REFERENCIA: SUÁREZ PÉREZ, J. M., FUENTES DÍAZ, A., BECERRA TORREIRA, Y.: "Arquitectura para Sistema Gestor de Procesos de Media"

En REVISTA DIGITAL SOCIEDAD DE LA INFORMACION, Nº 38, 2012. (http://www.sociedadelainformacion.com)

Arquitectura para Sistema Gestor de Procesos de Media

AUTORES: Jean Michael Suárez Pérez [1]*, Adnan Fuentes Díaz [2], Yesleny Becerra Torreira [3]

[1]Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, Km 2 ½. Boyeros. Ciudad de la Habana, Ciudad de La Habana, Cuba. jmsuarez@uci.cu

[2] Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, Km 2 ½. Boyeros. Ciudad de la Habana, Ciudad de La Habana, Cuba afuentesd@uci.cu

[3]Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, Km 2 ½. Boyeros. Ciudad de la Habana, Ciudad de La Habana, Cuba ybecerra@uci.cu

SOBRE LOS AUTORES

Jean Michael Suárez Pérez: Profesional graduado en el año 2009, labora en el proyecto Sistema Gestor de Procesos de Media del departamento Señales Digitales de la Universidad de las Ciencias Informática. En la producción cumple los roles de desarrollador y jefe de proyecto. Está categorizado de instructor.

Adnan Fuentes Díaz: Profesional graduado en el año 2011, labora en el proyecto Sistema Gestor de Procesos de Media del departamento Señales Digitales de la Universidad de las Ciencias Informática. En la producción cumple los roles de desarrollador, gestor de configuración y arquitecto, es el jefe del grupo de arquitectos del departamento. No tiene categoría docente.

Yesleny Becerra Torreira: Profesional graduado en el año 2009, asesor de calidad del departamento Señales Digitales de la Universidad de las Ciencias Informática. Está categorizado de instructor.

La Habana, octubre de 2012

RESUMEN

Las tecnologías grid permiten resolver problemas con gran demanda computacional y de datos, teniendo en su definición una característica muy importante dentro de los sistemas distribuidos, como es la compartición de recursos. Se propone una arquitectura grid para un producto de software orientado al procesamiento de materiales audiovisuales, que sea distribuido y descentralizado, capaz de aprovechar recursos de hardware geográficamente distribuidos, que realice un adecuado balance de cargas a los nodos de procesamiento y que garantice el funcionamiento constante del sistema, incluso cuando existan fallos en el mismo. El trabajo expone las tecnologías de desarrollo y de comunicación para los distintos sistemas, poniendo en práctica la arquitectura propuesta así como los resultados obtenidos después de desplegar la solución en varios entornos. Se muestran además alguna de las vistas importantes de la arquitectura para que todos los interesados en el desarrollo de la solución informática comprendan el proceso de desarrollo.

PALABRAS CLAVES: arquitectura, gestión de procesos, grid.

ABSTRACT

The grid technologies allow to solve problems with large computational and data demands, taking in its definition a very important feature in distributed systems, such as the sharing of resources. In this paper we describe the most significant characteristics of grid systems and its architecture. It proposes a grid architecture for a software product oriented media processing to be distributed and decentralized, able to harness geographically distributed hardware resources, to conduct a proper load balance the processing nodes and to ensure continued system operation even when no faults therein. The paper describes the development technologies and communication for different systems implementing the proposed architecture and the results obtained after deploying the solution in multiple environments.

KEY WORDS: architecture, grid, process management.

INTRODUCCION

La creciente complejidad de los proyectos científicos, en la actualidad, ha provocado que para su investigación y puesta en marcha es necesaria la implementación de novedosas y elevadas estrategias computacionales. Las aplicaciones en las que se desarrollan estos proyectos son cada vez más complicadas, demandantes de potencia de cálculo, así como grandes consumidoras de datos. Muchos de estos proyectos, además de requerir de una amplia capacidad computacional y de almacenamiento de inmensas cantidades de datos, necesitan la colaboración de varios sistemas informáticos que realizan labores como: codificación, indexación, captura, transferencia, notificaciones, transmisiones, etc. Estos sistemas, así como los recursos de los que disponen, pueden pertenecer a una misma área en la que realizan sus tareas, pero encontrarse distintamente distribuidos geográficamente.

La utilización eficiente de estos recursos es todo un reto, aunque los mismos se encuentren dispersos deben ser operados conjuntamente como un sistema. Con el presente trabajo se propone una arquitectura para un sistema informático que sea capaz de soportar la gran demanda de ejecución de procesos sobre archivos multimedia, permitiendo que se incremente la potencia del sistema (recursos de hardware) y la integración con terceros. Así mismo debe asignar las tareas a los computadores de acuerdo a sus capacidades de procesamiento, logrando un uso racional de los recursos disponibles.

