IT Production Assessment:

GemFire from GemStone Systems

Prepared by	Dennis Adams
Client	Dennis Adams Associates Limited
Sponsor	Dennis Adams Associates Limited
Document Date	08-Sept-2005
Last Revision	13-Sept-2005

Copyright

Dennis Adams Associates Limited

Dennis Adams CITP, MBCS, B.Sc., has over 20 years of IT experience in the area of IT Infrastructure Management, Monitoring, and Architecture and Strategy.

Based on this experience, Dennis Adams Associates Limited was incorporated in 2002, with the mission statement: -

“To Advise, Assist and Serve the IT Departments of Medium-Sized Companies, to enable them to implement Quality Production Readiness to their IT Infrastructures”

Dennis Adams Associates Limited
114 Pinner View, Harrow, Middx, HA1 4RL, UK
tel: +44 (0)7753 834 804
email: info@dennisadams.net
http://www.dennisadams.net
Registered in England and Wales, Company No. 4456808

Index

Preamble

The purpose of a Production Assessment document is to look at the practical issues relating to the Deployment and Production use of a specific piece of IT Technology. [1]

When a new Technology solution is being evaluated, the primary focus is typically on the Business or Application Development functionality (or "functional requirements") of the product. This is very understandable, since these functional requirements are basic to the decision whether to purchase the product.

However, what is sometimes missed is the significance of the "Production-Readiness" criteria for the proposed solution (often included as "non-functional requirements"). These criteria are important since the real value of an IT solution only becomes realisable once it has been deployed into Production.

Further, it is sometimes forgotten that costs of any solution consist not only of the initial implementation costs and customisation, but also of the day-to-day running and supportability costs. In fact, in some organisations, as much as 75% of the IT budget is spent on the "Support Paradigm", within which we include IT Production.

This document attempts to assess a proposed technology solution from the perspective of "Production-Readiness". It is envisaged that this type of document would provide valuable input and guidance to the non-functional requirements, and therefore assist clients in the overall assessment process.

In order to assess the product, we need to define a set of criteria against which the product should be evaluated. This list of criteria has been created on the basis of many years IT Production experience. The key criteria that we will use to define "Production-Readiness" are:

- *Scalability*
- *Reliability and Stability*
- *Resilience*
- *Backup and Recovery*
- *Security*
- *Monitoring and Management*
- *Supportability*

Following a brief description of the product, the following sections examine each of these criteria in detail.
In addition, we discuss the key issues of Performance and Efficiency, and a brief look at alternative competitive solutions.

[1] This is one of a series of assessment papers from Dennis Adams Associates Limited looking at different IT technologies from the specific perspective of "IT Production", that is to say the management, monitoring and supportability of solutions. These papers may be distributed and quoted free of charge, providing that suitable acknowledgement is given.
For more details on the "Production-Readiness" criteria defined, and further background material, refer to the company web site at http://www.dennisadams.net

Assessment Scope And Approach

Scope

The objective of this assessment is to determine whether or not the product in question would be supportable in Production.

In order to do this, the product is assessed against the "Production readiness" criteria.

This documents is not intended to assess whether the product meets the business or development functional requirements, since this questions would be the responsibility of the IT Development or Business team(s).

Therefore, this assessment should only form part of the full review of the product.

In a more thorough assessment of a product of this type, it would be necessary to include a section on the supplier, and to consider their financial viability, and commitment to the product. This has not been included in this document at this stage.

Approach

This assessment of GemFire was based on documentation and supporting material available from GemStone Systems. It was not possible to obtain copies of the software for physical evaluation, due to time constraints.

GemFire Enterprise Version 4.0 was released in July 2005, and version 4.1 of GemFire Enterprise was released in August. This is the version that has been reviewed for this paper.

Target Audience

The target audience for this document is primarily IT Production Managers and Technical Support Teams. Development issues are out of scope of this document.

Introduction: GemFire

GemFire from GemStone Systems Inc [2] is a high-performance distributed "data fabric" or in-memory cache technology for implementation in a network of distributed systems (Unix, Linux and Windows).

