Content Management Instance Architecture
An Oracle Whitepaper

DATE

December 2009

Version 2.0

1 Introduction

Oracle Universal Content Management (UCM) is the core product for content management solutions provided by Oracle. This includes the additional product components that extend UCM such as Universal Records Management (URM), Web Content Management (WCM), Digital Asset Management (DAM), and others. UCM is also the content service provider for application integrations, portals, and other solutions.

A common question is, "Can I have a single UCM instance to serve my enterprise?"

First, an instance is defined as an installation of UCM with a unique URL location, database, and file system. All of these attributes are defined during the installation process. Instances cannot share a database or a file system.[1] A clustered install does not count as separate instances because they share the same URL, the same database, and the same file system.

This document also uses the term "application". The term application is used to define a UCM implementation scope that is targeted to a specific user community, line of business, or set of capabilities. A UCM instance may support multiple "applications". For example an instance may be configured with the appropriate metadata, security, components, etc. to support an HR policies and procedure "application". It may also be configured to support a marketing collateral application.

So, the answer to the question posed above is "perhaps", but there are many things to consider when deciding if you should use one single instance or deploy multiple instances. For enterprises of any significant size or complexity the answer will almost always be that multiple instances are effective and recommended.

This whitepaper covers specific topics to consider when making the choice about how many instances should be deployed, and why. This whitepaper also discusses the hardware infrastructure best practices associated with multiple instance architectures.

2 Multiple Instance Considerations

This section enumerates the most common topics of discussion when determining an instance architecture. These are presented without value qualification in that any of these may or may not be important to your organization.

2.1 Architectural Considerations

2.1.1 Internal, DMZ, External

Some applications require access by external parties. This is most common for web content management related projects, but is not limited to those. This architecture will require a DMZ presence of some sort, which may be a reverse proxy – or it may mean having a UCM instance within the DMZ. This can be a significant architectural difference from applications that are

[1] Instances can share the same database SID (database instance) but require unique schemas. Likewise, they can share the same storage solution but must have a unique file system allocation within the storage solution.

intended to be entirely internal – such as HR or legal documents. In addition, external applications may require the use of HTTPS, while internal applications do not.

2.1.2 Capacity Requirements

Some applications are extremely heavy users of resources, including CPU, I/O, database, and network. Applications that fall into this category are often of an archiving nature, but not always. For example, an application may be developed that captures a constant stream of "documents" coming from a mainframe print spool. We often describe such an application as being "high volume". There a number of other reasons an application may be "high volume", including a significantly large consumer population, a significant publishing process, or a significant ingestion rate. Such applications will frequently be put on dedicated infrastructure so that they can be tuned and configured completely independently from other UCM instances and applications.

2.1.3 Contribution and Consumption

It is a best practice in web content management deployments to use two instances – one for contribution and one for consumption. This is a best practice for several reasons.

First, it separates the resource usage (CPU, I/O, Network, etc.) of the contributors from the consumers. The authoring and editing of content, workflow and approval processes, document conversion, and other processing is isolated from the consumption environment – which is likely to be tuned for optimal performance for consumers.

Second, it provides a separation of security models. The internal contribution instance will often have a finer grained security model (definition of who can edit what) than is required in consumption. In fact, it is often the case in consumption environments that everything is converted to public and read-only (although certainly not always).

Third, it allows for a separation of security integrations. The internal contribution instance will likely be integrated with a corporate directory such as LDAP or Active Directory. The external consumption instance may not be (again, another level of security) or may be integrated with a separate LDAP that contains external consumers.

Fourth, it allows the disabling of contribution activity on the consumption server so that it is generally more secure.

Fifth, it can provide a level of failover support in the event of a considerable "disaster" since the entire site is operational on the contribution server as well.

2.1.4 Service Level Agreements

Different applications will likely have different overall value to the business or organization. That is to say, some applications may be deemed mission critical, while others are not. Mission critical applications will often require redundant components throughout the architecture as well as tier 1 storage. The infrastructure for such applications can be costly and may not be necessary for all applications. Other instances may be departmental or have lower service level agreements in terms of recovery times. Such applications can be deployed on instances with fewer redundant components and lower tier storage to save cost.

2.1.5 Geography

Many businesses are geographically disbursed, and are sometimes comprised of multiple data centers. It is usually the case that applications are best deployed “near” the user community to achieve the best performance. This also means that usually application support and administration is “local”.

