is not real time monitoring.

Considerations for the Request Queue

- resolution of queues may not be required
- queues should be managed to avoid bigger problems later

© 2015 IBM Corporation



Resolution of queues may not be required if performance is within usage expectations. Examples of specific resolutions to manage queues include:

- add service instances (a benefit of SOA)
- provide more resources to process more requests for that particular local service
- in a distributed environment, monitor and tune process capacity for optimal request dispatching
- design queries and reports for better performance
- scale the system up or out by adding more hardware capacity

It is possible to have too many affinity connections, such as if there are too many report servers spending too much time figuring out what to do instead of just doing it.

For more information, refer to the IBM Cognos System Management Methodology (SMM) on developerWorks. At time of printing, this is available on developerWorks on the IBM Web site at http://www.ibm.com/developerworks/data/library/cognos/infrastructure/cognos_specific/page592.html. This is a group of documents and examples provided to administrators for information and techniques in using IBM Cognos Administration.

What is a Conversation?

- sequence of requests sent from the client to a specific service
- example of conversation, running an HTML report:
 - database connectivity prompts
 - parameter selection prompts
 - query execution status updates
 - paging through HTML output

© 2015 IBM Corporation



In many situations, an operation is processed using a sequence of requests sent from the client to a specific service, and it is this sequence of requests known as a conversation.

Explore the Benefits of Conversations

- avoid browser and network timeout
- allows user actions during execution (such as cancel)
- allows detection of server failure and recovery
- allows detection of client abandonment

© 2015 IBM Corporation



The system operates most efficiently when all of the requests in a conversation are handled by the very same instance of the targeted service. Request routing must support this as a priority.

Examine the Usual Conversation

- starts with primary request such as `run()`
- subsequent requests such as `nextPage()` within the same conversation are secondary requests
- if preferred service instance unavailable, secondary request can be fulfilled by any other instance of the targeted service

© 2015 IBM Corporation



A primary request, such as `run()`, can be fulfilled equally well by any instance of the requested service; these are known as low-affinity requests.

Secondary requests can be fulfilled best by the same instance of the targeted service that handled the previous primary or secondary request; these are known as high-affinity requests.

If the preferred service instance is not available, routing to the non-preferred service instance is known as fail-over. Requests fulfilled by the non-preferred service instance will run slower since the service must first recreate the conversation state.

What are Asynchronous Conversations?

- if primary or secondary request cannot complete within the Primary Wait Threshold, the service returns a status of Working in the response to the client, and processes the request asynchronously
- client then issues heartbeat requests such as wait() until the service has completed work on previous primary or secondary request
- dispatcher cancels asynchronous conversation if no heartbeats are received within Secondary Wait Threshold

© 2015 IBM Corporation



When the request is processed asynchronously, it is commonly referred to as going asynchronous. The default length of the Primary Wait Threshold is 7 seconds.

Heartbeats must be served by the very same service instance that still works the primary or secondary request (no fail-over is possible); these requests are known as absolute-affinity requests.

The dispatcher will cancel an asynchronous conversation if a heartbeat is not received from the client within the Secondary Wait Threshold, such as if a user has abandoned the conversation, or cancelled the request by calling cancel(). These requests are known as control-affinity requests, and are like absolute-affinity requests but are not queued by the dispatcher. The default length of the Secondary Wait Threshold is 30 seconds.

There is nothing that the end user should need to do when a conversation ends for whatever reason. The dispatcher and the other services are aware of abandoned/cancelled/completed conversations and will do whatever they need to do to release resources that they have allocated to that conversation. No cleanup needs to be done for abandoned conversations.

What are Grouped Conversations?

- conversations can be grouped
 - multiple runSpecification() requests may be issued to test reports
 - can leverage Report Server caching

© 2015 IBM Corporation



If during Report Studio authoring sessions, many runSpecification() requests are issued to test reports, you can leverage report server caching if all of these primary requests are sent to the same instance of ReportService. These requests are known as session-affinity requests.

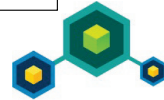
You can disable session caching at the server level or at the package or report level. Refer to the *IBM Cognos Business Intelligence Administration and Security Guide 10.2.2* for more information.

Describe Request Affinity

- requests handled by the dispatcher
- a request has affinity or is subject to load balancing
- primary requests are low affinity
- secondary requests usually have an affinity

```
<bus:conversationContext xsi:type="bus:conversationContext">
<id>yMdhMlysGsGhw9hqqqd24hhCdMvqClvv9hdd9G99</id>
<nodeID>2015-03-19_14:26:53.641_165</nodeID>
<processID>1</processID>
<affinityStrength>5000</affinityStrength>
<status>complete</status>
</bus:conversationContext>
```

© 2015 IBM Corporation



Conversations imply affinity, a measure of how important it is to route a request to a specific instance of an IBM Cognos service and hence to the dispatcher hosting that instance.

Request affinity is denoted in the suffix of the SOAPAction header, which is set and handled internally.

The `<bus:conversationContext>` element of a BIBus SOAP message contains information about the affinity request such as the dispatcher ID, affinity strength and status of the conversation.

The example in the slide is a sample conversationContext header.

Request affinity is different from server affinity, which will be covered later for BIBusTKServerMain (Report Server). Request affinity relates to requests being handled by the dispatcher. The affinity concept contradicts load balancing; a request either has affinity or it is subject to load balancing. Request affinity is handled internally and cannot be manipulated.

Define Request Affinity Assignments

- (none) or low affinity
- .high
- .session
- .absolute
- .control

© 2015 IBM Corporation



In the case of the assignment of no or low affinity, the dispatcher is free to route the request to any running instance of a service.

With high affinity, the dispatcher should try to route the request to the dispatcher that is specified by the node ID. If the requested dispatcher is not available then this request is treated as having no affinity. The node ID in the BIBusHeader is mandatory.

Session affinity is the same as high affinity, except the node ID in the BIBusHeader is optional. If no node ID is specified, this request is treated as having no affinity.

For absolute affinity, the dispatcher must route the request to the dispatcher specified by the node ID. If the specified dispatcher is not available, the request will fail and a SOAP Fault is returned. The dispatcher can queue these requests.

Control affinity is the same as absolute affinity, with the difference that it is never queued. This is reserved for system operations such as `cancel()`, and these requests cannot be queued at the dispatcher.

Review Request Routing

- only dispatchers implement request routing
- requests handled in conversations
 - primary and secondary requests
 - secondary requests usually have an affinity
- dispatchers can queue requests except control affinity requests
- request routing can be customized

© 2015 IBM Corporation



Gateways do not contribute to request routing, only dispatchers implement request routing. Dispatchers queue requests if they cannot be processed fast enough.