The purpose of Caching Mechanisms

The performance of modern distributed applications can sometimes be hindered by the time required to access information from relational databases, perform object/relational mapping, and parse XML-based data. Some of these problems could be resolved by re-developing key code segments. Others are a result of the inherent latency and overhead of networking protocols, RDBMS schema access etc.

For this reason, many application developers create some form of near-resident cache mechanism, to enable quick memory-based reading of common data. For example, in an Investment Banking environment, this could include:

- Common Static data, such as customer and bank contact information, settlement rules, core products etc.
- Real-time incoming feeds such as pricing spreads.
- Intermediate calculations and data enrichment, such as is required for a risk analysis system.

It has been proven many times that a well-designed and implemented Application Data Cache can result in spectacular increases in performance. In some cases, it can make the difference to the viability of the IT solution as a whole. In some fast-moving industry sectors, the ability to obtain information in seconds, rather than minutes, can make all the difference to the competitiveness of the Business.

In the past, these cache mechanisms were typically designed and developed "in-house", and required development resources to be switched from the principle focus of building business logic. In addition, in-house developed caching mechanisms require ongoing development support and maintenance, which also detracts from the development teams' core remit.

For this reason, it makes sense to have industry standards defined for caching mechanisms, and for commercial and open source products to be made available to support these standards.

[2] GemStone Systems Inc is a privately-owned software company based in Oregan, USA. The Gemstone web site is http://www.gemstone.com

Industry Standards for Caching Mechanisms

GemStone Systems, Inc was one of the members of the Expert Group that participated in the creation of the JCACHE Java Specification, JSR107 [3].

JCACHE ("Java Temporary Caching API") specifies an API and semantics for temporary, in memory caching of Java objects, including object creation, shared access, spooling, invalidation, and consistency across JVMs.

The original functional specification for JCACHE (also known as "Object Caching Service for Java" or OCS4J) was written by the Oracle Corporation and was based on work done on the Oracle 9i distributed caching mechanism. The specification is currently (September 2005) in the "Early Draft" process.

JCACHE is therefore likely to become part of a future JAVA standard, based on significant practical industry experience.

At this stage, there are a number of commercial and open source products available that implement all or part of the JCACHE standard. In practice, it is quite common for advance products to be released during the evolution of the standard.

[3] The Java Community Process Export Group for JCACHE was formed in April 2001. Details of the Java Specification Request JSR 107 can be found at http://www.jcp.org/en/jsr/detail?id=107.

GemFire Functionality

GemFire is a Distributed Caching technology, which implements many of the features of the JCACHE standard, including the following:

JCACHE Feature	Description
Regions	The ability to store cache data in segregated caches, according to application and/or function.
Distributed Cache Consistency	The caching service maintains cache-to-cache consistency using a broadcast reply mechanism.
Administrative Interface	An interface to enable system administrators to view the cache objects and manipulates them if necessary.
Discovery Address	A single IP address and port number can be used to access the cache. (In fact, the original specification allowed for multiple IP addresses where the cache is distributed, and GemStone have been able to retain a single IP "discovery node", even though there may be many machines involved in the caching.).

In addition, GemFire have implemented a significant amount of additional functionality to the basic caching mechanism.