“Las arquitecturas Grid [1] son muy utilizadas desde el año 2000, considerándose novedosas y poco costosas para los sistemas que la implementan. Los nodos se agrupan más allá del dominio de una LAN (Local Area Network), como sucede en los Clusters (1), y en donde los recursos pueden ser heterogéneos (diferentes arquitecturas, supercomputadoras, clusters, etc.). La idea de “La Grid” es un concepto más que ambicioso, ya que su fin es darle potencialidad a un sistema de nodos interconectados logrando que puedan procesarse operaciones desde cualquier nodo aunque este no cuente con los recursos suficientes siendo las tareas compartidas en todos los recursos que la componen. La Grid permite que instituciones de menos recursos tengan acceso a poder computacional de forma remota, o que diversas

instituciones puedan unir sus recursos computacionales para obtener uno más poderoso.

DESARROLLO

La arquitectura es "*la estructura de estructuras de un sistema, la cual abarca componentes de software, propiedades externas visibles de estos componentes y sus relaciones"* (**Software Architecture in Practice - Kazman**), es el nivel más alto de un sistema en su ambiente.

La solución de software que se expone se encarga de automatizar una serie de procesos que se realizan en las Televisoras. Después de estudiar el flujo de trabajo de estas empresas y de sistemas de software existentes en el mercado mundial como TdMpm, Pro MediaCarbon y TDIndexer, se han identificado algunos requisitos funcionales como:

- Codificar archivos multimedia de alta calidad y tiempo de duración, a numerosos formatos.
- Indexar video.
- Realizar análisis sobre Media.
- Crear flujos de procesos.
- Brindar servicios de monitorización y gestión de los procesos.

Identificar los requisitos no funcionales pudiera ser la garantía de lograr una solución de software robusta, es por ello que juegan un papel fundamental en el diseño de la arquitectura que se presenta. Se ha identificado que es necesario:

- Establecer prioridad entre los distintos flujos de procesos.
- Obtener respuestas en el menor tiempo posible.
- Establecer granja de servidores para el procesamiento de Media.
- Los servidores de la granja deben tener altas prestaciones de hardware.
- Implementar el descubrimiento de nuevos servidores dedicados.
- Ejecutar procesos concurrentemente.
- Balancear cargas entre los nodos de la granja.
- Tolerar fallos.

Descripción de la solución propuesta

Para tener una visión clara del funcionamiento de la arquitectura que se propone, se presenta en la Fig. 1 la estructura en forma de pizarra de control que presenta el diseño. El mismo debe permitir organizar y estructurar cómputos, así como software que se encargarán de realizar tareas de procesamiento sobre materiales audiovisuales.

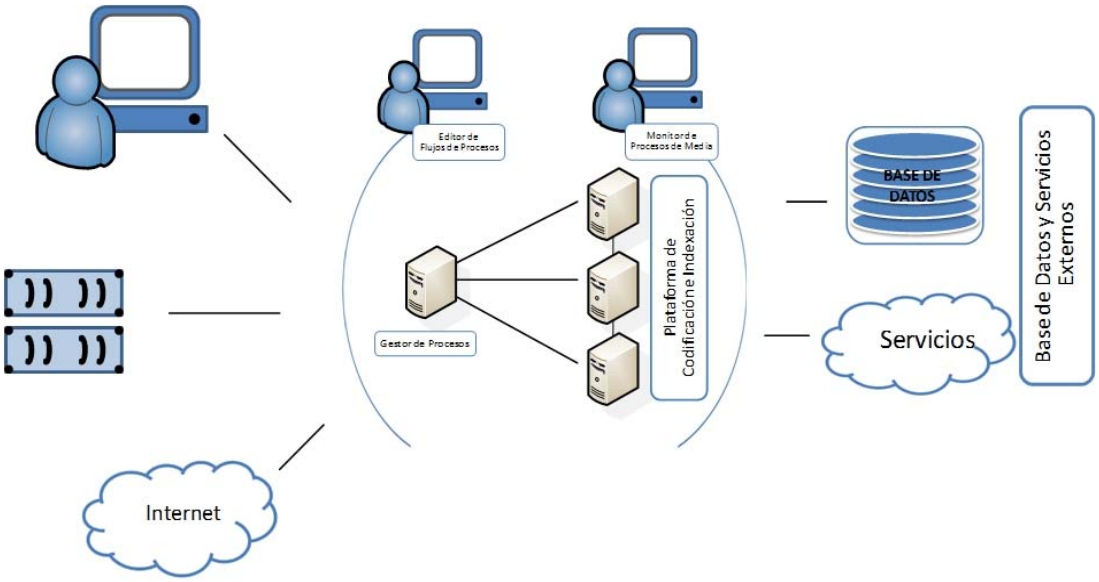

Figura 1: Estructura del Sistema Gestor de Procesos de Media.

