

Clustering Category

Company: Tieline Research

Team Members: Leigh Carey, Joanne Church, Andy Clark, Peter Fall, Luke Grose, Elester Johnson, Tobias Lauchenauer, Henry Lwin, Gareth Mahon, Alex Shaykevich, Anthony Sizer

Location: Malaga, Australia

Vertical: Broadcast Technology



1 Company description / size (basic company info)

Tieline is a medium sized company established in 1982 to introduce new forms of audio transport technology to the broadcast marketplace, with many success stories including world technology firsts and deployment into Olympic Games broadcasting. It employs staff in Australia and the USA with headquarters in Australia. In 2004, Tieline was awarded an Australian government research grant to explore the use of scalable J2EE technology for real time use in the application of audio and video surveillance for homeland security.

We demonstrated our application, *Shadow*, at a USA security tradeshow in 2005, recently trialed it at a major government agency, and we will roll out our first installation in June 2006. We anticipate significant sales of this product throughout the world.

2 Describe any business and/or technical challenges about the project.

We have faced and overcome both business and technical challenges in this project. These are outlined in the following paragraphs.

2.1 Business Challenges

Because we are building a product (as opposed to a one-off project), we needed to collect requirements from a number of potential customers and blend them into a product specification, while retaining the ability to adapt the product in an efficient manner and to interface it to a variety of legacy systems. Our target market is broad, spanning from many small customers with limited budgets to a select number of major institutions. We needed a scalable solution with low distribution (licensing) costs. We were also aware that larger organizations might mandate that we use their 'corporate choice' database, yet needed to be able to use open source databases for smaller customers.

For critical surveillance operations all but our smallest customers would need 24x7 non stop operation. Our challenge was to provide this in an affordable package.

In order to be credible in the evidence grade surveillance market, we needed to guarantee authenticity and integrity of the data. Additionally, perceived legal requirements make it impossible to store the media data in a database (it must be stored in files).

2.2 Technical Challenges

The major technical challenge that we faced was to tame the non-deterministic behavior of J2EE containers. We have upwards of 400 channels of audio/video data arriving at a various fixed rates. Every 20ms, data arrives from each channel, and the system must process it or the data piles up and gets lost. Data integrity is critical to evidence grade surveillance operations. Because media data needed to be stored in files rather than in the database, standard database clustering would not assist with load sharing and redundancy with respect to media storage. We needed to come up with a solution for clustering file systems in a reliable manner.

Part of our solution was to implement a J2SE front-end to reduce the frequency of application container requests. There were also technical challenges in clustering the J2SE servers.

3 What was your solution?

3.1 J2EE for adaptation

We selected the J2EE architecture as it offers many industry standard mechanisms for interfacing to third party systems. We elected to develop a Swing based thick client as this would provide the functionality and client side processing required. However, J2EE offers an easy expansion path to a thin client application and SOAP based interaction with other systems. Transaction management, security, and performance are key aspects of our system.

3.2 J2EE for scalability and clustering

Both scalability and clustering for load sharing and redundancy are key requirements. We decided to meet these through the use of J2EE technology. The system needs to scale to an arbitrary number of channels. We chose a stateless clustered solution to give us maximum flexibility.

3.3 JEMS because of its user base, stability and royalty-free distribution model

JEMS is an ideal product for us as it is relatively easy to find developers who are experienced with it, it is stable, well supported, and incurs no royalties for distribution. We take advantage of the pluggable and extensible nature of the stack by using extra components such as JBoss AOP, JBoss-Cache, and JBoss Serialization. We modify the microkernel to suit our needs.

3.4 Sequoia for database clustering

We considered a number of database clustering products: Oracle RAC, IBM DB2, MySQL cluster, EMIC, and Sequoia. We evaluated each of these against our criteria of license fees, stability, and ability for node recovery in a two node cluster. The Sequoia product stacks up well against these criteria. Sequoia uses controllers to perform replication functions across a number of database nodes. These controllers can be replicated in a controller load sharing configuration, or operated individually in a master / standby configuration. We found that the load sharing configuration was too slow – with the inter controller protocol being the bottleneck. However, the CPU loading of the controller is light, so loadsharing the controller was not mandated and we found it satisfactory to run the Sequoia controllers in master / standby mode. This still allowed sharing of read request loads across the clustered database nodes. Sequoia also provides table striping across nodes, although we have not used this feature.

3.5 JGroups for file system clustering

We have a business requirement to store media data (audio and video) in a file system, rather than a database (We use the database to store meta data for the media data). We needed then to provide a clustered filesystem so that the media data availability would not suffer from a single point of failure. Our solution was to use JGroups to manage the availability of a cluster of nodes and to provide a broadcast system for the issuing of file store requests.