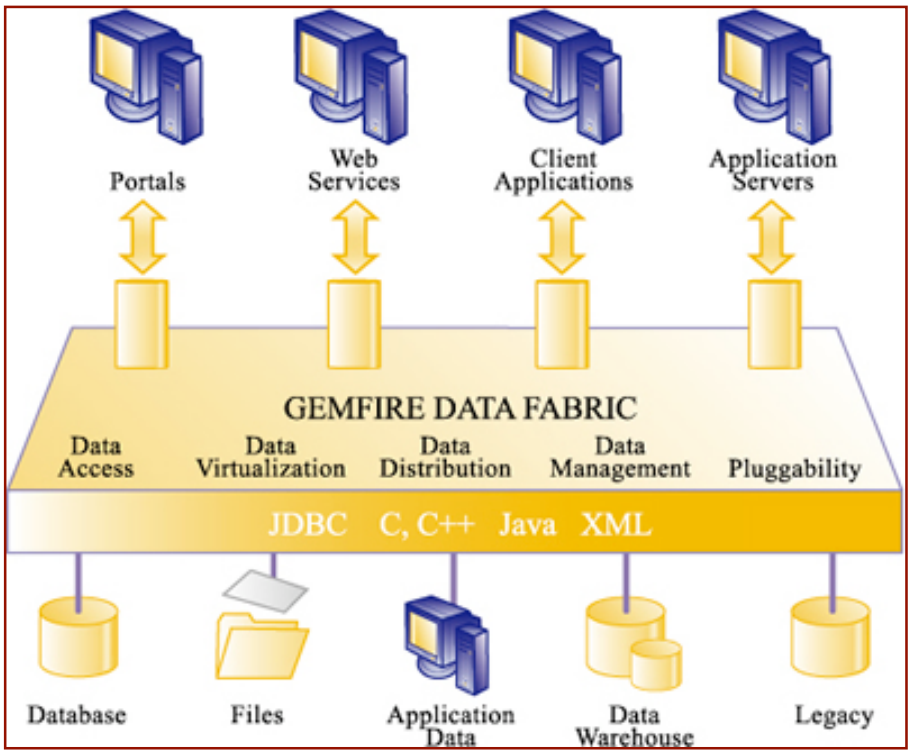
Some of the key features of GemFire are as follows:

GemFire Feature	Description
Data Caching	A distributed, synchronised cache, which can be deployed over multiple hardware nodes.
Data Distribution	As data in the cache gets updated, notifications can be sent to the applications that have explicitly registered interest in these updates across many nodes
Data Virtualisation	Many different data sources can be combined together to appear as a single data source, thus decoupling the logical data structures from the multiple physical data sources.
Multi-way Replication	A piece of data can be replicated to multiple other instances of the cache. This both removes any chance of a single point of failure, and also enables cache data to be located closer to the target "sink".
Active Events	An Event Processing mechanism enables data to be analysed when it arrives in the cache, and to run business logic directly within the cache system.
Distributed Cache API	A common API enables the entire distributed cache to be presented as a sinlge logical entity. This is either done using IP multicast, or by using a "Locator". This is the single IP address and port number that acts as the front-end to access the cache.
Support for multiple types of cached object	As well as JAVA objects, GemFire also supports caching of XML objects and other object types.
Distributed Locking mechanisms.	There is a hierarchy of locking mechanisms available to meet different latency and replication needs.
Security Mechanisms	SSL security can be used to ensure that only know "clients" access the cache.
Administrative tools and API	GemFire has implemented the JMX extension to enable monitoring and management of the fabric.

GemFire sells this solution as a “Data Fabric”. This is a good analogy. The Cache mechanism provides a virtualisation layer between the client applications and the data source, in much the same way as a physical fabric in. for example, a Switched Fabric SAN.

The analogy is also useful, since many of the attributes that are necessary in a SAN Fabric (no single point of failure, virtualisation, guaranteed delivery, ACID transactions, management tools, horizontal scalability etc.) are also necessary in a memory-resident cache.

The architecture topology of GemFire is shown below:



Scalability

Definition

One of the key benchmarks for deciding if a product is "Production-ready" is whether or not it can scale to the number of users / application instances etc. which may be required. More importantly, as the number of end-users or application instances increases, how much (proportionally) additional hardware etc. is required in order to deliver the extra capacity?

"Linear Scalability" by throwing hardware at an application is very unlikely. In practice, some near-linear scalability could be achievable. Sometimes the application architecture (or threading model) means that it works better on a single-CPU environment, rather than being able to take advantage of a multi-CPU system. In addition, sometimes applications are written with subtle in-built limitations that prevent them scaling. [4]

As well as "Scalability", we are also interested in "expandability", i.e. to what extent can the application be enhanced and expanded to adapt to possible future requirements.

Assessment

Scalability must be a key in a caching product, and GemFire appears to have addressed this very well.

There are several types or cache mechanism which can be implemented:

- Maintain the cache locally in the JVM "heap" (sharing the space with the application heap)
- Maintain the cache in a shared memory segment of the local Operating System. This involves running the "gemfire" program on the target system.

Typically, the second scenario is where GemFire will become useful for enterprises.