Para el desarrollo del Sistema Gestor de Procesos de Media es necesario hacer funcionar como un único sistema al Gestor de Procesos de Media (GPM), Monitor de Procesos de Media (MPM) y la granja de servidores dedicados que pueden ser diversos.

Los servidores dedicados se encuentran compartiendo sus recursos de hardware al GPM. Pueden ser descubiertos por la técnica estática o dinámica. En ellos se encuentran ejecutándose permanentemente algoritmos que realizan diversos procesos sobre materiales audiovisuales. Los mismos son funciones agregadas en forma de plugins (2), que permiten la inclusión de nuevas funcionalidades incrementando así la adaptabilidad del sistema. Todos estos procesos pueden ser monitorizados desde otros sistemas que consumen servicios de la pizarra de control GPM.

El GPM tiene definido los flujos de trabajo que el sistema puede gestionar y ejecutar en sus servidores de la granja. Para ello tiene descrito en un fichero XML la orquestación de los procesos que conforman un flujo y mediante servicios ofrece la posibilidad a terceros de integrarse con él. El XML puede ser editado para insertar nuevos flujos en dependencia de las necesidades de clientes que necesiten hacer uso de los mismos.

Las peticiones que llegarán al GPM las realizarán clientes terceros (sistemas) para pedir la ejecución de un flujo de trabajo o desde el MPM para monitorizarlos. Este último será un cliente de administración y monitorización que permitirá la visualización continua de los procesos que se están ejecutando en el GPM. Además permitirá ejecutar acciones de gestión como: encolado, eliminado, cancelado etc. Este cliente debe utilizar para consumir los servicios una API (3) de consumo desarrollado sobre tecnología orientada a objetos remotos o servicios web, dependiendo del utilizado por el GPM para la publicación de los mismos.

En el diseño la entrada de datos al GPM llega a través de servicios publicados. Estos pueden ser tanto servicios web como objetos remotos. La publicación de servicios para la comunicación entre aplicaciones es configurable, ya que utilizará plugins para la comunicación, lo que proporcionará grandes oportunidades de despliegue debido a la flexibilidad que presenta en cuanto al uso de tecnología de comunicación se refiere. Luego de crear el flujo correspondiente a la petición realizada, esta parte del sistema se encargará de planificar las tareas estableciendo prioridades entre ellas y ejecutándolas sobre los servidores dedicados al procesamiento de media que compartirán recursos a él. Este GPM constituye la pizarra de control de la arquitectura, pues es quien recibe las peticiones de procesamiento, delega el procesamiento a nodos especializados que comparten sus recursos y se mantiene actualizado del estado de los procesos, dando la posibilidad a aplicaciones externas como el MPM a realizar búsquedas y visualizar el estado real de cada uno de los procesos que se ejecutan en el mismo.

Uno de los puntales de la arquitectura que se presenta está constituido por las técnicas de descubrimiento de recursos distribuidos y de balanceo de cargas operacionales.

Dentro de las técnicas estudiadas para el descubrimiento de recursos la dinámica es la idónea ya que presenta altos niveles de portabilidad. Un nodo esclavo que desee compartir sus recursos al GPM será quien le comunique al mismo sus características y definiciones para ser usado como nodo esclavo de la granja de servidores. La técnica estática no se descarta ya que es necesaria la interacción del sistema a desarrollar con recursos de terceros (hardware y software), proporcionando a la solución alto grado de integración.

En el instante que se va a ejecutar un proceso el GPM hará uso de la información real y actualizada que posee de cada uno de los servidores de su granja para seleccionar cuál de ellos es el idóneo para realizar dicha tarea. Para esto hace uso de una técnica de balanceo de cargas operacionales (Conexión Menos Ponderado) que teniendo en cuenta varios parámetros decide cual de los servidores es el indicado. Este sistema es el encargado de tolerar fallos que puedan ocurrir ya que mantiene una caché física de todos sus procesos y define estrategias para varios de los fallos que se han identificado y que puedan presentarse.

"El procesamiento de video requiere alta capacidad de cómputo, es idóneo utilizar en este caso la arquitectura para un sistema distribuido, ya que permite que el procesamiento de la información se distribuya sobre varios ordenadores en vez de estar confinado solamente en uno" [2]. Los sistemas distribuidos poseen características que los hacen adaptables al tipo de situación que se presenta, ya que posibilitan la:

Compartición de recursos: Poseen la capacidad de compartir recursos de software y hardware asociados a los ordenadores de una red.