Consider Customization

- improve performance
- implement specific requirements
- implement by runtime configuration
- use care when customizing request routing

© 2015 IBM Corporation



Customizations can be implemented by runtime configuration, and include load balancing modes and advanced routing.

Customizing request routing is delicate, as a configuration based on uninformed decisions can hinder performance. Implementation complexity differs from system to system and should be evaluated beforehand, as there may be other approaches to consider.

Business Analytics software

What Load Balancing Modes are Available?

- Weighted Round Robin
- Cluster Compatible

Set properties - http://vclassbase:9315/p2pd
Help

General
Settings
Permissions

Specify the configuration settings for this entry. By default, an entry acquires its configuration settings from a parent. You can override those settings with the settings set explicitly for this entry.

Category: Tuning

Entries: 1 - 15

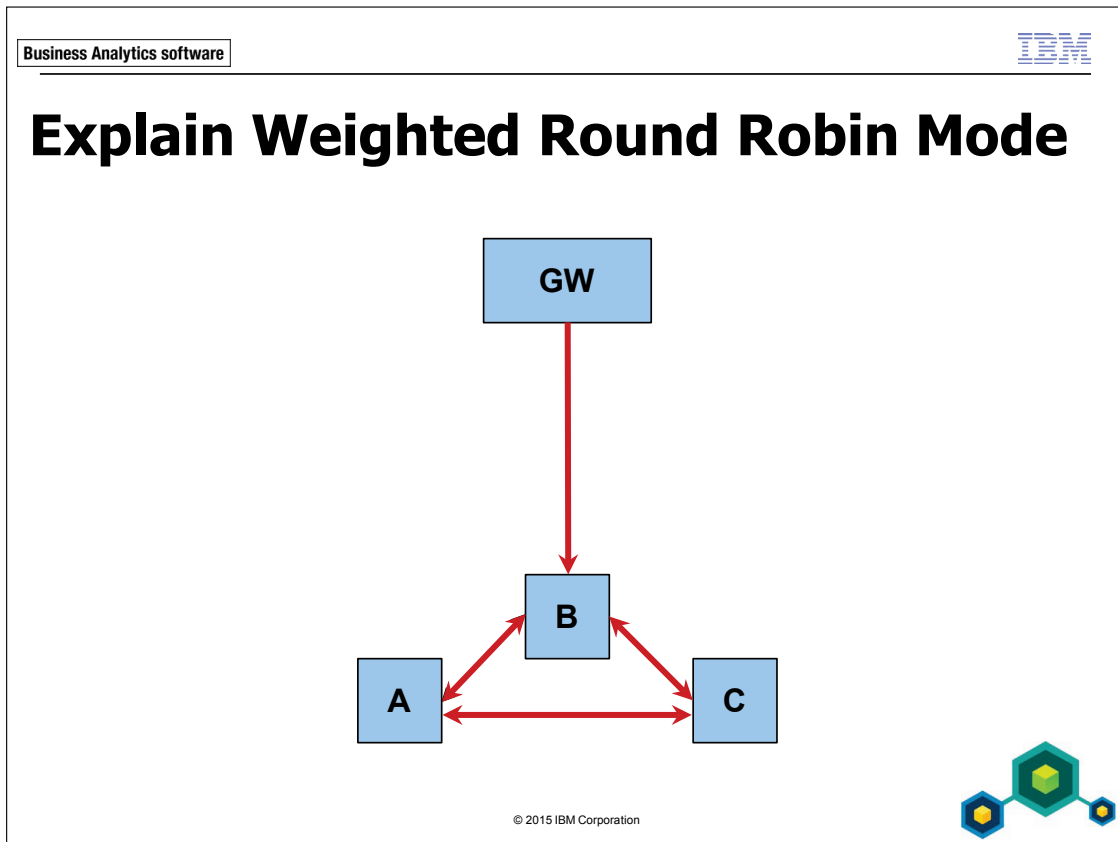
	Category	Name	Value	Acquired
<input type="checkbox"/>	Tuning	Processing capacity	1.0	Yes
<input type="checkbox"/>	Tuning	Load balancing mode	<div style="border: 1px solid black; padding: 2px;"> Weighted Round Robin Weighted Round Robin Cluster Compatible </div>	Yes
<input type="checkbox"/>	Tuning	Server group		No

© 2015 IBM Corporation

Only primary requests are subject to load balancing.

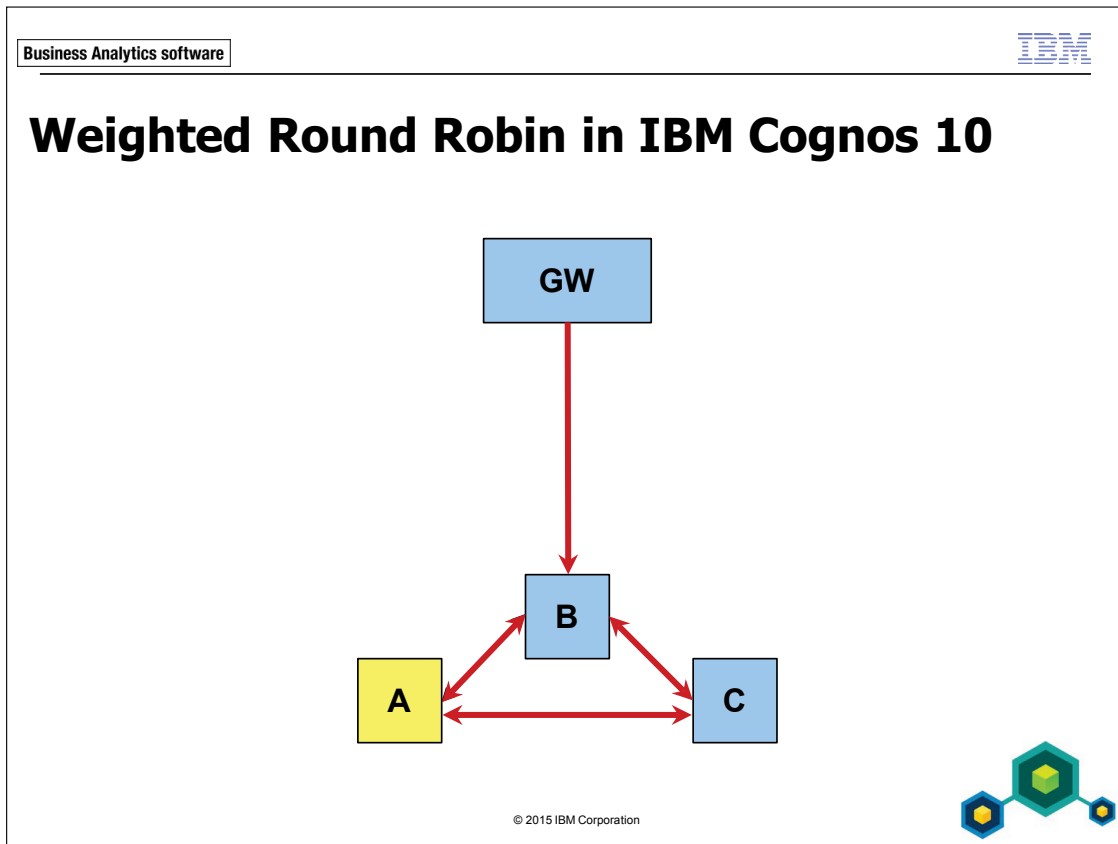
Load balancing is set in the Load balancing mode property of a dispatcher. All dispatchers in a single system must have the same setting.