In this configuration, the cache can scale to 1.7GB on Windows or many Linux systems. The maximum size under Solaris is 4GB.

If the cache tries to grow beyond these limits, it automatically overflows to disk, where it is stored using a hashed tree index. A "Least Recently Used" algorithm is used to determine which objects are flushed to disk.

One of the key features of GemFire is that it supports distributed caching, i.e. parts of the cache can be stored on different machines on the network, and appear to the end-user application to be part of a single massive cache. This means that the memory limit of any individual machine is no longer a practical limitation.

[4] Dennis Adams Associates web site quotes an example of where a database used a "process_id" counter, internally, to keep track of each user thread it was responsible for. This counter was, at one time, a 4-byte integer. Net result: it was physically impossible to have more than 32,768 users on the system at any one time - irrespective of the amount of hardware.

The GemFire distributed system consists of any number of member caches that are connected to one another in a peer-to-peer fashion, such that each member cache is aware of the availability of every other member at any time. The GemFire distributed cache API presents the entire distributed system as if it were just one logical cache, completely abstracting the actual location of the data or the data source from the developer. A single unique IP address and port is used to identify each distributed cache system. This creates a virtualisation layer, and has the benefit of providing a simple access path to the cache, and leaving the "Locator" to redirect clients to the appropriate physical system(s).

Each member cache is made up of one or more cache "Regions", which may further contain sub-regions, effectively creating hierarchical regions. Each cache region manages a collection of application objects. Each region carries a set of configuration attributes that control where (the physical location) the data is managed, the distribution characteristics and the eviction and consistency models. Use of Regions enables multiple applications to use the cache in a federated way.

When replicating data, different types of replication can be configured.

It is also possible to arrange caches in a hierarchy, with local caches being "fed" from a larger cache server. A request to the client cache which results in a cache "miss", leads to a request being delegated to the server cache. A "miss" on the server will typically result in the data being fetched from the origin data source.

Reliability and Stability

Definition

For the purposes of our assessment, Reliability is concerned with the extent to which the application will deliver the expected results in a consistent and repeatable fashion, irrespective of changed load and/or changed environmental circumstances.

In addition, we will define "Stability" as the ability to be able to run unattended for long periods of time without operational intervention. Reliability therefore has to do with predictable, repeatable behaviour, whereas Stability has to do with repeatable behaviour over time.

For example, some applications may be initially reliable, but their performance and/or reliability degrades over time, due to memory leaks, resulting in the necessity for the application to be restarted. This is a classic example of an application that is "Reliable" (i.e. behaves properly when it works), but at the same time it is not "Stable" (i.e. it degrades over time).

Assessment

We have not been able to obtain a copy of GemFire at this stage, so we are unable to comment on the stability or reliability of the product in practice. However, the technical description does give an indication of the likely behaviour.

GemFire can be configured to store the entire cache region persistently on the local disk. To optimize disk writes, the cached regions can be configured to write all changes to disk at regular intervals rather than synchronously. This option should only be used by applications that can tolerate incomplete recovery upon failure.

When configured to do distributed caching, it is important that the data is reliability synchronised, so that data is not lost or corrupted. GemFire implements the JCACHE semantics for distributed caching. Updates from one part of the cache to another can be done several ways:

- Asynchronous updates
- Synchronous updates
- Use of Global Locking.

In addition, GemFire supports a JMS plug-in to enable customised messaging to be implemented.

Resilience

Definition

"Resilience" can be defined as the ability to recover quickly from a failure of one or more components that make up an overall system.

Resilience therefore differs from Reliability and Stability. Reliability and Stability are concerned about behaviour under load and behaviour over time, assuming that all the components are in place. Resilience is concerned about how the system behaves if a component is lost.

A Resilience assessment takes the view that "if something could go wrong, it will do so". At a very basic level, Resilience of a Server is implemented by using dual power supplies, RAID storage controllers, Redundant Parity RAM etc. In general, it is expected that these low-level resilience issues are addressed adequately in modern high-performance production systems.

In the context of this paper, Resilience assessment is concerned with how to implement Clustering mechanisms to guard against the possibility of failure of an Operating System, and how to ensure that there is no single point of failure within the architecture.