2.2 Functional Considerations

2.2.6 Incompatibilities

Each instance of UCM has its own set of components that are deployed and its own set of system-wide configuration settings. Sometimes these components and/or configuration settings are not compatible with different applications. For example, it is possible to configure an instance of UCM to use what is referred to as “Fast Check-In”. This streamlines the check-in process to bypass some of the normal processing in exchange for speed. This includes bypassing the check for workflow, bypassing the check for notifications, etc. This behavior is likely not desirable in an application that is used for general document management.

These types of incompatibilities may include such things as document conversion definitions, rendition set definitions, components, security integrations, retention policies, customizations, and many others.

2.2.7 Application “Personalities”

While less concrete than other considerations discussed, this will often have a bearing. Instances are sometimes established around the application capabilities they intend to support. For example, a number of applications that tied to scanning, fax, and other types of similar ingestion, that require workflow routing, are often a good candidate for combining on an instance.

However, disparate applications such as records management vs. web content management may not share much in common. This will be evident in the architecture, the components required, the metadata, the security, and many other facets of the instance.

2.3 Metadata Considerations

2.3.8 Metadata Differences

While it is a good idea (and strongly recommended) to have a corporate level governance structure around taxonomy and classification (metadata) it often a very bad idea to implement the entire corporate metadata model in any UCM instance[2]. In fact, it is a best practice to implement a given application with a targeted and pragmatic set of metadata to streamline the application, resources used, and the user interface.

To this end, two different applications may have very different metadata models – even through they both fit into the corporate taxonomy structure. Applications that do not share a reasonable amount of metadata may be adding unnecessary resource usage, maintenance, and complexity to an instance.

[2] Increased maintenance with no business value add, increased database usage and complexity, increased complexity of rules and profiles, increased frustration and complexity for users, etc.

2.3.9 Metadata Complexity

Some applications have complex and even custom metadata behavior that is necessary to support the requirements for the application. Often this may create complexities or incompatibilities with other applications on the same instance.

2.3.10 Metadata Size

The number of metadata fields, whether from Oracle provided components or custom, can have an impact on an instance. There is a hard limit to the number of metadata fields that can be defined. This limit is set by the record size limitation of the underlying database. While this is fairly significant, if reached it is a hard limit. Breaking into multiple instances may be required as this limit is neared or reached.

2.4 Security Considerations

2.4.11 Security Model

Much like the metadata model, different applications are likely to have unique security model requirements. Also like metadata, it is strongly recommended that a governance structure around managing security models is established. Continuing the thought, it is also like metadata in that if two applications have significantly different security models there may be limited value in combining them on an instance.

2.4.12 Security Integrations and Provisioning

Applications may also be unique in their source of user provisioning. Some applications may use a corporate directory, others may use a directory designed for external users, some my combine these, some may use the internal content server provisioning. All of these are valid – but the deployment on a specific instance may create an incompatibility with other applications.

Likewise, some applications may deploy custom security integrations – either at the filter level (SSO integrations for example) or at the provider level. Again, the security customizations done for a specific business requirement (an application or set of applications) may not be applicable to other applications.

2.4.13 Complexity

Another consideration is the complexity or methods of security deployed on an instance. For example, the base security model may be extended with additional capabilities such as ACLs (Collaboration for example), Supplemental Markings (RM), Classified Security (RM), and custom security processing. The "*NeedToKnow*" component, for example, changes the security processing within UCM. Again, these types of security configurations may not be compatible with all applications you wish to deploy.

2.4.14 Performance

The security model, and any custom or product extensions to the security model, can have a direct impact on performance. For example, Oracle generally recommends that an instance have 50 or fewer security groups defined. The reason for this is because the content server takes a users roles and converts that to a "security clause" that gets appended to searches. While the content server generally does a nice job of optimizing the search, if the security model is sufficiently large or complex it can slow the search response times. Certainly database tuning, indexes, partitions, and other methods can be used to tune and improve search performance, it is still a good idea to plan on having as lean a security model as possible.

2.4.15 HIPPA, Auditing, and Other Regulations

In some cases instances may be segregated to be compliant with regulations. For example, in some industries external audits of systems are confined to the systems containing relevant data. If a system contains considerably more data than the target of the audit, the "other" data can be subject to discovery.

2.5 Organizational Considerations

2.5.16 Ownership

Occasionally it is the case that different business units, departments, or lines of business operate independently. This sometimes also means that individual lines of business have autonomy on managing IT infrastructure and applications. In cases like this it is common for UCM instances and applications to be deployed "close" to the owning organization.