The Processing capacity property is located in IBM Cognos Administration, on the Configuration tab, in the properties of the target dispatcher. The Processing capacity value is arbitrarily assigned based on hardware resources of the dispatcher host system. Although fractional numbers can be used they should be avoided.



Weighted round robin (WRR) is the default algorithm where requests are distributed equally between all dispatchers, one after another. The weighted variant allows a dispatcher to have higher capacity than another dispatcher. The processing capacity setting of a dispatcher determines the weight, with the default is set as 1.0.

If Dispatcher A has a weight of 2 and Dispatcher B has a weight of 1, Dispatcher A would receive twice as many requests as Dispatcher B. This method is used to account for different hardware resources in a multi-server environment.



In IBM Cognos 10 BI there is an additional factor to weighted round robin: In-Progress Request Factor (IPRF). This is configured as an advanced property of a dispatcher and applies to all services or particular service(s) of a dispatcher. IPRF specifies the factor by which the number of requests (of a particular service) that are executing or waiting at a particular dispatcher or dispatcher queue should influence the weighted round robin. This helps to prevent overloading slow dispatchers and service instances. IPRF allows adaptive influence to the algorithm and automatically adjusts to the system load.

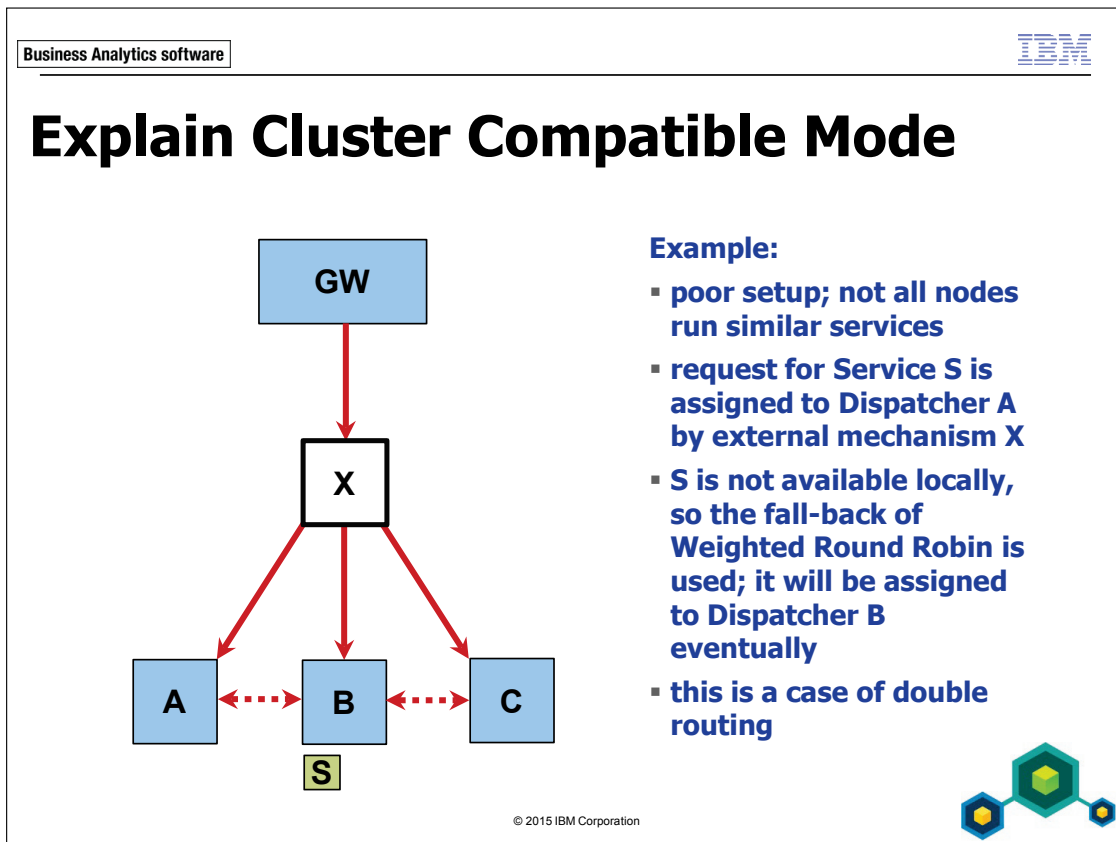
In example on the slide, assume that the ReportService on Dispatcher A has been assigned 4 long runners (for example, still executing after 10 minutes); it would be suboptimal to assign another request for ReportService to A even if WRR has concluded that it should go to A. The IPRF will negatively influence A's weight, thus leading to the request being assigned elsewhere.

For information on tuning the weighted round robin load balancing using the processing capacity and `inProgressRequestFactor`, go to:

- http://www-01.ibm.com/support/knowledgecenter/SSEP7J_10.2.2/com.ibm.swg.ba.cognos.cbi.doc/welcome.html, and navigate to: Administer and Deploy\Business Intelligence Administration and Security Guide 10.2.2\Server Administration\Tune Server Performance\Balancing requests among dispatchers

IPRF applies on top of WRR, is off by default and has to be enabled explicitly. IPRF can be set for a single service or all services. The best practice is to use a value of 2.0; 0.0 means normal WRR.

Example: WRR treats every primary request equally. There is no way the routing can know how long a certain request will take. This can lead to overloading a certain service instance/dispatcher although capacity is properly set; IPRF was introduced to alleviate this.



The load balancing mode of cluster compatible supersedes routing, as it gives precedence to local processing. The dispatcher tries to assign a request to a local instance of the requested service first, if this fails, the internal weighted round robin load balancing is used. Cluster compatible mode is used for environments where load balancing between dispatchers is performed externally, such as clustered WebSphere deployments.