This section should also include an assessment of how to implement Disaster Recovery mechanisms, and how to implement off-site recovery.

Assessment

GemFire can be configured to store the entire cache region persistently on the local disk. The I/O writes to disk can be either synchronous or asynchronous. Obviously, for maximum resilience, the synchronous writes option should be used.

The distribution system is designed such that members can join and leave at any time without impacting other member caches. However, it would be necessary to test this, to demonstrate what the impact would be (if any) on a cache member being suddenly withdrawn (for example due to an Operating System failure).

Data can be replicated between caches, so that there is a standby cache available in case of a failure.

For replicating data across many cache instances, GemFire offers the following options:

- "Replication on demand": Data object is replicated to where it's used. (A'PULL' model). In this model, the object resides only in the member cache that originally created it. Objects arrive to other cache members only when the connected applications request the object.
- "Key replication": Only the keys of the objects cached are replicated - A'PUSH' model. (This model can preserve network bandwidth and be used for low bandwidth networks)
- "Total replication": All data is replicated (A'PUSH' model)

In practice, only "Total Replication" would meet the requirements of a resilient standby system.

Backup and Recovery

Definition

Backup and Recovery should be supported by applications and systems for two distinctly different reasons.

Firstly, Backup extends the idea of Resilience - i.e. how to respond to failure of a component - to look at how to respond to the failure of all components. This is typically implemented by using backup & recovery techniques. [5]

Secondly, Backup can be used in order to recover the system to a known stage at a specific period of time. There may be a number of reasons for this. One reason might be that some business logic (or dependant application) has resulted in corruption and it is necessary to go back in time to recover. A second reason may be to build an "archive" or "historical" copy of the application for the purposes of analysing historical trends, or setting up a test or development environment.

"Backup and Recovery" can often has subtle implications for the design of systems. It is axiomatic that to just backup the disk contents at Operating System level is not adequate if the application is expected to perform "ACID" transactions. [6] *Therefore, it is necessary to have the ability to identify the begin-end point of business transactions so that data is consistent*

Assessment

Whilst this topic is fundamental to most IT Production Assessments, in the case of GemFire, there is very little which needs to be considered.

The nature of a cache is that it is a temporary copy of persistent data. Therefore, the key issue is whether or not the persistent data is maintained in a consistent way. The real issues with caching, therefore, have to do with Resilience, Stability and Reliability, which are covered in the sections above.

---

[5] For example, failure of an entire data centre may be addressed by implementing a live standby data centre (resilience), or by taking off-site backups which can be used to re-create the application on a replacement machine elsewhere (backup & recovery). In some cases, an application may be so resilient that there is no necessity for backup or recovery for the purpose of guarding against failure.

[6] The basic properties of a database transaction are: Atomicity, Consistency, Isolation, and Durability.

- Atomicity - The entire sequence of actions must be either completed or aborted. The transaction cannot be partially successful.
- Consistency - The transaction takes the resources from one consistent state to another.
- Isolation - A transaction's effect is not visible to other transactions until the transaction is committed.
- Durability - Changes made by the committed transaction are permanent & must survive system failure.

Security

Definition

"Security" is one of those aspects of IT development and deployment that can sometimes be ignored in the early stages of software design.

In this context, we are concerned not only with the security of the application as presented to the end user (e.g. the ability to implement IP fire walls, packet filters etc.), but also with isolation of the Production Application from any development / test versions. For example, some applications have a documented API that enables any developer to call the business logic. Under those circumstances, what is to prevent a developer (either deliberately or by mistake) from calling the Live Production business logic from within an application subnet?

This section is concerned with the following basic security principles:

- *Authentication – is the person or object attempting access who they say they are?*
- *Authorisation – is the capability of this person/object clearly defined & appropriately restricted?*

Assessment

GemStone have identified the importance of security for this type of widely distributed system, and have provided a means of using SSL security to communicate between different cache elements. This may well have a significant overhead, however, and it can be enabled or disabled as the client requests.

We would like to see some form of Registration / Firewall feature in GemFire, so that remote access is only allowable from pre-specified IP addresses. This would be a lower cost solution than using SSL. Although it would not prevent IP spoofing, it is suggested that this would be an adequate level of security for many medium- or large- enterprises.

