Understanding Where to Install the ODI Standalone Agent

Introduction

ODI is a true ELT product: no middle-tier server is required. Everything runs in the databases, and all the operations can be orchestrated by a very lightweight agent.
So the question is: without a dedicated server, where to install this agent?

If you look at the data integration environment, source systems are not ideal - they could be dispersed throughout the information system. Dedicated systems could work, but if they are independent of your ETL jobs, then you are dependent on physical resources that may not be tightly coupled with your processes… so installing the agent on the target systems makes sense. In particular if you are talking of a data warehousing environment, where most of the staging of data will already occur on the target system.

But in the end, "target" is a convenience, not an all be all. So rather than accepting this as an absolute truth, we will look into how the agent works and from there provide a more detailed answer to this question.

For the purpose of this discussion we are considering the Standalone version of the agent only – the JEE version of the agent runs on top of Weblogic, which pretty much defines where you would install the agent… but keep in mind that in the same environment you can mix and match standalone and JEE agents!

First we will look into connectivity requirements. Then we will look into how the agent interacts with the environment: flat files, scripts, utilities, firewalls. And finally we will illustrate the different cases with real life examples.

Understanding Agent Connectivity Requirements

The agent can have to perform up to 3 tasks for a process to run:

- Connect to the repository (always)
- Connect to the sources and targets (always)
- Provide JDBC access to the data (if needed)

Connection to the repository

The agent will connect to the repository to perform the following tasks:

- Retrieve the code that must be executed
- Finish the code generation that must be executed based on the *context* that was selected for execution
- Write the generated code in the operator tables
- After the code has been executed by the databases, update the operator tables with statistics and if necessary error messages returned by the databases or operating system.

To perform all these operations, the agent will connect to the repository using JDBC. The parameters for the agent to connect are defined when the agent is installed. For a standalone agent, you will find these parameters in the odiparams.sh file (or odiparams.bat on a windows platform).

What does this mean for the location of the agent?

Since the agent uses JDBC to connect to the repository, the agent does not have to be on the same machine as the repository. The amount of data exchanged with the repository is limited to logs generation and updates, but this can become somewhat consequent in near real time environments. It is highly recommended that the agent be on the same LAN as the repository. Beyond that, the agent can be installed on pretty much any system that can physically connect to the proper database ports to access the repository.

Connection to the sources and targets

Before sending code to the source and target databases for execution, the agent must first establish a connection to these databases. The agent will use JDBC to connect to all database sources and targets at the beginning of a session execution. These connections will be used by the agent to send the DDL (create table, drop table, create index, etc.) and DML (insert into… select…from… where…) that will be executed by the databases.

What does this mean for the location of the agent?

As long as the agent is sending DDLs and DMLs to the databases, once again it does not have to be physically installed on any of the systems that host the databases. However, the location of the agent must be strategically selected so that *it can connect to all databases*, sources and targets. From a network perspective, it is common for the target system to be able to view all sources, but it is not rare for sources to be segregated from one another: different sub-networks, firewalls getting in the way, you name it! If we do not have the guaranty that the agent can connect to all sources (and targets) if it is installed on a source system, then it makes more sense to install it on one of the target systems. Based on the activity described above, we can see that the actual activity of the agent (CPU, memory) is quite limited, so its impact on the systems will be quite negligible. Conclusion: from an orchestration perspective, the agent could be anywhere in the LAN, but it is often times more practical to install it on the target server.

Data Transfer Using JDBC if needed

ODI processes can use multiple techniques to extract from and load data into sources and targets: JDBC is one of these techniques. If the processes executed by the agent use JDBC to move data from source to target, then the agent itself establishes this connection: as a result the data will physically flow through the agent.

What does this mean for the location of the agent?

This is a case we have to pay more attention to the agent location. In all previous cases, the agent could have been installed pretty much anywhere as the performance impact of moving it was negligible. Now if data physically moves through the agent, placing the agent on either the source server or the target server will in effect limit the number of network hops required for the data.