2.5.17 Administrators & Maintenance

Application administrators and server/network maintenance may be divided within the business based on the line of business, it may be geographically, or some other mechanism that maps better to a multi-instance architecture.

3 What are the Advantages to Multiple Instances?

While this is likely not an exhaustive list, here are some of the commonly considered advantages.

- **Better performance**

  Instances, when they are properly planned and designed to contain "like" or "similar" applications, will have a security model and a metadata model that is leaner. Likewise, each instance will be operating on smaller data sets (number of content items and database sizes). Both of these points taken together means applications that generally perform better.

- **Better Resource Utilization**

  Instances that are operating efficiently and performing well will by definition be using resources well. However, the total utilization of the hardware, storage, and other components is also a consideration. By deploying multiple instances on shared infrastructure (see the Best Practice section below) it is possible to also better utilize the underlying infrastructure. Since presumably each independent instance is operating efficiently, the net result is a higher throughput of service requests overall (for everything running on the shared infrastructure) and good utilization of the infrastructure overall.

  The negative way to look at this is instances that have overly complex security or metadata models and/or large data sets. These instances will have lower throughput as the service calls, particularly searches, may require more time and resources. The end result is slower response times with lower overall infrastructure utilization[3].

- **Easier Management**

  This topic is specifically in regard to the metadata model, security model, and rules / profiles. By having divided the applications into smaller and less complex instances, the maintenance time required is reduced and the interactivity (impact of changes) to the environment is reduced.

- **Better User Experience**

  As more and more applications are combined into an instance, it generally becomes necessary to make the user interface more and more generic to encompass all of the applications. With similar applications grouped per instance, it is possible to better tailor or customize the user interface to better serve the consumers.

4 What are the Disadvantages to Multiple Instances?

Below are some of the common objections and disadvantages to multiple instances. It often these disadvantages that are weighed against the considerations and advantages to determine a final instance architecture.

- **Maintenance and Administration**

  One of the common objections to multiple instances is the necessity to administer and maintain each instance separately. This is a fair consideration. It is true that each instance will need to be individually administered. There are methods that can be used to mitigate the impact of this (such as Master / Proxy configurations), but in the end managing each instance has a cost associated.

  In fact, after reading the considerations for instances you may be wondering why not have one instance per application, and the "cost" for distributed administration is a consideration.

- **Search**

  Another common objection to multiple instances is that now users must search within individual instances to find things and thus don't truly have an "enterprise view" of all the content. First, it is true that each instance maintains its own search index, such that searches within a particular instance execute against the content stored in that instance.

  However, there are 3 important things to note.

  First, it is often the case that applications and instances are split in such a way that it doesn't really make business sense for most users to be searching across multiple repositories. Experience has shown us, having done hundreds of implementations, that this requirement will usually either be deleted or can be significantly reduced to some target users or groups.

[3] The utilization of the database may be very high as it is processing through the larger and more complex data sets. The rest of the infrastructure may be underutilized.

Second, assuming it *is* necessary to search across multiple instances in some cases, this can be done by connecting the desired instances using UCM's "Enterprise Search" capability. This provides the ability to execute a single search across multiple UCM instances.

Third, it may be that searching across multiple UCM instances is not even enough. Perhaps the real requirement is to be able to search "anywhere" in the enterprise – including UCM instances, shared drives, SharePoint, web sites, etc. For use cases like this an enterprise search tool, such as Oracle Secure Enterprise Search, should be used in addition to UCM.

- **Access by other instances**

  A key capability of UCM is the notion of content reuse. This is the ability to use a piece of content (perhaps in various renditions) across a number of consumption channels. A concern with multiple instances is often that content that needs to be reused in another instance. This may require replication of the content or "remote access" to the content. Again, this is an important consideration. While UCM provides tools to make this easier (replication, metadata mapping, retention, web services, etc.) any content reuse scenarios across instances will require careful planning.