One of the unique features of GemFire is "Active Events". This effectively means that Business Logic can be added to the Cache, to identify data changes and act accordingly. There are security implications in allowing Cache Configuration (typically a technical, administrative function) to be merged with Business Logic Configuration (a Developer or Business function). As a matter of best practice, we would recommend that this feature be not used. However, should it be required, it would need to be interfaced to Version Control, Change Control and other mechanisms, to ensure that there all changes are properly authorised, controlled and auditable.

The GemFire Cache is federated into Cache "Regions", which assists management and access. We would like to see some form of "security-by-region" implemented, to "Chinese Wall" the different regions. It is not clear whether this is possible.

Monitoring and Management

Definition

Monitoring and Management is a key part of the day-to-day function of any IT Production Team.

One of the purposes of monitoring is to pro-actively identify any adverse changes in the behaviour of the system and/or it's environment, in order to take appropriate corrective action before the change impacts the business client. For this reason, "Monitoring by exception" is most appropriate. Implementing Asynchronous SNMP traps are one way of achieving this.

A second form of monitoring is "trend analysis", the purpose of which is to extract time-series data in order to model the long-term behaviour of the system and to collate it against business trends for Capacity Planning purposes. [7]

Management is also another key role in IT Production. In this case, were are concerned with how easy it is to amend or adjust the configuration of the application, and adjust it's environmental behaviour. The important point is that such configuration should be as automated (and intuitive) as possible, in order to minimise IT Production costs for supporting the running application.

Assessment

It is refreshing to see that use of a management console and API included in GemFire as part of the fundamental architecture. The console enables the contents of the cache to be inspected by an administrator. This shows that GemStone have considered the needs of Production Administrators.

For Troubleshooting, it is possible to turn on Logging for each cache individually. Logging can be turned on at various levels of detail. The logging information generated from each cache can be consolidated to analyze the events across an entire distributed system. GemFire also provides a tool to merge log files allowing the developer to quickly troubleshoot a problem that may span several nodes. The logging system is also exposed via an API for applications to log messages. There is every indication that this has been a well thought-out solution.

To Quote the GemFire Documentation; *In addition to logging, GemFire gathers a lot of statistical information such as cache hit rate, number of gets, time for various operations, application process level stats, related OS stats, distribution message stats, etc., providing a very fine level of monitoring. The various system statistics are captured in memory and a daemon thread sweeps the stats at configurable intervals and archives these to a "statistics" file on local disk.* This appears to provide all that a Capacity Planner would require.

[7] The Capacity Planning role within IT Production needs to look at two aspects of IT growth:
- Trend analysis – i.e. what is the latent growth rate over time, and
- Business Metrics, i.e. how do changes to the no of business transactions correlate with the IT demands.

The console also provides a special charting tool to monitor, chart and correlate system statistics. This additional feature further demonstrates GemStone's commitment to enterprise-level Production deployment.

Very often, end-users need to integrate the management and monitoring into enterprise-wide tools such as CA Unicenter, HP OpenVIew, etc. This could be done by defining a set of managed beans (MBeans) under the JMX API, so that third-party adaptors can be plugged in. For example, it may be possible to add an adaptor to generate SNMP traps. There is also a JMX agent that can be used to help manage the system from a single entry point.

It is also possible to use JMX to start / stop the caches using in-house customised software.

Supportability

Definition

"Supportability" can be defined as the features which make the application or system able to be supported by a "Business as Usual" IT team. This is a general extension of the concepts of Monitoring and Mangement, above.

The significant issue with the "Supportability" assessment is whether the application can be supported at a reasonable cost. In practice, this means ensuring that we can minimise the amount of manual intervention required to keep to application at it's appropriate level of activity.

Assessment

It is difficult to assess supportability without the opportunity to try out the software in practice. However, it is clear from reading the documentation that GemStone have put some significant thought into how their product would be deployed enterprise-wide.

One small feature that would be of great help to developers is "gfgen". This is a simple utility which takes the XML description of an object structure, and generates "C" headers or Java Class definitions from it. Although not within scope of an IT Production Assessment, this sort of utility does show that GemStone have really thought about how the application will be used in real life.