Apertura: Son en su mayoría abiertos, lo que facilita al equipo de desarrollo que los diseña realizar el diseño sobre protocolos estándares, lo que permite la utilización y combinación de software y hardware de distintos proveedores.

Concurrencia: Pueden ejecutarse varios procesos al mismo tiempo sobre diferentes ordenadores de la red, en ocasiones estos procesos se comunican entre ellos mismos durante su ejecución.

Escalabilidad: Son escalables debido a la capacidad de estos de ir añadiendo nuevos recursos para cubrir nuevos requisitos del sistema [2].

Se propone utilizar el estilo en capas, estructurando la aplicación en varias capas, lo que permitirá llevar el desarrollo a varios niveles y delimitar bien las responsabilidades y el alcance de cada parte del software.

La solución cuenta con cuatro módulos:

Editor de Flujos: Se encargará de la creación visual de flujos de trabajos. Generará un fichero escrito bajo el estándar de orquestación de procesos de negocios BPEL que será necesitado por otros módulos.

Gestor de Procesos de Media: Se encargará de recibir las peticiones de ejecución de flujos de trabajos. Es el encargado de mandar a ejecutar todos los procesos de manera racional en cada uno de los servidores de procesamiento con que cuente el sistema. Es quien mantiene toda la información real del estado de los flujos y de los servidores de procesamiento.

Plataforma de Codificación e Indexación: Encargada del procesamiento que se despliega en los servidores dedicados de la granja. Permitirá agregar de manera fácil nuevas funcionalidades de procesamiento a la misma.

Monitor de Procesos de Media: Encargada de visualizar todos los flujos de procesos que se estarán llevando a cabo en el GPM. Permitirá tomar algunas acciones de gestión como son: cancelar, encolar, eliminar y refrescar. Además permitirá realizar búsquedas de flujos y procesos por distintos filtros, así como especificar a qué Gestor se desea conectar. Esta aplicación será totalmente distribuida por lo que permite realizar sus acciones desde cualquier localización geográfica.

El patrón arquitectónico a utilizar en el caso de las aplicaciones clientes corresponde al Modelo Vista Controlador (MVC), usado principalmente en aplicaciones que manejan gran cantidad de datos donde se requiere una adecuada estructuración del código. Este patrón separa los datos, la interfaz y la lógica de la aplicación en tres capas diferentes facilitando la programación y la reusabilidad del código.

Se pueden citar entre las ventajas del patrón MVC que facilita el mantenimiento en caso de errores, permite el escalonamiento de la aplicación en caso de ser requerido, factor que es de vital importancia en la aplicación que se desarrolla [3].

Tecnologías propuestas

Después de realizar un estudio minucioso de las características de este tipo sistemas, se identificaron un grupo de tecnologías y herramientas para la construcción del mismo, constituyendo tarea fundamental en la elaboración de la arquitectura de un software.

Se determinó que las aplicaciones de procesamiento (Gestor de Procesos de Media, Editor de Flujos de Procesos y Plataforma de Codificación e Indexación) sean de escritorio, lo que logrará un mejor aprovechamiento del hardware. En el caso del Monitor de Procesos de Media será una aplicación web para aprovechar la portabilidad y disponibilidad.

Para las aplicaciones web se proponen utilizar los lenguajes de programación PHP, Java Script y HTML y los frameworks (4) Symfony y Dojo. Symfony se basa en PHP5, es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja y sigue las mejores prácticas y patrones de diseño para la web [4]. Dojo permite la creación de interfaces de usuario amigables, provee de un grupo de componentes visuales de los que se puede hacer uso favoreciendo la usabilidad y minimizando el tiempo de desarrollo de la aplicación [5]. La utilización de estos dos marcos de trabajo permitirá la obtención de un producto funcional y bien acabado. Se propone el entorno de desarrollo integrado NetBeans por ser un editor de código preparado para las tecnologías de desarrollo seleccionadas, es libre y multiplataforma.

En el ambiente de escritorio será conveniente utilizar el lenguaje de programación C++, es orientado a objetos, con él se pueden obtener aplicaciones multiplataforma, ha demostrado ser ideal para el procesamiento de imágenes y materiales audiovisuales, pues permite optimizar la utilización de recursos de hardware de las computadoras. Se propone además el framework Qt, que utiliza a C++ de forma nativa. El entorno de

desarrollo integrado ideal en este caso constituye el QtCreator, posee una ayuda integrada sensible al contexto y es distribuido bajo la licencia GNU GPL v3.0 [6].

Para lograr la comunicación requerida en las aplicaciones de la plataforma se propone utilizar Internet Communication Engine (ICE), un middleware orientado a objetos que proporciona herramientas, APIs, y soporte de bibliotecas para construir aplicaciones distribuidas con el mínimo esfuerzo. ICE permite centrar los esfuerzos en la lógica de la aplicación, pues se encarga de las interacciones a bajo nivel de interfaces de red [7].