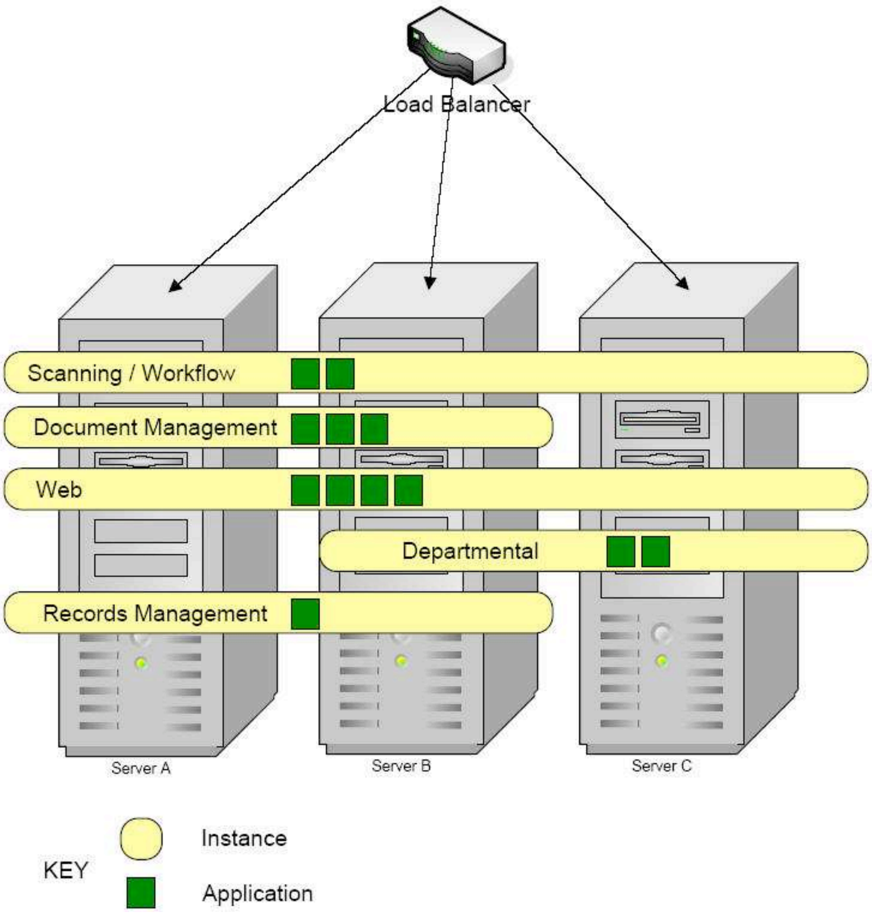
5 Best Practice Instance Architecture

To this point we've discussed many of the considerations for determining appropriate instance architectures. While the term "best practice" is not really applicable[4], there is an "appropriate practice" for your organization based on your requirements, infrastructure, organization, and the other considerations.

However, there are patterns to successful enterprise deployments of UCM that we can certainly draw knowledge from. Some of the patterns have been hinted at.

Avoid the extremes of having either one instance for your enterprise or having 1 instance per application. Neither of these approaches will likely produce optimal results for your organization in the long run. The "best" instance architecture is going to be in the middle, where applications share instances when it makes the most sense.

Another pattern is the deployment of multiple instances on the same hardware. It is entirely possible to install and run multiple UCM instances on the same server (or servers for clustered environments). These multiple instances can share a single web server[5] or each can have their own web server. The former is more common. There are several advantages to deploying instances in this "stack them on the same server(s)" approach (depicted in the diagram below).



[4] The term "best practice" assumes some common starting point that all implementations will fall under. This is simply not the case, and it is necessary to take into account all of the considerations listed in this document – as well as others that may be unique to your organization.

[5] Each instance must have a unique web root

First, this approach can be used to get good utilization out of the servers. Instances can be added to the infrastructure and traffic balanced via the load balancers to tune the overall environment. This is like a server consolidation method that allows for getting better value out of your hardware.

Second, this approach takes advantage of the way that Oracle licenses the software. Because the licensing is by CPU, there are no restrictions on the number of instances you can run on server(s) once the CPUs have been licensed.

Third, this approach (if multiple servers are used) allows for flexibility in leveraging the capacity of the servers. In other words, a smaller traffic instance with no HA requirements may run on one of the servers, while larger traffic instances can be spread across all the servers. If the instances are installed such they span all the servers in the cluster, it is possible to do this load balancing dynamically using load balancer configurations.

Fourth, this allows system monitoring tools to also be consolidated in the sense that they can monitor the servers in the cluster rather than a wider distribution of servers.

Finally, this allows for affective scaling. If additional capacity is needed it can be added by adding either additional server(s) (horizontal scaling) or by adding addition CPU / Memory capacity to the existing servers (vertical scaling). The existing applications can take advantage of the new capacity, as can additional instances.

6 Conclusion

Careful planning, with all the considerations in mind, is the best approach to deploying an enterprise content management instance infrastructure that will work for your organization.

Oracle has many customers that have done successful enterprise deployments that resemble the best practices described herein, and to them we are grateful.