3.6 JBoss Cache for J2SE server clustering

Media data is transmitted to the system by codec equipment every 20ms. In order to cope with this and provide an accurate timestamp for the data, we employ a J2SE component (Gateway Server) which performs the timestamping, initial processing, and packaging of the data into a *Quantum Frame*. The size of this Quantum Frame was determined empirically according to the load.

JBoss Cache is used within our Gateway Server component to store and replicate active session state across a loosely coupled cluster of J2SE servers. Each node within the cluster is aware of the others and holds a view of the other nodes within the group. If there is a change within the view, such as a node failing, then the current cluster controller (usually the longest surviving member of the cluster) distributes any sessions from the failed node to other surviving nodes, moving state information from the failed node's tree in the cache to the area belonging to the new owner of the session.

3.7 Data aggregation and buffering to address the non-determinism of the application container.

Empirical data showed that the application container response time was generally linear with load until a limit was reached after which asymptotic behavior was observed. Nevertheless, within the linear zone, the response time was significantly variant according to the instantaneous load. We determined that a reasonable hit rate on the application container was one record request every 2s for each recording stream.

In order to achieve this, we used the J2SE Gateway Server component (generally deployed on a separate node) to process the RTP media data that arrives every 20ms and aggregate it into a *Quantum Buffer*. Once this buffer is full, it is placed into a FIFO queue. A separate thread removes buffers from this queue and invokes the J2EE record request service. This decouples the variance of the application container response time from the real time requirement to process data every 20ms. We evaluated and selected BEA JRockit and tuned it to select low pause times for the garbage collector.

3.8 EJB 3.0 to provide application container distributed cache

In order to provide scalability and fault tolerance through application container replication, we selected EJB 3.0 as it allows us to use distributed cache invalidation to significantly improve performance. It also allows for more flexible modeling of our domain.

4 Describe your vendor selection process and reasons for choosing JEMS.

In selecting a middleware vendor, we considered the following factors:

- Ease of finding experienced developers
- Stability
- Availability of support if required
- Good documentation
- Royalty free distribution
- Open source means that we should be able to fix any problems

JEMS was the only product we evaluated that met all these criteria.

5 What role did JEMS products play in the solution?

JBoss Application Container

- We used JBoss to provide a scalable application container for use case processing.

JGroups File system clustering

- We used JGroups to implement node awareness and broadcast communications to implement a clustered filesystem for storing media data.

JBoss Cache J2SE clustering

- We used JBoss Cache to implement clustering of the Gateway Server. This provides a global state for the cluster and allows a gateway server node to assume responsibility for an incoming stream in the event of failure of a peer node.

6 What was the overall impact of the project on the business? (eg: ROI, competitive advantage, time to market?)

The use of JEMS products has provided us with the opportunity to use off the shelf components to solve many of our problems, and avoided the need to develop additional custom software. We were able to focus on developing our application specific software.

Our target markets span from a large number of customers needing a desktop solution, to a smaller number of customers needing large fault tolerant clustered solutions.

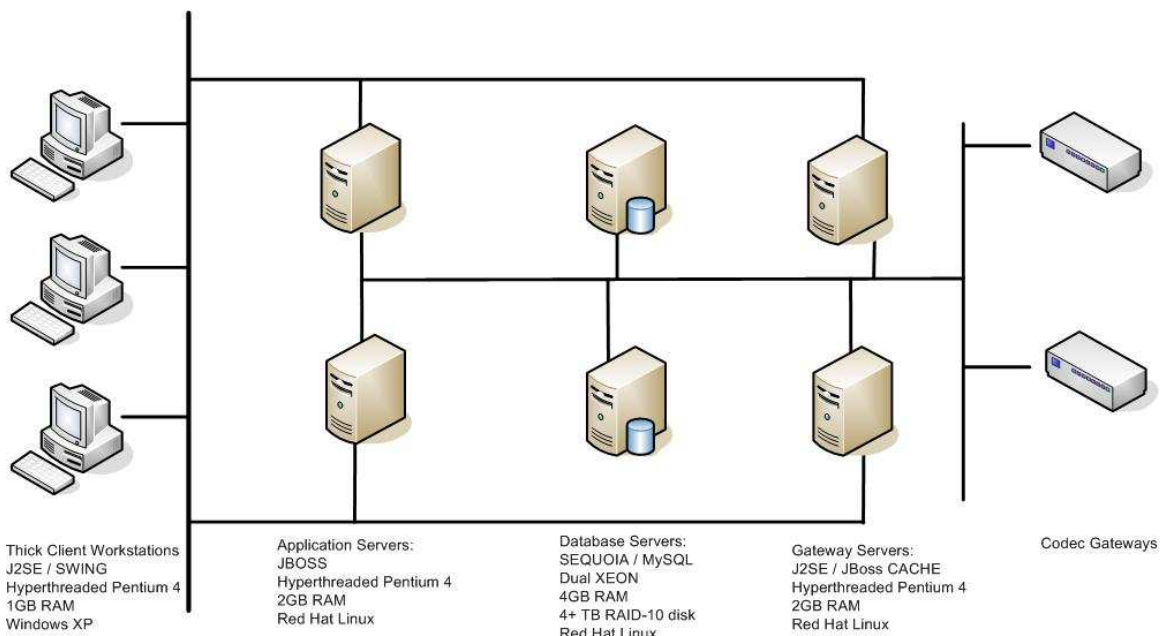
Selecting JEMS has allowed us to easily develop a single scalable solution which gives us a competitive advantage by allowing our customers a scalable path to expansion as their business grows.