In this mode, all dispatchers are required to host identical sets of IBM Cognos services. You should not modify the service configuration on the running instances. Requests which are not subject to load balancing will still get routed by the dispatcher.

Only WebSphere clustering is supported.

The concept of a cluster implies the same services are on each participating dispatcher because a single application tier files package is deployed to multiple instances of the application server. The application server treats them as equal and applies its own external load balancing based on its metrics. The application server is not aware of different service configurations per dispatcher/node which could occur due to configuration in IBM Cognos Administration. The phrase "external load balancing between dispatchers" can involve external load balancers, software or hardware, or even external load balancers in front of gateways which use different front dispatchers to forward their requests to.

Service configuration changes to an instance that is running could cause cluster compatible load balancing to fail as all of the instances will no longer be identical.

What is Advanced Routing?

- provides flexibility
- requires planning and testing
- increases maintenance complexity
- works with server groups and routing sets

© 2015 IBM Corporation



Advanced routing can serve advanced configuration requirements where execution type requests should be routed to specific sets of dispatchers based on the group or role that the executing user is a member of, or the package used for the execution. This is an example of a setup for charge-back accounts, which is used for tracking computing resource usage by business departments or projects, such as a Marketing Department or Summer Promotions project.

Advanced routing can also be based on data source availability and the package which is for specific data sources.

What are Server Groups?

- dispatchers arranged in server groups
- default server group contains all dispatchers
- define additional explicitly named server groups for advanced routing
- define associations for request execution, as in clusters
- saved in the content store

© 2015 IBM Corporation



Each dispatcher is a member of a single server group. There is an unnamed default server group that initially contains all dispatchers, but it is invisible and cannot be managed like other server groups.

Server groups define association for request execution which can include application clusters, Charge-Back scenarios (either by user or by application), and provide heterogeneous database access (not all machines may have access to all databases).

Server groups are saved in the Content Store (the central repository of the system) and are managed through IBM Cognos Administration. Folders defined in IBM Cognos Administration to contain dispatchers do not automatically imply server groups.

Whenever requests have a target Server Group defined, it will route to that group, but Server Groups will be ignored if there is no specific target Server Group, and instead, the current Server Group would be used.

Server Group Considerations

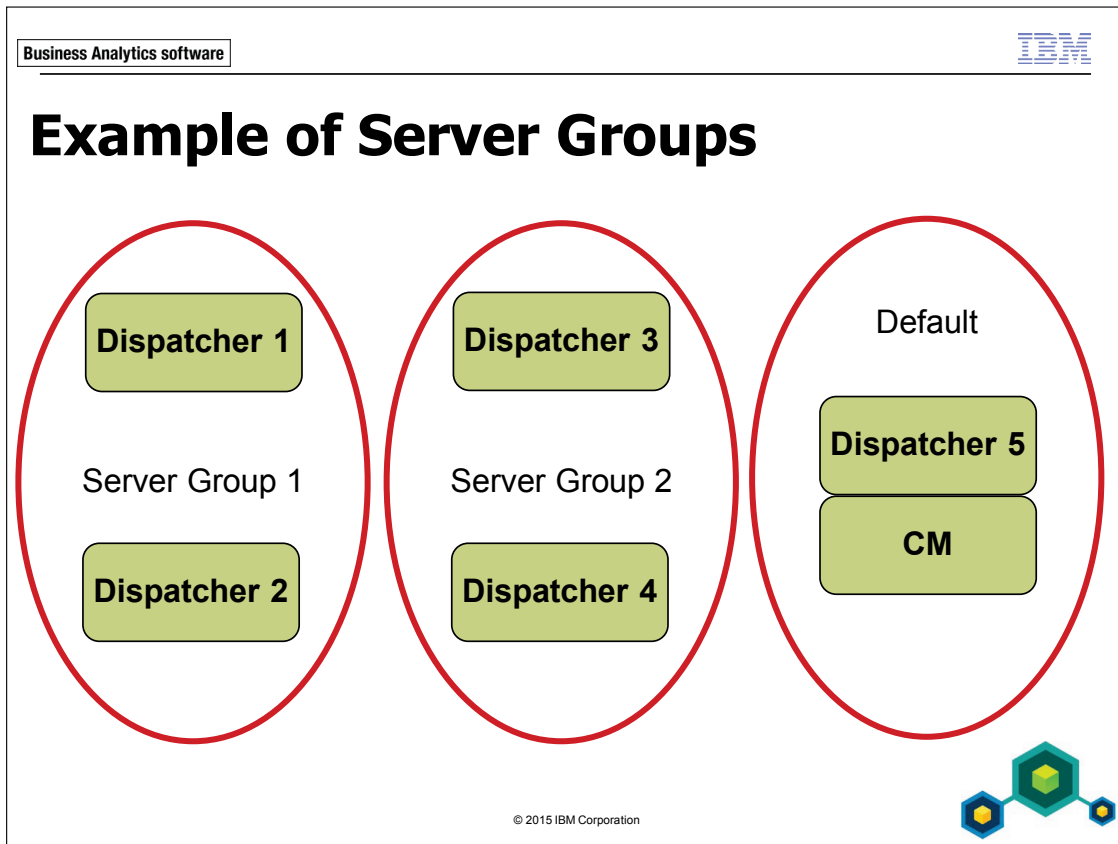
- Content Manager should be in the default server group
- what comes to the server group stays in that server group
 - advanced routing exception

© 2015 IBM Corporation




With the exception of advanced routing, what comes to a server group stays in that server group, where the request will be handled in the server group that the dispatcher to receive the request initially is a member of. Not all dispatchers can handle all types of requests due to services that are available.

From this it becomes clear that load balancing only happens within a server group.



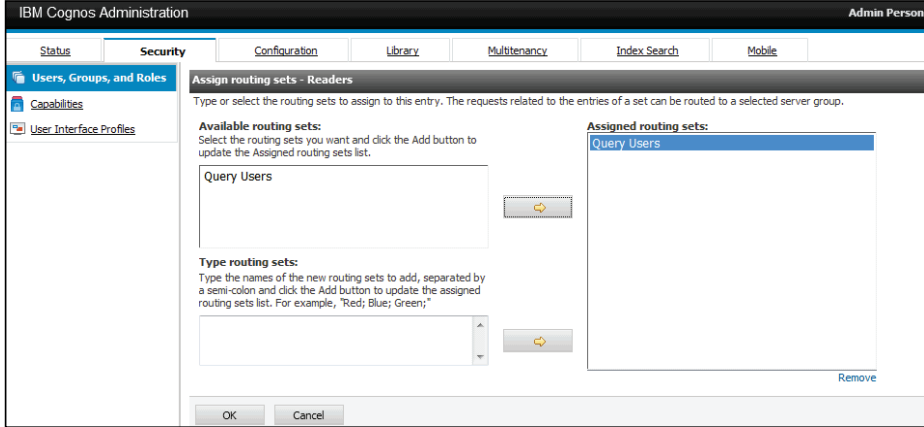
One scenario:

- Server Group 1: handles sales reports
- Server Group 2: handles marketing reports
- Default: Rest of the users

Business Analytics software


Describe Routing Sets

- group routing set
- role routing set
- package routing set



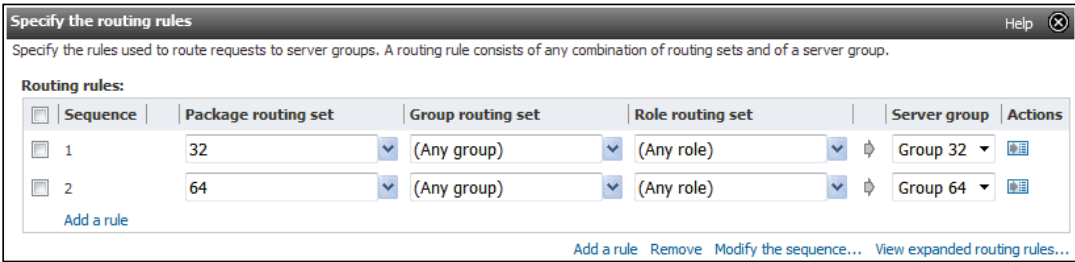
© 2015 IBM Corporation

Routing sets allow advanced routing to route execution type requests based on groups, roles or packages. A routing set is a label assigned to a set of requests based on different purposes. A group routing set is based on requests from a group from any namespace. A role routing set is based on roles from any namespace. A package routing set is based on packages.

Business Analytics software

What are Routing Rules?

- link routing sets to a server group
- stored in a global list
- parsed in sequence



© 2015 IBM Corporation

A routing rule consists of three conditions logically concatenated by AND, and a target server group. Each condition can reference a routing set of a specific type, or remain neutral (any package\group\role).

In routing rules, the first match is used, and subsequent rules are ignored. The request will fail if the target server group does not exist or the requested service does not exist in that server group.

The slide shows the ordered list of routing rules. You can specify the routing rules in IBM Cognos Administration\Configuration\Dispatchers and Service\Toolbar\Specify Routing Rules. In this example routing sets are defined to server groups named Group 32 and Group 64.

Routing Rule Considerations

- request matches routing rule if
 - executed by member of a group/role which is assigned to referenced group/role routing set
 - report executed based on package which is part of referenced package routing set

© 2015 IBM Corporation



Routing rules affect non-affine requests for all executing services such as:

- (Batch)ReportService
- PPESService (PowerPlay Enterprise Server)
- GraphicsService
- MetadataService (MDS)

Explain Advanced Routing

- content manager matches routing rules
- sender of execution request supplies target server group information
- advanced routing takes precedence over load balancing


© 2015 IBM Corporation



The sender of an execution request supplies the target server group information, such as needing the Presentation Service for interactive reports, or the Monitoring Service for batch reports. The sender queries the Content Manager Service for the object to execute, and the response will contain target server group information if applicable. The sender adds the target server group to the BIBusHeader of the request, and then passes the request to its local dispatcher.


The dispatcher routes accordingly, keeping in mind that advanced routing takes precedence over load balancing.

The matching of routing rules is performed by Content Manager, not by a dispatcher.

Business Analytics software


Describe Steps of the Routing Process

1. Identify target service.
2. Determine if request is for content manager service.
3. If absolute affinity request, route to requested dispatcher.
4. From cluster information, deduce set of potential target dispatchers request could be assigned to.
5. Within the cluster view, try assigning affine requests to desired dispatcher. If this fails, treat request as non-affine.
6. If cluster compatible load balancing enabled, look for a local instance of requested service. If local dispatcher is within the cluster view, assign there, else continue to Step 7.
7. Evaluate load balancing (weighted round-robin) within the cluster view.



© 2015 IBM Corporation

Step 1. The targeted IBM Cognos service is identified based on the following information in this order:

1. SOAPAction header (<http://developer.cognos.com/schemas/reportService/1>)
2. b_action variable in an HTML FORM or URL parameter
3. element in the path component of the URL of the request (such as /cgi-bin/cognosisapi.dll/gd/) used to retrieve pre-rendered output such as a chart

If none of the above, the request is forwarded to the Presentation service to display the IBM Cognos Connection home page.

Step 2. Is this a request for the Content Manager service? If so, forward request to the active Content Manager Service.

Step 3. If this is this is an absolute affinity request, route to the requested dispatcher.

Step 4. From the cluster information, deduce the set of potential target dispatchers the request could be assigned to. This set of target dispatchers is known as the cluster view (CV). If advanced routing is configured, this will be the set of dispatchers within the requested server group, otherwise the request goes to the default server group. Advanced routing will simply limit the set of potential target dispatchers. Since every dispatcher has the cluster information available, it can deduct the set of dispatchers belonging to a certain server group.

Step 5. Within the cluster view, try assigning affine requests to the desired dispatcher. If this fails, treat request as non-affine. Affinity means the dispatcher tries to assign the request to a certain dispatcher as indicated by the NodeID in the BIBusHeader. If that fails, it is treated as non-affine, or having no affinity.