Let's take the example where I would run the agent on my own windows server, with a source on a mainframe and a target on Linux. Data will have to go over the network from the mainframe to the windows server, and then from the windows server to the Linux box. In data integration architectures, the network is a limiting factor. Placing the agent on either the source or the target server will help allow us to limit the adverse impact of the network.

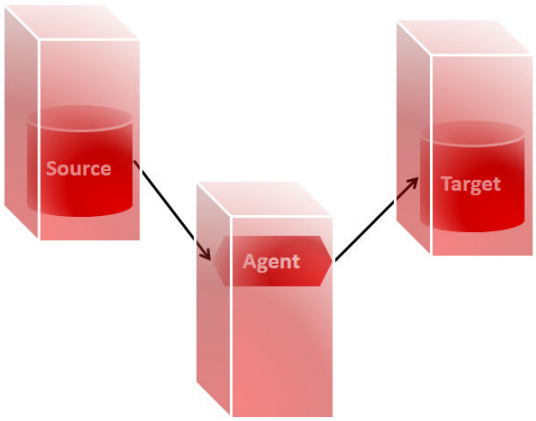

Figure 1: JDBC access with remote ODI agent

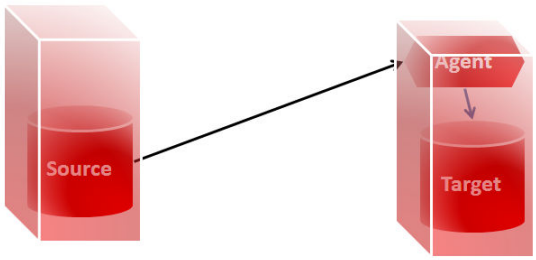

Figure 2: JDBC access with ODI agent on target

Other considerations: Accessing files, scripts, utilities

Part of the integration process often requires access to resources that are local to a system: flat files that are not accessible remotely, local scripts and utilities. A very good example is when you want to leverage the database bulk loading utilities for files located on a file server. In that case, how do you invoke the utilities? How do you access the files? With the ODI agent, the answer is quite simple: install the agent on the file server along with the loading utilities – or share the directories where the files and utilities are installed so that the agent can view them remotely.

What does this mean for the location of the agent?

It is actually quite common to have the ODI agent installed on a file server (along with the database loading utilities) so that it can have local access to the files. This is easier than trying to share directories across the network (and more efficient), in particular if you are dealing with disparate operating systems.

Another consideration at this point is that you are not limited to a single ODI agent in your environment: some jobs can be assigned to specific agents because they need access to resources that would only be visible to other agents. This is a very common infrastructure, where you would have a central agent (maybe on the target server) and satellite agents in charge of very specific tasks.

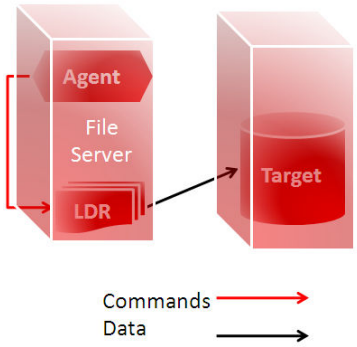

Figure 3: ODI agent loading flat files

Beyond databases: Big Data

A very good description of Hadoop is available here:
http://hadoop.apache.org/common/docs/current/hdfs_design.html.

In a Hadoop environment, execution requests are submitted to a NameNode. This Namenode is then in charge of distributing the execution across all DataNodes that are deployed and operational. It would be totally counter-productive for the ODI agent to try and bypass the NameNode. From that perspective, the agent would have to be installed on the NameNode.

Note: The Oracle BigData appliance ships with the ODI agent pre-packaged so that the environment is immediately ready to use.

Firewall Considerations

One element that seems pretty obvious is that no matter where you place your agents, you have to make sure that the firewalls in your corporation will let you access the necessary resources. More challenging can be the timeouts that some firewalls (or even servers in the case of iSeries) will have. For instance it is not rare for firewalls to kill connections that are inactive for more than 30 minutes. If a large batch operation is being executed by the database, the agent has no reason to overload the network or the repository with unnecessary activity… but as a result the firewall could disconnect the agent from the repository or from the databases. The typical error in that case would appear as "connection reset by peer". If you experience such a behavior, think about reviewing your firewall configurations with your security administrators.