Para el almacenamiento de los datos en el GPM se propone usar SQLite descartando los gestores de base de datos más conocidos. A continuación se detallan algunas razones para escoger SQLite como una herramienta de desarrollo:

- Tamaño: SQLite tiene una pequeña memoria y una única biblioteca es necesaria para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas.
- Rendimiento de base de datos: SQLite realiza operaciones de manera eficiente y es más rápido que MySQL y PostgreSQL.
- Portabilidad: se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- Estabilidad: SQLite es compatible con ACID, reunión de los cuatro criterios de Atomicidad, Consistencia, Aislamiento y Durabilidad.
- SQL: implementa un gran subconjunto de la ANSI – 92 SQL estándar, incluyendo sub-consultas, generación de usuarios, vistas y triggers.
- Interfaces: cuenta con diferentes interfaces del API, las cuales permiten trabajar con C++, PHP, Perl, Python, Ruby, Tcl, groovy, etc.
- Costo: SQLite es de dominio público, y por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente.

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca en C. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El

programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción. En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño [8].

Como lenguaje de modelado se propone utilizar el Lenguaje Unificado de Modelado (UML), Visual Paradigm como herramienta CASE y Subversion para el control de versiones, guiando el proceso de desarrollo la metodología Rational Unified Process (RUP).

Vistas de la Arquitectura

El uso de múltiples vistas de la arquitectura permite abordar los intereses de los distintos “stakeholders (5)” por separado: usuarios finales, desarrolladores, ingenieros de sistemas, administradores de proyecto, etc., y manejar los requisitos funcionales y no funcionales separadamente. Muestra como se descompone la estructura de software en componentes y la interacción entre los mismos.

Con el objetivo de lograr abordar los intereses de todos sus clientes la arquitectura se visualizará haciendo uso del modelo 4+1 vista. El mismo describe la arquitectura de software en cinco vistas concurrentes.

- La vista lógica describe el modelo de objetos del diseño cuando se usa un método de diseño orientado a objetos. Para diseñar una aplicación muy orientada a los datos, se puede usar un enfoque alternativo para desarrollar algún otro tipo de vista lógica, tal como diagramas de entidad-relación.
- La vista de procesos describe los aspectos de concurrencia y sincronización del diseño.

- La vista física describe el mapeo del software en el hardware y refleja los aspectos de distribución.
- La vista de despliegue describe la organización estática del software en su ambiente de desarrollo.

Vista lógica:

En esta vista se representa la funcionalidad que el sistema proporcionará a los usuarios finales. Es decir, se ha de representar lo que el sistema debe hacer, y las funciones y servicios que ofrece.

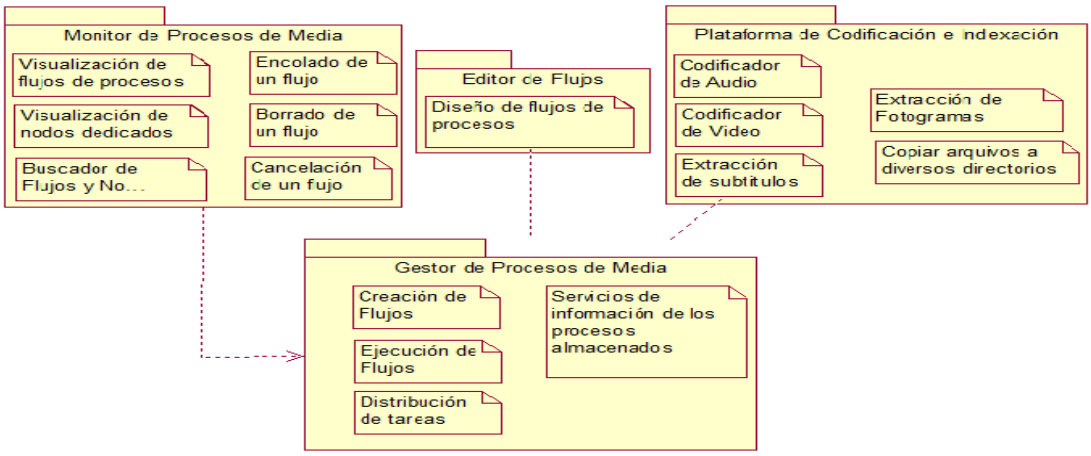

Figura 2: Vista lógica

Vista de Despliegue: En esta vista se muestra el sistema desde la perspectiva de un programador y se ocupa de la gestión del software; o en otras palabras, se va a mostrar cómo está dividido el sistema software en componentes y las dependencias que hay entre esos componentes.