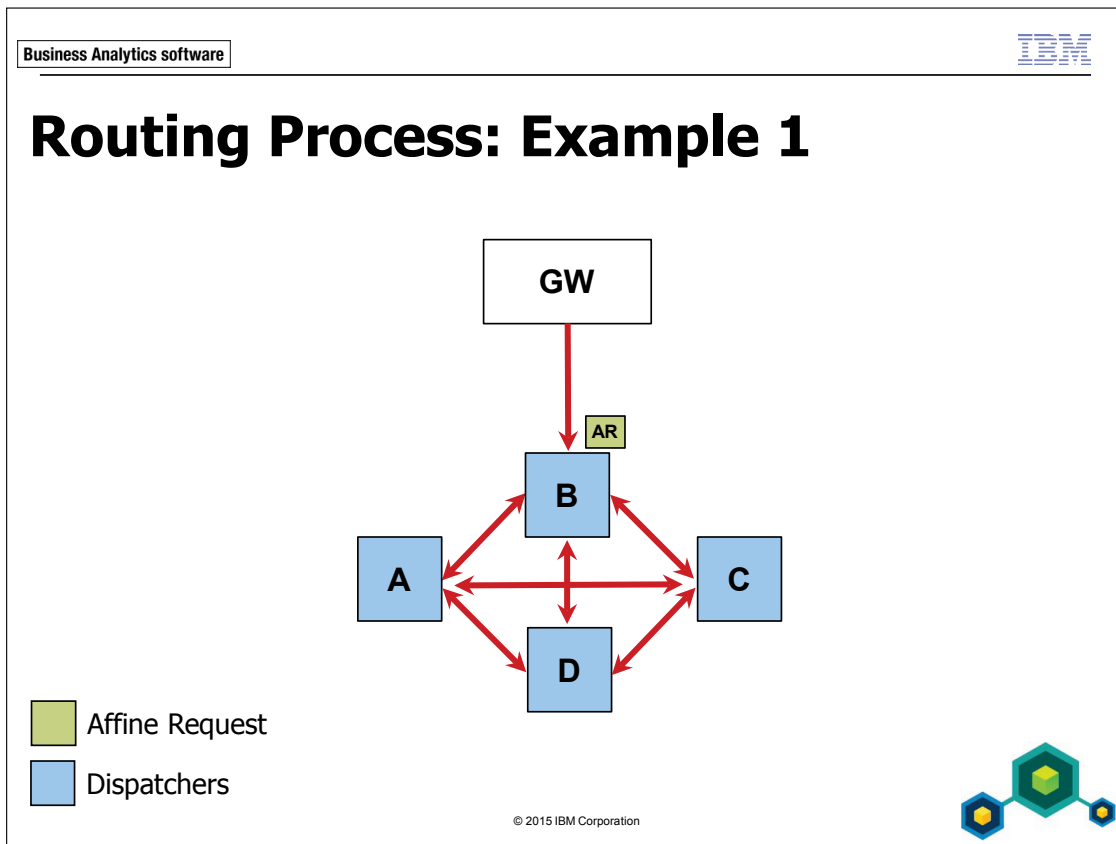
Step 6. If cluster compatible load balancing is enabled, look for a local (on this dispatcher) instance of the requested service. If the local dispatcher is within the cluster view, assign there, otherwise continue to Step 7. For cluster compatible load balancing there is always an attempt to assign the request to a local instance of the requested service. If this does not exist or the local dispatcher is not within the Cluster View, the request is load balanced. This will precede advanced routing because although the request should go to a specific server group, it could remain local. This adds to the point that it is illogical to have server groups in a cluster which are used to model different service layouts and/or resources. In a cluster, all nodes are considered equal.

Step 7. Evaluate load balancing (weighted round robin) within the cluster view. How many instances of the identified IBM Cognos service are active in the current server group?

= 0 - produce error

= 1 - assign request to IBM Cognos service

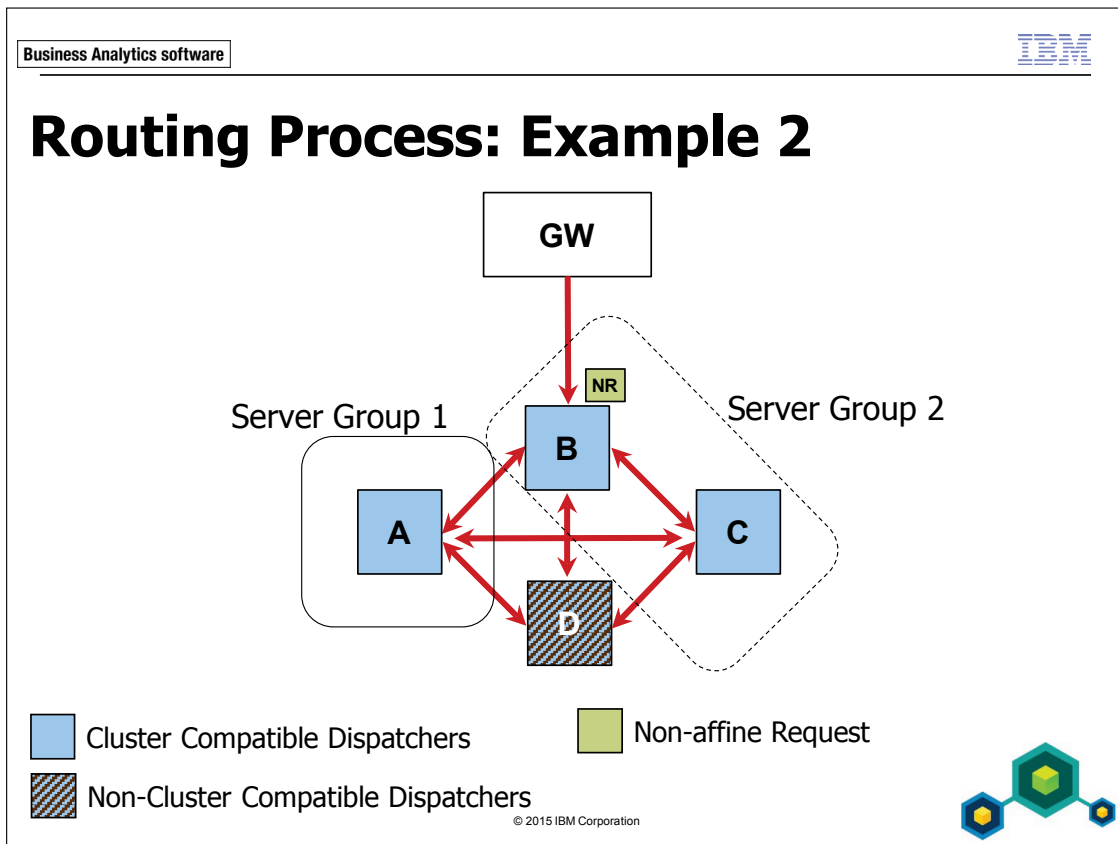
> 1 - evaluate weighted round robin and assign



Scenario: An affine request for Dispatcher C hits Dispatcher B.

Using the steps provided on the previous pages as a guideline, what happens? A possible explanation is provided on the next page.

1. Identify the target service.
2. Determine if the request is for the Content Manager service.
3. If it is an absolute affinity request, route it to the requested dispatcher.
4. From cluster information, deduce set of potential target dispatchers that the request could be assigned to. (Cluster view = A, B, C, D)
5. Within the cluster view, try assigning affine requests to the desired dispatcher. If this fails, treat the request as non-affine. (Try to route to C, but service is down.)
6. If cluster compatible load balancing is enabled, look for a local instance of the requested service. If a local dispatcher is within the cluster view, assign there. (Does not apply in this case.)
7. Evaluate load balancing (weighted round robin) within the cluster view. Evaluate load balancing within default server group and assign.



Scenario: Dispatchers A, B, and C are cluster-compatible.