Supportability is greatly aided by the fact that there is a very powerful GUI console, as well as command-line interface. It is possible to turn on logging, to gather statistics, and generate graphics to track the behaviour of the caches.

Performance and Efficiency

Although strictly outside the scope of this document, it is important to include a discussion of the Performance and Efficiency of GemFire.

Obviously, the fundamental reason for a cache mechanism is to provide fast up-to-date and reliable access to persistent data (such as relational databases etc.)

In this context, "Performance" is concerned with how well (typically; - how fast) the application delivers the required functionality. "Efficiency", on the other hand is concerned with amount of resources (IT or otherwise) that are required in order to deliver the functionality concerned. For example, it may be that an application has a very high performance, but only at the cost of a large investment in hardware and software. In this case, we would conclude that we have an application which is has a good Performance, but not very good Efficiency.

As a Caching Mechanism, GemFire appears to have been well thought-out.

However, we have a concern that the multiplicity of additional features that have been added to the standard Cache concept may have some adverse impact on the Performance of the final deployed solution. One of these features is "Active Events", which enables data enrichment within the cache itself. Effectively, this is introducing business logic to the cache. In our view, this feature could have an adverse performance impact on the cache itself. Sometimes, a KISS [8] approach is more appropriate for this type of technology.

Access to the Cache is via a single IP Address and port number. This then results in an IP multicast to "discover" all the physical members of the cache. Use of IP multicast would be forbidden in some IT Production environments. Therefore, GemStone have created an "Locator" service which is used to "discover" the nodes instead. This would (hopefully) have a lower impact on the network than IP multicast.

[8] KISS = "Keep It Simple, Stupid!". Neither the acronym nor the sentiment is unique to Dennis Adams Associates Limited. However, our many years experience in IT Production means that we are strong advocates of the benefits of a simple, modular approach to solutions.

Alternative Competitive Solutions

This section looks at some of the alternative competitive approaches or solutions which may be compared with the product being assessed.

JCS

The Apache Jakarta programme has a project called JCS which is an Open Source distributed caching system for server-side Java applications [9]

JCS is a very complex piece of software which contains many of the features which have been specified in JCACHE, including Disk overflow (and defragmentation), Element grouping, Quick nested categorical removal. Data expiration, Remote synchronization, Non-blocking "zombie" (balking facade) pattern, Optional lateral distribution of elements via HTTP, TCP, or UDP and Remote server chaining (or clustering) and failover.

However, JCS has not implemented all of the JCACHE features because they claim that there are some aspects of the JCACHE architecture that could lead to inefficiency (ex, the lateral distribution and net searches). These assertions would need to be tested in practice.

Also, JCS is designed for Java objects only. Unlike GemFire, it does not support other object types such as C++ objects, XML documents etc.

JCS comes with a very comprehensive user's guide, and appears to be easy to set up and configure. Like GemFire, the JCS developers have been concerned about the implicit single point of failure of the initial "Locator", and have implemented a different model.

JCS does not support Cache Loaders.

Times Ten (Oracle)

Times Ten is a RealTime data management system (or in-memory database) which was acquired by Oracle Corporation in June 2005.

Times Ten already used elements of the Oracle Caching architecture, and an option called "Cache Connect to Oracle". With this option, TimesTen users effectively set up an in-memory cache for their Oracle databases.

The disadvantage of Times Ten is that it is optimised for use with Oracle only, and does not support any other object types.

At this stage, it is not clear what the future of this technology will be under Oracle, although it is possible that it will sold as an alternative solution to the original Oracle RDBMS to Banks and other institutions.

[9] For more information on the Java Caching System, the appropriate Jakarta web site is at http://jakarta.apache.org/jcs/index.html

Conclusions

An efficient, industry-standard caching mechanism, which is IT Production-ready, would be a valuable addition to the IT Infrastructure for many large or medium size organisations.

For example, Risk Analysis systems and large business-to-business web services solutions require large amounts of data to be provided with very slow latency time.