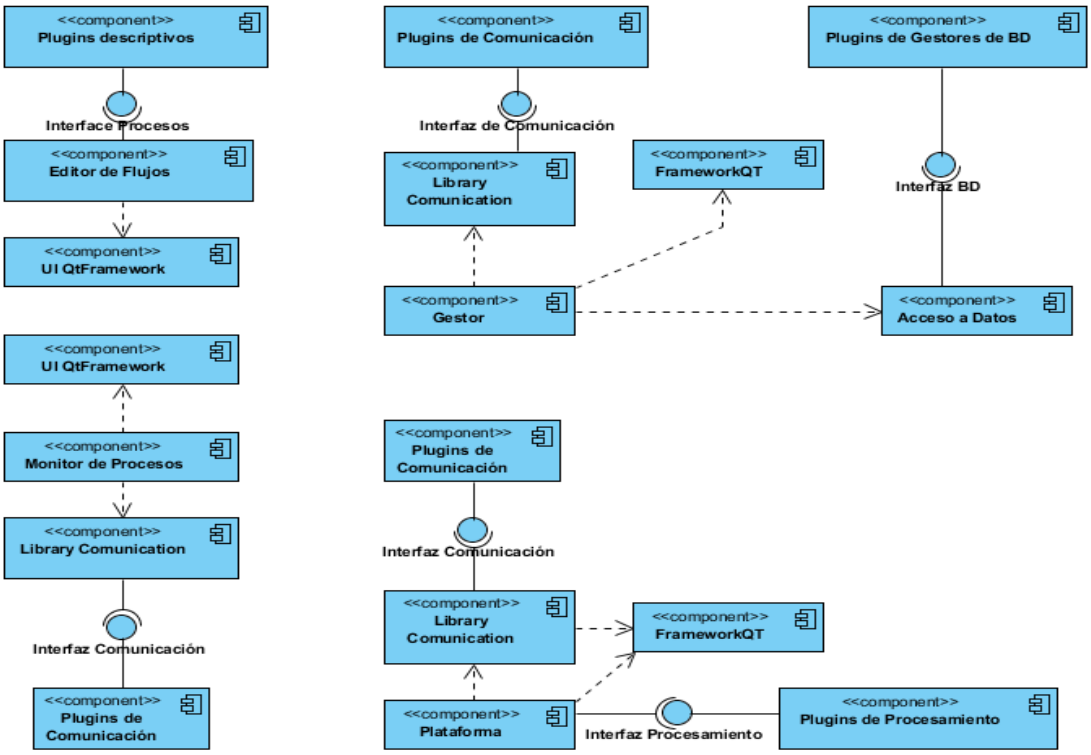

Figura 3: Vista despliegue

Vista de Física: En esta vista se muestra desde la perspectiva de un ingeniero de sistemas todos los componentes físicos del sistema así como las conexiones físicas entre esos componentes que conforman la solución.