Real life Examples

We will now look into some real life examples, and define where the agent would best be located for each scenario.

The case for Exadata (External tables)

We are looking here into the case where flat files have to be loaded into Exadata. An important point from an ODI perspective is that we first want to look into what makes the most sense for the database itself – then we will make sure that ODI can deliver.

The best option for Exadata in terms of performance will be to land the flat files on DBFS – this way the data loads will take advantage of the performance of infiniband.

Now for the data loads from flat files into Exadata, External Tables will give us by far the best possible performance.

Considerations for the agent

The key point here is that External tables can be created through DDL commands. As long as the files are on DBFS, they are visible to the database… (They would have to be for us to use External tables anyhow). Since the agent will connect to Exadata via JDBC, it can issue DDLs no matter where it is installed! If you do have a personal preference for the agent location, then you can do what you prefer. If you don't know where to install it, simply put it on Exadata and be done with it.

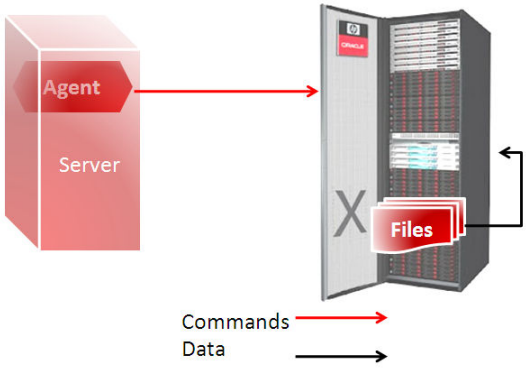

Figure 4: Remote ODI agent driving File Load with External Tables

The case for JDBC loads

There will be cases where volume dictates that you use bulk loads. Other cases will be fine using JDBC connectivity (in particular if volume is limited). Uli Bethke has a very good discussion on this subject here (http://www.business-intelligence-quotient.com/?tag=array-fetch-size-odi), even though his objective was not to define when to use JDBC or not.

One key benefit of JDBC is that it is the simplest possible setup: as long as you have the proper drivers and physical access to the resource (file or database) you are in business. For a database, this means that no firewall prevents access to the database ports. For a file, this means that the agent has physical access to the files.

Considerations for the agent

The most common mistake for files access is to start the agent with a username that does not have the necessary privileges to see the files – whether the files are local to the agent or accessed through a shared directory on the network (mounted on Unix, shared on Windows).

Other than that, as we have already seen earlier, locate the agent so as to limit the number of network hops from source to target (and not from source to middle tier to target). So the preference for database-to-database integration is usually to install the agent on the target server. For file-to-database integration, have the agent and database loading utilities on the file server. If you combine files and databases as sources then you can either have a single agent on the file server, or have 2 agents and thus optimize the data flows.

Revisiting the case for Exadata with file detection.

Let's revisit our initial case with flat files on Exadata. Let's now assume that ODI must detect that the files have arrived, and that this detection must trigger the load of the file.

Considerations for the agent

In that case, the agent itself will have to see the files. This means that either the agent will be on the same system as the files (we said earlier that the files would be on Exadata) or the files will have to be shared on the network so that they are visible on the machine on which the agent is installed. Installing the agent on Exadata is so simple that it is more often than not the preferred choice.

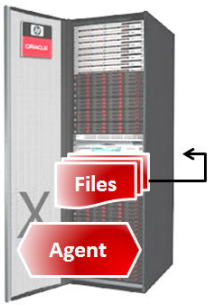

Figure 5: ODI agent on Exadata detecting new files and driving loads with External Tables

Conclusion

The optimal location for your agent will greatly depend on the activities you want the agent to perform. Keep in mind that you are not limited to a single agent in your environment – and more agents will give you more flexibility. A good starting point for your first agent will be to position it on the target system. Then look at your requirements, and add additional agents when they are needed.