Further, the Banking sector is increasingly moving towards a need for "Same Day Settlement", which results in the need for very high speed messaging systems to feed into complex transaction settlement systems. GemFire would be able to address such a need.

GemStone have raised the bar as far as Cache technology is concerned, since they have added the "Active Events" to the cache architecture. This effectively means that Business Logic can be added to the Cache, to identify data changes and act accordingly. The danger with this approach is that it may be attempting to be too clever, and in so doing may impair the pure "cache" functionality. For this reason, and for the security reasons, this feature should be used with caution.

In addition, there is a version of GemFire called "GemFire Real-Time Events", which is designed specifically for handling real-time data feeds., such as market data feeds (prices or bids etc.).

From an IT Production perspective, GemFire appears to be a very well engineered solution to these needs. Subject to actual system testing (which we have not been able to perform, given the time available), we consider that this is a very powerful product that could take its place within the IT Production Infrastructure.

Assessment Matrix

The Main conclusions of this assessment can be presented in a tabular format, as below:

Production Criteria	Theoretical Assessment	Test Results	Notes
Scalability	5	N/A	Theoretically very scalable solution but requires additional testing in order to prove.
Reliability and Stability	4	N/A	Theoretically very reliable solution but requires additional testing in order to prove.
Resilience	5	N/A	Cache can be spread over multiple systems.
Backup and Recovery	4	N/A	Very Resilient Cache, so backup and recovery are not significant
Security	3	N/A	SSL and certificates can be used for messaging.
Monitoring and Management	5	N/A	JMX and enterprise console.
Supportability	4	N/A	Very supportable enterprise-wide product

In most cases, it will be necessary to combine this Theoretical Assessment with specific Tests to identify and prove that the application meets the criteria mentioned. In the case of GemFire, we have not been able to do so because of time constraints. Therefore, there are no test results in this document at this stage.

The meanings of the scores are as follows:

1. Application or System is considered to be totally unsuitable for IT Production use. Costs of support are likely to be prohibitively high if the application or system were ever introduced into IT Production.

2. This Version of the Application or System is considered to be unsuitable for IT Production use, but could be used for software development, and additional discussions with the vendor should be held in order to introduce required features in a future version. Costs of support are likely to be very high if the application were ever introduced into IT Production.

3. Application or System is recommended for deployment into production with some additional customisation required by the client or vendor in order to improve supportability. Costs of support are likely to be in line with costs for other applications of this type.

4. Application or System is suitable for Production deployment, with very little additional customisation required. The client can implement any such customisation, without any necessity for involvement from the vendor. Costs of support are likely to be in line with costs for other applications of this type.

5. Application or System is suitable for Production deployment, with minimal customisation. The vendor has demonstrated a strong understanding of the principles of “Production Worthiness”, which are reflected in the design and implementation of the product. Costs of support are likely to be in line with, or less than, costs for other applications of this type.

In addition to the above, we have also considered the topic of Performance and Efficiency. These are criteria that we are unable to assess properly without the opportunity to conduct proper testing. However, given the high scores that GemStone have gained in the other sections, we have good reason to believe that this would be an Efficient and High Performance solution which would be very suitable for deployment in an IT Production environment.

Sources

This document has been prepared from information provided by GemStone Systems.

- GemFire Executive Summary.
- Gemfire and Ell: A New Approach to Distributed Enterprise Data Management.
- GenFire Technical White Paper. This is targeted at technical architects and developers evaluating or implementing GemFire as an enterprise data fabric. It addresses some important aspects that typically impede an architect such as architectural resilience, systems management, robustness, and security.

The GemStone Systems web site can be found at http://www.gemstone.com

In addition, following Analysts reports were reviewed in preparation of this assessment:

- IDC Analyst White Paper: “Overcoming the Data Bottleneck to Deliver Dynamic IT: GemStone’s GemFire with IBM’s BladeCentre” by Carl Olofson and John Humphreys.

- Forrester Report: "Selecting A Middle Tier Caching Architecture" by

About the Author:

Dennis Adams has over 20 years experience with Managers of IT Production Systems, and now provides a specialised Consultancy Service to help IT Production Managers take a more Strategic view of IT Production Management. He can be contacted via the company website at http://www.dennisadams.net