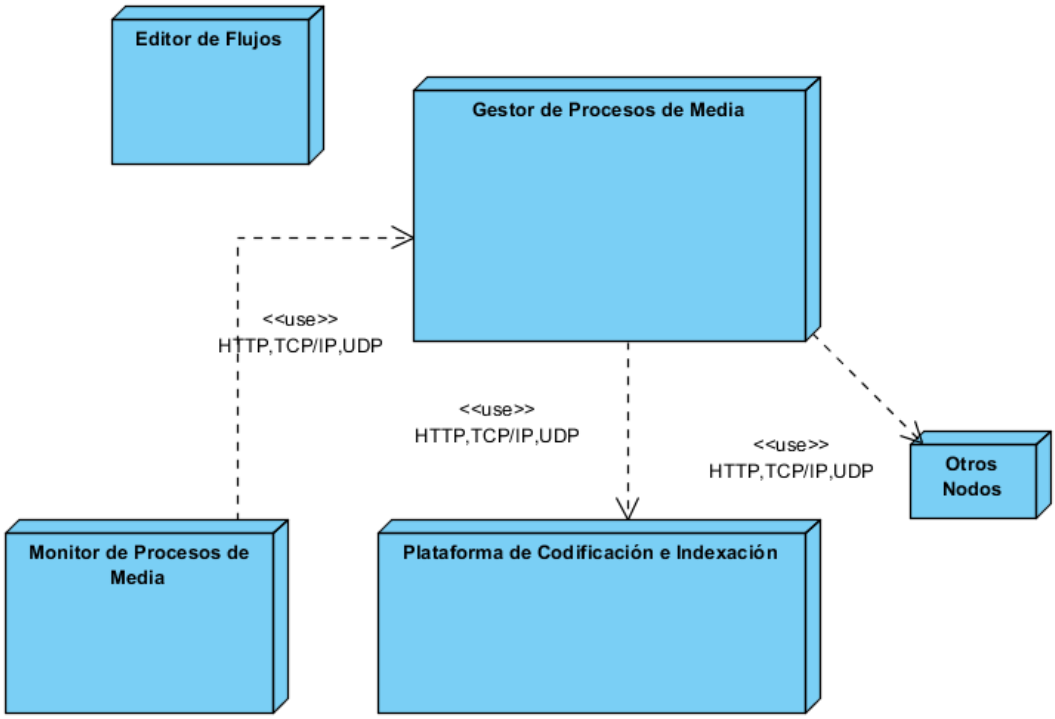

Figura 4: Vista Física

Vista de Procesos: En esta vista se muestran los procesos que hay en el sistema y la forma en la que se comunican estos procesos; es decir, se representa desde la perspectiva de un integrador de sistemas, el flujo de trabajo paso a paso de negocio y operacionales de los componentes que conforman el sistema. Mediante esta vista es posible identificar el paso de un requisito por el sistema hasta su cumplimiento.