7 With the savings gained from implementing JEMS, how did you reallocate your cost savings within your company?

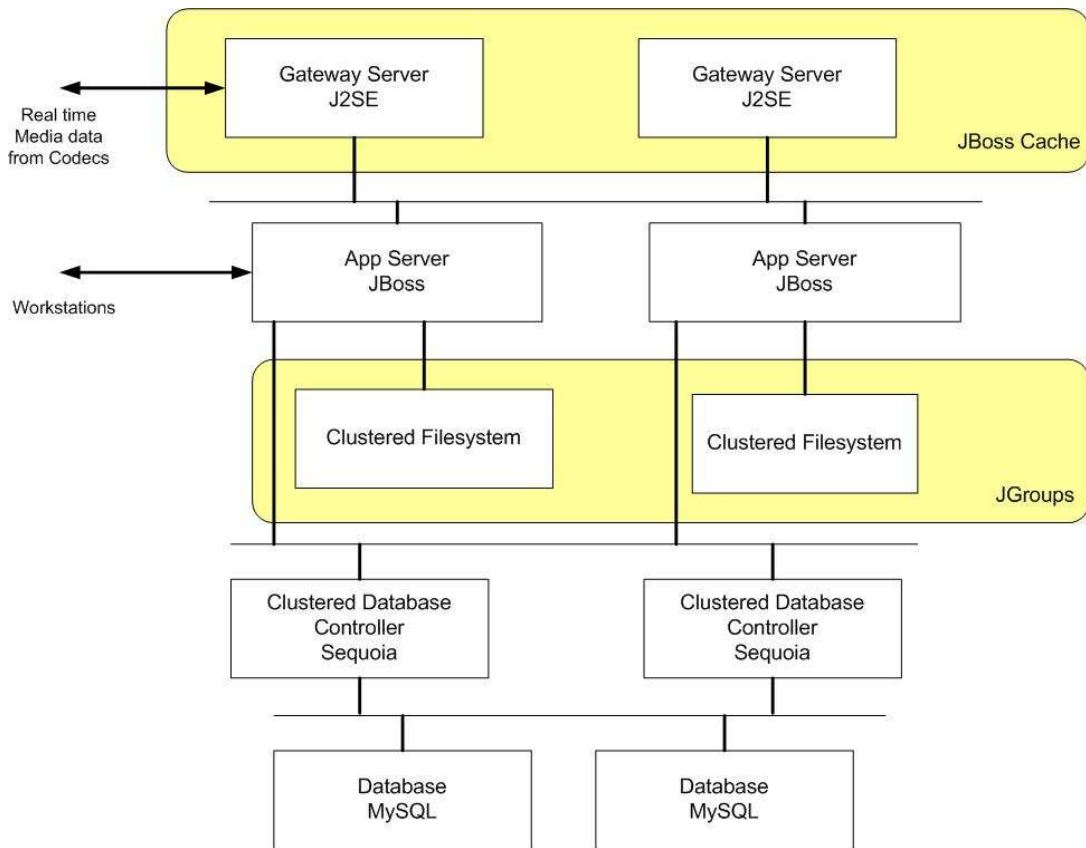
Cost savings have allowed us to tackle the project more aggressively and direct additional research and development funding into other product development areas and hire more staff. These cost savings have been instrumental to the viability of this project.

8 Technical description of implementation/size of deployment? (Hardware, applications, o/s, databases, etc.)?

The system is designed for many different deployment scenarios – from everything on a Laptop to a large fault tolerant scalable system. The diagram below shows a typical small fault tolerant deployment. Larger systems can be built by scaling up the number of servers in each tier. A private network keeps internal traffic off the corporate LAN.



Small fault tolerant deployment



Overview of Software architecture

9 Did you leverage JBoss support services, training or consulting? If so, how was your experience?

Our experience with the ease of use of JEMS and the quality of documentation meant that we have not yet needed to use the JBoss support services.

10 Advice to other companies considering JEMS

Our experience with the JEMS stack is that it is very easy to customize to meet our needs whilst being powerful enough to fulfill complex requirements. The forums are very active, so finding solutions is easy. Bugs are fixed quickly and issues are addressed in a logical manner. We encourage others to try the JEMS stack and hope they have as much success with it as we have.