Server Group 1: Dispatcher A

Server Group 2: Dispatchers B, C

External load balancing is assigned. A non-affine request for a service in Server Group 2 goes to Dispatcher A.

Using the steps provided on the previous pages as a guideline, what happens? A possible explanation is provided on the next page.

1. Identify the target service.
2. Determine if the request is for the Content Manager service.
3. If it is an absolute affinity request, route it to the requested dispatcher.
4. From cluster information, deduce set of potential target dispatchers that the request could be assigned to. (Cluster view = B, C)
5. Within the cluster view, try assigning affine requests to the desired dispatcher. If this fails, treat the request as non-affine. (Does not apply in this case.)
6. If cluster compatible load balancing is enabled, look for a local instance of the requested service. If a local dispatcher is within the cluster view, assign there. Try assigning to service on Dispatcher A. But Dispatcher A is not in the Cluster View, so this will fail, regardless whether the service does on Dispatcher A.
7. Evaluate load balancing (weighted round robin) within the cluster view. Evaluate load balancing within Server Group 2 and assign.

Things to Remember

- requests for Content Manager service get routed to active Content Manager
- what comes to a server group stays in that server group unless advanced routing applies
- load balancing only happens within a server group
- secondary requests are usually not subject to load balancing as they have an affinity assigned to them
- advanced routing applies to requests for execution type services

© 2015 IBM Corporation



There is only one active Content Manager in an IBM Cognos system.

For advanced routing, be aware of how the Cluster View is built.

Only service instances within the same server group are available for load balancing. Load balancing never routes a request to a different server group, as only advanced routing can do that, however advanced routing happens before load balancing takes place.

Advanced routing applies to requests for execution type services which include Batch Report service, Graphics service, PPES service, and Metadata service.

What is Content Manager Cache Service?

- each dispatcher maintains a local cache for objects retrieved from Content Manager service
- IBM Cognos Services will verify CMCS is available on the local dispatcher, and then will query the local CMCS instead of Content Manager service

© 2015 IBM Corporation



The local cache on a dispatcher for objects retrieved from Content Manager service is implemented by the ContentManagerCacheService (CMCS). This is used to decrease load on the active Content Manager and improve the speed execution. Objects subject to caching are Report, ReportCache, Package, Model, PackageConfiguration, ModelView, DataSourceSignon, Account and Session. A best practice is to enable this service on all dispatchers which run the Dispatcher service.

After having verified CMCS is available on the local dispatcher, IBM Cognos services will query the local CMCS instead of CM Service. This is supported by CC, QS, RV and others.

CC: IBM Cognos Connection

QS: Query Studio


RV: IBM Cognos Viewer, or Report Viewer

CMCS is a new feature since IBM Cognos 10.1.1. Memory is taken from the Java heap, and there is no JNI heap involved.

It is best practice to have CMCS enabled on all dispatchers which run the Dispatcher service or in other words all possible front dispatchers. Therefore, this excludes dispatchers of a Content Manager only install.


If this service is not available on a dispatcher which initially handles a request, there will be no caching for that request.

Business Analytics software



Describe CMCS Details

- serves a primary or secondary request
- CacheContext
- Validator



© 2015 IBM Corporation

CMCS either serves a primary or secondary request for some IBM Cognos service or requests sent by other CMCS instances.

Each data item in the cache has context (CacheContext). This context is used to evaluate if it matches a given query. The context may tie the item to a session or not, so data may be shared across sessions if not subject to security.

A Validator is an object attached to each data item by CM. It allows probing for the validity of cached data by using lightweight CM queries only. So it is validation over querying an actual CM object which again saves load to the CM.

When CMCS has found an object in its cache, it has to ensure freshness. For that, it validates the data if necessary. The CM query required for that is minimal though it exists. To reduce the amount of validation queries, Validators may linger, so within the linger timeout data will not be validated again. This is to serve high frequency queries to the same objects. It carries some risk though, which will be presented on the Explain CMCS Settings page.

Service a Primary Request

- if requested data is not in local cache or Validator is expired
 - Query CM to fill cache; CM will supply the data and the automatically attached Validator for it
 - create CacheContext
 - reset linger timer
- if linger timer is expired
 - validate the data by issuing a light-weight CM query
 - reset linger timer
- respond to query, attach CacheContext
- completed

© 2015 IBM Corporation



The slide lists a pseudo-code rundown of steps which the CMCS executes when serving requests.

Service a Secondary Request

- if requested data is not in local cache
 - if request has CacheContext attached
 - from CacheContext, retrieve address of remote CMCS with data and access of that instance rather than CM Service
 - add data to local cache and reset local linger timer
 - else
 - query CM to fill cache; CM will supply data and the automatically attached Validator for it
 - create CacheContext and reset linger timer
- if linger timer is expired
 - validate data by issuing a light-weight CM query
 - reset linger timer
- respond to query, attach CacheContext
- completed

© 2015 IBM Corporation



Linger time is how long the item in the cache lingers around.

Business Analytics software

Explain CMCS Settings

- memory limit
 - set memory limit to 0 to disable CMCS
- linger timer

© 2015 IBM Corporation

The memory limit for CMCS determines the maximum amount of Java heap taken from the JRE hosting the Dispatcher for the cache. The cache size is dynamic. To disable the CMCS, set the memory limit to zero (0).

Add an advanced parameter to CMCS, for the linger timer with the `DISP.contentManagerCacheService.cacheValidatorTimeToLinger` property. The value is in ms (milliseconds), with a default value of 30000. This setting determines the linger time before data is validated again. This implies that changes within 30 seconds are not automatically anticipated by CMCS.

Linger can lead to issues; if cache entries, for example, change frequently then lowering the linger timeout may be required.

Describe Timers

- validation timer:
DISP.contentManagerCacheService.cacheValidatorTimeToLive
- account inactivity timer:
DISP.contentManagerCacheService.accountInactivityTimer

© 2015 IBM Corporation



The validation timer determines how long data objects remain in the cache. After the validation timer expires, objects are considered invalid and have to be fetched from the CM again. Add the advanced parameter to CMCS for the validation timer, with the `DISP.contentManagerCacheService.cacheValidatorTimeToLive` property. The value is in ms, and the default value is 900000.

The account inactivity timer determines how long data for this session is cached. If the session is idle for longer than the inactivity timer, all data cached for that account is purged from the cache. This is a security and resource saving feature. Add the advanced parameter to CMCS for the account inactivity timer, with the `DISP.contentManagerCacheService.accountInactivityTimer` property. The value is in ms, and the default value is 900000.