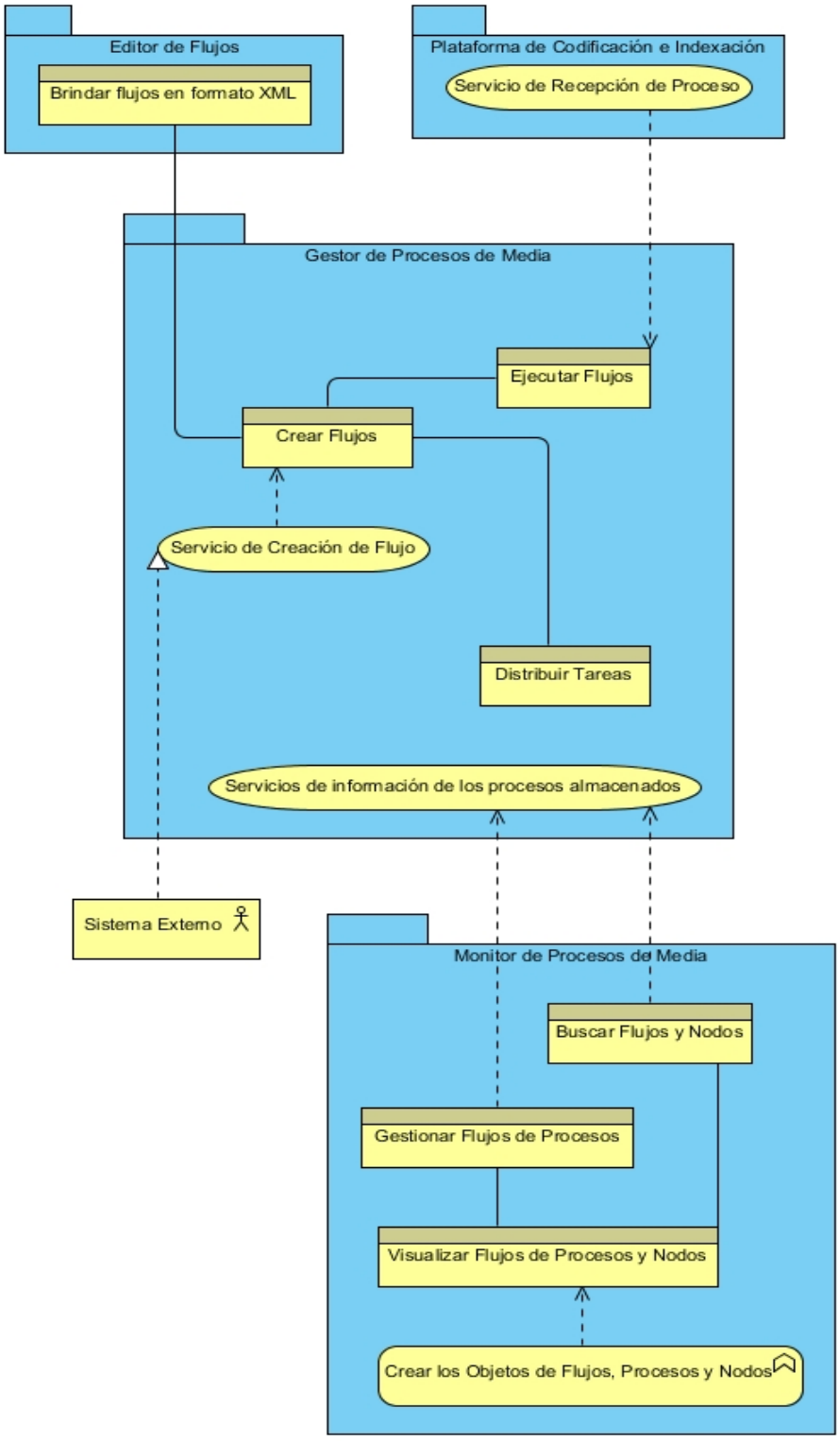

Figura 5: Vista de Procesos

RESULTADOS

La arquitectura propuesta se está utilizando en el departamento de Señales Digitales perteneciente al centro de desarrollo GEySED de la Universidad de Ciencias Informáticas (UCI), como pilar fundamental del SGPM. Se logró una satisfactoria integración entre los cuatro subsistemas obtenidos (Gestor de Procesos de Media, Monitor de Procesos de Media, Editor de Flujos de Media y la Plataforma de Codificación e Indexación) logrando que el producto de software distribuido y descentralizado funcione como un único sistema a los ojos del usuario final. Se logró un balance de cargas equilibrado haciendo uso de una de las técnicas más usadas en ambientes distribuidos. La arquitectura plantea alto grado de escalabilidad permitiendo se incrementen de manera sencilla nodos de procesamiento a la granja de esclavos o cómputos dedicados. Permite además que se utilice la técnica de réplica de servidores y funcionalidades, logrando de esta forma que el sistema sea muy tolerante a los fallos. Se obtuvo un producto de software que en la actualidad está desplegado en soluciones instaladas dentro de la Dirección de Televisión Universitaria de la Universidad de las Ciencias Informáticas y el Instituto Cubano de Radio y Televisión, además tiene como proyecciones para el próximo año su implantación en la Dirección Nacional de Unión de Jóvenes Comunistas y la Oficina de Información del Consejo de Estado.

CONCLUSIONES

La selección de los estilos arquitectónicos y de diseño seleccionados ante las necesidades de un sistema de alta demanda de datos, tributan a un resultado flexible, escalable que le permite interactuar con terceros de manera fácil y rápida, logrando un alto rendimiento en cuanto a velocidad de respuesta se refiere. Dicha arquitectura ha permitido que varias soluciones de procesamiento de media del departamento donde se desarrolla el Sistema Gestor de Procesos de Media, se integren y hagan suyo la lógica con la que cuenta el propio sistema. Se han realizado despliegues en escenarios reales obteniéndose resultados satisfactorios.

Notas:

(1) Un grupo de equipos servidores conectados en red tanto físicamente como mediante software con el fin de proporcionar características de clúster como tolerancia a fallos o equilibrado de carga.

(2) Extensión de un programa. Software que incorpora funciones no previstas en el desarrollo original de un programa. Permite el usuario la posibilidad de agregarle funciones especiales.

(3) Una interfaz de programación de aplicaciones o API (del inglés application programming interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

(4) Se le denomina framework en computación a una estructura conceptual y tecnológica de soporte definida, en base a la cual un software puede ser organizado y desarrollado.

(5) También son llamados interesados o involucrados en un problema determinado, y que necesitan una solución óptima.

Desde el punto de vista del desarrollo de sistemas, un "stakeholder" es aquella persona o entidad que está interesada en la realización de un proyecto o tarea, auspiciando el mismo ya sea mediante su poder de decisión o de financiamiento, o a través de su propio esfuerzo.

BIBLIOGRAFÍA

1. **Casaní, Álvaro Fernández.** *ARQUITECTURAS GRID.* Barcelona : s.n., 2004.

2. **Sommerville, Ian.** *Ingeniería del Software.* 2005.

3. **López, Alejandro Rivera.** Centro Interactivo de Recursos de Información y Aprendizaje. [En línea] 16 de Enero de 2008. [Citado el: 15 de Septiembre de 2011.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rivera_l_a/..

4. librosweb. [En línea] [Citado el: 15 de Septiembre de 2011.] http://www.librosweb.es/symfony/capitulo1/symfony_en_pocas_palabras.html..

5. **Testillano, Rodrigo Tordesillas.** IrisLibre. [En línea] 2011. [Citado el: 15, de Diciembre de 2012.] https://forja.rediris.es/docman/view.php/859/1322/Frameworks_Comparativa_forja.pdf..

6. **Pixelco, . [Cited: Septiembre 15, 2011.].** Pixelco. [En línea] 2011. [Citado el: 10 de marzo de 2012.] http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma.

7. **The Internet Communications Engine (Ice). [Online] ZeroC, . [Cited: Septiembre 15, 2011.].** The Internet Communications Engine (Ice). *2011.* [En línea] [Citado el: 14 de abril de 2012.] http://www.zeroc.com/ice.html..

8. QDiario 12 de 1 de 2011.]. [En línea]

9. **Kruchten, Philippe.** *Planos Arquitect´onicos: El Modelo de “4+1” Vistas de la.*