Summary

- At the end of this module, you should be able to:
 - describe IBM Cognos Dispatcher
 - describe request routing and the routing process
 - describe Content Manager Cache Service



Workshop 1: Review the Server Groups (Optional)

As an administrator, you have configured server groups in your environment to let you define advanced routing rules. You will review these settings in your environment using p2plbDiag, and you will review the routing sets.

You will use p2plbDiag to review:

- the dispatcher vclassbase:9315/p2pd settings: What server group does the dispatcher belong to?
- the dispatcher vclassbase:9320/p2pd settings: What server group does the dispatcher belong to?

You will use IBM Cognos Administration to review:

- the dispatcher vclassbase:9315/p2pd settings: What server group does the dispatcher belong to?
- the dispatcher vclassbase:9320/p2pd settings: What server group does the dispatcher belong to?

You will use IBM Cognos Connection to review the routing sets for the packages in:

- Public Folders\Samples\Models
- Public Folders\Samples_DQ\Models
- Public Folders\Samples_Dynamic_Cubes

You will use IBM Cognos Administration to review:

- the defined routing rules

For more information about where to work and the workshop results, refer to the Tasks and Results section that follows. If you need more information to complete a task, refer to earlier demos for detailed steps.

Workshop 1: Tasks and Results

The IBM Cognos 10 Full:9315 dispatcher and the IBM Cognos 10 DispCM:9320 dispatcher should be running at the beginning of this workshop.

Task 1. Review the server groups in p2plbDiag.

- Launch **Internet Explorer**, and then go to <http://vclassbase:9315/p2pd/servlet/dispatch/p2plbDiag>.
- Log in using **admin/Education1** credentials.

```
This dispatcher is: vclassbase:9315/p2pd/servlet/dispatch
GUID=2015-04-01-21.08.14.757907
Using CM: vclassbase:9320/p2pd/servlet
Current time: Apr 7, 2015 11:14:50 AM EDT
Configured dispatchers and services:
this dispatcher is : "/configuration/dispatcher[@name='http://vclassbase:9315/p2pd']"
this dispatcher is in serverGroup : "Group 64"

All known dispatchers:
  Dispatcher: /configuration/dispatcher[@name='http://vclassbase:9320/p2pd']
    name: vclassbase:9320/p2pd
    dispatcherID: 2015-04-02-20.26.05.342844
    capacity: 1.0
    SSL: false
    serverGroup: Group 32
    loadBalancingMode: weightedRoundRobin
    edition: 10.2.6100.68
    Services:
      Service name: repositoryService disabled? false
      Service name: dispatcher disabled? false
      Service name: contentManagerCacheService disabled? false
```

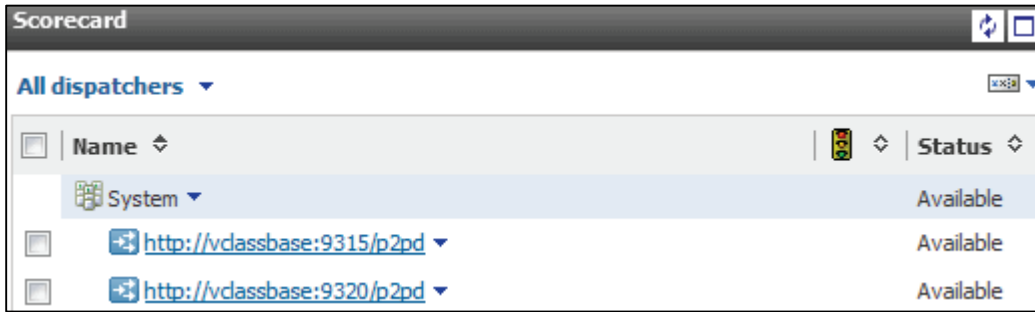
The server group name is displayed for this dispatcher: Group 64, and the second server group Group 32 is displayed for the vclassbase:9320/p2pd dispatcher.

- Close the browser window when you have finished viewing the results.

Task 2. Review the server group settings in IBM Cognos Administration.

- Launch **Internet Explorer**, go to <http://vclassbase:88/C10Full>, log on to the **LDAP_Dev** namespace with **admin/Education1** credentials, and then launch **IBM Cognos Administration**.
- On the **Status** tab, click **System**. It may take a few moments for the Status tab to refresh.

- In the upper-left corner of the **Scorecard** pane, click the **All servers** arrow to display the **Change view** menu, and then click **All dispatchers**.



You can also set the server group property at the system level, rather than at this level of detail.

- Next to the **http://vclassbase:9315/p2pd** dispatcher, click the **Actions** arrow, and then click **Set properties**.
- Click the **Settings** tab, and then from the **Category** list, click **Tuning**.
- Notice that for the **Server group** property, in the **Value** column box, **Group 64** is the name of the server group.

This is where you can configure the Server group name when defining your environment; you could also set this on the Configuration tab, on the Dispatchers and Services pane.

You can assign multiple dispatchers to a server group.

- Click **Cancel** to close the **Set properties** dialog box.
- Optionally, repeat the steps to review the Server group property for the **http://vclassbase:9320/p2pd** dispatcher.

These server group names can be used when you define routing rules.

Task 3. Review package routing sets that use the server groups.

- Launch **IBM Cognos Connection**, and navigate to **Public Folders\Samples\Models**.
- In the **Actions** column for the **GO Data Warehouse (analysis)** package, click **Set properties**.
- In the **Advanced routing** section, under **Routing sets**, click **Edit**.


Notice the current assigned routing set is 32.

This environment was configured to name the routing sets to align with the server group names. The other option available is 64. The server groups have dispatchers that are configured to handle either 32-bit ReportServer service queries or 64-bit ReportServer service queries, as you cannot run both on the same dispatcher.

All reports in this package, GO Data Warehouse (analysis) use the 32-bit ReportServer for queries.

- Click **Cancel** to close the **Assign routing sets** dialog box, and then click **Cancel** to close the **Set properties** dialog box.
- Optionally, repeat the steps to review the routing sets for any package in the following **Public Folders** packages in:
 - **Samples_DQ\Models**
 - **Samples_Dynamic_Cubes**

Task 4. Review the routing rules.

- Launch **IBM Cognos Administration**, and then click the **Configuration** tab\Dispatchers and Services pane.
- On the toolbar, click **Specify Routing Rules** .



Notice that in the **Routing rules** pane, there are two package-based routing sets defined.

Also notice that routing sets can also be defined for groups and for roles.

For packages using routing set 32, notice that it is routed to the server group "Group 32", which has dispatchers that are configured to use the 32-bit ReportServer for queries.

Packages using routing set 64 are routed to "Group 64", which has dispatchers that are configured to use the 64-bit ReportServer for queries.

- Click **Cancel** to close the **Specify the routing rules** dialog box, log off **Admin Person**, and then close the browser window.